A Probabilistic Model of the LMAC Protocol for Concurrent Wireless Sensor Networks

Esparza, Luz Judith R; Zeng, Kebin; Nielsen, Bo Friis

Published in:
2011 11th International Conference on Application of Concurrency to System Design (ACSD)

Link to article, DOI:
10.1109/ACSD.2011.20

Publication date:
2011

A Probabilistic Model of the LMAC Protocol for Concurrent Wireless Sensor Networks

Luz Judith R. Esparza Kebin Zeng Bo Friis Nielsen
Informatics and Mathematical Modelling
Technical University of Denmark
KGS. Lyngby, Denmark
Email: {ljre,keze,bfn}@imm.dtu.dk

***Abstract*—We present a probabilistic model for the network setup phase of the Lightweight Medium Access Protocol (LMAC) for concurrent Wireless Sensor Networks. In the network setup phase, time slots are allocated to the individual sensors through resolution of successive collisions. The setup phase involving collisions should preferably be as short as possible for efficiency and energy consumption reasons. This concurrent stochastic process has inherent internal non-determinism, and we model it using combinatorics. The setup phase is modeled by a discrete time Markov chain such that we can apply results from the theory of phase type distributions. Having obtained our model we are able to find optimal protocol parameters. We have simultaneously developed a simulation model, partly to verify our analytical derivations and partly to be able to deal with systems of excessively high order or stiff systems that might cause numerical challenges. Our abstracted model has a state space of limited size where the number of states are of the order $\binom{n+r+1}{n}$, where n is number of sensors, and r is the maximum back off time. We have developed a tool, named LMAC analyzer, on the MATLAB platform to assist automatic generation and analysis of the model.**

***Keywords*-probabilistic model; performance analysis; wireless sensor networks;**

I. Introduction

Wireless Sensor Networks (WSN) consist of widely distributed sensors that cooperatively monitor physical or environmental conditions, and have been used in widespread applications. WSN is one of the prime examples of networked embedded systems, where many modern computer science challenges exist, such as the challenges in distributed computing, wireless communication, and system integration. One major consideration in WSN is how to prolong the network lifetime. The Lightweight Medium Access Protocol (LMAC) was introduced in [15], designed as a multi-hop and energy-efficient protocol for WSN at the Medium Access Control (MAC) layer. In the LMAC protocol, the network is self-organizing in terms of time slot assignment and synchronization. The protocol uses Time Division Multiple Access (TDMA), where each time slot is assigned to a sensor. In this manner, the nodes can communicate collision-free after the network has stabilized. In this way, the protocol provides energy efficiency. The LMAC protocol gives a significant lifetime improvement compared to prior protocols, such as EMACs and SMAC. Thus, analyzing and reducing collisions in order to optimize the network have been some of the remaining challenges. As a consequence, we will concentrate on the part of the protocol that is responsible for the distributed and localized strategy of assigning time slots to sensors.

In previous works [7], [14], [17], the concurrent behaviour of the LMAC nodes was modelled using parallel decomposition. The complexity of the models rose drastically with the number of network nodes, limiting the analysis to small–sized networks far from reality. In addition, the random nature of the slot selection process required further analysis and modelling. In our approach, we will use a direct mathematical method with immediate abstraction of network details that does not influence the time for the network to stabilize.

II. Related Work

In [7], [14], formal verification of the LMAC protocol is investigated in the timed automaton model checker UPPAAL [1]. The LMAC network is modelled by parallel composition of single node behaviours. The properties for model checking primarily focus on the fundamental mechanisms. For example, checking whether collisions can be detected or a new choice of slots is initiated after collision. The UPPAAL model has been used to systematically investigate all topologies with 4 and 5 nodes. Based on this work, the LMAC protocol has been updated by patching discovered bugs, and problematic topologies with possible scenarios of unsolved collision have been identified. However, the UPPAAL model has encountered serious state space explosion, if the model contains more than 5 nodes. Moreover, probabilistic aspects of LMAC, e.g. optimal parameters, have been mentioned as important future work. This inspired our work and has now become one of our key contributions.

A study on probabilistic aspects of LMAC is given in [17] relying on timed automata in the probabilistic version of UPPAAL (UPPAAL PRO 0.2 [6]). The probabilistic choice has been made by pre-assigned weights to all possible transitions when nodes select back off time after collision. By changing the weights, various probability distributions

represent different back off strategies. It shows that if the back off time before starting to pick a new time slot increases, the number of collisions will decrease. In this work [17], slot selection is modelled in a deterministic way. Each individual sensor keeps trying from the first time slot and then the second, until it finds a free one. Thereby, later coming sensors unavoidably have a number of collisions before they settle down. This has negative influences on the overhead of the protocol, therefore we suggest a probabilistic solution. At last, the probabilistic UPPAAL model encounters even severer state explosion problem than the non-probabilistic version, which is capable of modelling a maximum of 4 nodes under the fully connected topology.

In summary, previous works in the slot selection phase of the LMAC protocol rely on parallel composition approach. Modelling the non-deterministic nature and solving the state space dilemma are very interesting topics. In our work, we propose a mathematical approach using a direct abstraction technique to solve these concurrent stochastic problems.

III. The LMAC Protocol

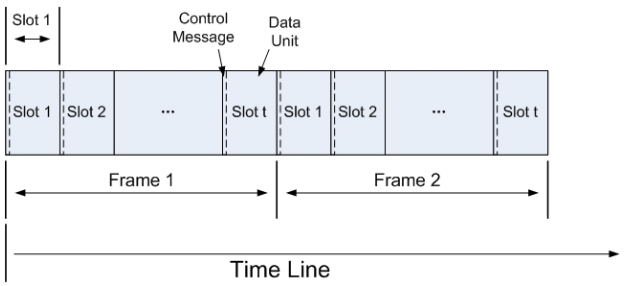

Figure 1. Time structure in LMAC

As a schedule-based MAC protocol, the time in LMAC is organized in *time slots*, which are grouped into *frames* (see figure 1). For each time slot, the controlling node always transmits a fixed length (12 bytes, [15]) *control message* in order to maintain synchronization. The control message also carries a node ID of the time slot controller, the size of the data unit and the intended receiver. In particular, the control message is critical for broadcasting information regarding the occupied time slots. For this reason, late coming nodes can pick only free slots. The remaining part of a time slot is an optional *data unit* if there are any needs. The current maximum size of the data unit is 256 bytes, [15]. During each frame, nodes can only transmit messages in their own time slot, for the rest of the time they can only receive messages. In this manner, energy consumption is minimized.

