Model-Based Validation of QoS Properties of Biomedical Sensor Networks

Simon Tschirner [a], Liang Xuedong [b] and Wang Yi [a,1]

[a] *Uppsala University, P.O. Box 337, 751 05 Uppsala, Sweden. Email:* {*simon.tschirner,wang.yi*}*@it.uu.se.*

[b] *Rikshospitalet University Hospital and University of Oslo, P.O. Box 1139, N-0316 Oslo, Norway. Email:* xuedongl@medisin.uio.no.

Abstract

A Biomedical Sensor Network (BSN) is a small-size sensor network for medical applications, that may contain tens of sensor nodes communicating over wireless channels. In this paper, we demonstrate that model-based techniques can be applied to the design and analysis of BSNs without investing on the hardware platform and the time-consuming deployment and measurements.

We present a timed automaton model for BSNs, where the sensor nodes communicate using the Chipcon CC2420 transceiver (developed by Texas Instruments) according to the IEEE 802.15.4 standard. Based on the model, we have used UPPAAL to validate and tune the temporal configuration parameters of a BSN in order to meet desired QoS requirements on packet delivery ratio and network connectivity. The network studied allows dynamic reconfigurations of the network topology due to the physical movements of sensor nodes and also their temporally switching to the power-down mode for energy-saving. Both the simulator and the model-checker of UPPAAL are used to analyze the average-case and worst-case behaviours. Our experiments show that even though the main feature of the tool is model checking, it is also a promising and competitive tool for efficient simulation and parameter tuning. The simulator scales well; it can easily handle up to 50 nodes in our experiments. Even the model checker can handle networks with 5 up to 16 nodes depending on the properties to be checked; these are reasonable numbers for BSN applications in medical care.

Finally, to study the accuracy of our model and analysis results, we compare simulation results by UPPAAL for two medical scenarios with traditional simulation techniques. The comparison shows that our analysis results coincide closely with simulation results by OMNeT++, a widely used simulation tool for wireless sensor networks.

[1] Corresponding author

1 Introduction

Wireless Sensor Networks (WSN) [1] contain hundreds or thousands of sensor nodes equipped with sensing, computing and communication devices. These sensor nodes are distributed in a large area and connected by short-range communication devices over wireless channels. WSNs have a lot of potential applications, e.g., battlefield surveillance, wild-life monitoring and medical applications. In these mission-critical applications, a certain set of QoS requirements on network performance must be satisfied. This poses a number of unique challenges on the design and analysis of WSNs. Due to the severe constraints on hardware platform, dynamic working environments, and self-organizing manner, a key design challenge is to tune the temporal parameters of sensor nodes and to evaluate the network performance without investing on the hardware platforms and the time-consuming deployment and measurement.

In this paper, we demonstrate that model-based formal techniques can be used as an alternative approach to the design and analysis of WSNs to complement traditional simulation-based techniques. We shall study Biomedical Sensor Networks (BSN), which are small-size WSNs for medical applications. A BSN may contain tens of sensor nodes distributed over a limited area. For example, in a medical-care application scenario, a sensor node may be used to measure the temperature of a patient with a certain period. Depending on the application, the sensor node may send the measured data immediately to a specified sink node or when it reaches a threshold value. Due to the hardware constraints on communication devices and limited power supply, the range for wireless communication is highly bounded. Thus a message often has to be forwarded by a number of nodes to reach its destination. For instance, on an accident site difficult to access, there may be many injured persons and the available medics are limited. In such a situation a quickly deployed BSN on the accident victims may be used to collect and transmit vital sign data to a centralized medical server for diagnose and analysis so that proper and efficient medical operations can be carried out. For a concrete application scenario of BSNs, we refer to [2]. Due to the life-critical nature of the application, certain QoS requirements on, e.g., network connectivity and packet delivery ratio must be guaranteed.

The essential difficulty in designing and analyzing a BSN is not in dealing with an individual sensor node in the network, which may be running a simple software. But as the network contains a number of nodes, and these nodes must cooperate to achieve some common goal, the behaviour of such a network is extremely more complicated and difficult to analyze due to non-determinism. For instance, the network topology may be changing dynamically. The sensor nodes may move, disappear, and new nodes may appear from time to time.

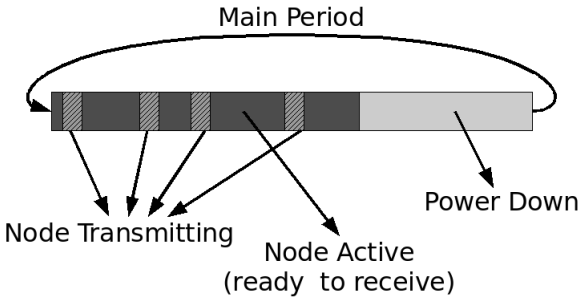


Figure 1. Timing parameters of a sensor node

To achieve the common goal, the sensor nodes in a network need to follow a suitable communication protocol. The IEEE 802.15.4 [3] standard for wireless communication is one of such protocols. It offers different modes for communication and algorithms for signal routing if no direct connection to the sink exists. However, the specification of the standard covers only the logical behaviour of a sensor node in wireless communication. Temporal configuration parameters, such as the active and standby period, of a node must be determined according to the application and the QoS requirements to be satisfied. For example, the application defines how often a sensor node should transmit measured data and the necessary bandwidth. The duration a node spends in the power down mode or in a mode for packet forwarding can also be carefully set to reduce energy consumption.

Fig. 1 illustrates the main timing parameters associated with a sensor node. It has a main period covering three main modes: transmission, reception and power down (sleeping). The behaviour of a node repeats over the main periods. It may, for instance, represent the measurement frequency of the sensor. The second main parameter is the active period. Within a main period, a node may stay active for some time and then switch to the power down mode. When it becomes active again, the node may begin to transmit data immediately or after a short delay. Within the active period, if a node is not transmitting data, it can receive data. The received data may need to be forwarded, which brings the node to the transmission mode again. The technical challenge here is to tune and validate the timing parameters such that the desired QoS requirements are satisfied. In a more complicated scenario, these parameters may be changing in an adaptive manner at runtime for each individual node. In this paper, we shall focus on the case of fixed parameters.

As a concrete example, we study the Chipcon CC2420 transceiver [4] developed by Texas Instruments, which is widely used as the radio communication unit in sensor nodes. The chip implements wireless communication services for sensor nodes, following the IEEE 802.15.4 standard [3]. We shall develop a formal model using timed automata for the transceiver. A BSN based on

such chips is modelled as a network of timed automata. The network studied allows dynamic reconfigurations of the network topology due to the physical movements of sensor nodes among fixed positions and also their temporally switching between active and inactive modes. We have used UPPAAL [5] to find the timing parameters and to validate QoS properties of the network. Both the simulator and the model-checker of UPPAAL are used to analyze the average-case and worst-case behaviours. To demonstrate the usefulness of the technique, we have focused on packet delivery ratio and network connectivity. Our experiments show that even though the main feature of UPPAAL is model checking, it is also a promising and competitive tool for efficient simulation and parameter tuning. The simulator scales well; it can easily handle up to 50 nodes in our experiments. We have also shown how to formalize and check QoS requirements on network connectivity and packet delivery ratio using the UPPAAL query language. Compared with simulations, the model-checker may provide a guarantee on whether (or not) a requirement is satisfied by all possible behaviours of the network. Our experiments show that the model checker installed on a notebook with 1.5GB main memory is able to deal with BSNs of moderate size with 5 up to 16 nodes depending on the properties checked. These numbers are reasonable for BSN applications in medical scenarios. Finally, to study the accuracy of our model and analysis results, we compare the simulation results by UPPAAL with traditional simulation techniques. The comparison shows that our analysis results coincide closely with simulation results by OMNeT++ [6], a widely used simulation tool for wireless sensor networks.

