Formal Analysis of a Scheduling Algorithm for Wireless Sensor Networks

Maissa Elleuch[1,2], Osman Hasan[2], Sofiène Tahar[2], and Mohamed Abid[1]

[1] CES lab, National School of Engineers of Sfax, Sfax University

Soukra Street, 3052 Sfax, Tunisia

Email: maissa.elleuch@ceslab.org

Email: mohamed.abid@enis.rnu.tn

[2] Dept. of Electrical & Computer Engineering, Concordia University

1455 de Maisonneuve W., Montreal, Quebec, H3G 1M8, Canada

Email: {melleuch,o_hasan,tahar}@ece.concordia.ca

Technical Report

February, 2011

Abstract

In wireless sensor networks (WSNs), scheduling of the sensors is considered to be the most effective energy conservation mechanism. The random and unpredictable deployment of sensors in many WSNs in the open fields makes the sensor scheduling problem very challenging and randomized scheduling algorithms are thus used. The performance of these algorithms is usually analyzed using simulation techniques, which do not offer 100% accurate results. In this paper, we overcome this limitation by using higher-order-logic theorem proving to formally analyze the coverage-based random scheduling; a variant of the random scheduling algorithm designed for wireless sensor networks. Particularly, we aim at formalizing this coverage-based random scheduling within the probabilistic framework developed in the HOL theorem prover. We also formally reason about the expected values of coverage intensity, the upper bound on the total number of disjoint subsets, given expected coverage intensity, the lower bound on the total number of nodes and the average detection delay inside the network.

1 Introduction

Wireless sensor networks (WSNs) [39] have been proposed as an efficient solution to monitor a field without any continuous human surveillance. Such networks are composed of small tiny devices wirelessly connected and deployed over the field. The main task of sensors consists in taking measurements of the monitored event. These measurements are physical attributes of the environment such as light, temperature, humidity, pressure and acceleration. According to these measurements, a decision procedure is made at the base station. WSNs are extensively being deployed these days in a variety of critical applications like detection of natural disasters, detection of chemical or biological attack and military tracking.

Since the sensors are usually battery-powered, using minimal energy for all the performed tasks becomes the main constraint. Scheduling [18] of the nodes is one of the most widespread solutions to preserve energy. It consists in splitting the network on several sub-networks which will work alternatively. Nevertheless, achieving scheduling for WSNs results in major challenges to avoid losing the coverage and connectivity. The coverage reflects how well an area is monitored or tracked by sensors. Hence, at any given time, the active sensors should cover most of the field while remaining connected. A lot of algorithms have been proposed to solve the scheduling problem. Most of them take into account only one of the constraint i.e. coverage [1, 34, 36, 38] or connectivity [5, 35] but rarely both. Moreover, many works have studied different techniques to determine an eligibility rule for the activation of the next sub-network [26] and [30].

For inhospitable fields, the sensors are deployed in a random manner, i.e., by throwing them from an airplane. The trend is then to use a random scheduling scheme to save energy. The idea of the random scheduling was first motivated by the work of [30] and has been proposed by [1] and [21]. As the study of random scheduling algorithms for WSNs is recent, the focus is to investigate more in developing new models that can satisfy both constraints of coverage and connectivity. In general, a theoretical paper-and-pencil based model of the proposed scheduling algorithm is developed and analyzed. After that, performance evaluation by simulation is done in order to illustrate the theoretical results. Nevertheless, the results obtained by simulation can never be totally accurate. Thus, simulation cannot be considered as a reliable solution for the probabilistic analysis of WSNs especially when applied to validate WSNs for mission-critical applications like military, buildings, health, disaster relief, area and industrial monitoring.

In order to overcome the common drawbacks of simulation, formal methods [9] have been proposed as an efficient solution to validate a wide range of hardware and software systems. Formal methods increase the system reliability by rigorously using mathematical techniques to analyze the mathematical model for the given system. They have the advantage to find out subtle errors that cannot be revealed by traditional simulation. The need of formal methods in the context of WSNs is illustrated in [25]. However, formal methods seem very restricted when used to validate probabilistic systems. The random components of the system cannot be directly modeled within traditional formal tools. For example, it will be impossible to reason precisely about statistical properties, such as expectation and variance, in the case of state-based approaches. Furthermore, huge proof efforts are usually expected to be involved in reasoning about random components of a wireless system in the case of theorem proving.

We believe that due to the recent developments in the formalization of probability theory concepts in higher-order-logic [16, 12], the analysis of a variety of wireless systems with random components in a higher-order-logic theorem prover [6] can be handled with reasonable amount of proof efforts. In this paper, we demonstrate the effectiveness of the proposed idea by formally verifying the coverage-based random scheduling algorithm of [22], which is basically a joint scheduling method composed of a random scheduling algorithm and an extra-on rule. This scheme has the advantage to solve at the same time the problem of sensing coverage and network connectivity without any location information. We particularly aim at formalizing this coverage-based random scheduling within the probabilistic framework developed in the HOL theorem prover [12]. The intent is to verify the expected values of coverage intensity, and deduce the upper bound on the total number of disjoint subsets, given expected coverage intensity. We will also verify the lower bound on the total number of nodes and the average detection delay inside the network.

The remainder of this paper is organized as follows. First, we present the probabilistic framework developed in the HOL theorem prover. Then, we give an overview of the coverage-based random scheduling algorithm for wireless sensor networks. In Sections 3 and 4, we provide the formal specification and verification of the coverage-based random scheduling algorithm respectively. Finally, we discuss related work and conclude the paper by exposing future works.

2 Preliminaries

In this section, we first present the probabilistic framework developed in the HOL theorem prover. Particularly, we will present the general methodology that we have to follow for analyzing the WSN scheduling algorithm. We also illustrate the formalization of discrete random variables in HOL, the notations used and the verified probabilistic properties that will be needed later. At the end of this section, we will give a brief overview of the HOL theorem prover.

2.1 Probabilistic Analysis Framework

The framework, given in Fig. 1, illustrates the inputs, outputs and the main components of the probabilistic analysis framework based on the HOL theorem proving system [12]. The

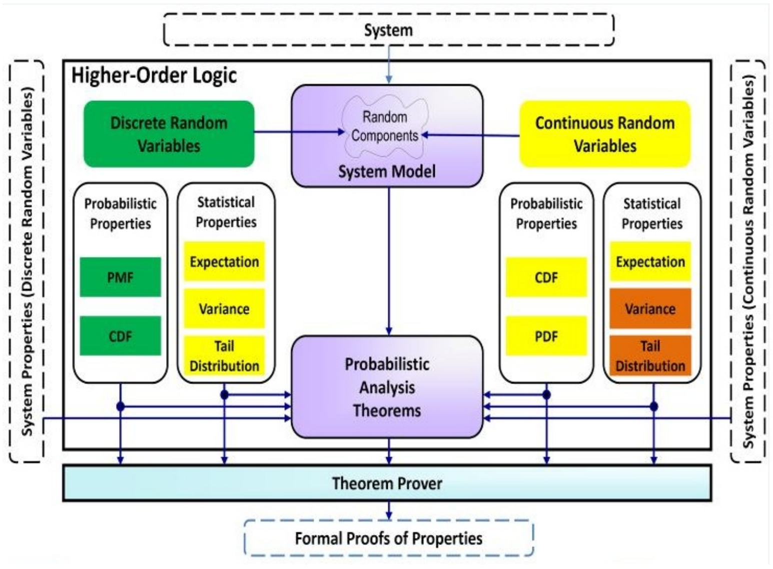

Figure 1: Theorem Proving based Probabilistic Analysis Framework for Systems [12].

shaded boxes in this figure represent the fundamental requirements for conducting probabilistic analysis in a theorem prover.

