MULE-based Wireless Sensor Networks: Probabilistic Modeling and Quantitative Analysis

Fatemeh Kazemeyni[1,2], Einar Broch Johnsen[1],
Olaf Owe[1], and Ilangko Balasingham[2]

[1] Department of Informatics, University of Oslo, Norway
[2] The Intervention Center, Oslo University Hospital, Oslo, Norway

Abstract. Wireless sensor networks (WSNs) consist of resource-constrained nodes; especially with respect to power. In most cases, the replacement of a dead node is difficult and costly. It is therefore crucial to minimize the total energy consumption of the network. Since the major consumer of power in WSNs is the data transmission process, we consider nodes which cooperate for data transmission in terms of groups. A group has a leader which collects data from the members and communicates with the outside of the group. We propose and formalize a model for data collection in which mobile entities, called *data MULEs*, are used to move between group leaders and collect data messages using short-range and low-power data transmission. We combine declarative and operational modeling. The declarative model abstractly captures behavior without committing to specific transitions by means of probability distributions, whereas the operational model is given as a concrete transition system in rewriting logic. The probabilistic, declarative model is not used to select transition rules, but to stochastically capture the result of applying rules. Technically, we use probabilistic rewriting logic and embed our models into PMaude, which gives us a simulation engine for the combined models. We perform statistical quantitative analysis based on repeated discrete-event simulations in Maude.

1 Introduction

Formal methods traditionally consider qualitative properties of models such as various correctness properties. However, many communities (additionally) expect quantitative analysis results, which can be difficult to obtain for such models. In contrast, approaches based on probability distributions over possible transitions are able to provide numerical results; for example, the probability of reaching a certain state with a given probability for message loss. *Probabilistic rewrite theories* [17] form a semantic framework for system specification which is capable of specifying both nondeterministic and probabilistic behaviors of systems, extending rewriting logic [21]. Probabilistic rewrite theories can be used instead of traditional rewrite theories to model networks with different probabilistic and nondeterministic behaviors. The execution of models given as probabilistic rewrite theories can be simulated using the Maude rewriting tool [6], which allows tool-supported analysis.

In this paper, we apply a combination of *operational specifications* of behavior, given as a transition system, with *declarative specifications*, given by probability distributions, using probabilistic rewrite theories. Abstract declarative specifications are used to underspecify behavior when it is difficult to predict the exact behavior of the model in terms of specific transitions, whereas operational specifications are used otherwise. This way, the probability distributions are not associated with the choice of transitions rules, but rather with the outcome of applying transitions. Combining declarative and operational specifications as a means for underspecification can in some cases remove oversimplifying assumptions from the operational model; this makes the resulting specifications more realistic while they can still be analyzed using quantitative techniques. Using Maude to simulate the basic behavior of models given as probabilistic rewrite theories, we apply a statistical quantitative analysis method based on discrete-event simulation, in order to obtain numerical results about the combined model.

The proposed modeling approach is illustrated by a case study in the domain of underwater wireless sensor networks (UWSNs). WSNs consist of small nodes with sensing, computing, and communication devices, which collaboratively monitor and collect data from the environment. Resource limitations in WSNs raise the importance of efficient communication protocols among sensor nodes. Especially, limitations of energy resources need to be considered in order to improve the longevity of the nodes [26]. Data transmission is expensive with respect to power, therefore, the management of communication between nodes is an important factor for network power efficiency. In UWSNs [7], communication uses acoustic data transmission through water. Due to its acoustic nature, transmission costs more power than in usual WSNs, and message loss may occur. One approach to UWSNs is Mobile Ubiquitous LAN Extension (*MULE*) systems [29]. A (data) MULE is a mobile object, such as a vehicle with large and replaceable energy resources. A MULE system consists of a three-tier architecture: (i) *sensor nodes*, in the lower level, which gather data; (ii) mobile agents as MULEs, in the middle level, which move around in the network area and collect nodes' data using single-hop short range transmission; and (iii) access points or *sink* nodes, in the upper level, which receive the data from the MULEs. MULEs move independently from the sensors, and in most cases randomly or following predefined paths. The MULE architecture is an energy efficient solution for data gathering in WSNs that is also scalable and flexible with respect to the network size [3].

This paper develops a probabilistic model that is a combination of declarative and operational models for data collection in a MULE-based WSN, extending a grouping protocol introduced in [16]. In this protocol sensor nodes form groups, using coalitional game theory, in order to save energy in the network. A group has a selected node called *leader* which is responsible for receiving data from the group members and for communication with the outside of the group. To further improve energy efficiency, MULEs gather the data from group leaders. We model MULEs by using a probability distribution of the MULEs' locations in order to abstractly model their movement and the rate of message loss. We combine this declarative specification of MULE-based communication with an

operational model of the grouping protocol in rewriting logic [21], and use the Maude tool [6] to simulate the stochastic behavior of the resulting model. Combining a series of Maude simulations, we obtain numerical insight about the behavior of this protocol. The numerical results show that using the grouping protocol is beneficial to MULE-based WSNs with respect to energy conservation.