The paper is organized as follows. Section 2 provides a brief survey on existing validation techniques for WSNs. Section 3 describes informally the architecture and main components of a BSN including the sensor node platform, communication protocols, and important QoS requirements for medical applications. In Section 4, we develop an automaton model to describe the behaviour of sensor nodes with focus on wireless communication over dynamically reconfiguring network topologies. Section 5 presents how the model and UPPAAL are used for validation of QoS properties and parameter tuning. Section 6 compares the analysis results with traditional simulation techniques. Section 7 concludes the paper with a summary of results and possible directions for future work.

2 Related Work

The WSN research community has developed a number of emulation tools such as ATEMU [7], Avrora [8], COOJA [9], EmStar [10] and TOSSIM [11]. An emulator provides a virtual operating environment to run the source code (or with minor changes) designed for the real sensor node platform. As it can be used to emulate specific sensor nodes instruction-by-instruction at the

level of clock cycle accuracy, detailed information, e.g., microcontroller status, power and memory consumption can be investigated; so the code can be tested and fine-tuned to achieve the best performance. While the drawback is also obvious, usually an emulator can only analyze a specific sensor node platform. For instance, both Avrora and ATEMU are designed to analyze the AVR microcontroller produced by Atmel and the Mica2 sensor node, which limits their application. Emulators are often used for quantitative evaluation of a individual node behavior, e.g., to precisely evaluate the energy consumption of different components of a sensor node [12].

For the analysis of network level performance of WSNs, currently the most used validation techniques are based on Discrete Event Simulation. There exist well-developed simulators such as NS-2 [13] and OMNeT++ [6] free for academic research, and OPNET [14] and QualNET [15] for commercial applications. These simulators have been further extended with accurate simulation models for various physical components and their access interfaces in WSNs, such as sensors and wireless channels, e.g., Castalia [16] based on OMNeT++ and SensorSim [17] based on NS-2. In these extended simulators, the simulation code (usually written in C or C++), defining the behaviour of sensor nodes and wireless channel configurations, can be executed in the simulation environment. Due to the accurate modelling of physical components, these tools can be used to validate distributed algorithms and communication protocols in a rather realistic setting.

Compared with classical simulation-based techniques, formal techniques are much less explored for the analysis of WSNs. As it is known, formal verification has a limitation with scalability. But it can be used in the early design phase, e.g., to check the correctness of protocols and to identify worst-case scenarios for systems of moderate size, which are very important for mission-critical applications. In [18], Olveczky and Thorvaldsen describe the application of Real-Time Maude for analysis of the OGDC algorithm [19]. The results show that simulations with Real-Time Maude provide a more accurate performance estimation for OGDC than NS-2. The paper also shows that all performance metrics of the algorithm can be measured, and the analysis required much less effort than using a specialized network simulation tool. Automata-based techniques have also been used recently for analysis of wireless communication networks and protocols. In [20], a probabilistic timed automata model of the CSMA-CA contention resolution protocol according to the IEEE 802.15.4 standard is presented. The probabilistic model checker PRISM is able to verify small scenarios of the data transmission in wireless networks. The work compares different configurations and abstractions of the model. In [21], the LMAC protocol is modelled in timed automata and a number of configurations for networks with four and five nodes are systematically analyzed using UPPAAL. The verification results are used to improve the protocol. However, to our best knowledge, there are no published works on validating the temporal

parameters and QoS properties of WSNs using a model checker and comparing with existing simulation techniques for WSNs.

3 Biomedical Sensor Networks

The integration of biomedical sensors with wireless networks has led to the emergence of BSNs [22], which have great potential applications in medical care. Body temperature, blood pressure, electrocardiogram (ECG), Pulse Oximeters (SpO2), and heart rate are sensed and transmitted to a medical center, where the data is used for health status monitoring, and medical analysis and treatment. The main function of BSNs is to ensure that sensed medical data can be delivered to the medical center reliably and efficiently without physical wire-connections. Thus a BSN may contain a number of sensor nodes with a sink node collecting packets for the medical center.

In this section, we describe the basic architecture of the hardware platform for medical sensor nodes and the mostly used protocols for wireless communication in a BSN. We shall also summarize the most common QoS requirements for medical applications of BSNs.

3.1 The Hardware Platform of Sensor Nodes

A sensor node usually consists of five parts: a microcontroller for data processing, sensor(s) for data collection, analog-to-digital converter (ADC) for signal conversion, a transceiver for wireless communication and a power supply unit. Fig. 2 illustrates the basic architecture of a sensor node.

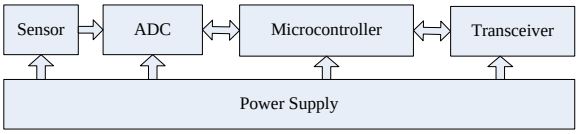


Figure 2. Hardware structure of a sensor node

Due to the severe constraints on power supply and small size requirements, sensor nodes usually use short-range radios for communication. In order to have the capacity of interoperability of sensor nodes from different manufacturers, IEEE Computer Society proposed the IEEE 802.15.4 standard to define the protocol and compatible interconnection for data communication devices in WSNs.

One of the widely used hardware transceivers for wireless communication is the Chipcon CC2420 transceiver, developed by Texas Instruments according

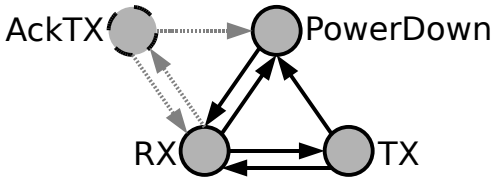


Figure 3. An Abstract State Machine of the Chipcon CC2420 Transceiver

to the IEEE 802.15.4 standard described in the following subsection. The CC2420 is a single chip designed for low-power and low-voltage wireless applications. It provides 250 kbps data rate with high receiving sensitivity (-95dBm). The reference manual of the CC2420 [4] defines the functionality of a CC2420 transceiver by a state machine. Fig. 3 is an abstract version of the state machine with four abstract states. The state machine may be seen as the abstract behaviour of a sensor node. The state transitions may be triggered by either command strobes or internal events, e.g., a timeout. Each of the abstract states represents a group of states in the original state machine. The PowerDown state combines the different energy saving states of a node, which may be entered from any state after the active period (see Fig. 1) of the node has ended. The working states of a node during an active period are abstracted as RX for reception and TX for transmission. RX covers those states of a node, where it may be searching for a signal on the channel and can receive a packet at any time. The abstract state TX covers those states of a node for transmitter calibration, preamble, and frame transmission.

3.2 Communication Protocols

Routing, medium access control (MAC), and physical layer protocols are the most important communication protocols in BSNs. Routing protocols are used to define how data packets are delivered to the sink node during multi-hop communication. As defined in the IEEE 802.15.4 standard, the physical layer is mainly responsible for data transmission and reception, the clear channel assessment (CCA) for carrier sense multiple access/collision avoidance (CSMA-CA), and activation (or deactivation) of the radio transceiver. The MAC sublayer handles all accesses to the physical layer channel and provides a reliable link between two peer MAC entities.

There are two operation modes in the IEEE 802.15.4 network, beacon enabled (slotted) and nonbeacon enabled (unslotted). In beacon enabled mode, communication is synchronized and controlled by a network coordinator, which

transmits periodic beacons to define the start and the end of a superframe. The superframe may consist of an active and an inactive period; the active part of the superframe is divided into 16 equally sized slots and consists of two parts: the contention access period (CAP) and an optional contention free period (CFP). In the contention access period, slotted CSMA-CA is used as channel access mechanism, where the backoff slot aligns with the start of beacon transmission. In the contention free period, time slots are assigned by the coordinator, devices assigned with specific time slots can transmit packets in this period. All communications must take place during the active part. In the inactive part, devices may switch to power down in order to save energy.

