2012 26th IEEE International Conference on Advanced Information Networking and Applications

Mixed observability Markov decision processes for overall network performance optimization in wireless sensor networks

Daniel L. Kovacs
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
dkovacs@mit.bme.hu

Wuyungerile Li, Naoki Fukuta, Takashi Watanabe
Faculty of Information
Shizuoka University
Hamamatsu, Japan
{li, fukuta, watanabe}@aurum.cs.inf.shizuoka.ac.jp

***Abstract*—Optimizing overall performance of Wireless Sensor Networks (WSNs) is important due to the limited resources available to nodes. Several aspects of this optimization problem have been studied (e.g. improving Medium Access Control (MAC) protocols, routing, energy management) mostly separately, although there is a strong inter-connection between them. In this paper an Artificial Intelligence (AI) based framework is presented to address this problem. Mixed-Observability Markov Decision Processes (MOMDPs) are used to effectively model multiple aspects of WSNs in stochastic environments including MAC in data link layer, routing in network layer, data aggregation, power management, etc. MOMDPs distinguish between full and partial observability, hence they are more efficient than other similar AI methods. The proposed framework provides global optimization of user-defined performance metrics, e.g. minimization of time delay, energy consumption and data inaccuracy. Near-optimal joint network policies are obtained via offline approximation of optimal MOMDP solutions and they are distributed among the individual nodes. Resulting node-policies place effectively no additional computational overhead on nodes in runtime. Experiments evaluate the framework by demonstrating near-optimal solutions for a small-scale WSN in detail in case of given tradeoff criteria. The proposed approach produces better joint network behavior in 5 out of 6 cases compared to other two standard methods in simulation by increasing overall network performance by more than 20% in average.**

Keywords-partially observable Markov decision processes; wireless sensor networks; mixed observability; overall performance optimization; decentralized; tradeoff optimization

I. INTRODUCTION

Wireless Sensor Networks (WSNs) [1] are used for decentralized measurement of real-time data. The recent application of such networks ranges from home automation, through industry control to outer space monitoring [2]. The overall efficiency of such networks is important in every application, yet there is still no general means to optimize it. The difficulty arises from the limited resources of nodes; the uncertain, dynamic environment; nodes' partial and asymmetric information about the current state of the environment; the decentralized nature of nodes' operation; and the strong inter-connection between different network layers such as Medium Access Control (MAC) protocols in data link layer; routing in network layer, topology control and data aggregation algorithms; power management schemes; sleep/wake and data generation policies, etc.

Several methods were proposed to cope with the above issues [3], [4], [5] and [6], but none of them provides a means to solve them together in general. The proposed solutions are either too specific (e.g. [3]), or of limited scope (e.g. [5][7]), or overly simplifying (e.g. [6]), or their (near)optimality is not guaranteed (e.g. [4]). There is also research about tradeoff optimization in WSNs [8], however it takes account of 3 fixed factors currently (delay, energy and accuracy), and has no guarantee about optimality.

In this paper the above issues are addressed in general. Well established optimization methods are borrowed from Artificial Intelligence (AI), and applied to find a multi-layer policy/program for each WSN node to altogether produce an approximately optimal joint network behavior according to a given, user-defined performance metric.

A straightforward choice for this reason is to apply Partially Observable Markov Decision Processes (POMDPs) [9][10] which guarantee optimal behavior in uncertain environments, but because of the complexity of real-world situations, approximations are needed in practice (currently optimal solutions of problems with around 10^{60} states can be approximated [11]). Beside approximation another approach is to factorize the problem description. One of the most recent techniques combining approximation and factorization is MOMDP (Mixed Observability MDP) [12], which intuitively divides the problem into a fully and a partially observable part, which are then vectorized further into individual inter-related variables. Beyond the convenience of use of this approach, it also allows a significant speed-up by reducing the dimensionality of the problem.

Because of the above advantages we chose MOMDPs to model and optimize WSNs in our work, but MOMDPs similarly to POMDPs are single agent concepts, while WSNs incorporate multiple agents (nodes). Decentralized POMDPs (DEC-POMDPs) [13] could offer a solution, but at the price of very high computational complexity. So we finally decided to model the whole network as a single agent with MOMDP, approximate an optimal MOMDP solution offline and then distribute this near-optimal solution (which is a near-optimal policy for the whole network) among the individual nodes to govern their individual behavior in runtime. Resulting node-policies are collections of simple non-numeric `if-then` rules, which place effectively no

1550-445X/12 $26.00 © 2012 IEEE
DOI 10.1109/AINA.2012.17

additional computational overhead on nodes. Thus WSNs operating according to near-optimal overall network policies can be realized in practice. Experiments demonstrate this approach in detail for a small size WSN, but the proposed method can also be scaled-up to optimize larger networks.