Using this probabilistic framework, it is possible to formally analyze probabilistic aspects of any wireless system following these steps:

- Building a model describing the system to verify by specifying its random components as higher-order-logic functions. Since random variables are always defined in term of their PMF and CDF, we need first to express these mathematic characterizations in the HOL theorem prover,

- Expressing the properties of the system as higher-order-logic theorems. The properties are mainly related to the expectation and variance of the corresponding random variable. Therefore, the mathematical definitions of expectation and variance for the two kinds of random variables (discrete and continuous) have to be formalized. The figure shows that the framework distinguishes the properties related to discrete random variables from the ones related to continuous random variables. This distinction is made for simplicity purposes,

- Formally verifying the theorems, developed in the previous step, using the HOL theorem prover. The output of the framework, depicted by the rectangle with dashed edges, represents the formal proofs of the system properties. These proofs certify the correctness of the formalized probabilistic properties for the given system.

2.2 Formalization of Discrete Random Variables and Verification of their PMF

A random variable is called discrete if its range, i.e., the set of values that it can attain, is finite or at most countably infinite [37]. Discrete random variables are mathematically specified by their PMF which is the probability that a random variable X is exactly equal to some value x, i.e., $Pr(X = x)$. In higher-order-logic, discrete random variables are formalized as deterministic functions with access to an infinite Boolean sequence $\mathbb{B}^\infty$; a source of infinite random bits with data type ($num \rightarrow bool$)[16]. According to the result of popping the top most bit in the infinite Boolean sequence, these deterministic functions make random choices. They may pop as many random bits as they need for their computation. At the end of the computation, they return the result along with the remaining portion of the infinite Boolean sequence to be used by other functions. Thus, a random variable that takes a parameter of type α and ranges over values of type β can be represented in HOL by the function:

$$\mathcal{F} : \alpha \rightarrow B^\infty \rightarrow \beta \times B^\infty.$$

As an example, the $Bernoulli(\frac{1}{2})$ random variable that returns 1 or 0 with equal probability can be modeled as follows

```
⊢ bit = λs.  (if shd s then 1 else 0, stl s).
```

where the variable s represents the infinite Boolean sequence and the functions `shd` and `stl` are the sequence equivalents of the list operation *'head'* and *'tail'*. The function `bit` accepts the infinite Boolean sequence and returns a pair with the first element equal to either 0 or 1 and the second element equal to the unused portion of the infinite Boolean sequence, which in this case is the tail of the sequence.

Random variables can also be expressed in a more compact form using the general state-transforming monad where the states are the infinite Boolean sequences.

```
⊢ ∀ a,s.  unit a s = (a,s)
⊢ ∀ f,g,s.  bind f g s = g (fst (f s)) (snd (f s)).
```

The HOL functions `fst` and `snd` above return the first and second components of a pair, respectively. The `unit` operator is used to lift values to the monad, and the `bind` is the monadic analogue of function application. All monad laws hold for this definition, and the notation allows us to write functions without explicitly mentioning the sequence that is passed around, e.g., function `bit` can be defined as

```
⊢ bit_monad = bind sdest (λb.  if b then unit 1 else unit 0).
```

where `sdest` gives the head and tail of a sequence `s` as a pair (*shd* `s`, *stl* `s`).

Some formalizations of the measure theory in [16] can be used to define a probability function called `prob` which operates from sets of infinite Boolean sequence to the set of real number between 0 and 1. The domain of `prob` is the set $\mathcal{E}$ of probability events. Consequently, the formalization of `prob` and $\mathcal{E}$ can be used together to prove probabilistic properties of random variables such as:

$$\vdash \texttt{prob } \{\texttt{s | fst (bit s) = 1}\} \texttt{ = } \frac{1}{2}.$$

where the HOL function `fst` selects the first component of a pair and $\{x|C(x)\}$ represents a set of all elements x that satisfy the condition C.

By following the methodology described above, most of the commonly used discrete random variables which are frequently used have been specified in the HOL theorem prover. The corresponding PMF of each of these discrete random variables has been also verified. For example, HOL definitions and PMF theorems for the Bernoulli, Uniform, Binomial and Geometric random variables can be found in [16, 12]. In this paper, we will need to use the first three random variables enumerated above, i.e., the Bernoulli, Uniform and Binomial. The HOL functions used to describe them are called respectively: `prob_bernoulli`, `prob_uniform` and `prob_binomial`.

2.3 Formalization and Verification of Expectation Properties for Discrete Random Variables in HOL

The expectation of a discrete random variable, which attains values in the positive integers only, is specified as follows [19]:

$$Ex\_fn[f(R)] = \sum_{n=0}^{\infty} f(n)Pr(R = n). \tag{1}$$

where R is the discrete random variable and f represents a function of the random variable R. The function f maps the random variable R to a real value. The above definition of expectation holds only if the summation is well defined, i.e., finite. The above equation can be formalized in HOL as follows:

```
Definition 1:
⊢ ∀ f R. expec_fn f R = suminf (λn.(f n) prob {s | (fst (R s) = n)}).
```

The definitions of `R` and `f` are the same as in (1). The HOL function `suminf` represents the infinite summation of a real sequence [11]. The function `expec_fn` accepts two parameters, the function `f` of type $(num \rightarrow real)$ and the positive integer valued random variable `R` and returns a real number.

The expectation of a discrete random variable that attains values in positive integers would be a particular case of the above definition where the function `f` is instantiated by the identity function (λ`n.n`). The corresponding HOL function is specified in Definition 2.

```
Definition 2:
⊢ ∀ R. expec R = expec_fn (λn.n) R.
```

After formalizing the expectation of a positive valued discrete random variable, it is very interesting to check the correctness of some related properties, which will greatly facilitate the probabilistic analysis. In fact, the proof of the linearity of expectation of a function of a discrete random variable, specified in (2), has been provided in [12].

$$Ex\_fn[af(R) + b] = aEx\_fn[f(R)] + b \tag{2}$$

The corresponding HOL theorem is expressed by Theorem 1.

```
Theorem 1:
⊢ ∀ a,b,R,f.
          (summable (λn.  (f n)prob bern {s | fst (R s) = n})) ∧
          (R IN indep_fn) ⇒
          (expec_fn (λs.(a×fst (f(R) s) + b, snd (f(R) s))) =
          &a×expec_fn (f(R)) + &b).
```

where `a` and `b` are two positive integers, `R` is a positive valued random variable and `snd` is the function that returns the second component of a pair. The operator `&` is used to transform a positive integer to its corresponding real value. The HOL function `summable` used in the assumptions ensures that the expectation of the function of the random variable `R` is well-defined, i.e., the summation of (1) is finite. The second assumption states that the random variable must be independent. Based on Hurd's formalization infrastructure, a random variable can be verified to be a measurable function if it accesses the infinite Boolean sequence using only `unit`, `bind` and `sdest` primitives.