At the beginning of the network setup phase, all of the nodes are unsynchronized. In order to get synchronization, one (or more) gateway node(s) will take initiative to start controlling the time slot(s), i.e. becoming the master node(s). Control messages from the gateway will be received by its one-hop neighbours. Once these nodes get their time slots, they will start sending control messages to the other hops. The network will stabilize once all nodes get their reserved time slots. Thereafter, nodes can communicate with each other in a collision free manner.

Figure 2 describes the behaviour in terms of the phases for an individual node in LMAC.

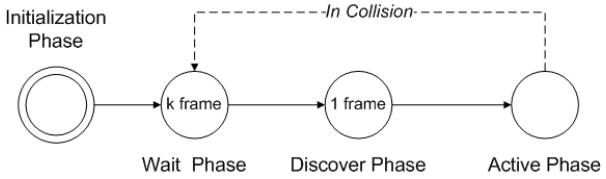

Figure 2. Phase diagram for an individual sensor

Initialization Phase: When a sensor node powers on, it is unsynchronized. In this phase, a node will try to detect its neighbouring nodes. As long as at least one neighbouring node is detected, the node will synchronize with it and go to the wait phase.

Wait Phase: The wait phase is designed with the purpose of reducing the number of nodes that pick slots at the same time, which helps reducing the probability of collision. In this phase, a node waits at random k frames, where k is an integer number from the set $S = \{0, 1, \ldots, r\}$, where r is the maximum back off time. After waiting k frames, the node will go to the discover phase.

Discover Phase: Before a node starts to pick a time slot, it registers all the currently available slots in order to pick only among those. This happens in the discover phase where nodes compute free slot information based on the control messages. Afterwards, it will randomly select one of the available slots and go to the active phase. A node stays one time frame in this phase.

Active Phase: After a node has picked a time slot in the discover phase, it will start to transmit a message and receive messages from neighbouring nodes. Here, if there are two or more nodes transmitting simultaneously, a collision occurs. Then neighbouring nodes will send control messages to ask them to give up their time slots and go to the wait phase.

In the LMAC specification, the back off mechanism for collided nodes is underspecified. Thereby, it is possible for nodes to start back off in either the current frame or the next. It depends on whether there are neighboring nodes to register collisions for the discover identities at the remaining time of the frame. This non-determinism can be interpreted as implementation freedom. Considering the worst case scenario, we resolve the non-determinism by assuming that back off always starts in the successive frame.

Moreover, to limit the number of time slots necessary in the network, the LMAC protocol allows for time slots to be reused at non-interfering distances. In [7] is proved that it is safe to share the same time slot after at least three hops.

The LMAC protocol uses a distributed algorithm described in [10] and [13] to manage the division of time slots.

IV. A PROBABILISTIC MODEL OF THE LMAC PROTOCOL

The component-based approach to model the concurrency in the stabilization process was applied in [7] and [17], where the behaviour of a single node is modelled as a basic component. Indeed, the system properties are represented in terms of parallel composition of individual nodes. The compositional way is inherently close to the protocol specification, which gives a detailed verification result. The state space explosion problem, however, restricts model checking experiments to 5 nodes. This is far from the WSN applications, where the number of sensors could be up to hundreds. Therefore, we will propose a lightweight model that will be valid for the verification task at hand.

In [7], the case with 5 nodes considering all the 61 topologies has been investigated. These different topologies can lead to dramatically varied verification results. Generally, it is hard to identify a representative one. Hence, throughout the paper, we will assume only the fully connected topology, which has been proven as one of the successful topologies.

A. System abstraction

Associated with parallel composition, the abstraction methods described in [3] and [5] produce abstracted models to reduce the state space problem of model checking. But, in order to do so, they have to start with a detailed model. As an alternative, our mathematical approach will attack the highly abstracted model directly, thereby obtaining a huge reduction of the state space. The system is abstracted by the statistical collection of system level information based on the given LMAC specification. The statistical collection is represented by a data vector which also represents the state of the system. In our implementation, we will propose an injective function, which maps a set of data vectors to a set of positive integers in order to construct a one dimensional discrete state space.

In section III, we introduced the LMAC protocol, which uses a distributed algorithm to divide time slots in order to reuse them in more than 2 hops. Thereby, it is sufficient to analyze the worst case of at most 2 hops distance to characterize the overall network. Thus, we will only model the behaviour for the worst case, defined by $\max\{\frac{n}{t} \mid \frac{n}{t} \leq 1; n, t \text{ within 2 hops}\}$, where n and t are the system variables defined below.

System variables

In the following we will define the system variables.

- n is the number of sensors.
- t is the number of time slots, where we assume $t \geq n$, since the number of time slots in each frame is, at least, equal to the number of nodes in the network.
- r is the maximum back off (waiting) time.

Now, assuming that the system is in frame $j \geq 0$, we define the following:

- $\mathbf{X}_j = (X_{j,0}, X_{d_j}, X_{j,1}, X_{j,2}, \ldots, X_{j,r})$ is the state vector which collects the system information, where
  $X_{j,0}$ is the number of sensors with a reserved slot.
  X_{d_j} is the number of sensors in the discover phase.
  $X_{j,s}$ is the number of sensors which will wait s more ($s \in \{1, \ldots, r\}$) frames.
- Y_j is the number of sensors that successfully get a slot in frame j.
- $Z_{j,s}$ is the number of sensors that collided in frame j and chose to wait s ($s \in \{1, \ldots, r\}$) frames. The vector $\mathbf{Z}_{j+1} = (Z_{j+1,1}, \ldots, Z_{j+1,r})$ is used to record results of the random choice for back off time from a multinomial distribution with parameters $X_{d_j} - Y_{j+1}$ and $\mathbf{p}$, where $\mathbf{p}$ is an r dimensional vector corresponding to a uniform distribution, i.e. $\mathbf{p} = (\frac{1}{r}, \ldots, \frac{1}{r})$. Based on various kinds of back off time selection strategies, however, $\mathbf{p}$ can be an arbitrary probability vector.