In nonbeacon enabled mode, no regular beacons are transmitted. Unslotted CSMA-CA is used as channel access mechanism. When a network device wants to transmit a packet to the network coordinator, it waits for a random number of backoff slots, which is chosen uniformly between 0 and $2^{\mathrm{BE}} - 1$, where BE is the backoff exponent. Then it checks if the channel is idle. If so, the network device begins to transmit the data packet; otherwise BE is incremented by 1, and the network device backs off again with the new value of BE before trying to access the channel. The procedure is repeated until BE exceeds a defined maximum. Similarly the number of iterations is also limited by a maximum number of backoffs before declaring a channel access failure.

3.3 QoS Requirements

In medical applications where data packets usually contain vital medical information on human health, the network used for communication should guarantee that these packets are delivered to the medical center with a certain packet delivery ratio for a given time period. This is one of the most common QoS requirements on BSNs for medical applications [23].

- *Packet Delivery Ratio:* Packet loss can be caused by channel access failure, packet collision, transmission error caused by thermal noise and external interference. The packet delivery ratio for a given node is the ratio of the number of packets received successfully at the sink node by the number of packets sent by the node.
- *Network Connectivity:* Each node should have a connection with the sink node within a certain time period, either connected directly or through multi-hop communication. There should not exist isolated nodes.
- *End-to-end Delay:* Data packets must be delivered to the sink node within a given time delay. The end-to-end delay is normally defined as the time difference that a packet is ready to be sent at a sensor node until it reaches the sink node.
- *Energy Consumption/Network Lifetime:* For long-term patient monitoring

networks, the power consumption of each node must be low to ensure a certain network life-time.
- *Network Throughput:* Network throughput is the amount of data per time unit delivered to the sink node. The network throughput is usually measured in bits per second (bits/s or bps).

In this paper, we will focus on the first two types of QoS requirements and tune parameters for the sensor nodes to ensure that these requirements are satisfied.

4 Modelling BSNs with Timed Automata

A timed automaton is a finite state automaton extended with real-time clocks [24]. UPPAAL [5] is a tool box for timed automata, which provides a modelling language, a simulator and a model checker. In UPPAAL, timed automata are further extended with data variables of types such as integer and array etc., and networks of timed automata, which are sets of automata communicating with synchronous channels or shared variables, to ease the modelling tasks. The modelling language allows to define templates to model components that have the same control structure, but different parameters, which is a perfect feature for modelling of sensor nodes. For a tutorial of UPPAAL and timed automata, we refer to [25,26].

In this section, we develop a UPPAAL model for a BSN, as a network of timed automata where each automaton models a sensor node. As all sensor nodes are implemented with the same chip for wireless communication, running the same protocol, we use a template to model the node behaviour with open timing parameters to be fixed in the validation phase. The network topology is modelled using a matrix declared as an array of integers in UPPAAL. Elements in the matrix denotes the connectivity between pairs of nodes.

4.1 Modelling the Chipcon CC2420 Transceiver

To study the network performance, we need to model only the transceiver of a sensor node for wireless communication. We assume that all sensor nodes use the Chipcon CC2420 transceiver as described earlier. We model the transceiver as a UPPAAL template based on the radio control state machine of the transceiver, described in its reference manual [4]. As described in the previous section, Fig. 3 shows an abstract view of the state machine.

The modelled template is shown in Fig. 4. For a detailed description of data,

clock variables, names of states etc. used in the template, we refer to Appendix A. Most of the states are of the same name as the radio control states in the original state machine for the transceiver. The functionality of the transceiver is modelled by the state transitions according to the reference manual. The timing behaviours, as shown in Fig. 1, are formalized with clock constraints on transitions where the two important timing parameters, the main period (`P_M`) and the active period (`P_W`), are used as clock bounds.

In the real hardware, the main period will be started by an external signal from the sensor with a fixed period `P_M`. The signal indicates that there is a packet to send. We model this simply by a transition with a clock constraint enforcing the periodic behaviour and a buffer assigned with the identity of the packet to be sent. Furthermore, in the real hardware, a node may send an acknowledgement after a successful reception of a packet depending on the configuration of the node. This is implemented implicitly by the dynamic routing scheme as described in the following subsection. Note also that we have added two extra states (i.e. `PreTX` and `Backoff`) to the part of the model concerning packet transmission. These states model the CSMA-CA back-off period in the communication protocol as described earlier.

4.2 Modelling the Network and Communication

Now we describe how data packets are transferred between nodes and how errors are modelled, that may occur during packet transmission. The description is mainly on the global data variables used by the template automaton.

The network topology – the spatial distribution of the sensor nodes – represents the direct connections between the nodes. It is the task of the routing protocol to find a path for a packet from one node to the sink. We model the network topology using a matrix (`topology`) referred as topology matrix. The dimensions of this matrix correspond to the number of nodes in the network. Every element stands for the connectivity from one node (row index) to another (column index). If the matrix should map the topology, negative values can be used, for instance, to represent that a pair of nodes is not connected and positive values can reflect the distance or signal strength between the corresponding nodes. The matrix can also be used to store routing information. In this case, some values can stand for a connection, where a node is in range but not on a routing path.

Using the topology matrix, it is easy to model a fixed routing scheme. The matrix also allows us to model dynamic reconfigurations of the network topology due to the movement of a node or the change of routing information at runtime. To study dynamic reconfigurations, we have modelled controlled

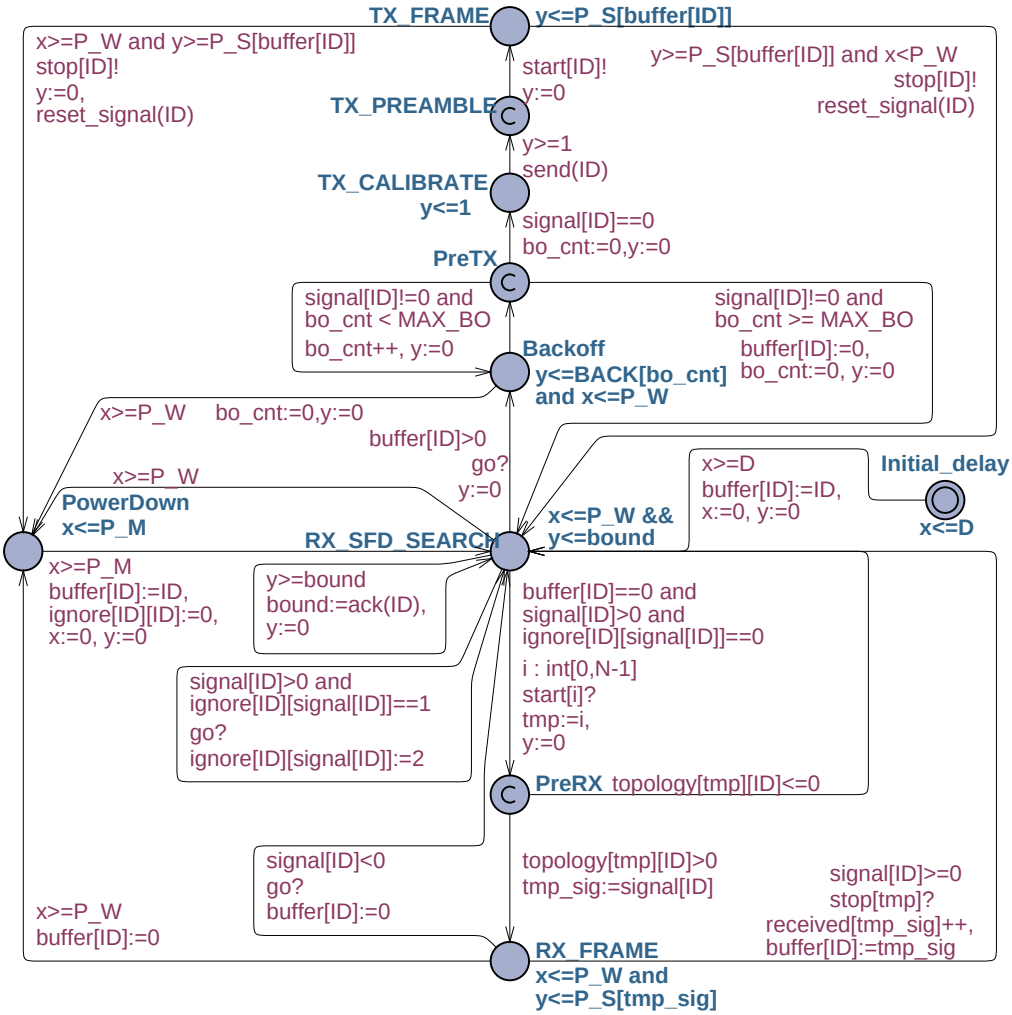


Figure 4. A UPPAAL template for wireless sensor nodes based on the Chipcon CC2420 Transceiver