The proof of the above theorem relies on two important intermediate results. The first one illustrates that the expectation of a random variable multiplied by a positive integer is equal to the positive integer multiplied by the expectation of the random variable (3). This theorem has been also verified in HOL [12].

$$Ex\_fn[af(R)] = aEx\_fn[f(R)] \tag{3}$$

The second theorem is related to the expectation of a sum of two random variables which is equal to the sum of their respective expectations. The generalization of this theorem is given in (4) where *Ex* designates the expectation.

$$Ex[\sum_{i=1}^{n} Ri)] = \sum_{i=1}^{n} Ex[Ri] \tag{4}$$

For illustration purposes, the formalization of expectation was used to verify the expectation of the Bernoulli, Uniform, Binomial and Geometric random variables [12].

2.4 The HOL Theorem Prover

The HOL theorem prover is a proof assistant of higher-order-logic [31]. The verification approach of HOL is composed of three main steps: describing the system verified in higher-order logic, formalizing the properties to be verified as theorems of higher-order-logic and finally verifying these properties within HOL. The HOL theorem prover includes a very rich library of theories. A theory can be defined as a set of pre-verified theorems for a given domain, function or operation. When needed, a HOL theory can be loaded and used. Thus, an experiment user can make proofs much easier by efficiently using the existing theories. In addition, users may be assisted by automatic proof procedures [10] which are sets of gathered steps executed in a single command.

Despite the existence of all these theories and automatic procedures, most of the time, proofs in HOL are interactive and require the intervention of user. Direct proofs are limited to very simple theorems. Various proof techniques can be used to achieve a proof such as rewriting, simplification, specialization, generalization and mathematical induction.

The table below summarizes some of the HOL symbols used in this paper and their corresponding mathematical interpretation [7].

HOL Symbol	**Standard Symbol**	**Meaning**
$\wedge$	*and*	Logical *and*
$\vee$	*or*	Logical *or*
$\sim t$	$\neg t$	Not t
SUC n	$n+1$	Successor of a *num*
x *pow* n	x^n	*real* x raised to *num* power n
ln x	*ln* x	Neperian logarithm on x
exp x	e^x	Exponential logarithm on x
$\lambda x.t$	$\lambda x.t$	Function that maps x to $t(x)$
$\{x \mid P(x)\}$	$\{\lambda x.P(x)\}$	Set of all x that satisfy the condition P

Table 1: HOL Symbols

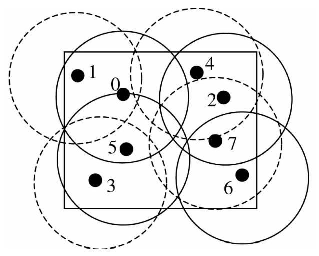

Figure 2: An example of the randomized coverage-based algorithm [22].

3 Formalization of the Coverage-based Randomized Scheduling Algorithm

According to the probabilistic framework given in [15], we have to specify the higher-order-logic functions describing the random components of the wireless system, and then verify the theorems which formalize the desired properties. In this section, we develop the HOL formalization of the coverage-based random scheduling algorithm for WSNs. This formalization is basically inspired by the paper-and-pencil based analytical analysis presented in [22].

3.1 Overview of the Coverage-based Randomized Scheduling Algorithm

The WSN considered deploys n sensors over a field of size a. All the sensors have the same task; gathering data and routing it back to the base station. The deployment of nodes over the two-dimensional field is random so that no location information is available. The size of the sensing area of each sensor is denoted r. A sensor can only sense the environment and detect events within its sensing range. We say that a point of the monitored field is covered when any event occurring at this point can be detected by at least one active sensor. The probability that each sensor covers a given point is r/a. This probability is called q.

Given a natural number k, the random scheduling of the nodes assigns each sensor to one of the k sub-networks with equal probability $1/k$. At time slot Ti, only the nodes belonging to the sub-network i will be active and can cover an occurring event. Hence, the disjoint sub-networks created will work alternatively. We denote also by: Si, the set of sensors that belongs to the sub-network i and covers a specific point inside the field, S, the set of nodes covering a specific point inside the field, and, c, the cardinality of S.

For illustration purpose, Fig. 2 shows how the scheduling algorithm splits arbitrarily a network containing eight sensor nodes to two sub-networks. The eight nodes, randomly deployed in the monitored region, are identified by IDs ranging from 0 to 7. The two sub-networks are called *S0* and *S1*. Each node chooses at random between 0 and 1 in order to be assigned to one of these two sub-networks. Suppose that nodes 0; 2; 5; 6 select the number 0 and join the subset S0 and nodes 1; 3; 4; 7, choose the number 1 and join the subset *S1*. Thus, the two sub-networks will work alternatively. In other words, when the nodes 0; 2; 5; 6, which sensing ranges are denoted by the solid circles, are active, the nodes 1; 3; 4; 7 illustrated by the dashed circles will be idle and vice versa.

3.2 Formalization of the Network Coverage Intensity

The challenge in the random scheduling algorithm described below, is to select a value of k so that the energy can be saved with a good coverage. Therefore, the performance of this algorithm depends essentially on the chosen value of k. A large k will imply a lot of sub-networks which would in turn result in few nodes in each of these sub-networks, and hence a poor coverage. However, a small k will imply few sub-networks with a lot of points covered simultaneously by a lot of nodes, so a waste of energy.

The random scheduling algorithm suggests several random variables. The first one distributes uniformly the nodes over the sub-networks. It is formalized by the HOL function `rd_subsets` which generates recursively a list of Uniform random variables, and accepts two parameters: c, the number of sensors that covers a specific point inside the field, and k, the number of sub-networks. In this definition, we use the predefined HOL function `prob_uniform` which takes as input a natural k and generates a Uniform random variable. The function `rd_subsets` is described as follows:

```
Definition 3:
⊢ (∀ k.  rd_subsets 0 k = []) ∧
   (∀ c,k.  rd_subsets (SUC c) k = (prob_uniform k)::(rd_subsets c k)).
```

Let X the random variable denoting the total number of non-empty *Sj*. X is defined as follows:

$$X = \sum_{j=0}^{k-1} Xj. \tag{5}$$

where *Xj* is the Bernoulli random variable describing a non-empty subset. The variable *Xj* is based on the recursive HOL predicate `subset_empty` which describes an empty subset by looking for an index j in a given list. The function `subset_empty`, given in Definition 4, takes as inputs a natural number denoted j and a list called L having the format $(h :: t)$, and returns true only if j is not an element of the list L. Later, when we call the predicate `subset_empty`, the list L will be substituted by the above function `rd_subsets`.

```
Definition 4:
⊢ (∀ j.  subset_empty j [] = UNIT T) ∧
  (∀ j,h,t.  subset_empty j (h::t) = bind h (λa.
                                          bind (subset_empty j t) (λb.
                                                unit (∼(a = j) ∧ b)))).
```

