Petri Nets as an Analysis Tool For Data Flow in Wireless Sensor Networks

Bruno Lacerda and Pedro U. Lima

Abstract— **We present a method to analyse the behaviour of wireless sensor networks (WSN) with a multihop architecture, where the sensor nodes are conveying gathered information into a sink node. In particular, we are interested in analysing the impact of data aggregation in intermediate nodes, comparing the increase of the time of arrival of messages to the sink node with the decrease of total messages sent by the whole network. One can use this approach to perform a pre-deployment optimization of the network parameters, so that its lifetime is increased while keeping the messages delay within a certain threshold.**

I. Introduction

Wireless sensor networks (WSN) are networks consisting of spatially distributed autonomous devices that use sensors and actuators to perform tasks. WSN are used in many areas including, for example, environment monitoring and home automation. With the increasing development of these kind of systems, in order to optimize its performance, the need for a systematic approach to its analysis and design is arising. Petri nets are proven to be well suited to model systems with distributed, concurrent and asynchronous features. In particular, generalized stochastic Petri nets (GSPN) [10] allow us to obtain performance measures such as probability of success of the modelled task and robustness of the system to component failures.

The GSPN framework have been used successfully to model robotic tasks, as seen in [5] and [2]. Petri nets have also been used in the scope of WSN, for example, in [9], where a model for the energy consumption of processors for WSN is presented and in [4], where an extension of Petri nets is used to build a graphical simulation system that enables the analysis of data propagation in a WSN where nodes receive data and immediately broadcast it to all the nodes within communication distance. Other analysis tools for WSN have been proposed. For example, in [7], Real-time Maude, a language supporting the formal specification and analysis of real-time and hybrid systems, is used to model and analyse WSN algorithms, particularly the *optimal geographical density control algorithm* (OGDC) [11], an algorithm that tries to maintain complete sensing coverage and connectivity of an area for as long as possible by switching nodes on and off, and in [1] a Markov chain based model is introduced to study WSN performance in terms of capacity, data delivery delay and energy consumptions as its sensor dynamics in sleep/active mode change. The impact of data aggregation in WSN, specifically how to balance communication costs and data delivery delay has also been studied, for example in [3].

We propose the use of GSPN as a tool to analyse several relevant properties for a WSN with a multihop architecture, where the sensor nodes are conveying gathered information to the sink node. The idea is, given a graph representing the network's topology and GSPN models of the nodes and the communication channels, to automatically build a GSPN that encompasses the GSPN node models plus the communication between them and proceed with the analysis of this GSPN, checking the impact of changing some of its parameters in the overall network performance. In particular, we will analyse the impact of data aggregation on network latency and usage of the communication channels. This analysis allows us, for example, to increase battery life by decreasing the number of messages sent while keeping the network latency below a certain threshold. To calculate this impact, we perform simulations using the TimeNET tool [12], a graphical tool that allows modelling, analysis and simulation of GSPN.

In Section II, we give a brief introduction to GSPN, followed by an explanation of how to use it to model WSN in Section III. In Section IV, we present an example of application and some results and, in Section V, we provide some conclusions and possible improvements to the method.

B. Lacerda and P. Lima are with the Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, Portugal {blacerda, pal}@isr.ist.utl.pt. The work of B. Lacerda was supported by the portuguese Fundação para a Ciência e Tecnologia through grant SFRH/BD/45046/2008.

II. Generalized stochastic Petri nets

In order to provide a clearer explanation of the formalism's basic rules, we start by defining place/transition nets (P/T nets) [6], the ordinary kind of Petri nets, which GSPN extend.

Definition 1 (P/T Net): A P/T net is a tuple $N = \langle P, T, W, M_0 \rangle$ where:

- P is a finite, not empty, set of places;
- T is a finite set of transitions;
- $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is the arc weight function;
- $M_0 : P \to \mathbb{N}$ is the initial marking.

A marking M can be seen as a vector of size $|P|$ that represents a state of the system, with $M(p) = q$ meaning that in M there are q tokens in place p.

Definition 2 (Preset and Postset of a Transition): Let $N = \langle P, T, W, M_0 \rangle$ be a P/T net structure and $t \in T$. The size $|P|$ vectors $t^\bullet$ and ${}^\bullet t$ are defined as:

$$t^\bullet(p) = W(t,p)$$

$${}^\bullet t(p) = W(p,t)$$

We refer to $t^\bullet$ as the postset of t and ${}^\bullet t$ as the preset of t.

