Multiple abstraction levels in performance analysis of WSN monitoring systems

Marco Beccuti
Dipartimento di Informatica
Università di Torino
Corso Svizzera, 185 Torino, Italy
beccuti@di.unito.it

Daniele Codetta-Raiteri,
Giuliana Franceschinis
Dipartimento di Informatica
Università del Piemonte Orientale
Viale T. Michel 11, Alessandria, Italy
{raiteri, giuliana}@mfn.unipmn.it

ABSTRACT

In this paper, we illustrate the use of different methods to support the design of a Wireless Sensor Network (WSN), by using as a case study a monitoring system that must track a moving object within a given area. The goal of the study is to find a good trade off between the power consumption and the object tracking reliability. Power saving can be achieved by periodically powering off some of the nodes for a given time interval. Of course nodes can detect the moving object only when they are on, so that the power management strategy can affect the ability to accurately track the object movements. We propose two models and the corresponding analysis and simulation tools, that can be used in a synergistic way: the first model is based on the Markov Decision Well-formed Net (MDWN) formalism while the second one is based on the Stochastic Activity Network (SAN) formalism. The MDWN model is more abstract and is used to compute an optimal power management strategy by solving a Markov Decision Process (MDP); the SAN model is more detailed and is used to perform extensive simulation (using the Möbius tool) in order to analyze different performance indices, both when applying the power management policy derived from the first model and when using different policies.

Keywords

Wireless Sensor Network, optimization, simulation, Markov Decision Well-formed Net, Stochastic Activity Network.

1. INTRODUCTION

A Wireless Sensor Network (WSN) is composed by several sensor nodes, each with the ability of acquiring information from the surrounding environment, of processing them locally and of transferring them to a collecting node (also called sink) by means of a wireless connection. The nodes are usually powered by a battery and use a multi-hop approach to transfer their data to the sink node; their organization can be completely distributed but can also be coordinated by one or more powerful nodes that are not battery operated (e.g. the sink node).

The characteristics of WSNs facilitate the deployment of a monitoring system based on this technology, however these characteristics also impose severe limitations on energy resources, computational power and reliability of communication, and make the network subject to malfunctions and unavailability. The battery limited duration aspect has been tackled in the literature by proposing various sleeping policies: in order to conserve energy the sensors may be put into a sleep mode (only its sensor and/or its transmitter) with a timer that determines the sleep duration. Adaptive sleep policies have been studied in different contexts [2, 8, 9]: typical goals are to maximize the probability of successful communication of relevant data to the sink (reducing the duty cycle of nodes while reducing the probability of collisions during transmission) and to maximize the probability of sensing relevant signals on the monitored field.

In this paper we propose a methodology for supporting the designer in finding good sleeping policies by trading the battery consumption and the reliability of the WSN system, for the probability of detecting the relevant events on the monitored field. The proposed methodology combines two Petri Net based formalisms featuring different analysis capabilities that can be used in a synergistic way to reach the goal: Markov Decision Well-formed Nets (MDWN) [3, 4] and Stochastic Activity Networks (SAN) [11]. The MDWN model is more abstract and is used to compute an optimal power management strategy by solving a Markov Decision Process (MDP) [10]; the SAN model is more detailed and is used to perform extensive simulation (using the *Möbius* tool [7]) in order to analyze different performance indices, both when applying the power management policy derived from the former model and when using different policies. The results of the first model can be integrated in the second one by semi-automatically encoding the optimal policy derived in the former into a function that drives the behavior of the latter. One of the challenges to be faced while developing the MDWN model is related with the state space exponential growth problem; this can be mitigated by adopting different strategies: keeping an high abstraction level, exploiting symmetries in the behavior, applying model decomposition so that analysis can be separately performed on smaller submodels. The SAN model can integrate more

WSNPERF 2009, October 23, 2009 - Pisa, Italy.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

details and represent the overall system.

The paper is organized as follows: Sec. 2 presents the case study on which the proposed methodology is illustrated; Sec. 3 presents the MDWN model, and the corresponding optimization results; Sec. 4 presents the SAN model; Sec. 5 presents the results obtained with the models presented in the previous sections; Sec. 6 concludes the paper and proposes some directions for future work.

1.1 Preliminary definitions

The two formalisms used in this paper are rather different, but they have common roots, in fact they both derive from Stochastic Petri Nets (SPN) [1]. We introduce here some preliminary notions that can help the reader in the sections dedicated to MDWN and SAN.

Discrete event concurrent systems can be quite naturally modeled by Petri nets. This formalism features a discrete distributed state representation, expressed as the number of *tokens* contained into the model *places*; where the places are graphically represented as circles, and the tokens as black dots within circles. The possible activities in the modeled system are represented by transitions (which can be timed or be associated with a probability and possibly with a priority level in SPNs). Transitions are graphically represented as bars or rectangles, and are connected with (input or output) arcs to the places: such arcs represent both the conditions on the state for the transition activity to be enabled, and the state change that is caused in the model by the end of the activity, an event that is called "transition firing".

High Level Petri Nets (HLPN) [5] have extended the expressivity of the formalism by allowing to associate *colors* (i.e. information) with tokens, so that they can be distinguished. This feature allows a more compact model representation and the possibility to put in evidence regularity (symmetries) in the model structure and behavior that can be exploited when the state space of the model is explored.

In HLPNs annotations are associated with places, transitions and arcs to specify respectively the type of information that can be attached to tokens, the parameters that must be instantiated to obtain a transition instance from a parametric transition (with their allowed range), and the functions used to obtain the (multi)set of *colored tokens* that must be withdrawn from/added into a place connected to a given transition when one of its enabled instances fires.