Now, the variable *Xj*, used in (5), is specified in Definition 5 through the function `subset_non_empty` which takes three parameters: j, a natural number, c, the number of sensors that covers a specific point inside the field, and k, the number of sub-networks. The event `{s | fst (subset_empty j (rd_subsets c k) s) = F}`, used in this function, designates the event *"The subset Sj is non-empty"*.

```
Definition 5:
⊢ ∀ j,c,k.  subset_non_empty j c k = bernoulli_num
    (prob bern {s | fst (subset_empty j (rd_subsets c k) s) = F}).
```

A given sub-network Sj is empty if the Uniform random variable never generates the number j, i.e., the random scheduling doesn't assign any sensor to the sub-network Sj. If applied to the set S defined below, the random scheduling will affect each sensor node in S to one of the k disjoint subsets. A subset Sj will be empty if all the c sensors of S miss this subset. Indeed, a node joins Sj with the probability $1/k$. It will miss the same Si with the probability $(1 - 1/k)$. Since the c sensors miss independently the subset Sj, the probability that Sj is empty, is then:

$$\left(1 - \frac{1}{k}\right)^{c}. \tag{6}$$

In order to define the random variable X given in (5), we define first the following function which recursively generates a list by accepting the following parameters: k, the length of the list, c, the number of sensors that covers a specific point inside the field, and m, the number of sub-networks. After that, a pre-defined function of the HOL probability theory, called `sum_rv_lst`, accepts this list of random variables and returns their sum as a single random variable.

```
Definition 6:
⊢ (∀ c,m.  subset_non_empty_lst 0 c m = [subset_non_empty 0 c m]) ∧
    (∀ k,c,m.  subset_non_empty_lst k c m =
          (subset_non_empty (SUC k) c m)::(subset_non_empty_lst k c m))).
```

The coverage intensity for a specific point *Cp* can now be defined as the average time during which the point is covered by the total length of the scheduling cycle. The variable *Cp* is expressed in (7).

$$Cp = \frac{E[X] \times T}{k \times T}. \quad (7)$$

where $E[X]$ designates the expectation of the random variable defined in (5). The variable Cp is formalized in HOL as follows:

```
Definition 7:
⊢ ∀ c,k.  cvrge_intsty_pt c k =
     (expec (sum_rv_lst (subset_non_empty_lst k c (SUC k))))/&(SUC k).
```

The above definition specifies the coverage intensity for a specific point using the HOL function `cvrge_intsty_pt`. This function takes as parameters: c, the number of sensors that covers a specific point inside the field, and k, the number of sub-networks. Added to the function `subset_non_empty_lst`, this definition uses two other predefined HOL functions which are `expec`, for the expectation of a discrete random variable (Definition 2), and `sum_rv_lst`, for the summation over random variables. More explanations about these two functions can be found in the preliminaries section and in [12].

It has been shown that Cp is equal to:

$$\left[1 - \left(1 - \frac{1}{k}\right)^{c}\right]. \quad (8)$$

We recall that the variable c is initially the number of nodes covering a specific point inside the field. Covering a point or not can be assimilated to a Bernoulli trial with the probability q. If we consider the variable c among the n nodes of the network, it becomes a Binomial random variable with the following probability:

$$Pr(c = j) = \frac{n!}{j!\,(n-j)!} q^{j} (1-q)^{n-j}. \quad (9)$$

where q is the probability that each sensor covers a given point.

Therefore, Cp is also a random variable. Particularly, Cp is a function of the random variable c. Since the random deployment strategy distributes independently the nodes over the area and the random scheduling makes a uniform distribution of the same sensors, the expectation of Cp for any point inside the area is the same and its value is Cn. The variable Cn is defined as follows:

$$Cn = Ex\_fn[Cp] \quad (10)$$

where Ex_fn designates the expectation of a function of a random variable. The corresponding HOL function formalizing (10) is:

```
Definition 8:
⊢ ∀ q,n,k.  cvrge_intsty_network q n k =
  expec_fn (λi.  1 + (-1)×(1 - 1/&(SUC k))^x) (prob_binomial_p n q).
```

The above function `cvrge_intsty_network` accepts as inputs q, the probability that a sensor covers a point, n, the number of sensors deployed inside the field, and k the number of sub-networks. This function specifies the expectation of a function of random variable and thus needs two parameters: the input function which basically describes the variable Cp and is specified by the first parameter of the above function `expec_fn`, and the random variable which is the Binomial of (9).

3.3 Formalization of the Average Detection Delay

The average detection delay is another performance metric which can be relevant in evaluating the random scheduling algorithm. It is defined as the expectation of the time elapsed from the occurrence of an event to the time when the event is detected by some sensor nodes. The average detection delay for an event arriving at any time slot with equal probability and lasting for duration longer than $(k-1) \times T$, is defined as:

$$delays = \sum_{i=1}^{k-1} \int_{0}^{T} \frac{1}{T} \times \Pr(H0 \cap H1 \cap ... \cap \overline{Hi}) \times (i \times T - t)dt. \tag{11}$$

where:

- Hi: the event that none of the c covering sensor nodes belongs to the working subset i
- $\overline{Hi}$: the event that at least one of the c covering sensors belongs to the working subset i
- T: the duration of a time slot
- k: the number of disjoint subsets

Table 2 shows the state of a WSN within a complete scheduling cycle. A scheduling cycle is a succession of time slots Ti. Time slot Ti corresponds to the working shift of the sub-network i, denoted Si. With the coverage-based random scheduling algorithm, the network is divided into k sub-networks. At any given time slot, only one subset is working. Let an event e happens at time t and detected within the time slot $T2$; the working shift of the sub-network $S2$. Detecting the happening event within the time slot $T2$ means that at least one of the c covering sensors belong to the subset $S2$. Consequently, none of the c covering sensors belongs to the past subsets which are $S0$ and $S1$. By generalization, at the

Table 2: The WSN state within a scheduling cycle

Time slot Ti	T_0	T_1	T_2	......	T_i	......	$T_{(k-1)}$
Working subset Si	S_0	S_1	S_2	......	S_i	......	$S_{(k-1)}$
Nbr. Remaining subsets	k	$(k-1)$	$(k-2)$	......	$(k-i)$	......	1

time slot *Ti*, the working subset is *Si* and it remains *(k–i)* subsets to which may belong the c covering nodes.

Subsequently, we clarify (11) through a concrete example. Particularly, we emphasize that the term *"expectation"*, used in the initial definition, means basically *"average"* and is not related to the expectation as a statistical quantity. We consider the same event e above. The exact time elapsed for the detection of the event e is then *(2T–t)*. This quantity must be weighted by the probability that *"at least one of the c covering sensors belongs to the working subset S2"* which is "$\Pr(H0 \cap H1 \cap \overline{H2})$". Hence, the average time for the detection of the event e is the average value of the function "$\Pr(H0 \cap H1 \cap \overline{H2}) \times (2T - t)$". This can be found by using the mean value theorem for integrals. Nevertheless, the event e can happen at any time slot of the scheduling cycle, a summation over all time slots is provided by generalization.