The structure of the paper is as follows: Section 2 presents the fundamentals of WSNs, POMDPs and MOMDPs; Section 3 starts the discussion of the main result: the application of MOMDPs for modeling and optimizing overall network performance of WSNs; Section 4 presents experiments and their evaluation in detail; Section 5 concludes the paper and gives an outline of future research.

II. PRELIMINARIES

In this section we introduce the fundamentals of WSNs, POMDPs and MOMDPs to an extent that is needed for understanding the later discussion of the presented results.

A. WSN

WSNs usually consist of a large number of battery-powered sensor nodes which organize into a network by themselves. Sensor nodes sense environmental events and generate data about these events. Then they transmit this data to a sink node via intermediate sensor nodes in multi-hop manner. The sink node usually connects to users' terminal nodes (TN) via Internet as shown in Fig. 1. If the battery/capacitor of a sensor node is exhausted, the network may collapse. Therefore, network lifetime is entirely decided by nodes lifetime. Hence, sensor networks should be energy efficient in light of the limited resources available to nodes. On the other hand, nodes' communication and computing abilities are also limited. Therefore current research in WSNs focuses on energy saving, data freshness, data security, error control, packet loss, throughput, data accuracy, QoS, network connectivity and tradeoffs among them. To achieve a higher performance, many researchers focus on improving e.g. MAC protocols, routing protocols, topology control algorithms, and data aggregation techniques, each making advances in its own subfield, but not addressing the performance optimization problem in its entirety. In this paper we try to solve this problem by applying methods based on Markov Decision Processes (MDPs) as follows.

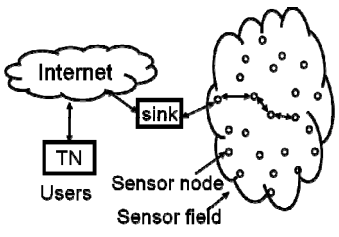

Figure 1. Typical use of WSNs

B. POMDP

POMDPs model single agents in uncertain environments. A discrete POMDP with an infinite time-horizon is specified as a tuple $(S, A, O, T, Z, R, \gamma)$, where S denotes the finite, non-empty set of environmental states; A is the finite, non-empty set of agent's actions; O is the finite set of agent's observations; $T: S \times A \times S \to [0,1]$ is the state transition function, which associates a $T(s, a, s')$ probability to every $s, s' \in S$ state and $a \in A$ action, meaning that if action a is executed in state s of the environment, then the next state of the environment will be s' with a probability $T(s, a, s')$. $Z: S \times A \times O \to [0,1]$ is the observation function of the agent – it associates a $Z(s', a, o)$ probability to every $s' \in S$ state, $a \in A$ action, and $o \in O$ observation, meaning that the probability that the agent observes o in a resulting state s' *after* executing action a is $Z(s', a, o)$. $R: S \times A \to \mathbb{R}$ is the real valued reward function of the agent, which associates an $R(s, a)$ payoff/reinforcement value to every $s \in S$ state and $a \in A$ action, defining the goodness of taking action a in state s. The above mentioned model-elements and their connections are summarized in Fig. 2.

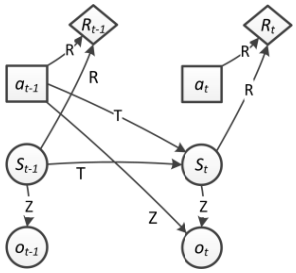

Figure 2. POMDP scheme

Fig. 2 illustrates POMDPs, where s_t denotes the state of the environment at time-period t; a_t denotes the action the agent chooses in s_t; o_t denotes the observation the agent receives in s_t after taking a_{t-1}; and $R_t = R(s_t, a_t)$ is the reward of the agent for taking a_t in s_t.

The goal of the agent is to plan its actions in advance to maximize its expected total reward, $\mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, where $\gamma \in [0,1)$ is a discount factor, which defines the importance/weight of future actions. The actions of the agent are assumed to be chosen by a deterministic policy $\pi: B \to A$, a mapping from the set of beliefs B to the set of actions A. Eventually $B = \Delta(S)$, i.e. the set of beliefs is the set of all probability distribution above S. Given a $b \in B$ belief, $b(s)$ denotes the probability of state $s \in S$. Thus an agent's policy in a POMDP prescribes an action to its every possible belief.

Agents actions are based on their beliefs $b \in \Delta(S)$, because they can't fully access state $s \in S$, otherwise we could model the situation with a MDP [14]. Nonetheless in our case MDPs are not enough, since beyond uncertain state transitions (e.g. random transmission failure), which can be modeled with MDPs, nodes in a WSN don't have full access to the current state in general (they can't observe farther nodes' actions, or detect events without noise, or check the availability of a channel without possible error), thus POMDPs are needed to describe them.