In this paper two PN based formalisms will be considered: Markov Decision Well-formed Nets (MDWN), a class of HLPNs extended with the possibility of specifying sequences of events with a probabilistic characterization of choices, and sequences of events where the choices are non deterministic: the semantics of a MDWN is a Markov Decision Process, and can be used to set up and solve optimization problems. The second formalism is called Stochastic Activity Networks (SAN): it does not allow colored tokens but it allows to have extended places, which correspond to typed variables (besides integers, also floats, arrays or other data structures are allowed). Moreover SANs allow to express both the enabling conditions and the state change caused by a transition firing using C functions. Finally the model is expressed as a hierarchical composition of several submodels, joined on shared places. Transitions have associated delays that are random variables characterized by a given distribution function. When some restrictions are satisfied, the semantics of a SAN is a Continuous Time Markov Chain; when such restrictions are not satisfied, the analysis of SANs is performed by resorting to discrete event simulation techniques.

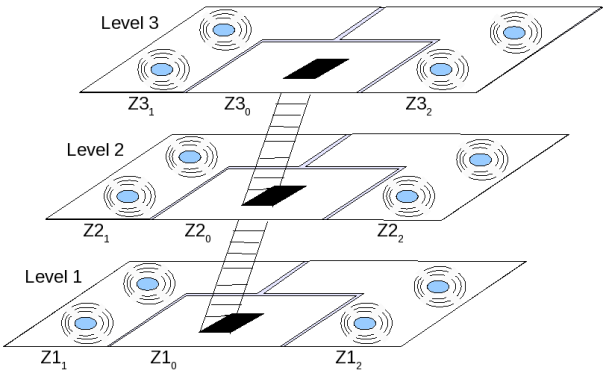

Figure 1: The area structure where the object can move.

2. THE CASE STUDY

In this section, the case study used to illustrate the methodology is presented: the monitoring system to be implemented as a WSN must track an object that can move inside an area structured on three levels as depicted in Fig. 1 according to probability distributions that are known a priori. Nevertheless it is easy to extend our work, so that these probability distributions are updated on-line. This extension will be discussed in Sec. 6.

On each level there are three zones: in one of them there are no sensors, while in each of the other zones there are J sensors. The location of the object is identified by two digits: the first one indicates the level (1, 2 or 3), the second one indicates the zone (0, 1 or 2). We assume that the zone containing no sensors is identified by 0 on any level. The object is initially located in $Z1_0$ (level 1, zone 0).

The object can move from one zone to any other on the same level. For instance, if the object is located in $Z1_1$, it can move to $Z1_0$ or to $Z1_2$. The object can move from a level to the adjacent one(s) only if it is located in zone 0. So, if the object is located in $Z1_0$, it can only move to $Z2_0$; if the object is located in $Z2_0$, it can move to $Z1_0$ or $Z3_0$; finally, if the object is located in $Z3_0$, it can only move to $Z2_0$. In the graph in Fig. 2, the possible locations of the object are represented in form of nodes, while the possible object movements correspond to the edges.

The object movements the are stochastic events. We assume that at each time unit, the object may stay in the current location or move to another one, according to a given probability distribution. Such distribution changes according to the current location of the object. On any level, the probability to move from the zone 0 to zone 1 or from zone 1 to zone 0 or from zone 2 to zone 1 is p, from zone 2 to zone 0 or from zone 1 to zone 2 is q, and from zone 0 to zone 2 is k. On any level, the probability to stay in zone 1 or 2 is equal to $1 - (p + q)$.

In zone 0, the probability to move to the upper level (if any) is u, while the probability to move to the lower level (if any) is d. The probability to stay in $Z1_0$, $Z2_0$ and $Z3_0$

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

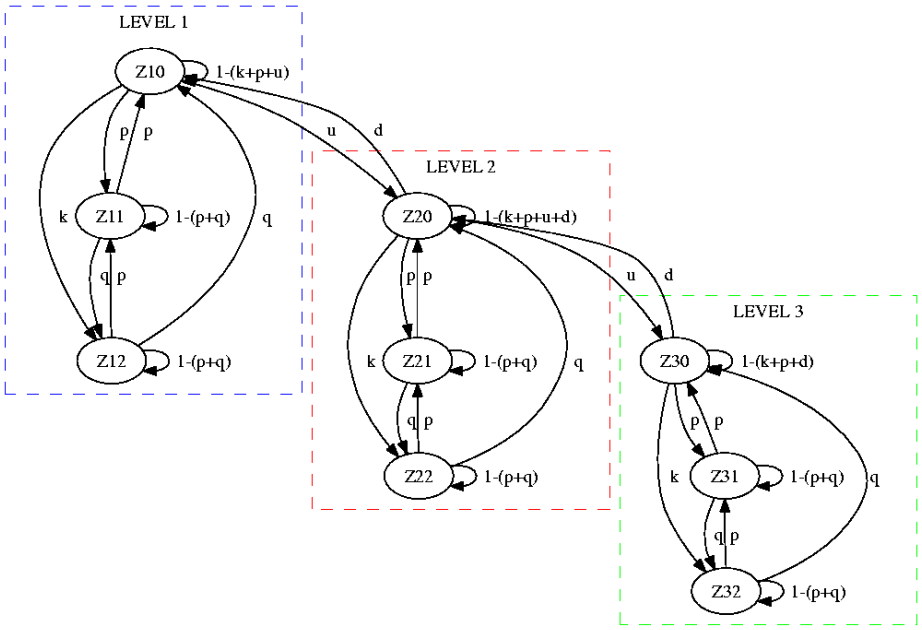

Figure 2: The possible locations and movements by the object.

is set to $1-(k+p+u)$, $1-(k+p+u+d)$ and $1-(k+p+d)$ respectively. In the graph in Fig. 2, the probability for each possible object movement is indicated close to the corresponding edge.

The role of the sensors is the detection of the current position of the object. The sensors may be active or dormant, and their state may change in time. This depends on the adopted power management policy: the aim is to reduce the power consumption. When a sensor turns dormant, it keeps such state for a certain period of time. During this period, the sensor cannot detect the presence of the object in the corresponding zone. At the end of the dormancy period, the sensor turns back to the active state. In our case study, we suppose that at each time unit, the current location of the object has to be detected by the sensors present in the zones 1 and 2 on any level. The probability to detect the object at each time unit depends on the number of sensors that are active in the zone where the object is currently located. We make the simplifying assumption that each sensor can communicate directly with the sink, so that each awake sensor can always send the object position to the sink when it detects it.