By analogy to the manner of computing the probability of an empty subset in (6), the probability of the event *Hi* stating that *"none of the c covering sensor nodes belongs to the working subset Si within Ti"* will be then:

$$\left(1 - \frac{1}{(k-i)}\right)^c. \tag{12}$$

Now, defining the HOL context for the verification of the average detection delay needs to specify correctly the set $(H0 \cap H1 \cap ... \cap H(i-1) \cap \overline{Hi})$ as a higher-order-logic function. The main idea consists in dividing this set into two parts: the first one must define the intersection of the $(i-1)$ first events while the second has to illustrate the event that *"the i^{th} working sub-network is non-empty within Ti"*. The intersection is introduced in HOL by the conjunction *"and"*.

The function `compl_intersection`, given in Definition 9, illustrates the first part of the required final set. It builds the intersection of events describing the $(i-1)$ first empty subsets. If we make a list of the needed random variables (function `subset_non_empty_rv_list`) by satisfying the independence criteria (function `indep_rv_list`), we can then create the disjunction and then the conjunction of all the elements of the list as required. The function `compl_intersection` takes as parameters: i, a natural index, c, the number of sensors that covers a specific point inside the field, and k, the number of sub-networks.

`Definition 9:`

```
⊢ ∀ i,c,k.  compl_intersection i c k =
            bind (indep_rv_list (subset_non_empty_rv_list i c k))
                                        (λx.  unit (disj_list x)).
```

The function `subset_non_empty_rv_list`, specified in Definition 10, generates recursively a list of `subset_non_empty` random variables. This definition seems to have the same goal as Definition 6. However, the inputs are different. Indeed, in this new definition, we have to take into account that the subsets are working alternatively within a given scheduling cycle and consequently the number $(k-i)$ of remaining subsets is decreasing when we move ahead in the cycle.

```
Definition 10:
⊢ (∀ c,k.  subset_non_empty_rv_list 0 c k = []) ∧
  (∀ i,c,k.  subset_non_empty_rv_list (SUC i) c k =
  (subset_non_empty (k-i-2) c (k-i))::(subset_non_empty_rv_list i c k)).
```

We ensure the independence of the random variables of a given list by the following function `indep_rv_list`.

```
Definition 11:
⊢ (indep_rv_list [] = unit []) ∧
  (∀ h,t.  indep_rv_list (h::t) = bind h (λx.  bind (indep_rv_list t)
                                                  (λy.  unit (x::y)))).
```

Finally, the function `disj_list`, provided in Definition 12, is used to create the disjunction of all the elements of a given list.

```
Definition 12:
⊢ (disj_list [] = F) ∧
  (∀ h t.  disj_list (h::t) = (&h = 1) ∨ disj_list t).
```

The second part of the final set can be described by the random variable `subset_non_empty` (Definition 5) which can express also the event of an empty subset. Thus, the final set is described by the following HOL function `final_set` which takes the same parameters as the function `compl_intersection`.

```
Definition 13:
⊢ ∀ i,c,k.  final_set i c k =
                        bind (compl_intersection i c k) (λx.
                        bind (subset_non_empty (k-i-1) c (k-i)) (λy.
                        unit (∼x ∧ (y = 1)))).
```

Regarding the integral used in (11), we need to define the corresponding primitive function. This primitive is specified by the HOL function `dint_val_fct`. Its inputs i, c and k have the same signification as in Definition 9. The parameter tm denotes the time slot Ti and x is the variable of the primitive function.

```
Definition 14:
⊢ ∀ i,c,k,tm,x.  dint_val_fct i c k tm x =
  ((1/tm)prob bern {s | fst (final_set i c k s) = T})
  ((&itmx)-((1/2)(x²))).
```

When computing the integral, we will need to substitute the variable x of the above definition by the edges of the integral. This operation is expressed in the following function `dint_vals` which takes the same parameters as the above function.

```
Definition 15:
⊢ ∀ i,c,k,tm.  dint_vals i c k tm =
       ((dint_val_fct i c k tm tm) - (dint_val_fct i c k tm 0)).
```

4 Formal Verification of the Random Scheduling Algorithm

We use the defined HOL functions in order to formally verify the main statistical properties regarding the network coverage intensity and the average detection delay. We will describe the verified theorems in a backward chaining approach, i.e., we will present first the main goal then we will detail as much as possible the intermediate needed theorems and the corresponding proofs.

4.1 Formal Verification of the Network Coverage Intensity

We have already noticed from the specification section that the network coverage intensity is defined as a statistical measure of the coverage intensity for a specific point (see (10)). Hence, we need to verify first that the coverage intensity for a specific point, defined in (7), is really equal to the expression given in (8). The HOL theorem formalizing this property can be expressed as follows:

```
Theorem 2:
⊢ ∀ c,k.  cvrge_intsty_pt c k = 1 - (1 - (1/&(SUC k)))ᶜ.
```

The verification of the above theorem is based on Theorem 3 which gives the value of the expectation of the sum of the random variables `subset_non_empty`.

```
Theorem 3:
⊢ ∀ c,k.  expec (sum_rv_lst (subset_non_empty_lst k c (SUC k))) =
          &(SUC k) × (1 - (1 - (1/&(SUC k)))^c).
```

The proof of Theorem 3 needed mainly three intermediate theorems. The first one, Theorem 4, rewrites the right hand side of Theorem 3, verifying that the expectation of the sum of the random variables `subset_non_empty` is the sum of their respective expectations. The proof of Theorem 4 uses Theorem 1, the pre-verified theorem formalizing (2), which further requires the verification of two other basic conditions. The first condition proves that each element of the list `subset_non_empty_lst` is independent while the second says that each of these elements is "`summable`".

```
Theorem 4:
⊢ ∀ c,k.  expec (sum_rv_lst (subset_non_empty_lst k c (SUC k))) =
        sum (0,LENGTH (subset_non_empty_lst k c (SUC k)))
        (λ.  expec(EL (LENGTH (subset_non_empty_lst k c (SUC k))-(x+1))
        (subset_non_empty_lst k c (SUC k)))).
```

Theorem 5, the second theorem needed for the proof of Theorem 3, determines the length of the list of random variables. The proof of this theorem was straightforward by induction.

```
Theorem 5:
⊢ ∀ c,k,m.  (1 ≤ m) ∧ (k ≤ m) ⇒
                    (LENGTH (subset_non_empty_lst k c m) = (SUC k)).
```

Finally, the following theorem is the third theorem needed for the proof of Theorem 3. It verifies the expectation value of each element of the list `subset_non_empty_lst`.

```
Theorem 6:
⊢ ∀ c,m,k,j.  (1 ≤ m) ∧ (k ≤ m) ∧ (j ≤ k) ⇒
  (expec (EL j (subset_non_empty_lst k c m)) = 1 - (1 - (1/&m))^c).
```

The proof of Theorem 6 requires first to verify that each element of the list `subset_non_empty_lst` is a Bernoulli random variable. This is expressed in Theorem 7.

```
Theorem 7:
⊢ ∀ c,m,k,j.  (1 ≤ m) ∧ (k ≤ m) ∧ (j ≤ k) ⇒
  (EL j (subset_non_empty_lst k c m) = bernoulli_num (1-(1-(1/(&m)))^c)).
```

The proof of Theorem 7 is done by induction besides using the PMF of the HOL function `subset_non_empty` which is given in Theorem 8. The assumptions of Theorem 8 are essentially related to the call of the function `rd_subsets` which in turn calls the Uniform random variable. The assumptions on this random variable must be hence satisfied. The HOL proof of Theorem 8 is based on some reasoning associated to the function independence, the transformation of probabilistic sets and classical real results.

```
Theorem 8:
⊢ ∀ j,c,k.  (1 ≤ k) ∧ (j ≤ k) ⇒
   (prob bern {s | fst (subset_non_empty j c k s)=1} = (1-(1-(1/&k))^c)).
```

At the end, we use Theorem 7 jointly with the expectation theorem for the Bernoulli random variable, verified in [12], and some mathematical reasoning in order to complete the proof of Theorem 6.