Hence, we have the basic identity

$$X_{j,0} + X_{d_j} + \sum_{s=1}^{r} X_{j,s} = n, \quad (j \geq 0)$$

as the sensors can only be in the active phase ($X_{j,0}$), the discover phase (X_{d_j}) or the wait phase ($\sum_{s=1}^{r} X_{j,s}$). With the variables defined above, we are able to capture the dynamics of the process.

Algorithm 1 The LMAC Simulation Algorithm

Require: $\mathbf{X}_0 := (0, n, 0, 0, \ldots, 0)$, $Y_0 := 0$, $j := 0$
repeat
  Generate $Y_{j+1} \leftarrow \mathbb{P}_{t-X_{j,0}, X_{d_j}}(Y = y)$
  $X_{j+1,0} \leftarrow X_{j,0} + Y_{j+1}$
  Generate $\mathbf{Z}_{j+1}$ from multinomial distribution
  for $s = 1$ to $r - 1$ **do**
    $X_{j+1,s} \leftarrow X_{j,s+1} + Z_{j+1,s}$
  end for
  $X_{j+1,r} \leftarrow Z_{j+1,r}$
  $X_{d_{j+1}} \leftarrow X_{j,1}$
  $j \leftarrow j + 1$
until $X_{j+1,0} = n$

The dynamic is driven by frames as the basic time unit where the vector $\mathbf{X}_j$ is used to record the system information. The network starts when all the sensors are unsynchronized and are attempting to get (unreserved) slots. As time elapses, an increasing number of sensors get a reserved slot. Eventually, the system stabilizes when all of the nodes have a reserved time slot. The whole process is modelled as a Discrete Time Markov Chain (DTMC) and the total time spent on the stabilization process is phase type

distributed ([11], [12]). The absorbing state of the underlying Markov chain is the state where all of the sensors have their reserved slots. Algorithm 1 illustrates the dynamics by pseudo code. Here, we identify the initial state of the system under the worst case, where all the sensors are trying to get their slots in the first frame. Indeed, the worst case gives the highest likelihood for the sensors to experience collisions at the beginning.

Table I exemplifies one scenario of the network dynamics of our abstracted model following Algorithm 1, where r = 3 and n = 3. The initial data vector is (0,0,0,0,0,3,0,0,0), where 3 sensors are in the discover phase, X_{d_0}. Unluckily, all of the three collide (2 sensors wait 2 frames and 1 sensor waits 3 frames), which is depicted by the second line of the table. Thereafter, time just passes and the sensors are waiting their turn to retry. In the end, all the sensors successfully get reserved time slots. Thereby, the number of sensors having reserved slots, $X_{j,0}$, becomes 3.

Table I
EXAMPLE FOR ALGORITHM 1 CONSIDERING $r = 3$ AND $n = 3$

Y_j	$Z_{j,1}$	$Z_{j,2}$	$Z_{j,3}$	$X_{j,0}$	X_{d_j}	$X_{j,1}$	$X_{j,2}$	$X_{j,3}$
0	0	0	0	0	(3)	0	0	0
0	0	2	1	0	0	0	2	1
0	0	0	0	0	0	2	1	0
0	0	0	0	0	2	1	0	0
0	2	0	0	0	1	2	0	0
1	0	0	0	1	2	0	0	0
0	1	1	0	1	0	1	1	0
0	0	0	0	1	1	1	0	0
1	0	0	0	2	1	0	0	0
1	0	0	0	(3)	0	0	0	0

B. Analysis of randomness

Two probabilistic choices occur in step 2 and 4 of Algorithm 1. In the following, we will formalize and calculate both assuming that the system is in arbitrary frame j.

1) Probability distribution of slot selection: In step 2 of Algorithm 1, we need to find the random number Y_{j+1}, which is the number of new sensors with reserved slots. Note that Y_{j+1} could be any integer value in the interval [0, X_{d_j}] except $\{X_{d_j} - 1\}$, which has a probability distribution described below.

To shorten notation, we define some new variables to assist us in calculating probabilities derived from the state vector $\mathbf{X}_j$:

- l is the number of current free slots, i.e. $l = t - X_{j,0}$, the number of current free slots is equal to the total available number of slots minus the number of reserved slots.
- k is the number of sensors in the discover phase, i.e. $k = X_{d_j}$ and $k \leq l$.
- y is the number of new sensors with reserved slots, i.e. $0 \leq y \leq k$.
- x is the number of sensors experiencing collision, i.e. $x = k - y$.
- h is the number of unreserved slots, i.e. $h = l - y$.

The probability distribution of slot selection (i.e. distribution of Y) depends on l, k, and y. Let $\mathbb{P}_{l,k}(Y = y)$ denote the probability that y sensors successfully get reserved slots, given the condition that there are l free time slots and k attempting sensors. Based on combinatorial theory, $\mathbb{P}_{l,k}(Y = y)$ is calculated by a rational function with parameters l, k, and y. The numerator counts the number of combinations where y sensors are reserved, and the denominator is the number of combinations for all possible values of Y.

When none of sensors collide, i.e $y = k$, we have that

$$\begin{aligned}\mathbb{P}_{l,k}(Y = k) &= \frac{1}{l^k}\binom{l}{0}\frac{h!}{(h-0)!}\frac{k!}{x!} \\ &= \frac{1}{l^k}\binom{k}{x}.\end{aligned}$$

Obviously, $\mathbb{P}_{l,k}(Y = k - 1) = 0$ since it is impossible to have collision involving just one sensor.