After an action is chosen, a state transition occurs, and an $o \in O$ observation is received. Based on this the agent updates its previous belief b given the executed action $a \in A$ and observation $o \in O$ to a new belief, $b' = \tau(b, a, o)$, where $\tau: B \times A \times O \to B$ is the belief update function, which is calculated for $\forall s' \in S$ as follows.

$$b'(s') = \tau(b,a,o)(s') = \eta(o)Z(s',a,o)\sum_{s\in S}T(s,a,s')b(s) \quad (1)$$

Equation (1) calculates the probability that after executing *a,* given *b,* we arrive to state *s'*, and then weights this with the probability of receiving observation *o* in *s'* after executing *a*. $\eta(o) = 1/[\sum_{s'\in S}Z(s',a,o)\sum_{s\in S}T(s,a,s')b(s)]$ is a normalizing constant.

Now based on b' the agent chooses its new action according to π. After executing $\pi(b')$ a transition occurs, and a new observation is received again. The agent updates its belief accordingly and the whole process repeats again. The initial belief b_0 is assumed to be given at the beginning.

The above concept can be used for *online* decision making when actions are chosen and observations are received in runtime accompanied with the appropriate $R(s,a)$ rewards, or it can be used *offline* by calculating contingencies in advance. In the latter case the agent receives only an expected reward, $r(b,a) = \sum_{s\in S}b(s)R(s,a)$.

The online approach does less calculations overall than the offline approach and also it can be used in case when the reward function is initially not known, but the agent needs to update the probability of every $s' \in S$ state at every step, so it may not be feasible for large state spaces in runtime with bounded resources. In this case it may be more suitable to use the offline approach. Either way the choice of actions is of central importance. *What action should we choose at a given time period? Are there optimal actions and/or policies?* To answer this we need to define optimality first. In case of POMDPs a policy π^* is optimal if it maximizes the expected total reward discussed earlier. This optimal reward value is defined recursively as follows.

$$V^*(b_0) = \max_{a\in A}\left[r(b_0,a) + \gamma\sum_{o\in O}z(b_0,a,o)V^*\left(\tau(b_0,a,o)\right)\right] \quad (2)$$

Equation (2) is the maximal expected total reward that is achievable starting from b_0; V^* is the optimal value function; and a policy π^* that achieves $V^*(b_0)$ from b_0, i.e. which produces the appropriate actions in Eq. (2), is an **optimal policy**. $z(b,a,o) = \sum_{s\in S}b(s)Z(s,a,o)$ for any $b \in B$.

Finding an optimal policy is intractable in practice for realistic problems with large state spaces, but it is known that V^* can be approximated arbitrarily closely by a convex, piecewise-linear function

$$V^*(b) \approx V(b) = \max_{\alpha\in\Gamma}(\alpha\cdot b)\ (\forall b \in B), \quad (3)$$

where Γ is a finite set of α-vectors, and $\alpha\cdot b$ is the scalar product of α, a vector of size $1\times|S|$, and b, the discrete vector representation of a belief of size $|S|\times 1$. An action $a(\alpha)\in A$ is associated with each $\alpha\in\Gamma$ vector, which is optimal for the given belief *b*, if α solves Eq. (3).

Several approximation methods exist to construct Γ in tractable time (e.g. [15]), so it can be a base of a near-optimal policy $\tilde{\pi}^*(b) = a(\arg\max_{\alpha\in\Gamma}(\alpha\cdot b)) \in A$.

C. MOMDP

MOMDP [12] is a recent approach trying to reduce the complexity of POMDP by dividing its representation of states into fully and partially observable parts, and then focus on solving the partially observable sub-problems of reduced dimensionality, i.e. in MOMDP the set of states is a Cartesian product $S = X\times Y$, where *X* is the set of values of fully observable state-variables, and *Y* is the set of values of partially observable state-variables. Thus $s\in S$ in MOMDP is a pair $s = (x,y)$, where $x\in X$ and $y\in Y$. As a consequence the state transition function *T* is also divided in two: $T_X: X\times Y\times A\times X\to[0,1]$ and $T_Y: X\times Y\times A\times X\times Y\to[0,1]$, where $T_X(x,y,a,x')$ and $T_Y(x,y,a,x',y')$ denote transition probabilities of fully and partially observable variables respectively. The observation function *Z* and the reward function *R* are also modified accordingly. The former is $Z: X\times Y\times A\times O\to[0,1]$, where $Z(x',y',a,o)$ is the probability of observing $o\in O$ in resulting state $(x',y')\in S$ after executing action $a\in A$, while the latter is $R: X\times Y\times A\to\mathbb{R}$, where $R(x,y,a)$ denotes the reward for taking action $a\in A$ in state $(x,y)\in S$. The listed relations among variables are summarized in the following figure.