Theorem 9 is the second main theorem. It verifies the network coverage intensity. The steps needed for the proof of Theorem 9 are detailed further down.

```
Theorem 9:
⊢ ∀ n,q,k.  (0 ≤ q) ∧ (q ≤ 1) ∧ (1 ≤ n) ⇒
   (cvrge_intsty_network q n k = (1 - (1 - (q/&(SUC k)))^n)).
```

As we have indicated in Definition 8, the HOL function `cvrge_intsty_network` is a function of random variable. The proof of Theorem 9 requires the application of the linearity of expectation property (see Theorem 1). The two supplied conditions for this theorem are related to the independence of the Binomial random variable which is already verified in [12] and the finite summation of the corresponding function multiplied by the probability.

Theorem 10 formalizes the finite summation of the corresponding function multiplied by the probability. We verify the following theorem in HOL and the proof is primarily based on the PMF of the Binomial random variable, verified in [12], and some mathematical reasoning related to the real summation.

```
Theorem 10:
⊢ ∀ q,k,n.  (0 ≤ q) ∧ (q ≤ 1) ∧ (1 ≤ n) ⇒
   summable (λi.  (1-1/&(SUC k))^i ×
   prob bern {s | fst (prob_binomial n q s)=i}).
```

In order to achieve the verification of Theorem 9, we need also to verify the expectation of the function `(λi.  (1-1/&(SUC k))^x)` of the Binomial random variable. This is illustrated in Theorem 11.

```
Theorem 11:
⊢ ∀ q,k,n.  (0 ≤ q) ∧ (q ≤ 1) ∧ (1 ≤ n) ⇒
   (expec_fn (λi.(1-1/&(SUC k))^i)(prob_binomial n q) = (1-q/&(SUC k))^n).
```

The proof of the above theorem is based on the Binomial theorem for reals, described in Theorem 12, which was not available in the existing HOL libraries and thus we also had to prove it.

```
Theorem 12:
⊢ ∀ a,b,n.  (n ≤ 1) ⇒
   ((a + b)^n = sum (0,SUC n) (λi.  &(binomial n i) × (a^(n-i)× b^i))).
```

The two corollaries expressed in (13) and (14) can be easily inferred from Theorem 9. Corollary 1 gives the lower bound on the number n of sensors such as the network coverage intensity is of at least t. It states that:

$$n \geq \left[\frac{\ln(1-t)}{\ln\left(1-\frac{q}{k}\right)} \right]. \tag{13}$$

which is verified in:

```
Corollary 1:
⊢ ∀ n k q t.  (1 ≤ n) ∧ (0 ≤ k) ∧ (0 ≤ q) ∧ (q ≤ 1) ∧ (0 ≤ t) ∧
               (t ≤ 1) ∧ (t ≤ cvrge_intsty_network q n k) ⇒
               ((ln (1 - t)/ln (1 - q)/&(SUC k)) ≤ &n).
```

The proof of the above corollary has required intermediate results associated especially to the two mathematical functions of power and logarithm.

Similarly, we can deduce Corollary 2 which states that for a given n and providing a network coverage intensity of at least t, the upper bound on the number of disjoint subsets k is:

$$k \leq \frac{q}{1 - e^{\frac{\ln(1-t)}{n}}}. \tag{14}$$

which is verified in HOL by:

```
Corollary 2:
⊢ ∀ n k q t.  (1 ≤ n) ∧ (0 ≤ k) ∧ (0 ≤ q) ∧ (q ≤ 1) ∧ (0 ≤ t) ∧
               (t ≤ 1) ∧ (t ≤ cvrge_intsty_network q n k) ⇒
               &(SUC k) ≤ (q /(1 - (exp (ln (1-t)/&n)))).
```

The proof of the above corollary was straightforward and is based on pre-verified theorems from the two HOL theories of real and exponential.

4.2 Formal Verification of the Average Detection Delay

The formal verification of the average detection delay has needed a good approach in order to be able to verify the gauge integral given in (11). In fact, the HOL reasoning support for the gauge is not strong enough to handle this function. Therefore, we have to go through theorems defined for the derivative. The derivative is formalized by the HOL function `diffl` and has many pre-verified results which can be used to facilitate our task.

The final goal giving the final average detection delay, is stated in Theorem 13. The proof of this theorem has required a lot of reasoning related to the derivative function and the real summation. Added to that, the proof is based on two important intermediate theorems which will be detailed further down.

```
Theorem 13:
⊢ ∀ c,i,k,tm.  (0 ≤ tm) ∧ (2 ≤ k) ∧ (2 ≤ c) ⇒
  (sum (1,SUC (k-2)) (λi.  dint_vals i c k tm) =
  1/2×tm×((&(k-1)/&k)^c + 2×sum (2,SUC (k-2)) (λi.  (&(k-i)/&k)^c))).
```

Theorem 14 is the first intermediate theorem verified. It expresses the complement of the probability specified in (12). The proof of this theorem is straightforward; it uses the PMF of the random variable `subset_non_empty` which has been already established in Theorem 8.

```
Theorem 14:
⊢ ∀ j,c,k.  (1 ≤ (k - i)) ∧ (j ≤ (k - i)) ⇒
  (prob bern {s | fst (subset_non_empty j c (k-i) s) = 1} =
  (1 - (1 - (1/&(k-i)))^c)).
```

The next theorem to verify must reduce the probability of a set of independent events to the product of their respective probabilities. This is expressed in Theorem 15. The proof of Theorem 15 has required reasoning related to the transformation of probabilistic sets and to the independence theorem of probability. Under some assumptions, this last theorem transforms the probability of the intersection of two independent events into the product of their respective probabilities. Besides that, the proof of Theorem 15 has needed another important intermediate theorem which gives the PMF of the HOL function `compl_intersection`. The function `product_val`, used in Theorem 15, is a recursive function giving the product of a sequence of elements of the same function.

```
Theorem 15:
⊢ ∀ i c k.  (2 ≤ k) ∧ (1 ≤ (k - i)) ⇒
  (prob bern {s | fst (final_set i c k s) = T} =
  product 0 i (λj.  (1 - (1/&(k-j)))^c)×(1 - (1-1/&(k-i))^c)).
```

Our results demonstrate the effectiveness of the approach based on theorem proving for the verification of randomized scheduling algorithms for WSNs. We have been able to demonstrate the most important probabilistic properties of interest associated to the network coverage intensity. The capabilities of theorem proving certify that the obtained results are 100% accurate; a novelty which is not available in simulation. This accuracy is due to the rigorous mathematical notations and the inherent soundness of the approach.