If 2 or 3 sensors are in collision, they can only collide in a single slot. However, for 4 or more sensors the collision can happen in more than one time slot. To consider a general case, we have the formula for $y \geq 2$

$$\begin{aligned}\mathbb{P}_{l,k}(Y = y) &= \frac{1}{l^k}\binom{l}{y}\sum_{i=1}^{\lfloor \frac{x}{2} \rfloor}\left[\frac{h!}{(h-i)!}k!p_i(x)\right] \\ &= \frac{1}{l^k}\binom{k}{x}\sum_{i=1}^{\lfloor \frac{x}{2} \rfloor}\left[\frac{l!}{(l-y-i)!}x!p_i(x)\right] \quad (1)\end{aligned}$$

where

- $\lfloor \cdot \rfloor$ is the floor function. The number of slots with collided sensors can vary from 1 up to $\lfloor \frac{x}{2} \rfloor$. E.g. if we have 7 collided sensors, these can be in at most $\lfloor \frac{7}{2} \rfloor$ = 3 slots.
- $p_i(x)$ is an iterative function that depends on x and i. The way of finding the explicit form of this function is described in appendix.
- $x!p_i(x)$ is the number of ways of putting x sensors into i slots, and these x sensors are under collision.

Thus, from (1), we are able to compute the distribution of slot selection analytically.

2) Probability distribution of back off time: Intuitively, to shorten the setup time we give high priority to small back off times but only in the case where there are only few sensors in the network ([17]). If there are a considerable number of sensors, it becomes necessary to have larger back off time options to reduce the probability of collision. Therefore, there exists an optimistic strategy which chooses

the probability distribution(s) of the back off time such that the expected time till absorption is minimum.

In [16] network latency is taken into account. Four types of strategies to ensure a low latency for the most common data traffic in WSN are proposed. We chose a uniform distribution of the back off times, which is one of the four classic strategies mentioned in [16].

C. Discrete time Markov chain

A Markov chain is a discrete stochastic process with the Markov property, i.e. the next state depends only on the current state. As frames are taken as the time unit, the model is a discrete time model. The model of LMAC is thus a DTMC. Let $E = \{1, 2, \ldots, \binom{n+r+1}{n}\}$ denote the state space of the underlying DTMC with transition matrix defined as follows

$$\mathbf{P} = (p_{m,m'})_{m,m' \in E}. \tag{2}$$

Now, let M be an injective function which maps state vectors $\mathbf{X}_j$, $j \geq 0$, to E in the following way

$$M(\mathbf{X}_j) = \sum_{s=0}^{X_{d_j}} \binom{n+r-s}{s} - \sum_{s=1}^{r} \sum_{i=X_{j,s}+1}^{a} \binom{a+r-s-i}{r-s},$$

where $a = n - X_{d_j} - \sum_{q=1}^{s-1} X_{j,q}$. Note that other mappings could have been used.

Now, suppose that $M(\mathbf{X}_j) = m$ and $M(\mathbf{X}_{j+1}) = m'$, the (m, m')-th element of $\mathbf{P}$ is computed as follows

$$\begin{aligned} p_{m,m'} &= \mathbb{P}(\mathbf{X}_{j+1} | \mathbf{X}_j) \\ &= \mathbb{P}_{t-X_{j,0}, X_{d_j}}(Y = X_{j+1,0} - X_{j,0}) \\ &\quad \cdot \binom{X_{d_j} - (X_{j+1,0} - X_{j,0})}{(X_{j+1,1} - X_{j,2}), \ldots, (X_{j+1,r-1} - X_{j,r}), X_{j+1,r}} \\ &\quad \cdot \left(\frac{1}{r}\right)^{X_{d_j} - (X_{j+1,0} - X_{j,0})}. \end{aligned}$$

As we can see, the transition probability from the state m to the state m', consists of two parts. The first part, $\mathbb{P}_{t-X_{j,0}, X_{d_j}}(Y = X_{j+1,0} - X_{j,0})$, is the probability that $X_{j+1,0} - X_{j,0}$ sensors get reserved slots. The remaining part of the formula calculates the probability that $X_{d_j} - (X_{j+1,0} - X_{j,0})$ collided sensors back off, which is multinomially distributed. Since the discovering sensors can either become reserved or back off, the multiplication of these two probabilities gives the overall transition probability.

With the above formulae, we have built a DTMC analytically using combinatorics to represent the dynamics at the LMAC setup phase. Figure 3 shows an example of a DTMC considering the parameters n = 3, r = 2, and t = 4. With our proposed mapping function M, the state vector is ordered with priority from X_{d_j} until X_r. The initial state of the network is defined with all sensors in the discover phase at the first frame, corresponding to the last row in the transition matrix. The absorbing state of the DTMC is state number 1 at the top row.

			Xdj	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	3
			X1	0	0	0	0	1	1	1	2	2	3	0	0	0	1	1	2	0	0	1	0
			X2	0	1	2	3	0	1	2	0	1	0	0	0	1	2	0	1	0	0	1	0	0
Xdj	X1	X2	M()	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	3	4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	5	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	2	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	2	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	2	1	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	3	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	11	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	12	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	13	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	1	15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	2	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	0	0	17	0.67	0	0.08	0	0	0.17	0	0.08	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	18	0	0	0	0	0.75	0	0.06	0	0.13	0.06	0	0	0	0	0	0	0	0	0	0
2	1	0	19	0	0	0	0	0	0	0	0	0	0	0.75	0	0.06	0	0.13	0.06	0	0	0	0
3	0	0	20	0.38	0	0.14	0.01	0	0.28	0.02	0.14	0.02	0.01	0	0	0	0	0	0	0	0	0	0

Figure 3. Example of the transition matrix $\mathbf{P}$ for $n = 3, r = 2, t = 4$

D. Simulation study

In this section we present the result of a simulation study using Algorithm 1. The network is configured with 4 sensors, 5 time slots, and 2 as maximum back off time. We have made statistical inference on the example, where the standard derivations we obtained indicate the reliability of the estimation from our simulation. Table II shows the probabilities of the system after 5 frames for all 35 states for both analytical and simulation results.

Besides the purpose of verifying the analytical model, simulation also provides an optional way of computing probabilities in the case when the model has high order or stiff systems are hard to be computed analytically.

V. Optimization Using Phase Type

In Section IV we have modelled LMAC by a DTMC using a direct mathematical abstraction. Now, we will investigate optimal parameter settings of the system in order to minimize the time for stabilization of the WSN. In particular, we will analyze the stabilization time when adding a minimal amount of excess capacity.