Related work. Protocol validation is mostly done with simulation-based tools, using NS, OMNeT+. Formal analysis techniques are much less explored in the development and analysis of WSNs, but start to appear. Among automata-based techniques, the TinyOS operating system has been modeled as a hybrid automaton [9] and UPPAAL has been used for analyzing the LMAC protocol [11] and the temporal configuration parameters of radio communication [30]. A recent process algebra for active sensor processes includes primitives for, e.g., sensing [8]. Ölveczky and Thorvaldsen show how a rich specification language like Maude is well-suited to model WSNs, using Real-Time Maude to analyze the performance of the OGCD protocol [24].

In this paper, we use probabilistic rewrite theories [17] as the formal modeling language and the Maude tool to develop a grouping protocol for MULE-based WSNs that exhibit probabilistic behavior, building on a protocol proposed in [16], which applies coalitional game theory but does not consider message loss and probabilistic modeling. From the modeling point of view, PRISM [19] is another probabilistic modeling language that comes with probabilistic model-checking and quantitative analysis tools [18]. Some process algebraic approaches to modeling, verification, and analysis of probabilistic models are the PEPA [13] and EMPA [5] frameworks, the Probabilistic KLAIM coordination language [25], and the Stochastic π calculus [27]. PMaude, the probabilistic extension of Maude, is a rewrite-based modeling language. PMaude offers a natural way to describe the structures considered in Stochastic CLS, so from a modeling perspective, it is more suitable for our purpose. In contrast to PRISM, PMaude cannot verify quantitative properties. The VeStA [28] tool, which support both PMaude and PRISM, fails when running as big state spaces as we have in our model. As a solution, we take Maude extended with probabilistic rules, using sampling from given distributions, and add a tailor-made external layer producing quantitative results by repeated probabilistic simulations. Consequently, we do not perform stochastic model checking as VeStA offers, but our analysis can provide some quantitative information as well as diagrams of attribute values during one simulation and the average of the values of different simulations, which are important for understanding and comparing a protocol's efficiency. The Real-Time Maude approach [24] has also been combined with probabilistic model-checking to analyze the LMST protocol [15]. They use VeStA to perform statistical model checking, while in our approach, a probabilistic rewrite theory is used to build the combined declarative and operational model with a simple discrete time model. The PVeStA tool [31] is a client-server-based parallelization of VeStA. The CaVi tool combines simulation in Castalia with probabilistic model-checking [10]. There are some works that follow the same approach as ours, but in different fields. For instance, the authors of [4] use stochastic ab-

straction and model checking for the communication system of the airplanes. We work on the higher layers of the network and use rewriting logic for our analysis, in contrast to the BIP toolset that is a component-based framework.

Different aspects of UWSNs have recently been studied. In [7] several research challenges in this area are discussed, while [26] provides an overview of networking protocols for UWSNs. Recent studies on the energy conservation in WSNs are surveyed in [3]. Cluster-based protocols have been studied in some research such as [22], which proposes a cluster-based routing protocol for UWSNs, regardless of the nodes' locations. A well-known work related to energy efficiency of WSNs is LEACH [12], a cluster-based protocol that uses randomized rotation of local cluster-based stations to distribute the energy load among the sensors. MULEs have not only been used in UWSNs, but also for other kinds of WSNs, see, e.g., [14]. We combine a MULE-based architecture and grouping of nodes, in order to increase the energy efficiency of WSNs.

Paper overview. Section 2 summarizes probabilistic rewrite theories. Section 3 describes the grouping protocol in MULE-based sensor networks, and Section 4 introduces our declarative model of MULE-based communication. Section 5 describes the proposed formal model, while the methods for statistical quantitative analysis are introduced in Section 6. The paper ends with Section 7, containing the conclusions and suggested future work.

2 Probabilistic Rewrite Theories and PMaude

Rewriting logic (RL) extends algebraic specification techniques with transition rules: The dynamic behavior of a system is captured by rewrite rules supplementing the equations which define the term language. A *rewrite theory* is a tuple (Σ, E, L, R) where the signature Σ defines the function symbols, E defines equations between terms, L is a set of labels, and R is a set of labeled rewrite rules. Rewrite rules apply to terms of given sorts. Sorts are specified in (membership) equational logic (Σ, E). When modeling computational systems, different system components are typically modeled by terms of suitable sorts defined in the equational logic. The global state configuration is defined as a multiset of these terms. From a computational viewpoint, a rewrite rule $t \longrightarrow t'$ may be interpreted as a *local transition rule* allowing an instance of the pattern t to evolve into the corresponding instance of the pattern t'. Formal models defined in rewriting logic [21] are executable in Maude [6]. Maude provides a tool framework that includes tools such as a reachability analyzer, an LTL model checker, and InVa (invariant model checker for infinite state-spaces).