Obviously, if the object is located in a zone containing no sensors ($Z1_0$, $Z2_0$ or $Z3_0$), then it can not be detected. If instead the object is positioned in any of the other zones, where J sensors are present, then the probability changes according to the number of active sensors in the zone: the probability to detect the object if i sensors are active is r_i, with $r_0 = 0$.

We assume that the WSN fails to detect the object, when the object is positioned in a zone where sensors are present and the object is not detected by them. So, the goal of the case study is to compute the optimal power management strategy that optimizes the trade-off between the energy savings and the tracking errors. Moreover we are interested to evaluate the probability of tracking failure and the average number of sensors that are active at the same time for such optimal power management strategy.

Parameter	value
p	0.30
q	0.60
k	0.15
u	0.30
d	0.20
r_1	0.80
r_2	0.90

Table 1: The probability distribution of movement direction and the probability of detecting the object given n active sensors

The parameters instantiation.

In order to perform our experiment, we have assigned the following values to the model parameters: the sensor number J for each zone Zi_j, with $1 \leq i \leq 3$ and $1 \leq j \leq 2$, is 2; the central control can order to a sensor to sleep only for one or two time units; the probability distribution of movement direction and the probability of detecting the object given i active sensors are reported in Table 1.

3. POWER MANAGEMENT STRATEGIES

In order to compute the optimal power management strategy we propose a model based on the Markov Decision Well-

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

formed Net (MDWN) formalism. MDWN is a high level formalism introduced in [4] to specify Markov Decision Processes (MDP) [10]. Its definition is based on Well-Formed Nets [5], an high level Petri net formalism that combines a powerful modeling language with the possibility to implicitly express in the model the behavioral *symmetries*, which are reflected also at the level of the state space and can be automatically exploited to reduce the state space size, by means of an algorithm for the construction of the so-called Symbolic Reachability Graph (SRG) [5].

The main features of MDWNs are the possibility to specify the general behavior as a composition of several components, that may have similar behavior, and some of which are controllable; moreover each MDP non deterministic or probabilistic transition can be composed by a set of non deterministic or probabilistic steps, each one involving a subset of components. Moreover, MDWNs allow to exploit the symmetries (due to the presence of similarly behaving components) by deriving from the SRG a MDP of reduced size w.r.t. the original one, on which the same results can be computed more efficiently.

3.1 Basic notions on MDWNs

In details, an MDWN model is composed of two parts, both specified using the Well-formed Net (WN) formalism [5]: the WN^{nd} net and the WN^{pr} net (describing the non deterministic (ND) and the probabilistic (PR) behavior respectively); the two subnets share the color classes definition and the set of places, while transition sets are disjoint. A subset of the color classes may be used to identify (sets of similarly behaving) system components. In both subnets the transitions are partitioned into "run" and "stop" subsets, and each transition has an associated set of components involved in its firing (possibly specified in a parametric way on the transition color). "Run" transition firings represent intermediate steps in a ND/PR transition at the MDP level, while "Stop" transitions represent the final step in a ND/PR transition, for all components involved in it. Transitions in WN^{pr} have a "weight" attribute, used to compute the probability of each firing sequence. An MDWN model behavior alternates between ND transition sequences and PR transition sequences, initially starting from a ND state. The PR sequences are determined according to the WN^{pr} structure, and include exactly one stop transition for each component; the ND sequences are determined by the WN^{nd} structure, and include exactly one stop transition for each controllable component plus a stop "global" transition.

The generation of the (reduced) MDP corresponding to a given MDWN consists of (1) a composition step, merging the two sub-nets in a single net, (2) the generation of the (S)RG of the composed net, (3) two reduction steps transforming each PR and ND sequence of the (S)RG into a single (reduced) MDP transition.

3.2 The MDWN model

The MDWN model proposed in this paper is an approximation of the case study where only one level is modeled in details, while the zones and the sensors of the others levels are not considered. This approximation allows to considerably reduce the size of the final (reduced) MDP, and it has not much impact on the quality of the computed power management strategy as shown in Sec. 5.

Observe that since the power management strategies for each level are independent by the sensors' status in the other levels, then we can compute the optimal power management strategy considering only the status of the sensors in the considered level. On the contrary an approximation of the optimal power management strategy may be introduced when we do not consider the exact position of the object in the other levels (e.g. we only keep track of the level where the object is).

The corresponding WN^{pr} net is shown in Fig. 3 and it can be divided in three subnets: $S1, S2, S3$. The subnet $S1$ models the behavior of the object that we want to track. For instance, if we are interested to compute the optimal power management strategy for the first level, then the subnet $S1$ has to model the automaton in Fig. 4 obtained by the automaton in Fig 2, by replacing the zones $Z2_0, Z2_1, Z2_2$ and $Z3_0, Z3_1, Z3_2$ with two macro-zones $MZ2$, $MZ3$. Hence, the marking of the place *Position* represents the current object position, while the firing of the transitions *BackwardZ1* and *ForwardZ1* models the object movement inside the level 1 from Z_i to $Z_{(i+1 \bmod 3)}$ and from Z_i to $Z_{(i+2 \bmod 3)}$ with $0 \leq i \leq 2$. Instead, the firing of the transitions *MoveZ0toZM2* or *MoveZMitoZMj*, with $i, j \in \{2, 3\}$ and $i \neq j$, models the object movement among the levels. The firing of transition *Idle* represents the fact that the object may not move in the current time unit.

All the transitions of this subnet are stop transitions for the object and their weights depend on the level that we are considering. For instance, for the level 1 their weights are the values labeling the corresponding arcs of the automaton in Fig. 4. Observe that the transition *Idle* is marking depended, so that its weight depends on the marking of the place *Position*.