flooding which is a dynamic routing scheme. A node broadcasts a packet to all its neighbours and remembers every received packet to control this flooding. If a node receives a packet that has been forwarded earlier, it will be ignored, which avoids cyclic forwarding. The model contains a matrix (`ignore`) with which every node remembers the packets it has received so far. The same matrix is used to remember if an acknowledgement is expected or received. In addition to dynamic routing, the flooding scheme offers the opportunity for an implicit acknowledgement: when a node has transmitted a packet, it will most likely receive it again after a short while, because the receiver(s) will broadcast it again. When a defined time after transmission has passed, a node will call a function (`ack`) to check if a packet has to be retransmitted.

For simplicity, we abstract away from the contents of packets. Every node has an unique identifier and if a node emits a packet, it is named by the identifier of the node. The identifier is also used to determine the length of the packet

(`P_S[ID]`). To transmit a packet, a node uses a function named `send`. The function walks through the topology matrix and updates the incoming signal of every node in range, where the incoming signals are modelled by an array named `signal`. Packet collisions that lead to packet losses are modelled with help of the signal array. If a node starts a transmission while another node in range is receiving a signal, the corresponding element in the signal array will be set to a negative value meaning that the packet is corrupted.

5 Validation Using UPPAAL

We consider a BSN as shown in Fig. 5, where the **S**-node is the sink node and the other nodes are modelled by the UPPAAL template presented above, with randomly chosen initial values for the timing parameters as listed in Table 1. The sink node is modelled as a simple automaton. It is not shown in the presentation as its essential behaviour is only to accept packets from the other nodes and keep track of the number of packets received for each node. The network topology is chosen randomly. We shall study the network performance and show how to tune the timing parameters such that certain QoS requirements are satisfied.

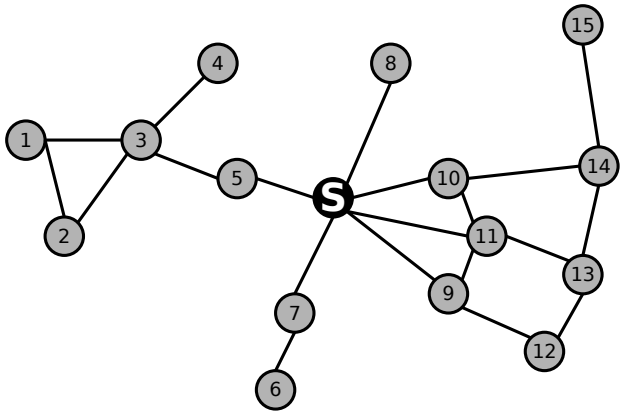


Figure 5. A random topology of a BSN with 15 sensor nodes.

5.1 Symbolic Simulation

We shall first use the symbolic simulator of UPPAAL to study the packet delivery ratio for the network in different settings including parameter tuning and dynamic reconfigurations of the network topology. We have observed that the simulator scales well; for example we can easily handle networks with

Table 1
Timing parameters of the sensor nodes in Fig. 5.

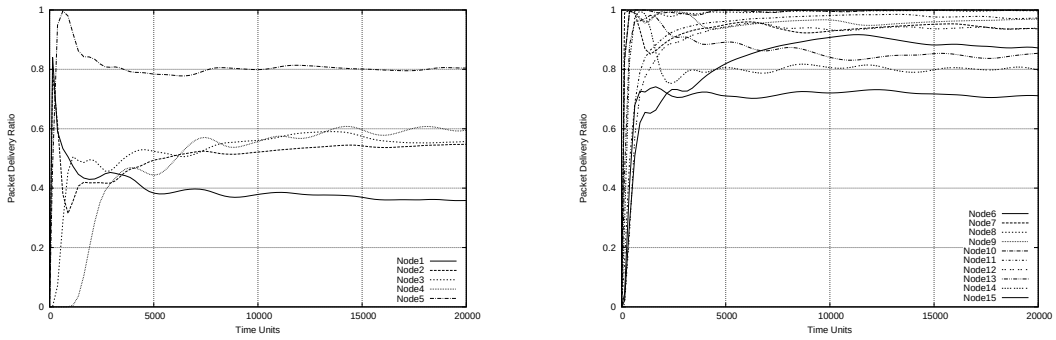
	Initial Parameters		Improved Parameters	
Node	Main Period	Active Period	Main Period	Active Period
1	180	120	180	120
2	240	160	240	160
3	240	160	240	235
4	300	200	300	200
5	300	200	300	290
6	200	100	200	100
7	200	100	200	100
8	240	160	240	160
9	300	200	300	200
10	300	200	300	200
11	180	120	180	120
12	200	100	200	100
13	240	160	240	160
14	300	200	300	200
15	300	200	300	200

50 nodes (and more), which is a big enough number for BSNs applications. However, for the presentation, we consider only the network shown in Fig. 5.

For a detailed description of the UPPAAL simulator, we refer to [25,26]. The simulator may be used to explore symbolic executions of a model. A symbolic execution is a sequence of transitions between symbolic states, corresponding to a collection of possible executions of the system modelled. A symbolic state contains the current control state, current values of data variables and possible clock values represented as clock constraints. As the symbolic states record all the changes of variables and clocks, they can be used to calculate performance metrics to validate all the QoS requirements summarized in section 3. For example to calculate the end-to-end delay for a packet, one may reset a clock in the original model to remember the time point when it is sent. When a symbolic state is found where the packet is delivered successfully, the bounds of the same clock in the found state represent the best- and worst-case delays for the packet. Another example is to extract clock constraints from the symbolic states, on the total time for all nodes staying active. The constraints can be used to estimate energy-consumption.

Parameter Tuning

Now we show how to use the simulator to find timing parameters such that given QoS requirements are satisfied. To show the usefulness of the technique, we focus on simulations for packet delivery ratios only. We simulate the network for 20.000 time units in which the nodes will complete between 65 and 110 main periods. The simulation takes about four minutes. The results are shown as diagrams in Fig. 6, where each curve illustrates the packet delivery ratio of a node, which is changing with time.



(a) Nodes 1 to 5 (b) Nodes 6 to 15

Figure 6. Simulation results for the network in Fig. 5 with initial timing parameters given in Table 1.

From the diagrams, we note that after a startup period, the packet delivery ratios for all nodes are stabilizing above a certain value, which indicates that the network performance is stable. For example, the packet delivery ratio stays above 80% for node 5, and 7 to 15, above 60% for node 6, above 50% for node 2 to 4, and under 40% for node 1, which is the worst of all.

Now if some or all the packet delivery ratios are not satisfactory according to the desired QoS requirements, we may tune the timing parameters of the nodes to influence or improve the network performance. Consider, for example, the QoS requirement: "the delivery ratio for all nodes should be above 60%". If we compare the curves with the positions of the nodes on Fig. 5, we see that node 3 and 5 are bottlenecks for the connection of node 1,2, and 4 to the sink. So we may increase the duration of the active period of node 3 and 5, and hopefully these nodes will be able to forward packets most of the time.