The dynamics of a P/T net are defined by the firing rule, which determines the flow of tokens between places.

Definition 3 (Firing Rule): Let $N = \langle P, T, W, M_0 \rangle$, $t \in T$ and $M : P \to \mathbb{N}$. Transition t is said to be enabled in M if for all $p \in P$, ${}^\bullet t(p) \leq M(p)$. A transition t enabled in a marking M can fire, resulting in the marking $M' = M - {}^\bullet t + t^\bullet$. This is denoted $M \xrightarrow{t} M'$.

Using the firing rule, one can define firing sequences and the set of reachable markings of a given P/T net.

Definition 4 (Firing Sequence): A firing sequence of $N = \langle P, T, W, M_0 \rangle$ from a given marking M is a sequence of transitions $\tau = t_1 t_2 ... t_n \in T^*$ such that there exists markings $M_1, ..., M_n$ such that:

$$M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{t_2} ... \xrightarrow{t_n} M_n$$

We also write $M \xrightarrow{\tau} M_n$ to denote that the firing sequence τ drives N from marking M to marking M_n.

Definition 5 (Reachable Markings): The set of all reachable markings by $N = \langle P, T, W, M_0 \rangle$ is denoted as:

$$R(N) = \{M : P \to \mathbb{N} \mid \text{exists } \tau \in T^* \text{ such that } M_0 \xrightarrow{\tau} M\}$$

GSPN are an extension of P/T nets, where there are two different classes of transitions: immediate transitions and timed transitions. Once enabled, an immediate transition fires in 0 time while timed transitions fire after an exponentially distributed time. Thus, a timed transition will only fire when none of the immediate transitions is enabled.

Definition 6 (Generalized Stochastic Petri Net): A GSPN is an 6-tuple $\langle P, T, W, M_0, D, S \rangle$, where:

- $P = \{p_1, p_2, ..., p_n\}$ is a finite, not empty, set of places;
- $T = T_I \cup T_E = \{t_{I1}, t_{I2}, ..., t_{Im}\} \cup \{t_{E1}, t_{E2}, ..., t_{En}\}$ where T_I is a finite set of immediate transitions and T_E is a finite set of exponential transitions, such that $T_I \cap T_E = \emptyset$;
- $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is the arc weight function;
- $M_0 : P \to \mathbb{N}$ is the is initial marking;
- $D : T_E \to \mathbb{R}^+$ associates each exponential transition t_{Ei} with a delay $\mu = D(T_{Ei})$[1];
- $S : T_I \to \mathbb{R}^+$ is a function that associates each immediate transition with a firing weight, thus forming the so called set of *random switches*[2];

When there are immediate transitions enabled, one of them fires immediately, with a probability given by the corresponding random switch. When there are only exponential transitions enabled, there is a "race" to decide which transition fires first. For example, assume that k exponential transitions $t_1, ..., t_k$ are enabled. From the properties of the exponential distribution, we know that the expected time transition t_i, $i = 1, ..., k$ takes to fire is $D(t_i)$. Hence, the probability that transition t_i is the one to fire next is given by $\frac{\frac{1}{D(t_i)}}{\sum_{j=1}^{k} \frac{1}{D(t_j)}}$.

The GSPN marking is a semi-Markov process with a discrete state space given by the reachability graph of the net for an initial marking [10]. A Markov chain can be obtained from the marking process, and the transition probability matrix computed by using the firing rates of the exponential timed transitions and the probabilities associated with the random switches. Given that the marking of the GSPN is equivalent to a Markov chain, it is possible to use the tools already available to analyse Markov chains directly with the GSPN. Unfortunately, this methods require *bounded* GSPN, i.e., nets where, for all reachable markings, the maximum amount of tokens in each place is bounded

[1]The delay of an exponential distribution is given by $\mu = \frac{1}{\lambda}$, where $\lambda > 0$ is the *rate* parameter of the distribution.

[2]Random switches associate probability distributions to subsets of conflicting immediate transitions. So, for example, if immediate transitions t_i and t_j are the only immediate transitions enabled, the probability of firing transition t_i is given by $\frac{S(t_i)}{S(t_i)+S(t_j)}$.