The subnet $S2$ models the sensors' behavior; the place *awake* contains the current sensors in the awake state, while the place *asleep* the ones in the asleep state. Instead, the marking of the place *SleepUnit* represents for each sleeping sensor the residual time that the sensor must still spend in the asleep state before waking up. The place *nextSleep* contains the pending sleep commands for the sensors. When an awake sensor receives a sleep command, it moves in the state asleep firing the transition *StartToSleep*. The firing of the transition *Wake-up* moves the sensor in its awake state. The inhibitor arc between the place *SleepUnit* and the transition *Wake-up* ensures that the sensor returns in its awake state iff its assigned sleeping time is consumed. Observe that all the transitions of this subnet are stop transitions for the sensors and they have higher priority than the transitions in the previous subnet.

The last subnet models the tracking error. A tracking error happens when the object is in $Z1_1$ or $Z1_2$ and no sensors are awake in the same zone (transition *NoFindZ*$_i$), or with probability 0.2 when only one sensor is awake in the same zone (transition *NoFindS1*), or with probability 0.1 when two sensors are awake in the same zone (transition *NoFindS1*). The transition *Clean* is used to clear the tracking error detected in the previous time unit. Such transition is a run transition for the global system and its priority is higher than all the other transitions. All the other transitions are stop transitions for the global system and have the lowest priority.

The WN^{nd} net is shown in Fig. 5 where all the places are shared with the WN^{pr} net; it models the central control decisions, so that the central control can decide whenever

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

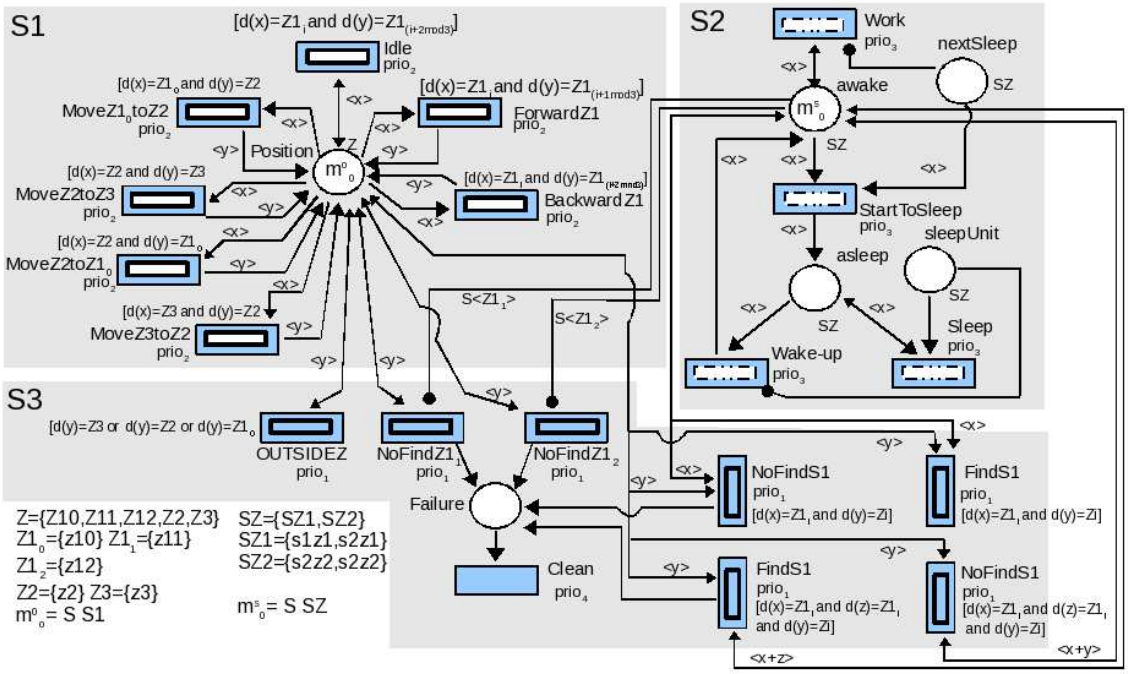

Figure 3: The WN^{pr} subnet of the MDWN model.

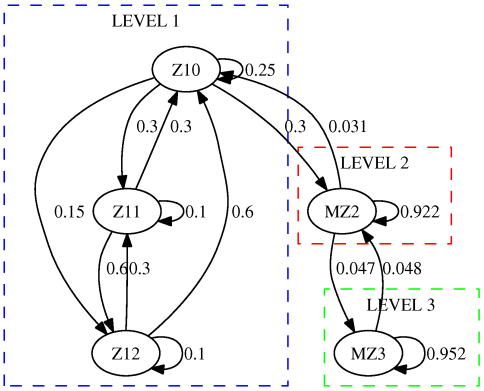

Figure 4: The automaton modeling the object behavior replacing the zones $Z2_0, Z2_1, Z2_2$ and $Z3_0, Z3_1, Z3_2$ with the two macro-zones $MZ2$ and $MZ3$.

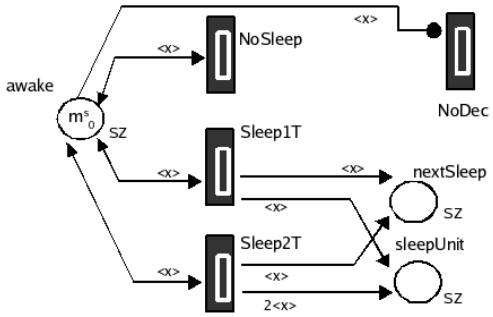

Figure 5: The WN^{nd} subnet of the MDWN model.

to send or not a sleep command to a sensor. Practically, the central control can take for each awake sensor one of the three following decisions: move into asleep state for one time unit (transition *Sleep1T*), or move into asleep state for two time units (transition *Sleep2T*), or stay in awake state for another time unit (transition *NoSleep*). The first two decisions require to explicitly transmit the command to the sensor. Instead only the transition *NoDec* can fire for a sensor which is in the asleep state (inhibitor arc between the place *awake* and *NoDec*). Observe that all the transitions are stop transitions for the sensors.