The critical property of interest is the expected time to stabilization. By definition, the expectation of a discrete random variable is calculated by

$$E(J) = \sum_{j=1}^{\infty} j\mathbb{P}(J = j), \tag{3}$$

where in our case j represents the frame number, and $\mathbb{P}(J = j)$ is the probability of absorption occurring exactly at frame j. The formula contains an infinite sum, therefore it is hard to compute the true value without truncation. Thus, an iterative method is required to guarantee the convergence. However,

Table II
Simulation probabilities considering n=4, r=2, t=5, obtained from 20000 iterations

State	True probability	Simulation	SD
1	0.81291	0.81710	0.00280
2	0.00000	0.00000	0.00000
3	0.00196	0.00240	0.00031
4	0.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000
6	0.00000	0.00000	0.00000
7	0.00392	0.00310	0.00044
8	0.00001	0.00000	0.00000
9	0.00000	0.00000	0.00000
10	0.02748	0.02900	0.00227
11	0.00001	0.00000	0.00000
12	0.00001	0.00000	0.00000
13	0.00044	0.00040	0.00015
14	0.00001	0.00000	0.00000
15	0.00005	0.00005	0.00005
16	0.04662	0.04510	0.00150
17	0.00000	0.00000	0.00000
18	0.00009	0.00015	0.00006
19	0.00000	0.00000	0.00000
20	0.05104	0.05020	0.00160
21	0.00018	0.00020	0.00009
22	0.00001	0.00005	0.00002
23	0.00158	0.00170	0.00028
24	0.00002	0.00005	0.00003
25	0.00018	0.00005	0.00009
26	0.04967	0.04655	0.00150
27	0.00000	0.00000	0.00000
28	0.00002	0.00005	0.00003
29	0.00169	0.00205	0.00029
30	0.00004	0.00005	0.00004
31	0.00037	0.00050	0.00014
32	0.00116	0.00080	0.00024
33	0.00000	0.00000	0.00000
34	0.00036	0.00035	0.00013
35	0.00018	0.00010	0.00009

it is very costly to obtain $\mathbb{P}(J = j)$ as many vector-matrix multiplications are required. We now present an alternative and simplified method to obtain (3).

It is easy to see that the time till absorption of the underlying Markov chain follows a phase type distribution. Phase type distributions were considered first in [11], [12], and are defined as distributions of absorption times in a Markov process with a finite number of transient states and one absorbing state. Using phase type distributions in our analysis gives computational advantages. By assuming that there is at least the same number of time slots as the number of sensors, we are able to guarantee absorption. Thus, we could use phase type distributions to accelerate computations.

First of all, we note that the transition matrix (2) can be rewritten into the form

$$\mathbf{P} = \begin{pmatrix} 1 & \mathbf{0} \\ \boldsymbol{T}_0 & \boldsymbol{T} \end{pmatrix},$$

where $\boldsymbol{T}$ is the transition matrix between transient states and $\boldsymbol{T}_0$ is the vector of probabilities of jumping to the absorbing state, i.e. $\boldsymbol{T}_0 = \mathbf{1} - \boldsymbol{T}\mathbf{1}$, where $\mathbf{1}$ is the column vector consisting of 1's.

Thus, the expectation can be computed efficiently from the phase type properties by

$$E(J) = \boldsymbol{\pi}(\boldsymbol{I} - \boldsymbol{T})^{-1}\mathbf{1}, \tag{4}$$

where $\boldsymbol{\pi}$ is the initial probability vector of the underlying Markov chain and $\boldsymbol{I}$ is the identity matrix (see [9]). The complexity of computing the formula given in (4) can be bounded by $O(n^2 + n^{2.376})$. Using the Coppersmith-Winograd algorithm [4], a matrix inversion has the complexity $O(n^{2.376})$.

Moreover, the variance of a random variable is defined as

$$\text{Var}(J) = E(J^2) - [E(J)]^2, \tag{5}$$

where $E(J^2)$ in our case, is computed by the second moment of a discrete phase type random variable (see [9]) given by

$$E(J^2) = 2\boldsymbol{\pi}(\boldsymbol{I} - \boldsymbol{T})^{-2}\mathbf{1}.$$

As we can see in this analysis, phase type distributions take the role as computational vehicle in solving optimization problems.

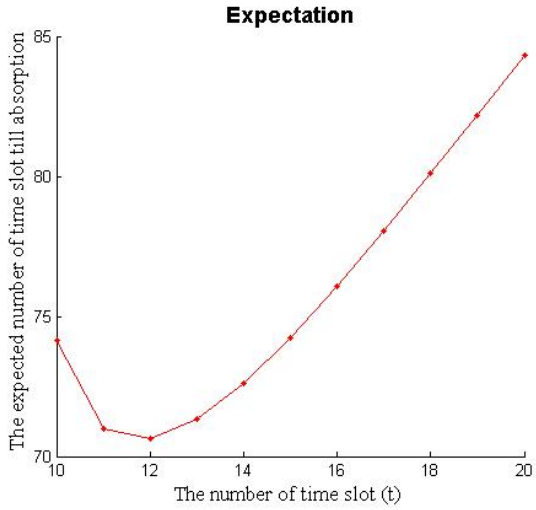

Figure 4. Choosing the optimal number of time slots for LMAC considering $n = 10, r = 2$

Figure 4 illustrates one optimization problem regarding the number of time slots. Here, the network contains 10 sensors and the maximum back off time is 2. In this case, 12 time slots provide the best possible stabilization rate, as it has the lowest expected stabilization time.