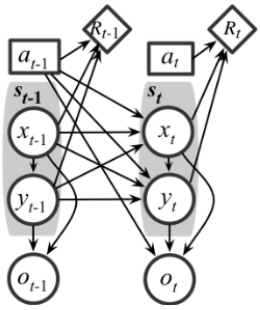

Figure 3. MOMDP scheme [12]

As hinted before an MOMDP model is solved for every possible $x\in X$ value of the fully observable state-variables by solving the respective POMDP sub-problem. Given $x\in X$, the set of possible beliefs is $B_Y(x) = \{(x,b_Y)|b_Y\in B_Y\}$, where $B_Y = \Delta(Y)$, the set of all possible beliefs above the values of partially observed variables *Y*. That means that there is no probability distribution given above *X*, since $x\in X$ is always observable. Thus approximation of the optimal value function V^* produces a separate set of α-vectors for each $x\in X$, which is denoted by $\Gamma_Y(x)$.

MOMDPs are more effective, when most of the variables are fully observable, which is exactly the case in WSNs (where there is mostly always a node that can observe the actual value of any variable except for e.g. environmental event occurrence, or the activity of far nodes). Given a belief (x,b_Y) the near-optimal policy $\tilde{\pi}^*$ should now produce an action $\tilde{\pi}^*(x,b_Y) = a\left(\arg\max_{\alpha\in\Gamma_Y(x)}(\alpha\cdot b_Y)\right)\in A$, i.e. an $a\in A$ action which is associated with the α-vector $\alpha = \arg\max_{\alpha\in\Gamma_Y(x)}(\alpha\cdot b_Y)$.

Beyond sets of α-vectors an MOMDP policy π can also be represented with a directed policy graph (c.f. Fig. 4) where vertices are $\langle(x,b_Y),\pi(x,b_Y)\rangle$ pairs, whose outgoing edges lead to vertices $\langle(x',b_Y'),\pi(x',b_Y')\rangle$ for every $x'\in X$ and $o\in O$, where the next belief $b_Y' = \tau(x,b_Y,\pi(x,b_Y),x',o)$ can be calculated according to Eq. (6) in [12] as a consequence of (1). Edges are labeled with 4-tuples

$\langle x', T_X(x,y,a,x'), o, z(x', b'_Y, a, o)\rangle$, where $z(x', b'_Y, a, o) = \sum_{y' \in Y} b'_Y(y') Z(x', y', a, o)$. These labels fix "observation" (x', o) according to which the belief is updated. The policy graph is constructed starting from the initial belief (x_0, b_{Y0}). The following figure illustrates MOMDP policy graphs.

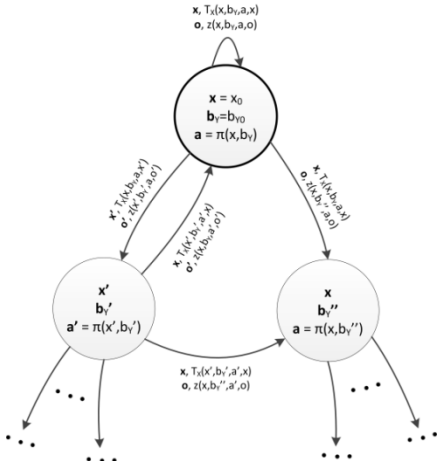

Figure 4. Illustration of MOMDP policy graphs

III. PROPOSED APPROACH

In this section we present an approach that uses MOMDP to model WSNs and optimize their overall network performance (time delay, energy usage, data accuracy, etc.). In Section III/A the modeled WSNs are presented, in Section III/B their MOMDP model is given, and in Section III/C the application of the solution of such a model is discussed.

A. Properties of modeled WSNs

The properties of the considered WSNs are common. We assume that one communication channel is available to nodes. The MAC protocol governing channel access is CSMA/CA. All nodes have the same type of sensor. Events occur according to Poisson distribution, which is a priori known. Sensor nodes can listen to events, but the detection may be false-positive (i.e. if there is no event, the node can still falsely detect it), or false negative with a given probability. Otherwise event detection is correct.

This is followed by data generation, which can also go wrong with a given a probability (so no data is generated). The same happens, when there is no event occurring. The goal of the network is eventually that every generated data is relayed to the sink node (maybe via multiple hops).

Sensor nodes can listen to events; generate, aggregate, send or receive data, or just be idle (but respond to incoming transmissions). They can execute only one of these actions at a given time. The sink node on the other hand can process and receive data simultaneously. The processed data disappears from the sink node. Processing always succeeds.

There is a non-zero probability of data loss during transmission. Sending a message drains energy from the sender, but reception may drain even more from the receiver. Energy usage is proportional to the size of transmitted data. Similarly listening and data generation also drain energy. The energy consumed by the network to generate and deliver all data to the sink is called *total energy*. The time needed for listening, data generation, aggregation and data transmission is assumed to take one period of time. The time of generating and delivering all data to the sink is called *total delay*.