Finally we have to introduce a cost function for the model: we assume that there is a penalty at each time when the object is in zone Zi_j with $1 \leq i \leq 3$, $1 \leq j \leq 2$ and is not detected, and a power consumption cost for every awake

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

Level	Tracking failure probability (DTMC)	Average number of active sensors (DTMC)	Tracking failure probability (SAN)	Average number of active sensors (SAN)
level 1	0.045737	1.1260	0.046±0.005	1.0±0.03
level 2	0.062249	1.1696	0.065±0.006	1.2±0.03
level 3	0.107020	1.6120	0.099±0.008	1.4±0.03

Table 2: The tracking failure probability and the average number of active sensors for each level

sensor in each time unit. Hence

$C_{state}(\mathbf{m}) = \#\mathbf{m}(Failure)*C_{notTrack}+\#\mathbf{m}(awake)*C_{power}$

where $\mathbf{m}$ is the current marking, $\#\mathbf{m}(PLACE)$ returns the number of token in a place for the current marking and $C_{not\_track}, C_{power} > 0$ are constants.

3.3 Optimal strategy

In order to compute the optimal power management strategy for the system, we have solved three instances of the above MDWN model, one for each level. Each instance is different from the others in weights associated with the transitions modeling the object movement; so that they have all the same SRG and MDP size (e.g. the SRG size is 37,657 symbolic states corresponding to 123,376 ordinary states and the size of the Lumped MDP is 580 states).

For each model we have computed the optimal power management strategy. Observe that the optimal strategy for this model is not trivial, because the central control cannot wake-up a sensor when it needs, but it can only decide whenever to send or not a sleep command to a sensor and the duration of such sleep. For these reasons the sleep decision for a sensor has a direct effect not only in the current time unit, but also on the next time units.

For instance if the object is in the zone $Z0_0$ and a sensor in zone $Z0_1$ has still to spend two time units in asleep status while the other only one time unit, and all the sensors in zone $Z0_2$ are active, then the optimal action is to move a sensor of the zone $Z0_2$ in the asleep status for two time units. Indeed, in this case the central control knows that in one time unit a sensor in the zone $Z0_1$ will be in awake state, so that the best action optimizing the trade-off between the power saving and the error tracking is to switch-off one sensor in the zone $Z0_2$ for two time units.

Instead, if the object is in the zone $Z0_0$ and all the sensors in zone $Z0_1$ have still to spend two time units in a asleep status and all the sensors in zone $Z0_2$ are active, then the optimal action is to move a sensor of the zone $Z0_2$ in the asleep status for only one time unit. Indeed in this second case, it knows that in two time units the sensors in the zone Z_1 will be in awake state, so that the best action optimizing the trade-off between the power saving and the error tracking is to switch-off a sensor in the zone $Z0_2$ for only one time unit.

Finally, the global optimal power management strategy is given by superposition of the three computed optimal power management strategies.

Moreover, we have also computed the tracking failure probability and the average number of active sensors for time unit solving the Discrete Time Markov Chain (DTMC) obtained from the underlying MDP assuming the optimal strategy is always applied. These values are reported in Table 2, where the first column shows the level, the second one the tracking failure probability and the last one the average number of active sensors. In the same table, these values are compared with the values obtained from the SAN model (Sec. 5).

3.4 MDWN model complexity

In this section, we discuss the states space growth of the MDWN model as a function of the number of sensors in each zone and the maximum sleep duration.

Unfortunately, the experiment results show that the model size grows very quickly while increasing the value of its parameters. Moreover, we observe that the parameter that has more impact on the model size is the number of sensors for zone. For instance, for the case 2-3 (Table 3), if we increase the number of sensors of one unit then the RG increases by a factor of 164, the SRG of 24 and the MDP of 5; on the other hand if we increase the maximum sleep duration of one unit the RG, the SRG and the MDP increase by factor of ~ 2.

These results are summarized in Table 3 where the first column shows the experiment parameters (e.g. $|S|$ the number of sensors for zone and $|C|$ the maximum sleep duration), while the second, third and fourth column are related to the MDWN model, and the last two columns to the underling reduced MDP model. In particular, the second, third and fourth column report the number of ordinary states (RG size) and symbolic states (SRG size), and the SRG generation time. Instead the last two columns show the number of states of the reduced MDP and its generation and solution time.

	MDWN			MDP	
$\|S\|$,$\|C\|$	|RG|	|SRG|	T.	St.	T.
2-1	19,253	6,356	5s	144	1s
2-2	123,376	37,657	56s	580	23s
2-3	229,661	67,001	75s	825	2m
2-4	527,768	149,708	5m	1,575	11m
2-5	1,050,757	292,324	10m	2,744	30m
3-1	920,981	55,508	2m	420	24s
3-2	7,818,304	379,840	16m	1,600	25m
3-3	37,737,589	1,623,725	52m	4,725	44m
4-1	45,246,989	345,200	14m	975	5m

Table 3: The state space growth of the MDWN model where $|S|$ is the number of sensors for zone and $|C|$ the maximum sleep duration.

4. EVALUATION OF STRATEGIES ON SAN

In the MDWN model presented in Sec. 3, we consider the internal zones of only one level, while the other levels are considered as a single locations, as depicted in Fig. 4. This assumption allows to reduce the computational cost of the MDWN analysis.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

We are now interested to evaluate the case study with more detail, by taking into account the movements by the object in all the zones on all the levels, according to Fig. 2. The MDWN is able to represent such movements in modeling terms, but the computation of measures on the MDWN model may become unfeasible because of the state space dimensions. For this reason, we need an alternative formalism to model the case study with all the possible object locations: we resort to Stochastic Activity Network (SAN) [11]. This formalism is Petri Net based, as MDWN, and is characterized by the possibility to rule the firing of transitions by means of particular nodes called *gates* and incorporating C code. So, the firing condition and effect can be specified inside a gate, in form of C code, possibly exploiting functions defined in external libraries. In this way, complex firing conditions or effects that would be complicated to express (or inexpressible) in a traditional Petri Net, can be specified in form of C code. We exploit this possibility in order to represent the optimal strategy derived from the MDWN, inside a gate of the SAN model. Therefore the SAN model of the case study represents all the possible movements by the object, and includes the optimal strategy to select the sensors to be switched off.