Figure 4 describes how to pick the optimal number of time slots given the number of sensors and the maximum back off time. By having this way of finding the optimal number of time slots, figure 5 depicts the growing trend by pairing the number of sensors and optimized number of time slots. Here, the maximum back off time is also 2. The result can be divided in linear segments, where the gap between the sensor

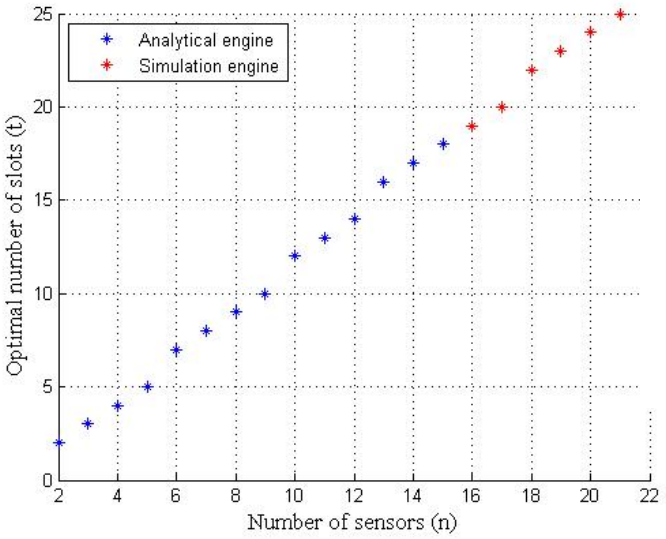

Figure 5. Matching the number of sensors with the optimal number of time slots considering $r = 2$

number and the slot number increases in larger networks (i.e. when the network has more sensors), because the probability of collision increases when more sensors are in the network. Therefore, figure 5 provides a guide for network designers to decide how to match the sensor number with the slot number in an optimal setting. For instance, 17 sensors should match 20 time slots to be optimal with maximum back off time being 2.

On the way of computing the plot, the state space is growing by having an increasing number of sensors. To deal with higher order models, we switched the computational engine from analytical to simulation at the point where we had around 1000 states. In figure 5, we distinguish the two engines by colours. Note, here we only intend to show how we used the simulation engine to assist the analytical engine in deriving solutions. Therefore, the point of switching would depend on your local computing power.

Furthermore, our abstracted model is able to analyze modification issues. In a predefined LMAC network, the number of sensors, the number of time slots, and the maximum back off are identified. For instance, sometimes it is necessary to add or remove a certain number of sensors in the current network. A question that may arise is how to adjust the parameters to keep the stabilization at an optimal speed. Figure 6 offers some recommendations regarding these issues.

It is clear that more time should be expected if there are more sensors in the network. However, the marginal cost varies due to the fluctuation of the probability of collision. Figure 6 has been plotted given 20 time slots with maximum back off time being 2. The vertical axis describes the incremental cost for plugging in one extra sensor. Because of

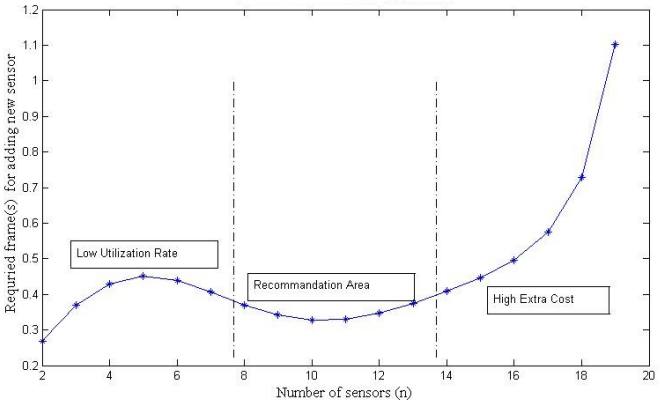

Figure 6. Modification price in the LMAC considering $t = 20, r = 2$

low utilization rate of the network, i.e. very low probability for collision, the price for incrementing sensors is higher at the beginning. For more than 14 sensors in the network, the cost rises dramatically. Therefore, the recommended number of sensors for the current configuration should remain in the middle region. If the required number of sensors has a high additional cost, it would be better to increase the number of time slots or the maximum back off time in order to keep efficiency. In contrast, the number of time slots or the maximum back off time should be reduced in order to raise the network utilization rate.

Evidently, there are many other questions about optimization that can be solved. For instance, to find the optimal number of sensors having a fixed maximum back off time and a fixed number of time slots.

VI. State Space Explosion

In some experiments, state space explosion emerges considering above 10,000 states even though the sparseness of the transition matrix of the underlying DTMC increases as well. With the MATLAB sparse matrix representation, the number of states in our model, given by $\binom{n+r+1}{n}$, crashed considering 10660 states (38 sensors and 2 maximum back off) on 2GB RAM memory. For maximum 3 and 4 frames waiting time, we are up to 19 and 15 sensors, respectively. Note that since we work on a relatively old computing platform, we expect our approach can handle a larger number of states than our experimental data.

Even though the current result is limited due to the state explosion problem, a pioneer study on different maximum waiting has been depicted in figure 7. In figure 7, the number of sensors and time slots is the same in all the sample points. It clarifies that maximum 2 frames waiting is the most favourable choice comparing with the cases of 3 frames and 4 frames given a number of sensors up to 10. It supposes that the favorable choice will switch to 4 frames after reaching a certain point because of the reduction of the probability

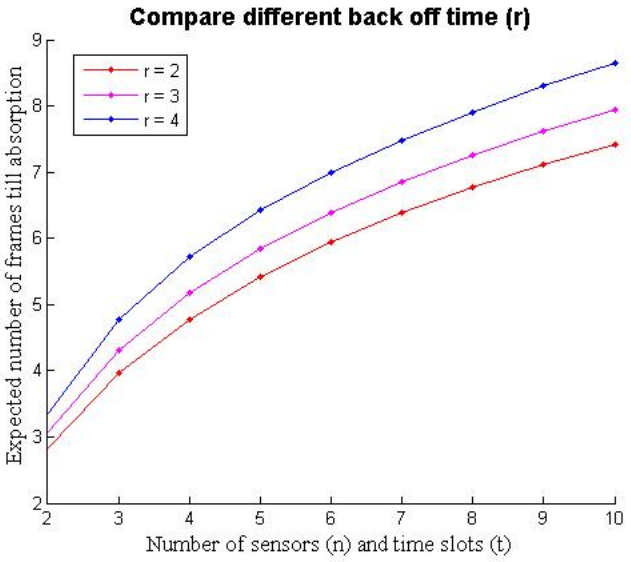

Figure 7. Measure different maximum back off time

of collision. Part of our future work will be to extend the current plot, i.e. to consider more sensors and time slots.