Aggregation aims to save energy by integrating all individual data which is present at a given sensor node at a given time into one message of size of a generated data. There is no redundant data at nodes (only the most recent version is stored). Energy usage of aggregation is negligible.

Data accuracy is the proportion of data generated in the network and the data size of messages received by the sink. The sink's energy is not considered in the model, since it is much less limited that the energy of sensor nodes.

B. An MOMDP model of WSNs

In this subsection we model the above informal specification of WSNs formally with MOMDPs. Such a model can then be solved via existing MOMDP/POMDP solvers, and the obtained near-optimal solution (joint network policy) can be applied to WSN optimization.

There are many possible ways to use MOMDP to model WSNs. Our approach takes into account that the solution of the model needs to be distributed among individual nodes to be implementable in practice (see. Section III/C).

Let us start by first defining the set of states, $S = X \times Y$, as discussed in Section II/C. To model the position of every generated data in the network (as a part of the state) we introduce the following Boolean variables representing the truth about data of node *i* being at node *j*: $data_{i,j} \in \{false, true\}$ for every $i \in 2^{N\setminus\{sink\}}\setminus\{\emptyset\}$ and $j \in N$, where *N* is the non-empty set of network nodes (including the *sink* node, which isn't generating data), so data *i* can be any non-empty element of the power-set of set $N\setminus\{sink\}$. This is because of aggregation, which can produce any combination (subset) of data generated at sensor nodes. So for example variable $data_{\{1\},2}$ represents the fact about data generated at node 1 being at node 2; $data_{\{1,2\},1}$ on the other hand denotes the fact that data $\{1,2\}$, which is an aggregation of data generated at node 1 and 2, is at node 1.

We assume, that on a network level all these variables are fully observable, because there is always a node that can observe the exact value of any of these variables (node *j* can observe the ever actual value of any $data_{*,j}$ variable). Thus the set of fully observable state-variables, *X* is the Cartesian product of the domain of any possible $data_{i,j}$ variable.

$$X = \prod_{i \in 2^{N\setminus\{sink\}}\setminus\{\emptyset\}, j \in N} \mathrm{Dom}(data_{i,j}). \quad (4)$$

In case of two sensor nodes and a sink there would be $|X| = 2^9 = 512$ possible fully observable states of the environment. Thus, since the number of different data is exponential in the number of nodes, and the set of fully observable states is exponential in the number of different $data_{i,j}$ variables, the number of fully observable states is $|X| = 2^{|N|(2^{|N|-1}-1)}$ in case of only Boolean variables, which means that it is super-exponential in the number of nodes (a hint on the complexity of the problem). But this is the case with any environment, whose state-space is a Cartesian

product of state-variables' enumerable domains, and the number of state-variables is exponential (a quite typical scenario for realistic models of environments). In our approach it is the duty of the solver to overcome this complexity (e.g. by approximation), although the model could also be simplified.

The partially observable part of the environment, *Y*, in accordance with Section III/A, consists only of variables describing local event occurrence at sensor nodes: $event_i \in \{false, true\}$ represents the fact, if an event is occurring at sensor node $i \in N\backslash\{sink\}$. Thus the set of states, *S*, is the Cartesian product of sets *X* and *Y*. In case of our previous example with two sensor nodes and a sink node, the size of this space would be $|S| = |X||Y| = 2^9 \cdot 2^2 = 2048$, which would be divided by MOMDP into 512 partially observable subspaces each having 4 possible states (representing joint event occurrence at the two sensor nodes).

Initially $data_{i,j} = false$ should hold for every $i \in 2^{N\backslash\{sink\}}\backslash\{\emptyset\}$ and $j \in N$, since there is no generated data in the network. The a priori probability of event occurrence at sensor nodes $i \in N\backslash\{sink\}$, $p(event_i = true)$ should be given.

The joint set of observations is $O = \prod_{i\in N\backslash\{sink\}} O_i$, where $O_i = \{false, true\}$, whose values represent the fact, if sensor node *i* detects an event. The probability of an observation is defined by observation function *Z*, and depends on the state and nodes' actions - only nodes listening are able to detect events. Beside listening, sensor nodes are also able to generate data, to send data to neighboring nodes (this includes the reception of data by the receiver), and to aggregate all their data into one message, or just stay idle (*do_nothing*). Thus a sensor node $i \in N\backslash\{sink\}$ has a set of actions $A_i = \{do\_nothing, listen, generate, aggregate\} \cup \ldots \ldots \cup \{send_{p,q}\}_{p\in 2^{N\backslash\{sink\}}\backslash\{\emptyset\}, q\in N(i)}$, where $send_{p,q}$ means sending data *p* to node *q*, and $N(i)$ denotes the set of neighbors of *i*. The sink node's actions are: $A_{sink} = \{do\_nothing, process\}$. The set of joint network-actions is thus $A = \prod_{i\in N} A_i$.