The SAN model of the case study has been designed and simulated by means of the *Möbius* tool [7]; the purpose is the evaluation of the tracking reliability and the power consumption in terms of quantity of active sensors. In this section, first some notions about the SAN formalism are provided, then the SAN model of the case study is described.

4.1 Basic notions on SANs

SANs can be considered as a particular form of Stochastic Petri Nets (SPN). So, a SAN model contains *places*, *activities* (transitions) and arcs. Places graphically appear as circles, and in SANs, the marking of a place can correspond to an integer variable, as in SPNs, or to another type of variable (float, character, structure, array, etc.). In the second case, we talk about extended places.

Activities graphically appear as vertical bars. An instantaneous activity completes (fires) as soon as it is enabled; a timed activity instead, completes after a certain amount of time. Such time can be deterministic or random. The completion (firing) of an activity is enabled by a particular condition on the marking of a set of places. Such marking can be expressed by connecting the activity to the standard places by means of oriented arcs, as it is possible in SPNs. The effect of the activity completion on the standard places can be specified in the same way. Another way to express the condition enabling a certain activity consists of using *input gates*, graphically appearing as red triangles. An input gate is connected to an activity and to a set of standard or extended places; the input gate is characterized by two expressions: (1) a *predicate* consists of a Boolean condition expressed in terms of the marking of the places connected to the gate; if such condition holds, then the activity connected to the gate is enabled to complete. (2) a *function* expresses the effect of the activity completion on the marking of the places connected to the gate.

Besides input gates, a SAN model can contain *output gates* as well; they appear as black triangles. An output gate has to be connected to a certain activity and to a set of standard or extended places. The role of an output gate is specifying only the effect of the activity completion on the marking of the places connected to the output gate. Therefore, an output gate is characterized only by a function.

In a SAN model, it is possible to set several completion cases for an activity; each case corresponds to a certain effect of the completion and has a certain probability: when the activity completes, one of the cases happens. A case graphically appears as a small circle close to the activity; from the case an arc is directed to an output gate or a place. Further information about the SAN formalism can be found in [11].

4.2 The SAN model

The SAN model of the case study is shown in Fig. 6; each part of the model represents a particular aspect of the case study:

the sensors state: the set of sensors is represented by the extended place *WSN* whose marking is an array having three dimensions corresponding to the level, the zone and the sensor respectively. So each element of the array represents a certain sensor in a certain zone in a certain level, and is a structure composed by two fields, *active* and *time*, representing the fact that the sensor is active or not, and the remaining sleep time in case of inactivity, respectively.

Activities synchronization: in the MDWN model, the time is discretized and at each time unit, the following events occur in this order: the object movement, the deactivation of sensors, the reactivation of sleeping sensors, and the object detection (Sec. 3). The SAN model has to be coherent with the MDWN, so it contains the timed activity *clock* which completes at each time unit, moving tokens inside the place *action*. This determines the completion of the instantaneous activities *move*, *sleep*, *wake* and *detect* (in this order) modeling the events listed above.

The object position and movement: the current position of the object is represented by the extended place *position* whose marking is a structure composed by the fields *level* and *zone*. The activity *move* determines the computation by the input gate *I_move*, of the probability distribution of the object to stay or to move to a certain zone or level, according to the current position of the object, as described in Sec. 2. Besides this, the activity *move* introduces one token in the place *movement* enabling the immediate completion of the activity *change_pos* which determines the effective movement: this activity has several alternative effects corresponding to the possible movements by the object, and modeled by the several completion cases of the activity. The probabilities of the completion cases are given by the distribution computed by the input gate *I_move*. Each case is ruled by a specific output gate determining the change of the object position in terms of zone or level. This is done by modifying (or maintaining) the marking of the extended place *position*.

The sensors deactivation: the *sleep* activity decides the sensor(s) to be deactivated. This is done by means of the corresponding input gate *I_sleep* whose internal code implements the strategy determined through the MDWN model (Sec. 3.3). Such strategy provides for each possible state of the WSN, the actions to be performed in order to minimize the reward function taking into account the power consumption and the object detection failures. In order to be coded inside a SAN gate, the strategy in such form has been abstracted first in terms of sensor states, then in terms of SAN place markings. When the activity *sleep* completes, the field *active* of the chosen sensors is set to 0, while the field *time*

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

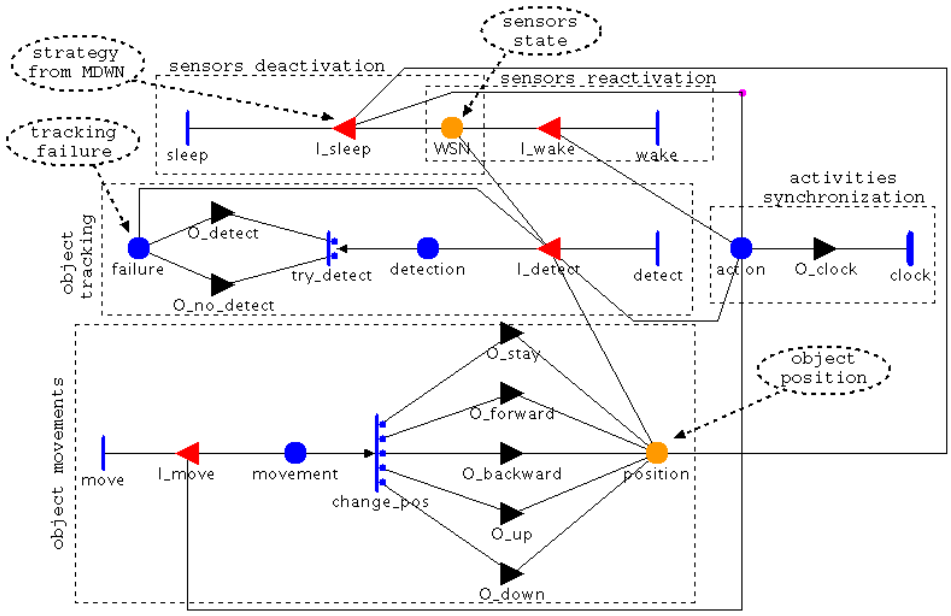

Figure 6: The SAN model of the WSN.

is set to the sleeping time (place *WSN*).