While other techniques like simulation make restriction on the number of simulated nodes or the number of disjoint subsets, our results are completely generic. These results are in fact valid for all values of n, k and q. It is clear that other techniques such as simulation can never have this kind of flexibility. Similarly, due to the high expressibility of higher-order logic, we have been able to formally reason about statistical properties of the problem that cannot be analyzed using a probabilistic model checker.

The above mentioned additional benefits, associated with the theorem proving approach, are attained at the cost of the time and effort spent, while formalizing the randomized scheduling algorithm and formally reasoning about its properties, by the user. This analysis consumed approximately 200 man-hours and 1500 lines of HOL code by an expert user. The fact that we were building on top of already verified probability theory related results helped significantly in this regard.

5 Related Work

Hurd's PhD thesis [16] can be considered as a pioneering work in the context of implementing a theorem proving based probabilistic analysis framework. It presents a methodology for the formalization and verification of probabilistic algorithms in the HOL theorem prover. Random variables are basically probabilistic algorithms and thus can be formalized and verified, based on their probability distribution properties, by following the methodology proposed in [16]. In fact, [16] presents the formalization of some discrete random variables along with their verification, based on the corresponding PMF properties. Building upon Hurd's formalization framework [16], the sampling algorithms of few continuous random variables based on their Cumulative Distribution Function (CDF) properties have been successfully verified in [12]. For comparison purposes, it is frequently desirable to summarize the characteristic of the distribution of a random variable by a single number, such as its expectation or variance, rather than an entire probability distribution function. For example, it is easier to compare the performance of two wireless communication protocols based on the expected values rather than the CDFs of their message transmission delays. In [12], Hurd's formalization framework has been extended with a formal definition of expectation. This definition is then utilized to formalize and verify the expectation and variance characteristics associated with discrete random variables that attain values in positive integers only.

To show the practical effectiveness of the probabilistic framework, Hurd successfully verify the Miller-Rabin primality test; a well-known and commercially used probabilistic algorithm [17]. Recently, the HOL probabilistic theorem proving framework has been greatly enriched so that it has been efficiently used to verify the stop-and-wait protocol [14], a stuck-at fault model for reconfigurable memory arrays [13] and the automated repeat request (ARQ) mechanism at the logic link control (LLC) layer of the General Packet Radio Service (GPRS) standard for Global System for Mobile Communications (GSM) [15]. In all these case studies, the authors illustrate the formal verification of some key statistical properties through the already formalized statistical quantities like the expectation and variance.

Probabilistic model checking can be an alternative solution to probabilistic theorem proving. It is one of the first formal methods to be used for probabilistic analysis of wireless systems [29]. It has the same principle as the traditional model checking: the mathematical model of the probabilistic system is exhaustively tested to check if it meets a set of probabilistic properties. This technique has been successfully used to validate many aspects of wireless sensor networks. For example, the probabilistic model checker PRISM [28] has been frequently used for the verification of MAC layer protocols designed for WSNs. Thus, this tool has verified the S-MAC [2] and the ECO-MAC [40] protocols. In addition to its accuracy, the main advantage of probabilistic model checking method is its mechanization. However, it also suffers from some major shortcomings, such as the common problem of state space explosion [3] and the inability to reason accurately about statistical properties.

Simulation is the most common technique to perform the analysis of random scheduling algorithms designed for wireless sensor networks. In general, the approach consists in building a computer based mathematical model of the given algorithm and then evaluating it through rigorous sampling. Since the scheduling algorithms proposed are probabilistic, the simulation tools must essentially provide some probabilistic features in order to perform realistic simulations. Hence, the random elements have to be modeled by functions which approximate random variables for probability distributions, and the algorithm to be evaluated has to be analyzed using computer simulation techniques [4], such as the Monte Carlo Method [23] . The main idea of this method is to approximately answer a query on a probability distribution by analyzing a large number of samples. Statistical quantities, such as expectation and variance, may then be calculated, based on the data collected during the sampling process, using their mathematical relations in a computer.

In the literature, many works deal with the coverage-based random scheduling algorithm of [22]. In the original work, the algorithm has been analyzed by a mathematical model which has been then evaluated on a Java simulator. The same work has been simulated in MATLAB where the authors have focused on the second part of the algorithm by showing the efficiency of the proposed extra-on rule [24]. Finally, the work of [20] has tried to enhance the coverage-based scheduling model by eliminating some blind points. The simulation results have been made on a Java simulator used in [22]. The authors have shown that they can significantly improve the coverage quality of the randomized scheduling. In [32], a similar scheduling scheme is proposed. The effectiveness of this scheme in terms of energy has been established by using the simulation tool NS-2. Another probabilistic algorithm proposed in this context is called the optimal geographical density control (OGDC) algorithm for WSNs [27]. The authors advocate that the OGDC algorithm can maintain complete sensing coverage and connectivity of a region based on the nodes scheduling. When simulated in the NS-2 simulator, the OGDC algorithm has shown its great robustness as it outperforms all the past algorithms. Nevertheless, OGDC is considered to be very sophisticated because it has to take into account a lot of geometric information such as the relative location of each node which is a very costly information in the scope of WSNs. The same OGDC algorithm has been also simulated and formally validated in the Real-Time Maude [33] tool where the authors have successfully analyzed the common performance metrics such as the network coverage intensity and lifetime [27]. The Real-Time Maude tool provides a spectrum of analysis methods including simulation, reachability analysis and temporal logic model checking. Although the formal analysis has shown the effectiveness of the RT-Maude tool within WSNs, two major limitations persist. First, the number of simulated nodes was limited because of the state-based approach. Second, the probabilistic aspect of the OGDC algorithm has not been really taken into account. Therefore, the authors of [27] suggest the use of PMaude [8] as an alternative tool to perform the probabilistic analysis of OGDC.

Due to the inherent nature of simulation coupled with the usage of computer arithmetic, the probabilistic analysis results attained by the simulation approach can never be termed as completely accurate. Moreover, the model checking approach usually suffers from the common limitations cited above. To the best of our knowledge, none of the past works about the random scheduling algorithm for WSNs or one of its variant have incorporated a formal probabilistic technique based on model checking or theorem proving. In this paper, we overcome the limitations of both simulation and model checking techniques by using the probabilistic framework developed in the HOL theorem prover to validate a variant of the randomized scheduling of nodes in the context of WSNs.

6 Conclusions

Due to the deployment constraints of WSNs, we are more motivated to provide algorithms characterized by a probabilistic behavior. Such a characteristic is impossible to cover using classic validation procedures like simulation, which do not ascertain 100% accuracy. In this paper, the purpose was to provide reliable validation by using an accurate verification tool, which is the probabilistic framework developed in HOL. We formally analyzed the coverage and the average detection delay of a variant of a nodes scheduling algorithm designed for randomly deployed wireless sensor networks. We particularly verify the expected values of the coverage intensity, the upper bound on the total number of disjoint subsets, the lower bound on the total number of nodes and the average detection delay inside the network.