Probabilistic rewrite theories form a general semantic framework for the specification of systems with both nondeterministic and probabilistic behavior [17]. In [17] it is shown that probabilistic rewrite theories represent a *unifying* semantic framework, i.e., that certain mappings exist between several different probabilistic modeling formalisms and probabilistic rewrite theories. This framework is an extension of *rewrite theories* [21], capturing the evolution of a system through a series of *conditional probabilistic rewrite rules* with the syntax

$$t(\vec{x}) \longrightarrow t'(\vec{x},\vec{y}) \textbf{ if } cond(\vec{x}) \textbf{ with probability } \vec{y} := \pi(\vec{x}), \tag{1}$$

where $\vec{x}$, $\vec{y}$ are sets of variables and $t(\vec{x})$, $t'(\vec{x},\vec{y})$ are terms in an algebra of fully simplified terms, with respect to a membership equational theory and a collection of structural axioms [21]. Also, $cond(\vec{x})$ is a condition that needs to be met for the rewrite (1) to take place and π is a probability distribution over a set of substitutions for $\vec{y}$, possibly depending on the variables $\vec{x}$ of the term $t(\vec{x})$. Such rules are nondeterministic, as the variables $\vec{y}$ in their right-hand side do not also appear in the left-hand side. The notation := in (1) can be understood as a standard `let` expression in functional languages, allowing us to specify the probability distribution which the variables $\vec{y}$ follow.

PMAUDE is introduced in [1] as a specification language for general probabilistic rewrite theories. In general, probabilistic rewrite rules such as (1) are nondeterministic, as the variables $\vec{y}$ in their right-hand side do not appear in the left-hand side, rendering them *nonexecutable* in Maude. However, Maude can be used to *simulate* a PMAUDE specification, provided that all variables $\vec{y}$ in rules like (1) are replaced with actual values *sampled* from the probability distribution $\pi(\vec{x})$. Thus, the executable Maude conditional rewrite rules have the form

$$t(\vec{x}) \longrightarrow t'(\vec{x}, \texttt{sampleFromPi}(\vec{x})) \textbf{ if } cond(\vec{x}),$$

where $\texttt{sampleFromPi}(\vec{x})$ is an operation that samples from the probability distribution π in (1). The same paper [1] introduces a technique, namely an Actor PMAUDE module, which can be used to create executable PMAUDE specifications that are free from any source of nondeterminism. This is achieved by considering the current state of the system as a multiset of *objects* and *messages*, in which, time is made explicit through a global floating point value. In an executable PMAUDE specification all rewrite rules are *scheduled* to execute at random moments of time, with the interval between two consecutive executions following an exponential probability distribution. Recall that the exponential distribution has cumulative distribution function $F(x) = 1 - \mathrm{e}^{-\lambda x}$, where $\lambda \in \mathbb{R}$ is called the *rate* parameter. As shown in [1], a stochastic time model can be implemented in the following manner: A `Configuration` is the sort of the state of a subsystem, to which the rewrite rules typically apply. In order to handle scheduling of the concurrent objects, *time* is added to the global configuration of the system, and the sorts *execution mark* and *scheduled execution mark* are added as subsorts of `Configuration`.

```
subsort Time ExecMark ScheduledExecMark < Configuration .
op time: Float → ExecMark .
op execute :Oid → ExecMark .
op [_,_] :Float ExecMark → ScheduledExecMark .
```

Here, `Oid` is the sort of object identifiers. The scheduled execution marks form the main ingredient of the stochastic time model introduced in [1], making it possible to quantify and resolve nondeterminism. A *tick* operation then makes the system evolve by unwrapping the scheduled execution marks into unscheduled ones and rendering exactly one object active. `Config` is the sort of the

global system, obtained from terms of sort `Configuration` by adding a pair of curly brackets:

```
op tick : Config → Config .
op {_} : Configuration → Config .
```

The motivation for having a *global configuration* sort is that, in order to specify the scheduling mechanism, the whole current configuration of the model must be considered. The semantics of the *tick* operation follows that of Actor PMAUDE [1], selecting the next object for execution in chronological order:

```
op tickAux : Float ExecMark Configuration → Config .
var CF : Configuration . vars T T' : Float . vars E E' : ExecMark .

eq tick({[T, E] CF}) = tickAux(T, E, CF) .
eq tick({CF}) = {CF} [owise] .
ceq tickAux(T, E, [T',E'] CF) = tickAux(T', E', [T,E] CF) if T' < T .
eq tickAux(T, E, CF time(T')) = {E CF time(T)} [owise] .
```

Here, **owise** equations are used only when no other equations apply and **ceq** indicates conditional equations. The global system configuration will contain exactly one time object `time(T)`. Execution marks of form `execute(O)` are added to the left-hand sides of all rewrite rules for an object `O`, as well as *scheduled* execution marks of form `[T+`δ`,execute(O)]` to their right-hand side, in order to make the new subconfiguration active at a later time, after a random interval of time δ has passed, following an exponential probability distribution with some fixed rate parameter, in our case 0.1. The random length of this interval is generated using a Maude operation denoted `sampleExpWithRate` (see Section 5). In the current implementation, the rates corresponding to the exponentially distributed waiting times of all scheduled execution marks are equal to 0.1. However, these rates can be given different values for each sensor, to simulate different sensor processor speeds. The tick rule `{ CF } ⟶ tick({ CF })`, used when `CF` contains no execution mark, is built into our analysis through the script producing quantitative results. The tick rule advances time T and creates an execution mark.

3 Grouping Nodes in MULE-based Sensor Networks

In WSNs, when a large number of sensor nodes are placed in the environment, neighbor nodes may end up being very close to one another. In this case, the transmission power level for communication with a neighbor can be kept low by using short-range multi-hop communication. Since nodes can *cooperate* to transmit data, multi-hop communication in sensor networks is expected to consume less energy than traditional single-hop communication [2]. Furthermore, multi-hop communication can effectively overcome some signal propagation effects experienced in long-distance wireless communication.