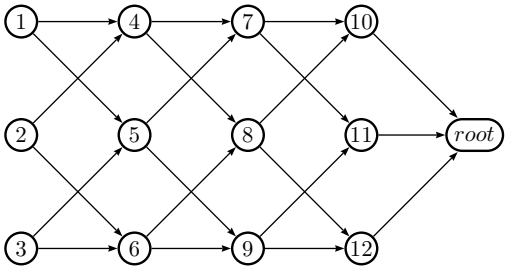

Fig. 1. Example of a network topology

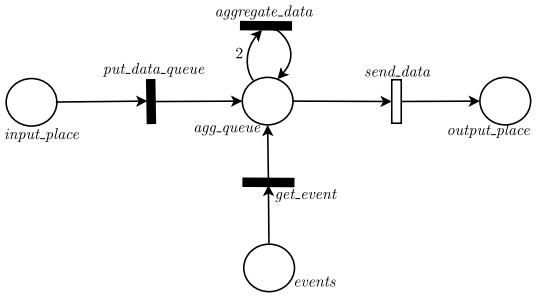

Fig. 2. A simple GSPN model for a node with data aggregation capabilities

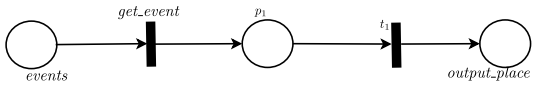

Fig. 3. Model for a leaf node, with no aggregation capabilities

by an integer, which is not the case of our models. Hence, our results are obtained by performing Monte Carlo simulations on the steady-sate of the GSPN model for the WSN, a possibility provided by the TimeNET tool.

III. Modelling the WSN

In this Section we discuss how to build the complete WSN models from the models of its nodes and the channel communication, according to a given topology of the network. We assume that the network is monitoring a given environmental conditional (e.g., temperature). Each node forwards the retrieved information to the nodes it is connected with. The topology is given as a directed graph, where each arc represents a channel connection between the source node and the target node. An example of a topology is given in Figure 1.

Each node is modelled by a Petri net, that describes the ability of the node to read data from the environment and the processing executed within it. In this work, we only model the processing related with data aggregation. We assume *total* data aggregation, where all data packets are assumed to have the same size and the aggregation of two packets in a node yields a single packet, i.e., the result of the aggregation of two different sets of data has the same size as one of the original sets of data to be aggregated. In Figure 2, a simple model for a node that performs data aggregation is depicted.

We provide a brief explanation of the meaning of the model components. The place *input_place* represents incoming data received from other nodes. This data is immediately placed in place *agg_queue*, where it waits for more data to be aggregated with. The aggregation is represented by the transition *aggregate_data* that simply retrieves two tokens from the *agg_queue* place and then places one token there. We model the aggregation function as an immediate transition because the time it takes to execute it is negligible. After a given amount of time, defined by the delay of the exponential transition *send_data*, the data goes to place *output_place*, to be sent to the target nodes in the topology.

The models for the leaf nodes - nodes that only send messages - and the sink node are much simpler. For the leaf nodes, depicted in Figure 3, we only model the reading of data from the environment and the sending of that data, due to the fact that we are only modelling the flow of data from the WSN to the sink node, the model of the sink node is simply one place that models the receiving of the incoming data.

One should notice that the models contain a place called *events*. We assume that the WSN is monitoring some kind of data, and all the nodes read from the environment at the same time, with a given rate. Hence, when constructing the complete WSN model, we add a transition with a given delay that puts one token in each of the *events* place of each node model, so that all the nodes read data from the environment at the same time. This data immediately goes to place *agg_queue*, to be aggregated with other data - if more data is already waiting or arrives while the node is waiting to aggregate - and is eventually sent towards the sink node.

The sending of messages is also modelled by a GSPN representing the channels, as seen in Figure 4. The place *input_place* represents messages to be sent. If the channel is not busy, the message starts being sent immediately, taking a time defined by the delay of the exponential transition *send_complete*. While a message is being sent, the channel is busy and other messages to be sent wait in place *input_place*. This model does not take into account possible failures in the sending

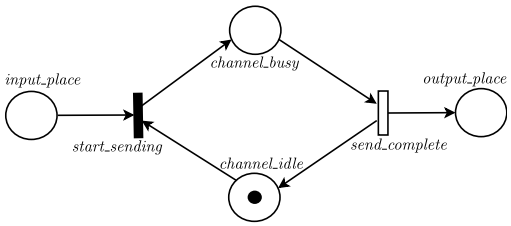

Fig. 4. Petri net model for the communication channels

of messages, but one could add another exponential transition, representing communication timeouts, with *channel_busy* as input place and with a larger delay than the *send_complete* transition. The delay of this timeout transition would be related with the reliability of the channel.