The two new parameters for node 3 and 5 are given in Table 1 where the new ones are in boxes with the rest unchanged. With the new set of parameters, the packet delivery ratios from simulation are shown in Fig. 7. We notice an increase of about 10 up to 40 percentage points for the delivery ratio of node 1 to 4, and the delivery ratios for all nodes are stabilizing above 60% satisfying the above requirement. However, we should be aware that the increased active

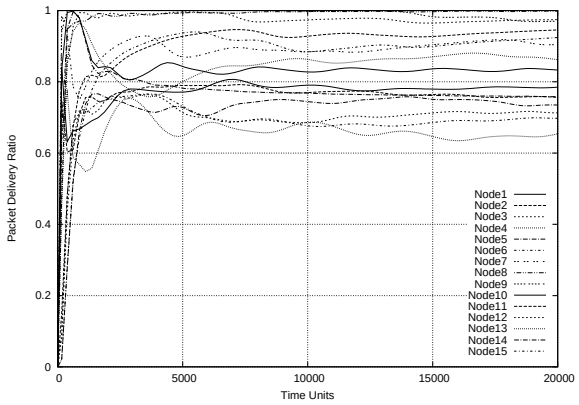


Figure 7. Simulation results for the network in Fig. 5 with improved parameters in Table 1.

periods may lead to a higher energy consumption; one needs to find the right trade-off. In this paper, we will not consider QoS requirements on energy consumption.

Dynamic Network Topologies

Note that the above simulations are dealing with a half-static network in the sense that, at any time point, some of the nodes may switch to the power-down state and disconnect some of the connections such that the network topology changes.

Now we consider the case when some of the nodes are moving among fixed positions. We study the influence of movements of mobile nodes on the behaviour of the whole network. For this purpose we use a timed automaton to manipulates the topology matrix to simulate the movement of mobile nodes.

For example, we allow node 2, 7, and 11 to move according to Fig. 8. The simulation result is shown in Fig. 9. A comparison of Fig. 9 with Fig. 6 shows no essential differences in the packet delivery ratios for all nodes except for node 6. The reason is obvious: depending on the movement of node 7, node 6 may be isolated from all other nodes. We also see that the turning points follow the time points of movements for node 7.

5.2 Verification of QoS Properties

The networks we are dealing with are extremely non-deterministic; any node can communicate with any other node directly or indirectly at any time. With

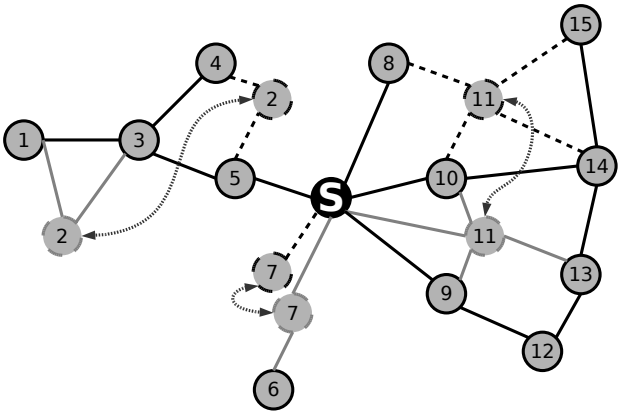


Figure 8. Example movements of mobile nodes.

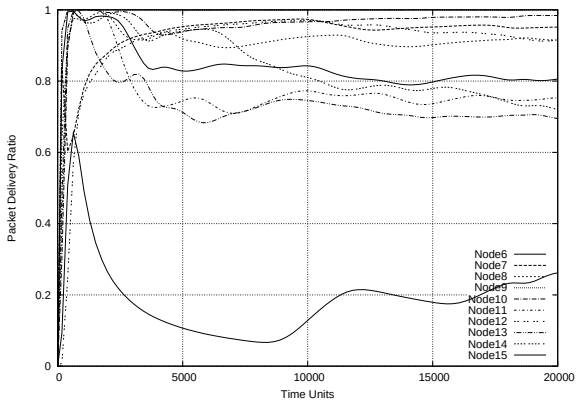


Figure 9. Simulation results for the network with mobile nodes as shown in Fig. 8 and initial parameters given in Table 1.

a simulator, we can explore only possible behaviours to study the average-case performance of a network. To reveal the worst-case scenarios and to check that some requirements are guaranteed by all possible behaviours of a network, we use the model checker of UPPAAL. However the goal here is not to show how powerful the tool is, rather to show that model checking is a useful technique to complement simulations.

We shall use the UPPAAL query language to formalize QoS requirements concerning network connectivity and packet delivery ratio. It is a subset of TCTL formula [27] for timed automata. The formulas are of the following forms:

- `A[]` ϕ — Invariantly ϕ.
- `E<>` ϕ — Possibly ϕ.
- `A<>` ϕ — Always Eventually ϕ.

- `E[]` ϕ — Potentially Always ϕ.
- ϕ `-->` ψ — ϕ always leads to ψ, the shorthand for `A[]` ϕ `imply A<>` ψ.

where ϕ, ψ are local properties on states, i.e., boolean expressions over predicates on control states, integer variables and clock constraints in symbolic states. For further details on the query language, we refer to [25,26].

We have used UPPAAL installed on a notebook with 1.5GB main memory to check the formalized requirements. The model checker can handle networks with 5 up to 16 nodes depending on the properties to be checked, that are reasonable numbers for BSN applications in medical scenarios. The verification results are summarized in Table 2, and a more detailed list of checked example requirements is given in Table B.1 in the Appendix.

Absence of Deadlocks

In general a deadlock means the situation when different processes block each other. In a wireless network this could be caused, for instance, when a node is waiting for an acknowledgement. We should also mention that the deadlock check is very useful for validating the model itself. For example, timing errors in a model may result in deadlocked states. If a clock guard or an invariant of an automaton is violated, the automaton may reach a state where time can not pass, and there are no enabled transitions either. The query for checking deadlock-freeness is: `A[]!deadlock`.

Network Connectivity

As described in Section 3, for BSNs, we are interested in network connectivity to guarantee that each node is connected with the sink node. For this purpose, in the model, we have used an array `received`. Each element of the array (initialized with 0) is a counter associated with a node and incremented whenever the sink receives a packet emitted by the according node. For a node with identity X, we use the query `A<>received[X]>0` to prove or disprove, whether X can eventually establish a connection to the sink. This allows us to find improper timing parameters which result in that some nodes are isolated.

As an experiment to discover disconnected nodes, we change the topology matrix such that node 4 is disconnected; the verification results are shown in Table B.1.

Note that `A<>received[X]>0` states that there will be a connection eventually without a time bound. To estimate the maximal delay, we use the number of main periods of node X. We modify the model such that the counter `periods[X]` is reset whenever the sink receives a message from node X. Then

we can use the query `A[]periods[X]<Y` to prove that node X is connected to the sink at least within Y periods. The array for the number of received packets has no impact on the properties verified here and thus it is declared as a meta variable.

Table 2
Example verification results.

Property	Network Size	CPU Time (Sec)	Memory (MB)
Deadlock-freeness	6 Nodes	2740.44	1620.5
Connectivity	11 Nodes	315.67	73.45
Bounded Connectivity	6 Nodes	157.71	36.7
Packet Delivery Ratio	6 Nodes	252.80	38.6

Packet Delivery Ratio

Recall that the packet delivery ratio of a node is the ratio of the number of packets delivered to the sink by the number of packets sent from the node, and the later is the number of main periods. These numbers are denoted by the counters `received[X]` and `periods[X]` in the model. Ideally we may want to check that over time, the packet delivery ratio of certain packets is over 90%. Unfortunately, in UPPAAL we can not use the query language to specify such properties concerning mean values or duration properties. However, we may run a number of checks to approximate the packet delivery ratio. We may check at least N out of M packets sent will be delivered successfully using the query, `A[]periods[X]>=M imply received[X]>=N`. For instance, for ten packets sent, we may check whether a packet delivery ratio of at least 90% is reached using the query `A[]periods[X]>=10 imply received[X]>=9`. We reset `periods[X]` and `received[X]` when the bounds are reached to assure that the property is not only satisfied after the first ten periods, but whenever ten periods have been completed. We may change the bounds on the numbers of packets sent and received to achieve better approximations.