Grouping is a method of cooperation between nodes, to transfer data, in which nodes belong to distinct groups [20]. Each group has a *leader*; i.e., a node which is responsible for receiving data from the group members to route it to the

sink, and also for communicating with other leaders. Outside the group, nodes always use their maximum transmission power. Instead, by cooperating with the group members, nodes can use their minimum transmission power to reach the group leader, and consequently decrease the power consumed for communication inside the group. There are different approaches to group formation. The grouping can be done based on distance. For better grouping, other factors such as signal interference may also be considered. We use the grouping algorithm based on coalitional game theory proposed in [16], considering the grouping problem for WSNs as a coalitional game, in which the sensor nodes are the players and the game is concerned with whether a node should join a group or not, as well as which group is more beneficial to join. By using this algorithm, sensor nodes in our model can find a suitable group to join after each movement. In the model, nodes move to different locations according to a predefined set of movements.

4 A Declarative Model of MULE-based Communication

In WSNs, nodes gather data from the environment and transmit them to sink nodes using data messages. We consider an extension of the grouping protocol in [16], in which nodes send messages to their group leaders and MULEs are responsible for moving around leaders to collect these messages and transmit them to sink nodes, in order to decrease the overall energy consumption of the network. Leaders always use their minimum power to communicate with MULEs. Also, nodes can send data messages at different rates. In general, it is better for the network to have a *fair message propagation*, in which nodes have equal message transmission rates, as it causes fair distribution of the power consumption in the network. Thus, in order to model the propagation of data messages, we assume that the next node to send a data message is selected uniformly from the set of all sensor nodes. According to this distribution, at each time tick, a node can send a data message with the same probability as the other nodes, namely $1/N$ where N is the number of nodes in the network. Single nodes communicate directly with the MULE using the maximum amount of power P^{max}. However, the nodes which belong to groups can send their data messages to the group leaders using minimum power P^{min}, and the leaders will send them to the MULE through short range communication.

Besides the modeling of data messages, the movements of MULEs are modeled using an abstract probabilistic approach to underspecify their concrete movements. The general assumption is that the MULE's movement is either random or mostly predefined [29]. Thus, we do not attempt to model a MULE's specific movements, but rather assume that the MULE always moves around the leader nodes, to increase the chance of successfully receiving messages. More precisely, Fig. 1 shows the type of probability density that we assume for locating the MULE at different coordinates. In this example, we considered three leaders at positions $(2, 3)$, $(10, 6)$, and $(4, 9)$. This probability is equal to the probability of successful message transmission between a data MULE and a leader. Outside the communication range of the leaders, the probability density breaks down to a

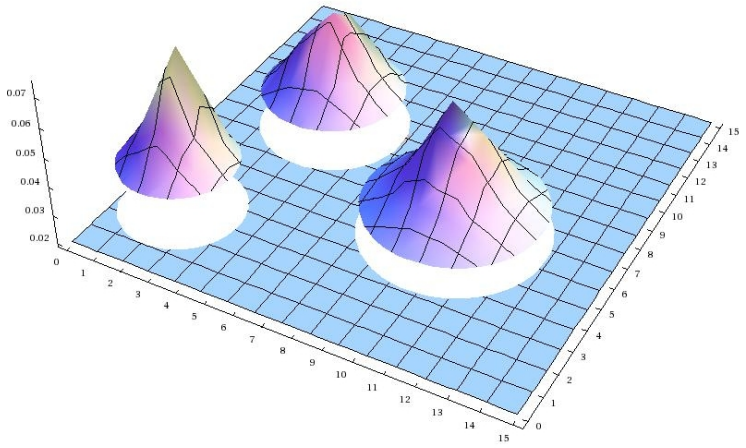

Fig. 1. Three-dimensional plot of the probability density function $f_{X,Y}(x, y)$ giving the probability of successful message transfer between a MULE found at position (x, y) and one of the leaders.

small constant value (in our case study 0.02). Figure 2 shows a two-dimensional density plot of the probability of successful message transfer between a data MULE found at polar coordinates (r, θ) with respect to a leader which is the *pole* of the polar coordinate system. The darker gray towards the center of the circle indicates higher values of the probability density function, while lighter gray indicates lower values. Notice from this diagram how the distance r between the leader and the MULE is calculated, as well as the angle θ between them. We may write

$$P = c \sum_{i=1}^{l} W_i, \tag{2}$$

where $c \in \mathbb{R}$ is a normalizing constant and $W_i \in \mathbb{R}$ is a weight corresponding to the chance of the MULE to be in the communication range of leader $i \in \{1, 2, \ldots, l\}$. We suggest to define this weight through the following formula

$$W_i = \int_0^{2\pi} \int_0^{Rmax_i} w_i(\theta, r) \, \mathrm{d}r \, \mathrm{d}\theta, \tag{3}$$

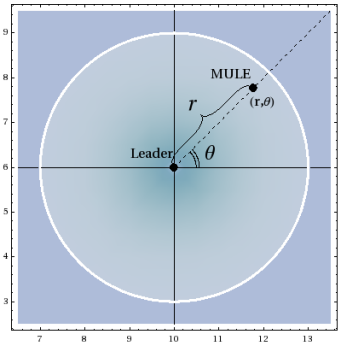

Fig. 2. Two-dimensional plot of the probability density function.