The sensors reactivation: the reactivation of those sensors whose time to sleep has expired, is modeled by the activity *wake* whose effect is ruled by the input gate *I_wake*. This activity checks the field *time* of inactive sensors: if *time* is equal to 0, the sensor becomes active again, by setting the field *active* to 1 for the corresponding element in the array related to the place *WSN*. If instead the value of *time* is greater than 0, it is simply decreased by one.

The object tracking: the detection of the object is modeled by the activity *detect*. Each time this completes, if the object is not inside the zone 0 of any level, the corresponding input gate *I_detect* computes the current probability to detect the object. This value depends on the number of active sensors in the zone where the object is positioned, as described in Sec. 2. Besides this, the input gate *I_detect* produces one token inside the place *detection* enabling the immediate completion of the activity *try_detect* which represents the actual tracking of the object. This activity has two completion cases corresponding to the fact that the object is detected or not, respectively. The probability of such cases are set to the probability computed by the gate *I_detect* and to its inverse value respectively. The effect of the completion cases is ruled by the output gates *O_detect* and *O_no_detect*, with the aim of setting the marking of the place *failure* to 0 or 1 respectively.

The tracking failure: if the object is currently located in any zone 0 (the field *zone* of the extended place *position* is equal to 0), then the gate *I_detect* simply sets the marking of the place *failure* to 0, in order to express that this situation is not considered as a tracking failure. Actually in Sec. 2, we stated that the object tracking fails only if the object is located in a zone where sensors are present and none of them detects the presence of the object. So, the role of the place *failure* is indicating the failure or the success of the object position tracking: if this place contains one token, then the detection of the object position by the sensors has failed.

5. EXPERIMENTAL RESULTS

Several measures have been evaluated by simulating the SAN model in Fig. 6 by means of *Möbius* setting a confidence level of 95% and an accurancy of 0.1:

1) the probability of object tracking failure: this measure estimates the WSN reliability in locating the current position of the object. It has been computed as the probability that the object is not detected and the object is not present in the zone 0 of any level. In terms of place markings, this probability corresponds to the probability that the place *failure* contains one token and the field *zone* of the extended place *position* is different from 0. By simulating the SAN model, the value of this measure converges to 0.21 ± 0.01.

2) The mean number of active sensors: this measure indicates the quantity of sensors that are active and as a consequence, the degree of power consumption by the WSN. It is computed as the sum of the values of the field *active* for each element of the array representing the marking of the extended place *WSN*. Each element of such array corresponds to a certain sensor in the WSN, while the field *active* is equal to 1 if the sensor is active, 0 otherwise. The value of this measure converges to 3.6 ± 0.07.

3) The mean number of sensors active on the level where the object is currently present: this measure is useful to control that only a subset of the sensors is active on the level where the object is present. It is computed as the sum of the values of the field *active* for each element of the array in the place *WSN* such that the corresponding sensor is located on the same level where the object is currently present. Such level is given by the current value of the field *level* of the structure representing the marking of the extended place *position*. The value of this measure converges to 2.1 ± 0.03. So, we

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

Strategy	#Levels	tracking failure probability	#active sensors	#active sensors / (4 · #Levels)
optimal	3	0.21±0.01	3.6±0.07	0.3
optimal	10	0.21±0.01	7.7±0.07	0.1925
optimal	50	0.21±0.01	31±1.5	0.155
ad-hoc	3	0.21±0.01	3.8±0.09	0.3167
ad-hoc	10	0.21±0.01	11.7±0.04	0.2925
ad-hoc	50	0.21±0.01	57±2.4	0.285

Table 4: Experiment results.

verified that only a subset of the sensors on the current level is active.

4) The probability of object tracking failure on the first, the second and the third level: it has been computed in the same way as the first measure, but requiring also that the field *level* of the extended place *position* is equal to the level of interest.

5) The mean number of sensors active on the first, the second and the third level: it has been computed in the same way as the second measure, but requiring also that the field *level* inside *position* corresponds to the level of interest. The results for the fourth and the fifth measure are shown in Tab. 2, together with the values obtained on the DTMC derived from the MDWN model according to the optimal strategy (Sec. 3.3). Both approaches return similar results. All the measures are influenced by the power management strategy derived from the MDWN model (Sec. 3).

In order to verify the efficiency of the optimal power management strategy, we defined an ad-hoc strategy (without the support of model optimization) such that the tracking reliability is close to the one obtained from the optimal strategy. We compared the mean number of active sensors determined by both strategies, with the aim of showing that the optimal strategy reduces such measure. In the ad-hoc new policy we assume that if the object is in ZJ_1 or ZJ_2 then we switch-off only an active sensor in such zone for one time unit; otherwise if the object is in ZJ_0, all the active sensors in the level J move into asleep state for one time unit. All the sensors not in the same object level are switched off for two time units.

To model this, we modified the code inside the output gate *I_sleep* in order to realize the ad-hoc strategy. This version of the SAN model has been simulated obtaining that the mean number of active sensors converges to 3.8 ± 0.09, while the probability of tracking failure is still 0.21 ± 0.01. This means that the optimal strategy derived from the MDWN actually reduces the number of active sensors and consequently the power consumption. This reduction becomes more evident if we increase the number of levels in the case study (Tab. 4).

The SAN model has been modified to this aim, by increasing the size of the array dimension representing the number of levels (place *WSN*). In particular, we simulated the SAN model for 10 and 50 levels, first assuming the optimal strategy. In every experiment, the value of the tracking failure probability and the value of the mean number of active sensors on the current level have been confirmed. In the case of the optimal strategy, the proportion between the mean number of active sensors and the total number of sensors (4 · #Levels) converges to 0.15 as the number of levels grows. This can be observed in Tab. 4 reporting the results for each experiment. So, the effect of the strategy on the tracking reliability and on power consumption, does not seem to change as the number of level varies.