6 Comparison With Discrete Event Simulation

One of the main concerns in applying model-based techniques is to develop faithful models of systems to obtain faithful analysis results. To study the accuracy of our model, we compare simulation results by UPPAAL with the traditional discrete event simulator OMNeT++, which is widely accepted in the WSN community. We shall see that for two typical application scenarios of BSNs, the UPPAAL simulation results for packet delivery ratio using our model coincide closely with simulation results by OMNeT++. However, we

have also observed some minor differences due to the simplifications in the modelling of packet transmissions and collisions.

6.1 Simulation Settings

To ease the comparison, we study two fixed network topologies as shown in Fig. 10 and 11, corresponding to two typical application scenarios of BSNs in medical care. The first topology for networks with one-hop communication is usually used for in-field patient monitoring where the sensor nodes are deployed in a small area, while the second for multi-hop communication is often used in a large area, where data packets cannot be transmitted to the medical server directly.

In the study with OMNeT++, we use the Castalia WSN simulator [16]. The WSN simulator is configured to follow the physical layer and MAC sublayer protocols as defined in the IEEE 802.15.4 standard. For sensor nodes, we adopt two types of sensors, ECG and temperature sensors with fixed sampling rate and packet size, which are often used in medical care. The ECG sensors emit 5 packets with a size of 132 bytes every second, and the temperature sensors emit 1 packet with a size of 34 bytes every second.

In the study with UPPAAL, the timing parameters including the main periods and transmission delays of our model for the Chipcon CC2420 transceiver are initialized according to the sampling rate and packet size of ECG and temperature sensors. The topology matrix is fixed according to the network topologies shown in Fig. 10 and 11.

Table 3 lists all necessary parameters for simulating sensor networks that are IEEE 802.15.4 compliant. These parameters are used in the configuration of the WSN simulator for OMNeT++, and our transceiver model for UPPAAL simulation. In particular, macMinBE is the initial value of backoff exponent, aMaxBE is the maximum number of backoff exponent, and macMaxCSMABackoffs is the maximum number of backoffs that the CSMA-CA algorithm will attempt before declaring a channel access failure. Note that for simplicity, we have used a fixed *packet overhead* for all packets on physical, MAC and application layer. Note also that for UPPAAL simulation, we have used a simplified collision model – listed as *simplified* in the table – meaning that if more than one sensor nodes within communication range transmit simultaneously, collision happens and all the packets are corrupted.

Table 3
Simulation parameters

Parameters	**OMNeT++**	**UPPAAL**
channel model	log shadowing wireless	fixed bit error rate
path loss exponent	2.4	N/A
collision model	additive interference	simplified
data transmission rate	250 kbps	250 kbps
simulation time	300 s	300 s
packet overhead	32 bytes	32 bytes
macMinBE	3	3
aMaxBE	5	5
macMaxCSMABackoffs	4	4

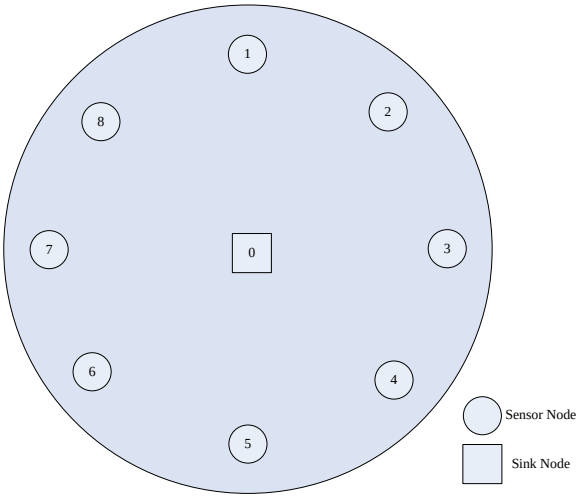


Figure 10. Network topology for one-hop communication

6.2 Experiment 1: one-hop Communication

We fix a one-hop communication network with the topology shown in Fig. 10. In the network, all nodes including the sink node are one-hop neighbours to each other, which means that each node is connected directly to the sink node and all nodes can communicate with each other with one-hop communication. Node 1, 3, 5, 7 are ECG sensors and node 2, 4, 6, 8 are temperature sensors. Node 0 is the sink node to collect data packets sent by the sensors at fixed rate as described above.

Table 4 lists the Packet Delivery Ratio for each node according to OMNeT++ and UPPAAL simulations, respectively, where **Sent** stands for the number

of packets sent by the respective sensor node, **Received** for the number of packets received by the sink node and **PDR** for Packet Delivery Ratio.

Table 4
Packet Delivery Ratios from Simulations with OMNeT++ and UPPAAL for the one-hop communication network shown in Fig. 10.

	OMNeT++			UPPAAL		
Sensor	**Sent**	**Received**	**PDR (%)**	**Sent**	**Received**	**PDR (%)**
Node 1	1500	1441	96.1	1502	1496	99.6
Node 2	300	288	96	301	288	95.7
Node 3	1500	1443	96.2	1504	1499	99.7
Node 4	300	285	95	301	298	99.0
Node 5	1500	1435	95.7	1502	1496	99.6
Node 6	300	289	96.3	301	293	97.3
Node 7	1500	1438	95.9	1502	1497	99.7
Node 8	300	290	96.7	301	294	97.7

6.3 Experiment 2: Multi-hop Communication

Fig. 11 shows a multi-hop communication network. Node 0 is the sink node, node 1 and 2 are ECG sensors, node 3, 4 and 5 are temperature sensors, respectively. Data packets are routed to the sink from node to node by multi-hop transmission.

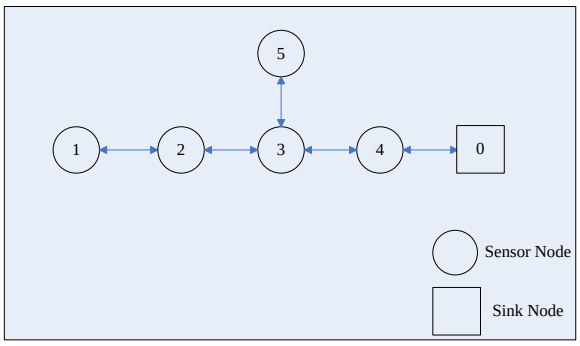


Figure 11. Network topology for multi-hop communication

Table 5 lists the packet delivery ratio for each node according to simulations by OMNeT++ and UPPAAL respectively.

Table 5
Packet Delivery Ratios from Simulation with OMNeT++ and UPPAAL for the multi-hop communication network shown in Fig. 11.

	OMNeT++			UPPAAL		
Sensor	**Sent**	**Received**	**PDR (%)**	**Sent**	**Received**	**PDR (%)**
Node 1	1500	1128	75.2	1546	1508	97.5
Node 2	1500	1266	84.4	1540	1504	97.7
Node 3	300	273	91	308	301	97.7
Node 4	300	285	95	308	308	100
Node 5	300	258	86	308	286	92.8

6.4 Comparison and Discussion

In the first experiment, the results of delivery ratio for each node from UPPAAL and OMNeT++ are similar, indicating that the UPPAAL model is faithful in capturing the behavior of the CC2420 transceiver compared with OMNeT++ simulations. In the second experiment, for both OMNeT++ and UPPAAL, the delivery ratios for node 4 (one hop to sink) are 95% and 100% respectively. Other nodes through multi-hop communication perform worse than node 4 with only one hop. The main difference is that in OMNeT++ the performance of node 1 is the worst, while in UPPAAL the performance of node 5 is the worst. Again the main reason is that the simplified model of wireless channels in UPPAAL, makes the delivery ratio of a node depend less on the number of hops to the sink; whereas the wireless channel is modelled in a more realistic way in OMNeT++. This is explained as follows.