with the intuition that the value $w_i(r, \theta) \in \mathbb{R}$ corresponds to the chance of successful communication between leader i and the MULE, where the polar coordinates of the MULE are given by $(r, \theta) \in [0, +\infty) \times [0, 2\pi)$ and considering that the leader is the *pole* of the polar coordinate system. Thus, the double integral in (3) calculates the "accumulated" weight associated with the leader i over the interior of the circle centered at i, with radius equal

to $Rmax_i$, the communication range of i. The energy consumption of the leader i, necessary to communicate with the MULE at a distance $r > 0$, is directly proportional to r^2 [23]. By making the natural assumption that the probability p_i of successful communication between i and the MULE is inversely proportional to the consumed energy, we obtain that p_i is inversely proportional to the squared distance r^2, which is the same order of magnitude as $(1+r)^2$. We prefer the latter expression since $1/(1+r)^2$ is well-defined for all $r \geq 0$, while $1/r^2$ is undefined for $r = 0$. Thus, we consider the weight function

$$w_i(r, \theta) = \frac{1}{2\pi} \cdot \frac{1}{(1+r)^2} \tag{4}$$

where the factor $1/(2\pi)$ corresponds to the assumption that there is an equal chance for the MULE to be located at any angle $\theta \in [0, 2\pi]$ around the leader i. In this case, we obtain a closed form expression for the weight W_i in (3):

$$W_i = \int_0^{2\pi} \int_0^{Rmax_i} \frac{\mathrm{d}r\, \mathrm{d}\theta}{2\pi(1+r)^2} = \frac{Rmax_i}{1 + Rmax_i} \tag{5}$$

The signal range of each node is limited by its transmission power P_i. Following [23], the maximum distance $Rmax_i$ where the MULE can still receive messages from node i, using transmission power P_i, is given by $Rmax_i = \sqrt{P_i}$. When using the grouping protocol, we assume that P_i denotes the minimum receiving power of leader i, otherwise we assume that it corresponds to its maximum receiving power. Replacing the maximum distance $Rmax_i$ by $\sqrt{P_i}$ in (5), we obtain the following expression for the weight W_i:

$$W_i = \frac{\sqrt{P_i}}{1 + \sqrt{P_i}} \tag{6}$$

The constant $c > 0$ is calculated such that (2) holds, i.e., $c = P \,/\, \left(\sum_{i=1}^{l} W_i\right)$, which allows us to define the probability $p_i = cW_i$, where $p_i \in [0, P]$, for the MULE to be in the range of leader i and to successfully communicate with it. We use these probabilities to model the behavior of the MULE when receiving or dropping messages. The main advantage of using probability distributions is that we obtain an abstract view of the MULE and ignore unnecessary details about the actual movements of the MULE vehicle and its physical communication with the sensors. In addition, our probabilistic approach for message propagation and MULE movement allows us to collect useful quantitative information for network analysis. Using discrete-event simulation, we obtain data related to the behavior of the network and to the amount of lost messages. Furthermore our model is flexible, i.e., it is easy to reuse it for different network configurations and MULE scenarios by just replacing the probability distribution in our model with another suitable distribution. In this sense, our formalism can be used as a *framework* for testing different MULE scenarios and algorithms.

In this paper, we used probabilistic rewrite theories [17] to model our grouping protocol, the propagation of data messages and also to model MULE be-

haviors. The next section describes how we can use this formalism to model the grouping protocol, while also incorporating probabilistic information.

5 Combining Declarative and Operational Models

In this section, we define a formal model of our proposed protocol in probabilistic rewriting logic. Our assumptions are: messages do not expire, and the number of nodes in the network is fixed, although they may move. The network is defined as a *system configuration*, a multiset of objects and messages, allowing the specification of local rules, for example to send data messages, as well as global rules, such as those used in the object scheduling mechanism. Following rewriting logic conventions, whitespace denotes the associative and commutative concatenation operator for configurations. The term $\langle$ `O: Node` | `leader:` L, `rpow:` E, `pow:` P, `buf:` B $\rangle$ denotes a `Node` object, where `O` is the object identifier, L its leader, E the remaining power, P the power capability, and B the message buffer.

As in [16], *unicast* messages have the form `(M from O to O')` where `M` is the message's body (possibly with parameters), `O` the source and `O'` the destination. A message will not reach its destination unless it is within the node's transmission range. *Multicasting* is modeled by allowing a set of destinations and equations which expand the destination set. *Wireless broadcasting* uses messages `(M from O to all)` where `all` is a constructor indicating that the message is sent to all nodes within range. We abstract from the actual data content of messages, and use a constant value for the message content.

In sensor networks, data is sensed from the environment continuously, and it should be transferred to the sink node. This process starts as soon as the network starts running and continues until all nodes run out of energy. Message passing is modeled by rewrite rules that can be applied at any time while the system is running, either during the grouping process or afterwards. These rules nondeterministically apply to enabled nodes in the network, so the nodes have an equal chance to pass messages to other sensor nodes.