The WSN model is the composition of the node models and the channel models. This composition is obtained by, for each channel between 2 nodes, merging the *output_place* of the sender node with the *input_place* of the channel and the *output_place* of the channel with the *input_place* of the receiver node. In Figure 5, we show an example of a sender node that sends messages to 2 receiver nodes.

Using this method to construct the model, one can calculate the expected value of the number of tokens in places *channel_idle* for each channel and the expected value of the number of tokens in the *agg_queue* place for each aggregating sensor node. This measures indicate, respectively, the average usage of the channels and the probability of a set of data being waiting for more data to arrive, in order for the aggregation to be performed.

IV. Application Example

We present a simple example of application of this method to a simple WSN. The WSN is composed of 16 sensor nodes, divided in four levels, and one sink node. Each node of each level broadcasts the data to all the nodes in the next level, e.g., node 1 in level 1 sends the data it reads from the environment to nodes 5, 6, 7 and 8 in level 2. The nodes of the last level send the data to the sink node. We assume that the nodes retrieve information from the environment following an exponential distribution of $delay = 1$. The sending of the messages also follows an exponential transition, with $delay = 0.05$, i.e., for each channel model, the parameter associated with the exponential transition *send_complete* is $D(send\_complete) = 0.05$. One could add different delays for different channels, taking into

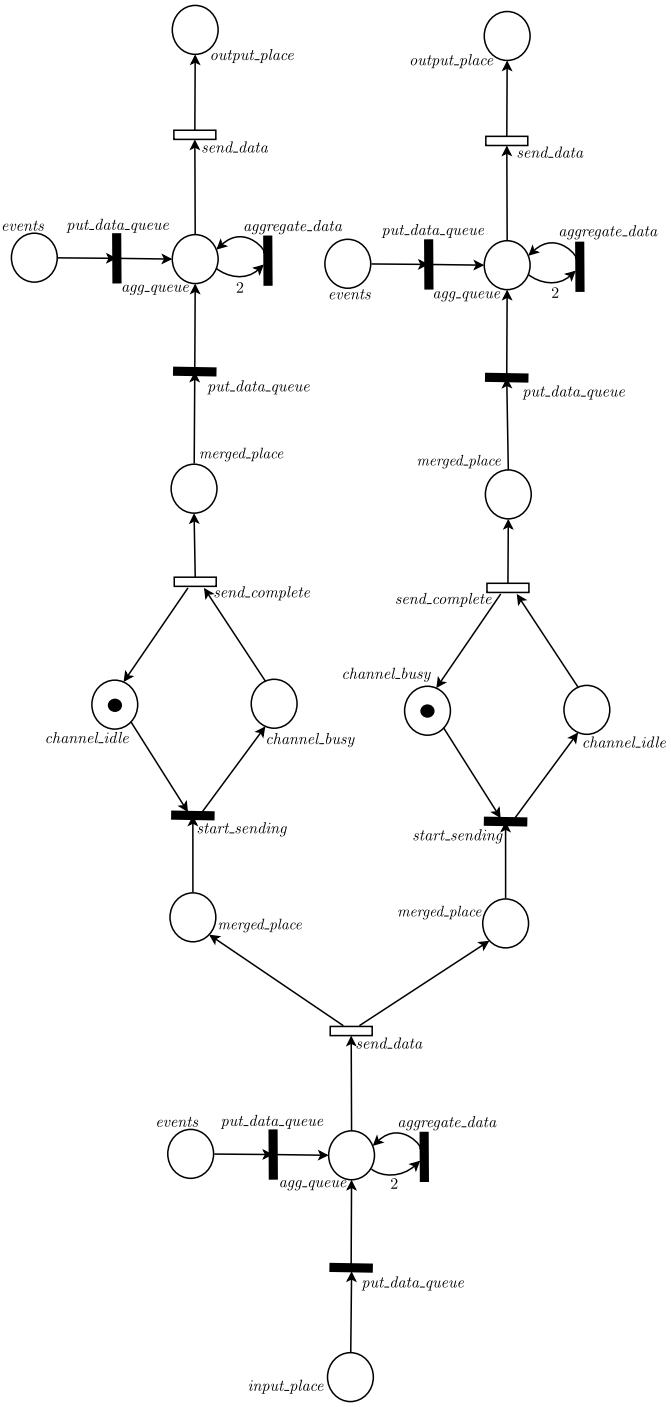

Fig. 5. Connection between one sender node and two receiver nodes

account the distance between nodes for example, but, to simplify the description of the example, we defined the same value for the delay of all *send_complete* transitions.