We can now define the state transition function *T*, which in MOMDP is divided into T_X and T_Y. We will further divide both of these functions by separately defining transition probabilities for each respective variable. Thus for example the probability, that given state $s = (x, y) \in S$ and action $a \in A$ the value of fully observable variables will be $x' = \left(data'_{i,j}\right)_{i\in 2^{N\backslash\{sink\}}\backslash\{\emptyset\}, j\in N} \in X$ after transition is the product $T_X(x, y, a, x') = \prod_{i\in 2^{N\backslash\{sink\}}\backslash\{\emptyset\}, j\in N} T_X\left(x, y, a, data'_{i,j}\right)$.

We need to define transition probabilities for every fully and partially observable variable. The main principle of this is the following. Concerning $data_{i,j}$, we can distinguish 5 cases depending on the value of *i* and *j*.

1) $|i| = 1, j = i$*:* in case of non-aggregated data *i*, if *j*=*i*, the probability that $data'_{i,j} = true$ (i.e. that node *i* will have its own locally generated data after transition) depends on whether it or its neighbors have it actually, if it has others' individual data, what action $a_i \in A_i$ it takes, and what actions its neighbors take. It can send its data away (if the other node is not busy), or receive it from some other node, or just generate it with a given probability, or aggregate it into a message with other data (if it has that other data).

2) $|i| = 1, j \neq i, j = sink$*:* in case 1) *j* couldn't be the sink node, since it isn't generating data (c.f. Section III/A), but if *j* is the sink node, the probability that $data'_{i,j} = true$ for data *i* depends on whether the sink node or its neighbors have *i*, what action the sink node takes, and what actions its neighbors take. It can receive *i* from a sensor node, or "lose it" by processing it, or receive a new version while processing the previous. The transmission probabilities should be defined accordingly to cover data loss, etc.

3) $|i| = 1, j \neq i, j \neq sink$*:* in case of a single data *i* being at another node (not which generated it), but not at the sink, the probability of $data'_{i,j} = true$ depends on whether *j* or its neighbors have *i*, if *j* is sending *i* away, or receiving it from a neighbor (which should have it for that), or if *j* is just aggregating *i* with other data (in case it has that other data).

4) $|i| > 1, j = sink$*:* in case *i* is an aggregated data, the probability that $data'_{i,j} = true$ holds, depends on if the sink has *i*, if its neighbors have it, if it is processing it and/or receiving it, or if it is just idle. The probability of the sink having *i* next in case a neighbor is sending it to the sink, and it is not having *i*, should be equal for example to the case when the sink has *i*, but it is processing it, while a neighbor is sending it. This is since processing is always successful.

5) $|i| > 1, j \neq sink$*:* the probability that a sensor node *j* will have an aggregated data *i* after state-transition depends on whether *j* or its neighbors have *i*, whether *j* is sending *i* away, or receiving *i* from some neighbor, or whether *j* creates *i* by aggregating the necessary single data.

The value of the partially observable variables, $event_i$ ($i \in N\backslash\{sink\}$), is not depending on anything – it is stochastic. This means that the probability $\mathrm{p}\left(event'_i = \mathrm{true}|\ldots\right)$ is unconditional, and should be equal to the initial probability $p(event_i = true)$ for $\forall i \in N\backslash\{sink\}$.

Now let us focus on the observation function *Z*, and the reward function *R*. The probability that node *i* detects a local event after nodes take joint action $a \in A$ and arrive to a state $s' = (x', y') \in S$ is $p\left(o'_i = true|x', y', a\right)$, which depends only on whether node *i* was listening, and if the event in the resulting state *s'* is occurring. Thus we should define only the following probabilities: $p\left(o'_i = *\,|event'_i = *, a_i = listen\right)$. Otherwise $p\left(o'_i = false|event'_i = *, a_i = *\right) = 1$ should hold. We should allow false-positive and false-negative observations as discussed in Section III/A.

Concerning the reward function, we propose to divide it also among the nodes, i.e. $R(x, y, a) = \sum_{i\in N} R_i(x, y, a)$ should hold, where $R_i(x, y, a)$ denotes the individual reward of node $i \in N$ in case nodes choose action $a \in A$ in state $s = (x, y) \in S$. By default the reward should be zero, but in case node *i* is sending a message (without interfering with other nodes) the reward should be *negative*. In case node *i* is receiving a message, the reward should be also negative, since sending and receiving drain energy. Listening, and generating data should also have a negative reward, but less negative, than sending/receiving, and it should be independent of success. In case of the sink node reward shouldn't depend on other nodes' activity, only on its own and the data it has. If it does nothing (is idle, or receives data), its reward should be zero. But if it processes data, then it should get a *positive* reward,

which should compensate the cost of relaying that data to the sink. We suggest that processing a collection of single data should have a greater reward than processing the same data in aggregated form, to put an emphasis on data accuracy.