If the selected node is a member of a group, then this node sends the data message to its group leader, using minimum power. Otherwise, it will send the message directly to the MULE, using maximum power. Fig. 3 describes the main MULE-based message passing rules in our model. The other rules in our model, such as those related to the underlying grouping protocol, are described in [16].

The *MsgFromNode* rule shows the message generated by a node. In this rule, `time(T)` is the current time, while `Pmin` and `Pmax` are defined by two equations that calculate minimum and maximum transmission power of nodes based on a value `P`, specific to each node, which we call the *power capability*.

The leaders transfer the data messages which they have received from their group members to the MULE. Rule *MsgFromLeader* represents the nondeterministic selection of one of the leaders that will pass the data message to the MULE. When a leader receives a data message from a node, it saves the message in its buffer `buf`. As soon as the buffer becomes full, the leader sends all messages to the MULE. This sending is modeled by means of a function `sendAll`,

```
rl [MsgFromNode]: ⟨ O : Node | leader: L, rpow: E, pow: P ⟩ execute(O) time(T)
⟶ if (L ≠ nil)
then ⟨ O: Node | leader: L, rpow: E-Pmin(P), pow: P ⟩ (msg from O to L)
else ⟨ O: Node | leader: L, rpow: E-Pmax(P), pow: P ⟩ (msg from O to "MULE") fi
[T+sampleExpWithRate(0.1), execute(O)] time(T).

rl [MsgFromLeader]: (M from O' to O) execute(O) time(T)
 ⟨ O : Leader | rpow: E, pow: P, buf: B ⟩
⟶ if #B+1≥Buffersize
then sendAll(⟨ O: Leader | rpow: E-Prec(P), pow: P, buf: push(B,M) ⟩ )
else ⟨ O : Leader | rpow: E-Prec(P), pow: P, buf: push(B,M) ⟩ fi
[T+sampleExpWithRate(0.1), execute(O)] time(T).

rl [MuleReceiveMsg]: (M from O' to "MULE") time(T) execute("MULE")
 ⟨ "MULE" : MULE | RecMsg: B, NumOfLostMsg: Y ⟩
⟶ if sampleBerWithP(probability)
then ⟨ "MULE": MULE | RecMsg: B, NumOfLostMsg: (Y+1) ⟩
else ⟨ "MULE": MULE | RecMsg: push(B,M), NumOfLostMsg: Y ⟩ fi
[T+sampleExpWithRate(0.1), execute("MULE")] time(T).

eq sendAll(⟨ O: Leader | buf: empty ⟩ ) = ⟨ O: Leader | buf: empty ⟩ .
eq sendAll(⟨ O: Leader | rpow: E, pow: P, buf: push(B,M) ⟩ ) =
(M from O to "MULE") sendAll(⟨ O: Leader | rpow: E-Pmin(P), pow: P, buf: B ⟩ ).
```

Fig. 3. Rules for MULE-based communication. Each rule considers an object ready for execution, and reschedules the object using sampling. Irrelevant node attributes are omitted. Buffer operations include the constructor `push` and # for length. As in Maude, we assume multiset matching. Variables are capitalized. `msg` is here a constant.

defined by two equations, which gives immediate sending of all messages in the buffer, since equations represent timeless actions in rewriting logic. In the rule, `Prec(P)` is defined by an equation calculating the power that a specific node consumes to *receive* a message, based on its power capability `P`.

The MULEs move and gather data messages which are sent by leaders or single nodes. The movement of the MULE causes some message loss, captured by the probability distribution of successful message passing (cf. Section 4). By using this probability distribution, we abstract from the movement of the MULE. In our model, every message sent to the MULE is received with a probability calculated by Equation 6 in Section 4, otherwise the message is lost (i.e., removed from the system configuration). Rule *MuleReceiveMsg* represents this process, with the `probability` variable giving the actual probability of successfully receiving a data message; i.e., $p_i = cW_i$, as defined in Section 4. The `sampleBerWithP` operation samples from the Bernoulli distribution; i.e., it returns `true` with a given probability p and `false` with probability $1 - p$.

We assume that a MULE transmits all the received messages to the sink. So in our model, there is no need for additional rules capturing the communication between the sink and the MULE. Further details about modeling the grouping and the routing protocols in rewriting logic can be found in [16]. In the present work, we extended all of the rules in the cited work to probabilistic rewrite rules, as well as added new equations. The validation of the group membership

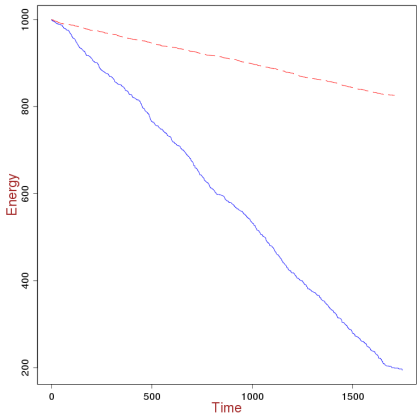

Fig. 4. The remaining energy of a node.