Our goal is to adjust the delay of the *send_data* transitions, so that we minimize the variables *agg* and *usg*, respectively the probability of having one token in the *agg_queue* places - data sets waiting for more data to be merged with - and the probability of having one token in the *channel_busy* places - usage of the communication channels. These two variables are conflicting, because if the data spends less time waiting to be aggregated, then more messages must be sent, thus increasing the usage of the communication channels, and vice-versa. Hence, one needs to define a function f of *agg* annd *usg* to be minimized. To simplify the example, we merely define f as a convex combination of both variables:

$$f(agg, usg) = \alpha agg + (1 - \alpha) usg$$

The α parameter can be adjusted to give more importance to decreasing the communication channel usage, or to avoid a big impact of the aggregation on the network latency. To simplify, we choose $\alpha = 0.5$. The function to be minimized can always be defined by the designer, taking into account the performance goals for the WSN.

We will perform the analysis by level, in an increasing order. This is done because the nodes in each level do not send data to nodes in lower levels, hence the value obtained for the delay of the *send_data* transitions on lower level nodes is not affected by changing the delay of the *send_data* transitions of the higher level nodes.

The first level of nodes does not receive data from other nodes, hence they are modelled as leaf nodes, as explained in the previous Section. Thus, for these nodes there is no need to perform any kind of evaluation.

For the second level of nodes, we arbitrarily fix the delay of the *send_data* transitions on higher levels and perform simulations varying the delay of the *send_data* transitions in this level from 0.01 to 0.1, with a step of 0.01. The results of the simulations are depicted in Figure 6.

One can see that, as expected, the value of *agg* increases with the increase of the time the node waits to aggregate data before sending, while the value of *usg* decreases. The minimum of the function $f(agg, usg)$ is attained when the delay is 0.03, hence, we fix the delay at level 2 to this value.

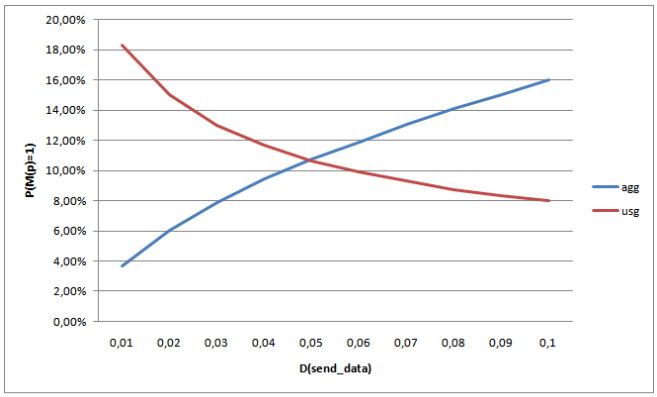

Fig. 6. Results for level 2

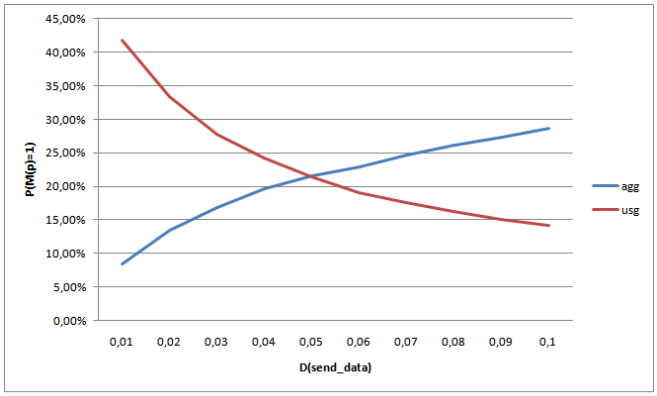

Fig. 7. Results for level 3

For the third level of nodes, similar simulations as the ones for level 2 were performed, fixing the delay of the second level nodes as 0.03 and arbitrarily fixing the delay of the nodes in the fourth level. The results can be seen in Figure 7.

The nodes in this level have larger values for both *agg* and *usg*, as a result of this level being closer to the sink. This is a usual property of WSN, since nodes near the sink typically have to forward larger amounts of data. The value of the delay that minimizes $f(agg, usg)$ is 0.06. The larger delay when compared with the nodes in level 2 is justified intuitively by the increase of messages received and sent by the nodes in this level.

Fixing the delay for nodes in level 2 as 0.03 and the delay for the nodes in level 3 as 0.06, simulations were run for the nodes in level 4. The result are depicted in Figure 8.