VII. LMAC ANALYZER

Based on the results described in the previous sections, we have created a tool, called LMAC analyzer, which is a prototype on the MATLAB platform. Figure 8 is a screen shot of the running prototype. Users are required to input the desired LMAC parameters (the number of sensors, the number of time slots, and the maximum back off time), and the tool will automatically output a probabilistic model, in the form of a probability transition matrix of a DTMC. The expected time to stabilization is also calculated. It is our intention that the next version of the tool will support plotting in order to solve different optimization problems, such as those mentioned in section V.

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

In this paper, we have reported a probabilistic model for the analysis of a medium access protocol LMAC for WSN. The model abstracts the concurrent behaviour of the setup process in LMAC, where individual sensors are allocated to time slots, preferably in an efficient way to minimize the price. There are two sources of randomness in this concurrent stochastic process. First, when the nodes randomly pick time slots, and second when the collided nodes randomly pick back off time. Considering the worst case scenario, we resolved the non-determinism, which comes from the underspecification of the protocol by probabilistic choices. The probabilities are calculated using combinatorial theory with a uniformity assumption. Moreover, our simulation engine verifies the analytical model, and it can optionally

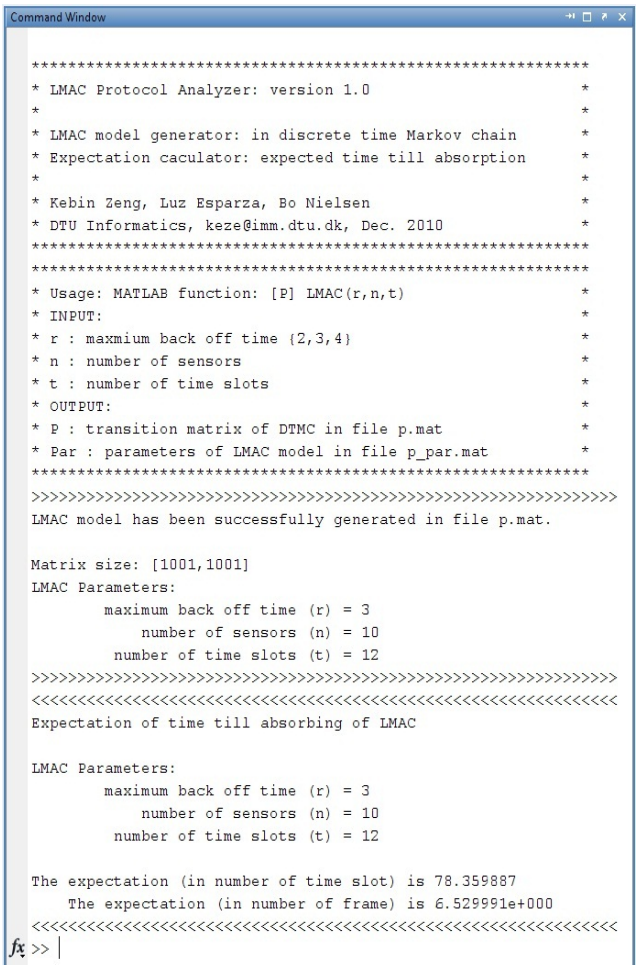

Figure 8. Screen shot of the LMAC Analyzer

compute solutions whenever numerical challenges appear for the analytical model. After obtaining the model, we have analyzed the performance of the process and we have calculated optimal parameters for network configuration.

Contrary to previous work ([7], [14], [17]), our contribution is an alternative approach of system abstraction, by which parallel composition can be avoided. The inherent advantage is that no detailed model is required in order to do the abstraction. Thereby, it is possible to handle larger systems directly. Note that our model has a moderate state space $\binom{n+r+1}{n}$, which depends only on the number of sensors n and the maximum back off time r.

B. Future Work

Our model is based on a uniformity assumption for the back off time selection, as one of the strategies that was proposed in [16] in order to ensure a low latency for the most common data traffic in WSN. It is interesting to see how all these strategies will perform using our abstracted model. One straightforward way is to apply different weights

on each back off option, such as the approach given in [17]. In addition, different back off distributions might lead to the same average waiting time, which could be considered as an optional way of substituting the maximum waiting time r. For example, maximum 2 frames waiting with probabilities $\frac{1}{4}$ and $\frac{3}{4}$ respectively, has the same average waiting time as maximum 3 frames waiting with probabilities $\frac{2}{4}$, $\frac{1}{4}$, and $\frac{1}{4}$, respectively. Thereby, 2 frames maximum waiting (r = 2) could be used to represent 3 frames maximum waiting (r = 3). If we could prove the mean of back off waiting is a crucial factor in the stabilization process, we could build our model with a smaller maximum back off time, which will directly reduce the huge number of states.

One of the shortcomings of the abstracted model is that it is less expressive. So far we have found difficulties expressing the model under other topologies. Handling arbitrary topologies will produce a more general method, which will be able to attack problems regarding expressiveness. However, most likely it comes with costs in form of increased state space.

Abstraction techniques are used to strip away information that is not relevant for the verification at hand, which lead to a simplification of verification models. Inspired by the work given in [2] and [3], we have started to look for the connections between the mathematical approach and the compositional approach for the abstracted LMAC protocol. We have worked on some small cases that can be handled by the Probabilistic Symbolic Model Checker PRISM [8]. For instance, when the network has 3 sensors, 3 time slots, and the maximum waiting is 3 frames. With some stochastic interpretations of the behaviour from the system of parallel composition, PRISM is able to generate the same abstracted model. We have got promising results where the DTMC obtained from our model is exactly the same as the one from PRISM. One direction for our future work will be to study and formalize the relationship between these two approaches.

At last, we aim to initialize a prominent treatment of generalizing our approach. Contrary to the direction of abstraction, the model could be concretized in order to cooperate with the inherent advantage of having small state space from our direct method. The main object of concretization is to enhance the expressiveness of the model. For example, try to keep the non-determinism and specify our model in terms of others topologies.