Also the ad-hoc strategy has been evaluated for a growing number of levels. The results for this strategy confirm that it maintains the reliability level with respect to the optimal strategy, but the ad-hoc strategy is still worse than the optimal one in terms of number of active sensors. This can be observed by comparing the results obtained for each strategy reported in Tab. 4: for instance, assuming 50 levels, the proportion between the mean number of active sensors and the total number of sensors is equal to 0.155 according to the optimal strategy, while it is equal to 0.285 according to the ad-hoc strategy.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an approach to support the design and the analysis of a WSN using as a case study a monitoring system that must track a moving object within a given area.

The proposed methodology combines MDWN and SAN, two Petri Net based formalisms with different analysis capabilities that can be used in a synergistic way to reach the goal of finding good sleeping policies that optimizes the trade-off between the power saving and the error tracking. Practically, the MDWN model is more abstract and is used to compute an optimal power management strategy by solving a Markov Decision Process (MDP). The SAN model instead, is more detailed and is used to perform extensive simulation (using the *Möbius* tool) [7] in order to analyze different performance indices, both when applying the power management policy derived from the first model and when using different policies. By comparing the results obtained from both strategies, we verified the optimality of the strategy returned by the MDWN analysis. The optimal strategy for this case study is not trivial to derive without the MDWN support, because the sensors can wake-up only when the sleep time expires, even if they might be needed earlier.

The case study evaluated in this paper can be extended by increasing the number of possible locations by the object. In particular, a single zone may be divided into several sub-zones where the object can move, while the number of sensors in the zone may remain the same. In this way, the probability to detect the object presence may not only depend on the number of sensors that are active in a certain zone, but also on the distance of the object from the sensors.

Another possible extension of the case study may take into account the power consumption with more detail: in this paper, we simply related the power consumption with the number of active sensors; in the future, we may introduce explicitly a residual battery charge variable, to evaluate the lifetime of the system.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736

Besides this, we plan to perform several simulation experiments where the parameters such as the detection probability or the movement probability distribution, vary in each experiment. The *Möbius* tool for the SAN design, analysis and simulation, allows the user to easily manage a set of experiments on the same model, as the parameters change. Moreover, *Möbius* provides a particular utility called *design of experiments* (DOE); by means of this, the set of experiments can be automatically generated in such a way that the relevant parameters are identified and tested for their significant values. Such generation can be done according to experiment design types [6].

Finally, three extensions of our work will be investigated. First, we are going to extend our approach, so that the probability distributions modeling the object movements can be updated on-line. Practically, this extension can be implemented considering initially all the possible object movements equiprobable and then updating at regular intervals these statistics according to the observed object movements. Obviously, the optimal power management strategy must be re-generated every time that these statistics are updated. The second extension consists to model the message routing between a sensor and the sink under the assumption that not all the sensors are connected directly to the sink. In this context an awake sensor may not succeed in communicating with the sink due to the fact that can not reach the sink (e.g. if all its neighbors are sleeping). Therefore, in the computation of the optimal power management strategy we have also to take into account this aspect. The last extension consists to use Partially Observable MDP (and hence an extension of MDWN in this direction) in order to take into account the fact that the actual system in operation can not distinguish a state where the object is in a zone not covered by the sensor nodes and one where the object is in a zone that is potentially covered by a sensor node, but it is sleeping; a similar approach has been adopted in [8].

Acknowledgments.

This work has been partially supported by the MIUR funded Project PRIN07-WiseDemon, and by local research funds.

7. REFERENCES

[1] M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. J. Wiley and Sons, 1995.

[2] C. Alippi, G. Anastasi, C. Galperti, F. Mancini, and M. Roveri. Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In *Proc. of the IEEE Int. Workshop on Mobile Ad hoc and Sensor Systems for Global and Homeland Security (MASS-GHS 2007)*, Pisa, Oct. 2007.

[3] M. Beccuti, D. Codetta-Raiteri, G. Franceschinis, and S. Haddad. A framework to design and solve Markov Decision Well-formed Net models. In *Proc. of the 4th Int. Conf. on Quantitative Evaluation of Systems (QEST)*, pages 165–166, Edinburgh, UK, Sept. 2007.

[4] M. Beccuti, G. Franceschinis, and S. Haddad. Markov Decision Petri Net and Markov Decision Well-Formed Net Formalisms. *Lecture Notes in Computer Science*, 4546:43–62, 2007.

[5] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications. *IEEE Trans. on Computers*, 42:1343–1360, 1993.

[6] T. Courtney, S. Gaonkar, M. G. McQuinn, E. Rozier, W. H. Sanders, and P. Webster. Design of Experiments within the Möbius Modeling Framework. In *Proc. of the 4th Int. Conf. on the Quantitative Evaluation of SysTems (QEST)*, pages 161–162, Edinburgh, UK, Sept. 2007.

[7] D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. Doyle, W. Sanders, and P. . G. Webster. The Möbius Framework and its Implementation. *IEEE Trans. on Software Engineering*, 28(10):956–969, 2002.

[8] J. A. Fuemmeler and V. V. Veeravalli. Smart sleeping policies for energy efficient tracking in sensor networks. *IEEE Trans. on Signal Processing*, 56(5), May 2008.

[9] M. Gribaudo, D. Cerotti, and A. Bobbio. Analysis of on-off policies in sensor networks using interacting Markovian agents. In *Proc. of the 4-th Int. Workshop on Sensor Networks and Systems for Pervasive Computing - PerSens 2008*, pages 300–305, 2008.

[10] M. Puterman. *Markov decision processes : Discrete Stochastic Dynamic Programming*. J. Wiley & Sons, 1994.

[11] W. H. Sanders and J. F. Meyer. Stochastic activity networks: Formal definitions and concepts. *Lecture Notes in Computer Science*, 2090:315–343, 2001.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7736
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7736