Eventually the MOMDP model of the WSN discussed in Section III/A is complete this way, except for the definition of the discount factor γ. We suggest to set it around $\gamma = 0{,}95$ by default, then run some tests to calibrate it appropriately.

C. Applying the solution of the MOMDP model for WSNs

After solving the MOMDP model constructed in Section III/B with an existing MOMDP solver, it is not trivial to apply the solution in practice, since it is a *joint policy for the whole network*, which first needs to be distributed among the individual nodes. The problem is that nodes can't observe the observations of other nodes; the local state-variables of other nodes, which are fully observable only to those nodes; their actions (if they are out of reach); and consequently others' beliefs, on which their decisions are based.

To cope with the above mentioned difficulties two methods for the distribution of joint MOMDP policies are proposed. The first approach assumes that local information can be exchanged between nodes to form a consistent global picture of the true state at every time-period at every node so they can use the joint policy to decide about actions, while the second method isn't assuming this possibility and thus the joint policy needs to be really divided among the nodes.

1) Every node using the same joint policy for decision: The following pseudo-code summarizes the program governing the individual behavior of a WSN node in case of local information exchange.

Algorithm 1: WSN node program based on joint MOMDP policy π

```
1: NODEPRG(i, G_π = (V, E), v_0 = ((x^0, b^{Y0}), a^0))
2:   t ← 0
3:   while true
4:     execute [a^t]_i
5:     t ← t + 1
6:     observe o_i^t ∈ O_i and [x^t]_i
7a:    construct and broadcast Hello_i^t
7b:    wait to receive Hello_j^t of every j ∈ N\{i}
7c:    relay selected Hello_j^t (j ∈ N\{i})
8:     construct (x^t, o^t) from {Hello_j^t}_{j∈N\{i}} and ([x^t]_i, o_i^t)
9:     select v_t = ((x', *), a') from V
         where x' = x^t and (v_{t-1}, ⟨x^t, *, o^t, *⟩, v_t) ∈ E
10:    a^t ← a'
11: end-while
```

Algorithm 1 has 3 inputs **(#1)**: the identifier *i* of the WSN node in the MOMDP model; the G_π graph of a joint MOMDP policy π; and the root vertex of the graph, v_0. After initialization of time-period to *t=0* **(#2)** the algorithm goes into an infinite loop **(#3)**. Here at first action $[a^0]_i$ prescribed to node *i* by joint action a^0 at v_0 is executed **(#4)**. After the execution of $[a^0]_i$ time-period *t* is updated to *t=1* **(#5)**, and observation o_i^1 and the value of node *i*'s local, fully-observable variables, $[x^1]_i$ is received **(#6)**. Now all nodes should exchange their observations and the value of their local fully-observed state-variables in compact Hello-messages **(#7a-7c)**. Such a message of node *i* at the beginning of time period *t* can be the following.

$$Hello_i^t = \langle i, t, o_i^t, \{(j, \{m\})\}_{j \in \mathrm{N}\setminus\{\mathrm{sink}\}:\mathrm{data}_{m,i}=true \wedge j \in m} \rangle \qquad (5)$$

Here *i* is the identifier of the node; *t* is the ID of the new time period; o_i^t is the ID of the observation node *i* received after acting at *t-1*; this is followed by a set of $(j, \{m\})$ pairs, where *j* is a sensor node ID, and $\{m\}$ is a set of message IDs which are present at node *i* at the beginning of period *t* and which contain data generated at node *j*. If a message contains just the singular data generated at node *j*, then its ID should be *m=0*. What is not mentioned in the Hello-message of node *i*, is assumed not to be present at node *i*. The exchange of Hello-messages can be done via existing, efficient protocols, but in case of a larger network this may lead to a non-negligible time and energy overhead. In this case Algorithm 2 may be considered instead of Algorithm 1.

After a node received the most recent Hello-messages from every other node, it can exactly identify the v_t vertex in the joint policy graph, in which the network should be **(#8-9)**. During the selection of v_t, beliefs and probabilities are not playing a role, so they can be arbitrary (*), or for efficiency reasons they could even be omitted from G_π. The node can now drop all the received Hello-messages, and act according to a^0 in v_t **(#10, #4...)**. Since every node in the WSN acts according to Algorithm 1, they jointly realize π.

2) Decentralization of a joint policy for the nodes: If the mutual exchange of local information is not possible (e.g. the network is too large; or the real time-interval corresponding to a time period is so small, that the additional overhead is significant; or if generated data size is comparable to the size of Hello-messages), then we should enable nodes to act based only on their local information.