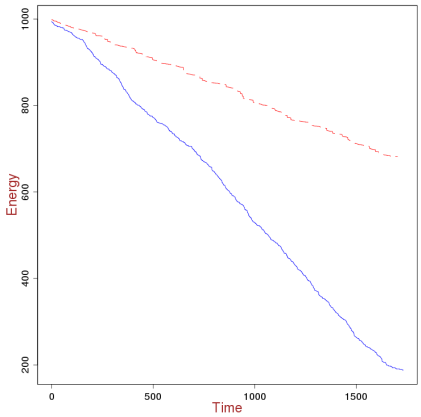

Fig. 5. The remaining energy of a leader.

protocol can be achieved by using Maude's model checking tools. In [16], Maude's `search` tool has been applied to verify the correctness of the grouping protocol.

6 Quantitative Analysis of the Combined Model

This section proposes an approach to obtain quantitative results by guiding and monitoring Maude simulations. The basic idea is to control the run of the Maude model and monitor the system configuration at each tick. The desired data is extracted from the configuration, including numerical data stored inside each node. After the simulations, all the data extracted from the model is gathered and analyzed. To automate this process, we have implemented a Python script which extracts quantitative information from a system configuration of our model by parsing the configuration after the application of each tick rule, and extracting numerical data as queried by the user. This way, the script gathers data resulting from the application of a specified number of ticks. Finally, the script analyzes the data and provides a plot diagram showing the graph of a given system parameter against time. In addition, several simulations of the Maude model can be combined, producing a graph which averages the data obtained from each simulation. More precisely, we use a modified linear interpolation procedure that is able to precisely combine data from a set of graphs.

Thus, we used Maude to simulate our model of MULE-based WSNs, driven by the grouping protocol proposed in [16]. Our topology contains a MULE and two groups of six nodes each. Each node starts with 1000 units of energy. In the beginning of the model execution, the nodes start sending data messages. During the execution, they can move and join a new group. We capture the remaining energy of each node at every tick of the simulation, as well as the number of sent and received messages. We ran simulations for two distinct scenarios; namely, when the WSN uses the grouping protocol vs. when it does not. Our purpose is to compare the energy consumption of the nodes and the leaders, in each scenario.

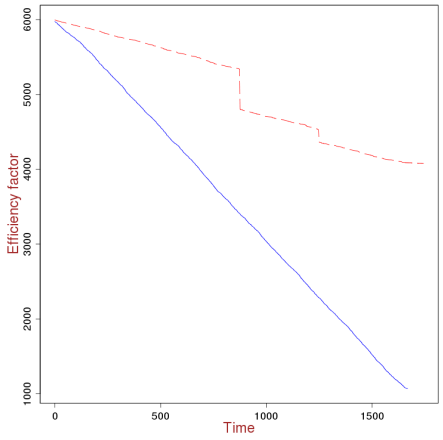

Fig. 6. The graphs of the F factor (for 6 nodes), when the MULE-based WSN is using (dashed line) and when it is not using the grouping protocol (solid line).

In addition, in order to obtain a better understanding of the network's efficiency, we define an *efficiency factor* F with the following expression $F = \frac{1}{LM} \sum_{n=1}^{N} E_n$, where N is the total number of sensor nodes, E_n is the remaining energy of node n and LM is the total number of messages that the MULE has lost. The efficiency factor F represents a ratio between the energy consumption and the message loss in the network. More efficient networks, in terms of energy consumption and performance in message delivery, have higher values of F.

Figures 3 and 4 show the saved energy of a sensor node and of a leader, respectively, in a MULE-based WSN with (dashed line) and without (solid line) using the grouping protocol. We have also calculated the value of the F factor for a run, as the average of 5 simulations, and displayed the results in Fig. 6, in the case when the WSN uses the grouping protocol, as well as when it does not. By comparing the two graphs in Fig. 6, we observe a considerable improvement in the efficiency of the network when the grouping protocol is running. To generate each of the graphs, we ran 5 simulations (each simulation lasting for 1000 ticks).

7 Conclusion

This paper applies a combination of declarative and operational specification, using a probabilistic approach for underspecification in the operational model. Technically, this is achieved using the framework of probabilistic rewriting logic and PMaude. We demonstrate the approach on a grouping protocol for MULE-based WSNs and show how the declarative specification provides an abstract and flexible solution to model both fair message passing and underspecified MULE behavior in WSNs. Furthermore we use a statistical method for quantitative analysis of Maude models, which provides useful data sets and graphs for network analysis and performance evaluation of protocols. The obtained numerical results allow the energy efficiency of the network to be compared, with and

without using the considered protocol. We have shown that using the grouping protocol improves the energy efficiency of the network. The particular choice of parameter values used in the probabilistic modeling is based on our preliminary experience, and can easily be readjusted to fit better with reality. The approach taken provides a framework for further experimentation.

In future work, we intend to build on our current Maude model as well as to extend it, to capture *real-time* aspects of WSNs. Furthermore, we plan to subject our model to statistical model checking, to be able to statistically prove the correctness of large models. It is known that, due to their huge state space, it is practically impossible to verify such models using traditional model checking techniques. We also plan to make an integration of our current Maude implementation with the VeStA/PVeStA tool, which allows for *probabilistic reasoning* via statistical model checking and statistical quantitative analysis. Using VeStA, we would be able to verify the *statistical correctness* of the protocol proposed in this paper, as well as to make more precise quantitative analysis.