In a WSN, whether a packet can be received successfully by a receiver depends on the Signal to Noise Ratio (SNR), which depends dynamically on the transmitting power, receiving sensitivity, thermal noise, propagation loss, and interferences. For multiple concurrent transmissions, even nodes that are not in the same communication ranges may interfere each other by increasing the background noise level. That is, there are still interferences even though no collision occurs.

For OMNeT++, the results are quite reasonable. Node 1 needs four hops to the sink, during the relaying procedure, packet loss can be caused by channel access failure, collision, and interference from multiple concurrent transmissions. Multi-hop has significant impact on the performance of packet delivery ratio. This explains why the delivery ratio for node 1 with four hops to the sink is not as good as node 5 with one hop less.

For UPPAAL, we observe that the delivery ratio for node 1 is better than

for node 5. The reason is as follows: As the model for wireless channels considers only collision and fixed transmission bit error, propagation loss and interferences due to concurrent transmissions are abstracted away, the channel appears to be perfect if there is no collision. Thus, multi-hop communication has less impact on network performance than it should be. However, the possibilities of collision are modelled close to reality and thus have a relatively higher impact on the packet delivery ratio. From the network topology, we see that collision occurs most likely because node 5 sends a packet to node 3 simultaneously with node 2, which is either transmitting its own packet or forwarding a packet of node 1. Because this collision can only happen, when node 5 is transmitting a packet, node 5 will lose the same number of packets as node 1 and 2 together. Due to the different frequencies for packet emissions of node 1, 2, and 5, within the same time period, node 1 and 2 emit considerably more packets than node 5, thus the number of lost packets of node 5 has a higher impact on the according packet delivery ratio than the number of lost packets for node 1 and 2.

From the simulation results, we may conclude that our model for the CC2420 transceiver is reasonably accurate compared with the WSN simulator of OMNeT++. While the wireless channel model should be improved in UPPAAL – especially for networks with multi-hop communication. However, since the simulation requirements and applied situations are different, certain abstraction must be made to achieve the trade-off between accuracy and verification capacities.

7 Conclusions and Future Work

One of the main challenges in designing a BSN (a small-size sensor network for medical applications) is to validate the network performance such that certain QoS requirements can be satisfied, without investing too much effort on hardware platforms, deployment and measurements for the final implementation. We have shown that model-based techniques can be used to complement the traditional simulation techniques to tune the temporal parameters of sensor nodes and to validate QoS properties of a BSN using automata-based models.

The main contributions of this paper include: (1) We have developed a formal model using timed automata for the Chipcon CC2420 transceiver, which is one of the most used hardware chips for wireless communication in sensor networks. To our best knowledge, this is the first model for such transceivers. We believe that the model can be extended easily to model and validate other transceivers (or wireless communication devices), and communication protocols, that are not necessarily limited to be IEEE 802.15.4 compliant. (2) We have shown how to use the UPPAAL tools to tune and validate the tim-

ing parameters of the sensor nodes such that the desired QoS requirements are satisfied. (3) To study the accuracy of our model and analysis results, we have compared the simulation results by UPPAAL with traditional simulation techniques. The comparison shows that our analysis results coincide closely with simulation results by OMNeT++, a widely used simulation tool for wireless sensor networks. We also observed that using OMNeT++, it is very time-consuming to implement the simulation code in C++, which has the advantage to be more precise for simulating low-level implementation details; whereas with UPPAAL simulations, one can easily tune the model to study network-level performance in the early design phase.

As future work, we shall study the other three types of QoS requirements on end-to-end delay, energy-consumption and bandwidth as well as Jitter i.e., the variation of delay experienced by the sink node. We shall also investigate the mobility degree of a BSN and its impact on QoS properties. Another interesting direction for future work is to develop a logic and extend the UPPAAL model checker to fully capture the five types of QoS requirements. The challenge is to deal with properties concerning mean values efficiently, such as "over the life time of a network, the energy-consumption per time unit is bounded by a certain value".

A Further Description of the Transceiver Model in Fig. 4

A.1 States

- `Initial_delay`: In a real system not all nodes are turned on at the same time; thus we have randomly chosen an initial delay for every node before the first active period. This is to model the initialization of a sensor node.
- `RX_SFD_SEARCH`: In this state, the transceiver is listening to incoming signals.
- `PreRX`: A node that starts packet transmission uses a broadcasting channel provided by UPPAAL to inform nodes in state `RX_SFD_SEARCH` of a beginning transmission. UPPAAL does not allow to broadcast only to a limited number of nodes, thus the `PreRX`-state is needed. If the notification (i.e. the broadcast) comes from a node out of range, it will be ignored.
- `RX_FRAME`: A node will enter this state when it receives a valid signal, and then it will leave the state if the whole packet is successfully transmitted or corrupted.
- `PowerDown`: This state represents the power saving mode of the transceiver.
- `Backoff`: In this state, a node waits for a random backoff concerning the IEEE 802.15.4 CSMA-CA.
- `PreTX`: This state is to model that the transceiver is performing the clear channel assessment.

- `TX_CALIBRATE`: This state models the delay that occurs in the real hardware to set up a transmission.
- `TX_PREAMBLE`: The frame preamble is transmitted during this state. It is modelled by the broadcast channel mentioned above.
- `TX_FRAME`: In this state, the transceiver is transmitting the payload of a packet.

A.2 Constants and Parameters

- `ID`: Each node has a unique ID. This ID is used to identify messages progressing through the network.
- `P_M`: It denotes the main period of a sensor node.
- `P_W`: It denotes the active period of a sensor node.
- `D`: In a BSN, nodes are usually not activated at exactly the same time. This is modelled using an initial delay D.
- `P_S[]`: This array contains the duration of a transmission for every packet type. It can be derived from the packet size.
- `BACK[]`: The bounds for the length of a backoff period depend on the number of re-transmissions, as explained in Section 3.2.
- `MAX_BO`: The maximum number of backoff periods before declaring a channel access failure.

A.3 Global Variables

- `buffer[]`: This is the buffer for outgoing packets of a node. It is globally accessible for statistical purposes.
- `signal[]`: This array models the incoming signal of a node. It is set by the sending nodes in range. If it is 0, it indicates that no node in range is transmitting. If two nodes in range are transmitting, the value for the corresponding receiving node is set to be negative. Otherwise it is set to the packet that is being transmitted.
- `topology[][]`: This is the topology matrix.
- `ignore[X][Y]`: This is to model the controlled dynamic routing scheme with acknowledgement. Each element may have three values: $0, 1$ and 2 denoting the three situations respectively: node X has not received packet Y, node X has received Y for forwarding, and node X has forwarded Y and it has been acknowledged.
- meta `periods[]`: This array contains a counter for the number of main periods of each sensor node. In general, its value is equivalent to the number of packets emitted by a node.
- meta `received[]`: This array contains counters for the number of different packets received by the sink.

A.4 Local Variables and Clocks

- Clock `X`: It is used to model the main period of a sensor node.
- Clock `Y`: It is used to model the transmission times for packets, the back-off period, and the timeout period for acknowledgements.
- `bound`: If a node waits for an acknowledgement, this variable is set to the period for the timeout. Otherwise it is set to be the main period of a sensor node.
- `tmp`: This variable is used to store the ID of the sending node during packet reception.
- `tmp_sig`: This variable is used to store the incomming signal during packet reception.
- `bo_cnt`: This is a counter for the number of passed backoff periods, i.e. the number of failed transmission attempts.