To the best of our knowledge, this paper presents the first formal analysis of a randomized scheduling problem using a probabilistic formal method. The successful reasoning about statistical properties clearly demonstrates the practical effectiveness of the proposed approach compared to probabilistic model checking, where such a feature is not available. Once the HOL probabilistic framework is enriched with possibilities to reason about statistical properties of multiple continuous random variables, it will be promising to extend the formal analysis of the coverage-based scheduling algorithm. We can for example think to formally verify the network lifetime which is a crucial aspect in the WSNs context or the impact of clock asynchrony on the coverage quality.

References

[1] Z. Abrams, A. Goel, and S. Plotkin. Set k-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks. In IEEE Press, editor, *International Symposium Information Processing in Sensor Networks*, pages 424–432, 2004.

[2] P. Ballarini and A. Miller. Model checking medium access control for sensor networks. In *Proceedings of the Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pages 255–262, Washington, DC, USA, 2006. IEEE Computer Society.

[3] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 2000.

[4] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[5] P. Godfrey and D. Ratajczak. Naps: scalable, robust topology management in wireless ad hoc networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 443–451, New York, NY, USA, 2004. ACM.

[6] M. J. C. Gordon. Mechanizing Programming Logics in Higher-0rder Logic. In *Current Trends in Hardware Verification and Automated Theorem Proving*, pages 387–439. Springer-Verlag, 1989.

[7] M. J. C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.

[8] A. Gul, M. José, and S. Koushik. Pmaude: Rewrite-based specification language for probabilistic object systems. *Electron. Notes Theor. Comput. Sci.*, 153:213–239, May 2006.

[9] A. Gupta. Formal hardware verification methods: a survey. *Form. Methods Syst. Des.*, 1.

[10] J. Harrison. Formalized Mathematics. Technical Report 36, Turku Centre for Computer Science (TUCS), Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland, 1996.

[11] J. Harrison. *Theorem Proving with the Real Numbers.* Springer, Heidelberg, 1998.

[12] O. Hasan. *Formal Probabilistic Analysis using Theorem Proving.* PhD thesis, Concordia University, Montreal, QC, Canada, 2008.

[13] O. Hasan, N. Abbasi, and S. Tahar. Formal probabilistic analysis of stuck-at faults in reconfigurable memory arrays. In *Proceedings of the 7th International Conference on Integrated Formal Methods*, IFM '09, pages 277–291, Berlin, Heidelberg, 2009. Springer-Verlag.

[14] O. Hasan and S. Tahar. Performance analysis and functional verification of the stop-and-wait protocol in hol. *J. Autom. Reason.*, 42.

[15] O. Hasan and S. Tahar. Probabilistic analysis of wireless systems using theorem proving. *Electron. Notes Theor. Comput. Sci.*, 242.

[16] J. Hurd. *Formal Verification of Probabilistic Algorithms.* PhD thesis, University of Cambridge, Cambridge, UK, 2002.

[17] J. Hurd. Verification of the miller-rabin probabilistic primality test. *Logic and Algebraic Programming*, 50(1-2):3–21, 2003.

[18] S. Jain and S. Srivastava. A survey and classification of distributed scheduling algorithms for sensor networks. In *007 International Conference on Sensor Technologies and Applications*, SENSORCOMM '07, pages 88–93, Washington, DC, USA. IEEE Computer Society.

[19] A. Levine. *Theory of Probability.* Addison-Wesley series in Behavioral Science. Addison-Wesley Publishing Company, 1971.

[20] J.W. Lin and Y.T. Chen. Improving the coverage of randomized scheduling in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 7(12):4807–4812, 2008.

[21] C. Liu. Randomized scheduling algorithm for wireless sensor neworks. Technical report, University of Victoria, Canada, 2004.

[22] C. Liu, K. Wu, Y. Xiao, and B. Sun. Random coverage with guaranteed connectivity: Jointscheduling for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(6):562–575, 2010.

[23] D. J. C. MacKay. Introduction to Monte Carlo methods. In *Learning in Graphical Models, NATO Science Series*, pages 175–204. Kluwer Academic Press, 1998.

[24] M. Mahdavi, M. Ismail, K. Jumari, and Z. M. Hanapi. Performance of a connected random covered energy efficient wireless sensor network. *IJECE*, 5(2):216–220, 2010.

[25] A. K. McIver and A. Fehnker. Formal techniques for the analysis of wireless networks. In *the Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pages 263–270. IEEE Computer Society, Washington, DC, USA, 2006.

[26] C. Mihaela and D. Ding-Zhu. Improving wireless sensor network lifetime through power aware organization. *Wirel. Netw.*, 11:333–340, May 2005.

[27] P.C Ölveczky and S. Thorvaldsen. Formal modeling and analysis of the ogdc wireless sensor network algorithm in real-time maud. In Marcello M. Bonsangue and Einar Broch Johnsen (Eds.), editors, *the 9th IFIP WG 6.1 international conference on Formal methods for open object-based distributed systems.*

[28] PRISM. www.cs.bham.ac.uk/∼dxp/prism, 2006.

[29] J. Rutten, M. Kwaiatkowska, G. Normal, and D. Parker. Mathematical Techniques for Analyzing Concurrent and Probabilisitc Systems. *CRM Monograph*, 23, 2004.

[30] S. Slijepcevic and M.Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, pages 472–476, New York, NY, USA, 2001. IEEE Press.

[31] The HOL theorem prover website. http://hol.sourceforge.net/.

[32] D. Tian and N.D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 32–41, New York, NY, USA, 2002. ACM.

[33] The Real-Time website. //www.ifi.uio.no/realtimemaude/.

[34] K. Wu, Y. Gao, F. Li, and Y. Xiao. Lightweight deployment-aware scheduling for wireless sensor networks. *Mobile Networks and Applications*, 10(6):837–852, 2005.

[35] Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. Estrin. Topology control protocols to conserve energy in wireless ad hoc networks. Technical report, Center for Embedded Networked Computer, University of California, USA, 2003.

[36] T. Yan, T. He, and J. Stankovic. Differentiated surveillance for sensor networks. In *International Conference on Embedded Networked Sensor Systems*, First, pages 51–62, ACM Press, New York, 2003. IEEE Press.

[37] R. D. Yates and D. J. Goodman. *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers*. Wiley, 2^{nd} edition, 2005.

[38] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. Peas: A robust energy conserving protocol for long-lived sensor networks. In *10th IEEE International Conference on Network Protocols*, pages 28–37. IEEE Press, New York, 2002.

[39] J. Yick, B. Mukherjee, and B. Ghosal. Wireless sensor network survey. *Computer Networks*, 52, 2008.

[40] H. Zayani, K. Barkaoui, and R.Ben Ayed. Probabilistic verification and evaluation of backoff procedure of the wsn eco-mac protocol. *Wireless & Mobile Networks*, 2(2):156–170, 2010.