As expected, since this is the level closer to the sink, its nodes are the ones with higher values of both *agg*

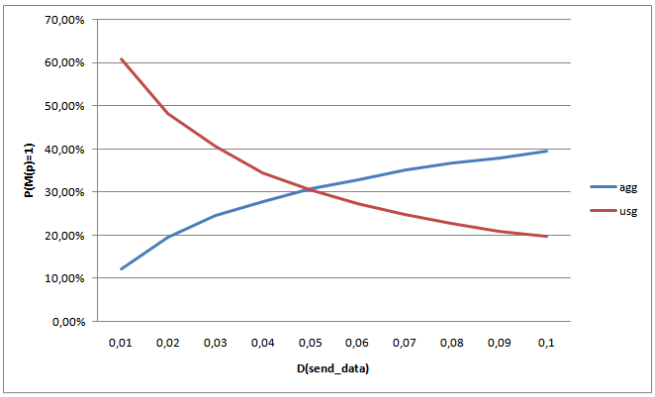

Fig. 8. Results for level 4

and *usg*. For this level, $f(agg, usg)$ is minimized when the delay is equal to 0.09.

V. Conclusions and Further Work

We presented a modelling framework for the data flow in WSN that allows the study of several network properties. The network sensor nodes and data channels are modelled as GSPN, a formalism that provides several analysis and simulation methods for an array of quantitative measures that have a physical interpretation in the WSN scope. We provided an application example where the channel usage and data waiting to be aggregated in each node was calculated and a function of these two metrics was minimized in order to optimize the network efficiency.

There is a number of improvements to the work presented. Firstly, improving the models in order to achieve a more accurate representation of the real network and to allow the possibility of calculating other relevant metrics such as the average time between the arrival of new data and its delivery to the sink node. One can also easily add transition failures to the channel models in order to analyse the network's robustness to communication failures. Another idea is to use this method to compare different topologies for the same network, regarding a set of relevant metrics calculated from the GSPN models, such as the ones presented in this work. Finally, one should validate the GSPN models using a suitable real WSN, for example the vibration monitoring application described in [8].

References

[1] Carla-Fabiana Chiasserini and Michele Garetto. An analytical model for wireless sensor networks with sleeping nodes. *IEEE Transactions on Mobile Computing*, 5:1706–1718, 2006.

[2] Hugo Costelha and Pedro Lima. Modelling, analysis and execution of multi-robot tasks using Petri nets. In *AAMAS '08: Proceedings 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1449–1454, Estoril, Portugal, 2008.

[3] Peter Korteweg, Alberto Marchetti-Spaccamela, Leen Stougie, and Andrea Vitaletti. Data aggregation in sensor networks: Balancing communication and delay costs. In Giuseppe Prencipe and Shmuel Zaks, editors, *Structural Information and Communication Complexity*, volume 4474 of *Lecture Notes in Computer Science*, pages 139–150. Springer Berlin / Heidelberg, 2007.

[4] Yan Luo and Jeffrey J. P. Tsai. A graphical simulation system for modeling and analysis of sensor networks. In *ISM '05: Proceedings of the 7th IEEE International Symposium on Multimedia*, pages 474–482, Irvine, CA, USA, 2005.

[5] Dejan Milutinovic and Pedro Lima. Petri net models of robotic tasks. In *ICRA '02: Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 4059–4064, Washington, D.C., 2002.

[6] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[7] P.C. Ölveczky and S. Thorvaldsen. Formal modelling and analysis of wireless sensor network algorithms in real-time maude. In *IPDPS '06: 20th International Parallel and Distributed Processing Symposium*, Rhodes, Greece, 2006.

[8] Luis D. Pedrosa, Pedro Melo, Rui M. Rocha, and Rui Neves. A flexible approach to WSN development and deployment. *International Journal of Sensor Networks*, 6(3/4):199–211, 2009.

[9] A. Shareef and Yifeng Zhu. Energy modeling of processors in wireless sensor networks based on Petri nets. In *ICPP-W '08: Proceedings of the 2008 International Conference on Parallel Processing - Workshops*, pages 129–134, Portland, Oregon, USA, 2008.

[10] N. Viswanadham and Y. Narahari. *Performance modeling of automated manufacturing systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[11] H. Zhang and J. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc & Sensor Wireless Networks*, 1(1-2), 2005.

[12] Armin Zimmermann and Michael Knoke. TimeNET 4.0: A software tool for the performability evaluation with stochastic and colored Petri nets. *http://pdv.cs.tu-berlin.de/timenet/*, 2001.