APPENDIX

The number of ordered arrangements of n objects, in which there are k_1 objects of type 1, k_2 objects of type 2, ..., and k_m objects of type m, such that $k_1+k_2+\cdots+k_m = n$, is given by

$$\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1!k_2!\cdots k_m!}.$$

This number is called the *multinomial coefficient*. For instance, suppose that we have 16 sensors and we want to put them into 6 slots, with 2, 2, 2, 3, 3, and 4 sensors respectively, then

$$\binom{16}{2,2,2,3,3,4} = \frac{16!}{2!2!2!3!3!4!},$$

is the number of ordered ways of doing so. If we do not distinguish the sensors, we have to eliminate the ordering, i.e., we will have that

$$\frac{\binom{16}{2,2,2,3,3,4}}{3!2!1!} = \frac{16!}{2!2!2!3!3!4! \times 3!2!1!}$$

is the number of non-ordered arrangements.

In order to use these arguments and put them into our case, we define the following function

$$A(\mathbf{v}, g, j) = \frac{1}{\prod_{s=1}^{x}(\mathbf{v}'[s])!},$$

where $\mathbf{v}$ is an input vector with dimension x, $\mathbf{v}' = \mathbf{v} + \mathbf{e}_j + \mathbf{e}_{g-j}$, where $\mathbf{e}_j$ is a vector with 1 in the j-th entry, and 0 otherwise. This function corresponds to the term $\frac{1}{3!2!1!}$ in the previous example, and it is used to eliminate the ordering of any arrangement.

Having the definition of A, we will give the general form of $p_i(x)$, for $i \geq 1$.

When the collision only happens in a single slot, we have that

$$p_1(x) = \frac{1}{x!}.$$

According to the interpretation of the multinomial coefficient, we have only one slot within all the x sensors.

Now, define the function

$$q_2(g, \mathbf{v}, s) = \sum_{j=s}^{\lfloor \frac{g}{2} \rfloor} \frac{1}{j!(g-j)!} A(\mathbf{v}, g, j), \quad s \geq 2,$$

which indicates that collisions can happen in two slots. The parameter g stands for the number of collided sensors. Note that in order to eliminate the ordering we call the function A. Within the summation, j is the number of sensors in the first slot, and $g-j$ is the number of sensors in the second slot. Thus, we get that $p_2(x)$ is a particular case of q_2 given by

$$p_2(x) = q_2(x, \mathbf{0}, 2).$$

In general, the function $q_i(g, \mathbf{v}, s)$ for $i \geq 3$, indicates that there are i slots with collisions. It is calculated using recursive calls as follows

$$q_i(g, \mathbf{v}, s) \quad = \quad \sum_{j=s}^{\lfloor \frac{g}{i} \rfloor} \frac{1}{j!} q_{i-1}(g-j, \mathbf{v} + \mathbf{e}_j, j), \quad s \geq 2.$$

We compute this function by putting j sensors into the first slot (corresponding to the term $\frac{1}{j!}$), and putting the others

$(g-j)$ sensors into $(i-1)$ slots, which we can get it from the previous q_{i-1}.

Thus, the general form of $p_i(x)$ for $i \geq 3$, is given by

$$p_i(x) = \sum_{j=2}^{\lfloor \frac{x}{i} \rfloor} \frac{1}{j!} q_{i-1}(x-j, \mathbf{e}_j, j).$$

ACKNOWLEDGMENT

The work is conducted in the VKR centre of Excellence - MTLAB. We would like to thank Villum Foundation and Velux Foundation for financial support. And we would like to thank Michael James Andrew Smith and Flemming Nielson for the helpful discussion with their work on compositional model of the LMAC protocol in PRISM.

REFERENCES

[1] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on uppaal. *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT 2004)*, pages 200–236, 2004.

[2] Ed Brinksma. Verification is experimentation. *International Journal on Software Tools for Technology Transfer (STTT)*, pages 107–111, 2001.

[3] Edmund M. Clarke, Orna Grumberg, and David E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16, 1994.

[4] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, pages 251–280, 1990.

[5] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of program by construction or approximation of fixpoints. In *the 4th Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, New York, 1977. ACM.

[6] Alexandre David, Arild Haugstad, and Kim G. Larsen. Uppaal pro: Uppaal for probabilistic timed automata. http://www.cs.aau.dk/~arild/uppaal-probabilistic.

[7] Ansgar Fehnker, Lodewijk van Hoesel, and Angelika Mader. Modelling and verification of the lmac protocol for wireless sensor networks. In *Integrated Formal Methods (IFM 2007)*, volume 4591 of *LNCS*, pages 253–272. Springer, 2007.

[8] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

[9] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA, SIAM, 1999.

[10] Thomas Moscibroda and Roger Wattenhofer. Coloring unstructured radio networks. *Distributed Computing*, 21(4):271–284, 2008.

[11] M.F. Neuts. *Probability distributions of phase type*. University of Louvain, 1975.

[12] M.F. Neuts. *Matrix geometric solutions in stochastic models*. Jhons Hopkins University Press, 1981.

[13] T.Nieberg, S.Dulman, P.Havinga, L. van Hoesel, and J. Wu. *Collaborative Algorithms for Communication in Wireless Sensor Networks*. Kluwer Academic Publishers, 2003.

[14] L.F.W. van Hoesel. *Sensors on speaking terms: schedule-based medium access control protocols for wireless sensor networks*. PhD thesis, University of Twente, 2007.

[15] L.F.W. van Hoesel and P.J.M. Havinga. A lightweight medium access protocol (lmac) for wireless sensor networks: Reducing preamble transmissions and tranceiver state switches. In *1st International Workshop on Networked Sensing Systems (INSS 2004)*, pages 205–208. Society of Instrument and Control Engineers(SICE), 2004.

[16] L.F.W. van Hoesel and P.J.M. Havinga. Design aspects of an energy-efficient, lightweight medium access control protocol for wireless sensor networks. Technical report, Centre for Telematics and Information Technology, University of Twente, Enschede, 2006.

[17] M.S. Vighio and A.P. Ravn. Analysis of collisions in wireless sensor networks. In *21st Nordic Workshop on Programming Theory (NWPT 2009)*, Lyngby, Denmark, 2009.