Acknowledgment. We would like to thank Lucian Bentea for his contribution to this paper, and in particular for his help with the implementation.

References

1. G. Agha, J. Meseguer, and K. Sen. PMaude: Rewrite-based Specification Language for Probabilistic Object Systems. *ENTCS*, 153(2):213–239, 2006.
2. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.
3. G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537–568, 2009.
4. A. Basu, S. Bensalem, M. Bozga, B. Delahaye, and A. Legay. Statistical abstraction and model-checking of large heterogeneous systems. *Software Tools for Technology Transfer*, 14(1): 53–72, 2012.
5. M. Bernardo and R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202(1–2):1–54, 1998.
6. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, editors. *All About Maude - A High-Performance Logical Framework*, vol. 4350 of *LNCS*. Springer, 2007.
7. I. A. Dario, I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: Research challenges. *Ad Hoc Networks*, 3(3):257–279, 2005.
8. J. S. Dong, J. Sun, K. Taguchi, and X. Zhang. Specifying and verifying sensor networks: An experiment of formal methods. vol. 5256, *LNCS*, pages 318–337,2008.
9. S. C. Ergen, M. Ergen, and T. J. Koo. Lifetime analysis of a sensor network with hybrid automata modelling. *Proc. 1st ACM Int. Workshop on Wireless Sensor Networks and Applications*, pages 98–104. 2002.
10. A. Fehnker, M. Fruth, and A. McIver. Graphical modelling for simulation and formal analysis of wireless network protocols. *Methods, Models and Tools for Fault Tolerance*, vol. 5454, *LNCS*, pages 1–24. 2009.
11. A. Fehnker, L. van Hoesel, and A. Mader. Modelling and verification of the LMAC protocol for wireless sensor networks. *Proc. 6th Intl. Conf. on Integrated Formal Methods (IFM'07)*, vol. 4591, *LNCS*, pages 253–272. 2007.

12. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. 33rd Hawaii Int. Conf. on System Sciences, vol. 8*, page 8020, 2000.
13. J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
14. S. Jain, R. C. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile Networks and Applications*, 11(3):327–339, 2006.
15. M. Katelman, J. Meseguer, and J. C. Hou. Redesign of the LMST wireless sensor protocol through formal modeling and statistical model checking. *Proc. 10th Intl. Conf. on Formal Methods for Open Object-Based Distributed Systems (FMOODS'08)*, vol. 5051, *LNCS*, pages 150–169. 2008.
16. F. Kazemeyni, E. B. Johnsen, O. Owe, and I. Balasinham. Grouping nodes in WSNs using coalitional game theory. *Formal Techniques for Distributed Systems, Proc. FMOODS/FORTE'09*, vol. 6117, *LNCS*, pages 95–109. 2010.
17. N. Kumar, K. Sen, J. Meseguer, and G. Agha. Probabilistic Rewrite Theories: Unifying Models, Logics and Tools. Technical report UIUCDCS-R-2003-2347, Dept. of C. S., Univ. of Illinois at Urbana-Champaign, 2003.
18. M. Kwiatkowska, G. Norman, and D. Parker. Quantitative analysis with the probabilistic model checker PRISM. *ENTCS*, 153(2):5–31, 2006.
19. M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic Model Checking for Performance and Reliability Analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.
20. J. Lloret, C. E. Palau, F. Boronat, and J. Tomás. Improving networks using group-based topologies. *Computer Communications*, 31(14):3438–3450, 2008.
21. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.
22. A. Muhammad and A. Azween. Dynamic cluster based routing for underwater wireless sensor networks. *ITSim'10*, 3, June 2010.
23. M. Noori and M. Ardakani. A probabilistic lifetime analysis for clustered wireless sensor networks. In *Proc. WCNC'08*, pages 2373–2378, 2008.
24. P. C. Ölveczky and S. Thorvaldsen. Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in Real-Time Maude. *Theoretical Computer Science*, 410(2-3):254–280, 2009.
25. A. D. Pierro, C. Hankin, and H. Wiklicky. Probabilistic Linda-based coordination languages. *Proc. 3th Int. Symp. on Formal Methods for Components and Objects (FMCO 2004)* vol. 3657, *LNCS*, pages 120–140. Springer, 2005.
26. D. Pompili and I. F. Akyildiz. Overview of networking protocols for underwater wireless communications. *IEEE Communications Magazine*, 47(1):97–102, 2009.
27. C. Priami. Stochastic π-calculus. *Computer Journal* , 38(7):578–589, 1995.
28. K. Sen, M. Viswanathan, and G. Agha. VESTA: A Statistical Model-checker and Analyzer for Probabilistic Systems. In *Proc. 2nd Int. Conf. on the Quantitative Evaluation of Systems (QEST '05)*, page 251, USA, 2005.
29. R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA* , pages 30–41, 2003.
30. S. Tschirner, L. Xuedong, and W. Yi. Model-based validation of QoS properties of biomedical sensor networks. Int. conf. on Embedded software, pages 69–78. 2008.
31. M. AlTurki and J. Meseguer. PVeStA: A Parallel Statistical Model Checking and Quantitative Analysis Tool. *Proc. 4th Int. Conf. on Algebra and Coalgebra in Computer Science*, vol. 6859, *LNCS*, pages 386–392. 2011.