A.5 Channels

- `start[]`: This is a broadcast channel to model the beginning of a packet transmission.
- `stop[]`: This is a broadcast channel to model the end of a packet transmission.
- `go`: This is an *urgent* channel to model urgent transitions.

A.6 Functions

- `send`: This function is called, when a node wants to start a packet transmission. It check through the topology matrix and updates the incoming signals of all nodes in transmission range. In addition the function implements the functionality needed to model collisions.
- `reset_signal`: This function is the complement to the `send`-function. It sets back the value of the incoming signal for each node in transmission range.
- `ack`: This function manages the implicit acknowledgement as mentioned in Section 4.2. First, the function checks for a node if its own packet has been acknowledged. If it is not, the packet will be retransmitted. Otherwise it checks if there is any other packet which has not been acknowledged and puts it for retransmission.

B Example QoS Properties

Table B.1 shows a list of example QoS properties with a more detailed view on the scenarios checked, CPU time and memory requirements for the verifications. The memory consumption is measured by polling the process statistics provided by the Linux operating system. Thus they are only an approximation.

Table B.1
Example QoS Properties Verified.

Property	Scenario	Satisfied	CPU Time (Sec)	Memory (MB)
`A[]!deadlock`	5 Nodes	Yes	0.33	36.66
`A[]!deadlock`	6 Nodes	Yes	2740.44	1620.5
`A<>received[4] > 0`	11 Nodes with node 4 connected	Yes	315.67	73.45
`A<>received[4] > 0`	16 Nodes with node 4 disconnected	No	56.52	111.1
`A[] periods[2] < 4`	6 Nodes	No	157.71	167.61
`A[] periods[5] < 2`	6 Nodes	No	5.59	40.14
`A[] periods[5] < 3`	5 Nodes	Yes	0.23	36.66
`A[] periods[5] < 2`	5 Nodes	No	0.12	36.53
`A[] periods[3] < 3`	5 Nodes	Yes	0.24	36.54
`A[] periods[3] < 2`	5 Nodes	No	0.11	36.53
`A[] periods[1] < 6`	5 Nodes	Yes	0.26	36.66
`A[] periods[1] < 5`	5 Nodes	No	0.24	36.66
`A[]periods[5]>=10 imply received[5]>=8`	5 Nodes	Yes	2.28	38.59
`A[]periods[5]>=10 imply received[5]>=9`	5 Nodes	No	0.28	36.66
`A[]periods[5]>=10 imply received[5]>=9`	6 Nodes	No	252.80	285.98
`A[]periods[4]>=10 imply received[4]>=8`	5 Nodes	Yes	2.26	38.60
`A[]periods[4]>=10 imply received[4]>=9`	5 Nodes	No	0.28	36.66
`A[]periods[1]>=10 imply received[1]>=4`	5 Nodes	Yes	2.27	38.59
`A[]periods[1]>=10 imply received[1]>=5`	5 Nodes	No	0.21	36.66

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (4) (2002) 393–422.

[2] X. Liang, B. Østvold, W. Leister, I. Balasingham, Credo: Modeling and analysis of evolutionary structures for distributed services – user driven requirements, diliverable D6.1, EU IST project, number 33826 (March 2007).

[3] Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs), IEEE Std. 802.15.4 (2003).

[4] Texas Instruments Incorporated, Dallas, Texas, USA, 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. B), CC2420 data sheet (March 2007).

[5] K. G. Larsen, P. Pettersson, W. Yi, Uppaal in a Nutshell, Int. Journal on Software Tools for Technology Transfer 1 (1–2) (1997) 134–152.

[6] A. Varga, The OMNeT++ discrete event simulation system, in: Proc. of the 15th European Simulation Multiconference (ESM'01), Prague, Czech Republic, 2001, pp. 112–118.

[7] J. Polley, D. Blazakis, J. McGee, D.Rusk, J. Baras, ATEMU: a fine-grained sensor network simulator, in: Proc. of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04), Los Angeles, California, USA, 2004, pp. 145–152.

[8] B. L. Titzer, D. K. Lee, J. Palsberg, Avrora: scalable sensor network simulation with precise timing, in: Proc. of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05), Los Angeles, California, USA, 2005, pp. 477–482.

[9] F. Osterlind, A. Dunkels, J. Eriksson, N. F. T. Voigt, Cross-level sensor network simulation with COOJA, in: Proc. of the 31st IEEE Conference on Local Computer Networks, Tampa, Florida, USA, 2006, pp. 641–648.

[10] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, T. Schoellhammer, A system for simulation, emulation, and deployment of heterogeneous sensor networks, in: Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04), Baltimore, MD, USA, 2004, pp. 201–213.

[11] P. Levis, N. Lee, M. Welsh, D. Culler, TOSSIM: accurate and scalable simulation of entire tinyos applications, in: Proc. of the 1st international conference on Embedded networked sensor systems (SenSys'03), Los Angeles, California, USA, 2003, pp. 126–137.

[12] O. Landsiedel, K. Wehrle, B. Titzer, J. Palsberg, Enabling detailed modeling and analysis of sensor networks, Praxis der Informationsverarbeitung und Kommunikation 28 (2) (2005) 101–106.

[13] Computer Science Division, University of California, Berkeley, The NS Manual (2007).

[14] X. Chang, Network simulations with OPNET, in: Proc. of the 1999 Winter Simulation Conference (WCS'99), Squaw Peak, Phoenix, AZ, 1999, pp. 307–314.

[15] Scalable Network Technologies, Inc., QualNet 3.9.5 User's Guide (2006).

[16] H. N. Pham, D. Pediaditakis, A. Boulis, From simulation to real deployments in WSN and back, in: Proc. of the 8th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'07), Helsinki, Finland, 2007, pp. 1–6.

[17] S. Park, A. Savvides, M. B. Srivastava, SensorSim: a simulation framework for sensor networks, in: Proc. of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (ACM MSWiM 2000), Boston, Massachusetts, USA, 2000, pp. 104–111.

[18] P. C. Olveczky, S. Thorvaldsen, Formal modeling and analysis of the OGDC wireless sensor network algorithm in Real-Time Maude, in: Proc. of the 9th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS '07), Paphos, Cyprus, 2007, pp. 122–140.

[19] H. Zhang, J. C. Hou, Maintaining sensing coverage and connectivity in large sensor network, Wireless Ad Hoc and Sensor Networks 1 (1-2) (2005) 89–123.

[20] M. Fruth, Probabilistic model checking of contention resolution in the ieee 802.15.4 low-rate wireless personal area network protocol, in: T. Margaria, A. Philippou, B. Steffen (Eds.), Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006), Paphos, Cyprus, 2006.

[21] A. Fehnker, L. F. W. van Hoesel, A. H. Mader, Modelling and verification of the lmac protocol for wireless sensor networks, Technical Report TR-CTIT-07-09, Centre for Telematics and Information Technology, University of Twente, Enschede (February 2007).

[22] Y. Guang-Zhong (Ed.), Body Sensor Networks, Springer, New York, 2006.

[23] D. Chen, P. K. Varshney, QoS support in wireless sensor networks: A survey, in: Proc. of the 2004 International Conference on Wireless Networks (ICWN'04), Las Vegas, Nevada, USA, 2004, pp. 227–233.

[24] R. Alur, D. L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (1994) 183–235.

[25] J. Bengtsson, W. Yi, Timed Automata: Semantics, Algorithms and Tools, Lecture Notes on Concurrency and Petri Nets LNCS 3098 (2004) 87–124.

[26] G. Behrmann, A. David, K. G. Larsen, A tutorial on UPPAAL, in: M. Bernardo, F. Corradini (Eds.), Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004, no. 3185 in LNCS, Springer–Verlag, 2004, pp. 200–236.

[27] R. Alur, C. Courcoubetis, D. L. Dill, Model-checking for real-time systems, in: LICS90, IEEE press, 1990, pp. 414–425.