The problem with this is that if we would try to decompose the MOMDP model into $|N|$ separate MOMDPs for every node, and then solve these models separately, then in case of node *i*, for example, we couldn't overcome the need to specify the probabilities of other nodes' actions in node *i*'s MOMDP model, since those actions can influence node *i*'s observations and the value of its resulting local fully observable state-variables (the latter happens in WSNs because of data transmission from node to node, and the former may also happen if we include observation of neighbors' actions). Now other nodes' actions depend on the solution of their separate MOMDP models, which also need to incorporate the probabilities of other' actions, including those of node *i*. This would mean that node *i*'s solution depends on others' solutions which depend on its solution, which is an infinite regress. This is partly the reason why transition and observation independent DEC-POMDPs were proposed [13], but that assumption is not holding for WSNs as stated just before. So we could try to find a single MOMDP policy for every node, but each one can have different local fully observable variables, so the domain of

that single MOMDP policy may need to differ in case of different nodes, which is not possible.

Mainly these are the reasons why we chose in case 2) to solve the MOMDP model just as a POMDP, and then distribute this POMDP solution among the agents according to Algorithm 2.

Algorithm 2: Decentralization of a joint factored POMDP policy π

```
1: DECPOL(N,(S,{S_i}_{i∈N},{A_i}_{i∈N},{O_i}_{i∈N},T,Z,{R_i}_{i∈N},γ),b_0,π,t_max)
2:  t ← 0,  a^0 ← π(b_0)
3:  foreach i ∈ N
4:    B_i ← ∅,  B_i^0 ← {b_0},  v_0 ← (b_0,[a^0]_i),  V_i ← {v_0},  E_i ← ∅
5:    foreach a_i ∈ A_i\{[a^0]_i},  p(a_i^0 = a_i) ← 0,  end-for
6:    p(a_i^0 = [a^0]_i) ← 1
7:  end-for
8:  while (t < t_max and not ∀i: B_i^t ⊆ B_i)
9:   t ← t + 1
10:  foreach i ∈ N
11:    B_i^t ← ∅,  B_i ← B_i ∪ B_i^{t-1}
12:    foreach a_i ∈ A_i,  p(a_i^t = a_i) ← 0,  end-for
13:    foreach b ∈ B_i^{t-1}
14:     a^{t-1} ← π(b),  v_{t-1} ← (b,[a^{t-1}]_i)
15:     foreach o_i ∈ O_i and s_i^t ∈ S_i
16:      b_i^t ← τ(b,[a^{t-1}]_i,s_i^t,o_i),  B_i^t ← B_i^t ∪ {b_i^t},  a^t ← π(b_i^t)
17:      p(a_i^t = [a^t]_i) ← p(a_i^t = [a^t]_i) + p(o_i,s_i^t|b,[a^{t-1}]_i)
18:      v_t ← (b_i^t,[a^t]_i),  V_i ← V_i ∪ {v_t},  E_i ← E_i ∪ {(v_{t-1},s_i^t,o_i,v_t)}
19:     end-for
20:    end-for
21:  end-for
22: end-while
23: return {G_i = (V_i,E_i)}_{i∈N}
```

Algorithm 2 has 5 inputs **(#1)**: ***(i)*** the set N of WSN nodes; ***(ii)*** a factored POMDP having a factored set of actions $A = \prod_{i\in N} A_i$, a factored set of observations $O = \prod_{i\in N} O_i$ and a factored reward function $R(s,a) = \sum_{i\in N} R_i(s,a)$ for every $s \in S$ and $a \in A$. The possible values of those state-variables which are fully observable to node i, denoted by S_i, are also given. It should be noted that $S = \prod_{i\in N} S_i$ may not necessarily hold in general, i.e. we don't assume, that every state-variable is observable by one and only one agent, i.e. in $S = S_i \times S_{-i}$, S_{-i} is not necessarily the set of values of state-variables fully observable by agents $-i$ (other than i), but in general it is the set of values of state-variables not fully observable by agent i; ***(iii)*** an initial belief b_0; ***(iv)*** a POMDP policy π; and ***(v)*** a maximum number of iterations, t_{max}.

Initially the algorithm sets the time-period to *t=0*, and the joint action a^0 to the recommendation of joint policy π for initial belief b_0 **(#2)**. This is followed by further initialization steps for every node i **(#3-7)**, where the set of all previously reached beliefs of node i, B_i is initially empty; the set of i's possible beliefs at *t=0*, B_i^0 contains only b_0; the set of vertices V_i of node i's graph G_i contains only vertex $v_0 = (b_0,[a^0]_i)$, where $[a^0]_i$ is the action recommended by the joint action a^0 for node i; the set of directed edges E_i of G_i is empty; and the probability $p(a_i^0 = a_i)$ that node i will execute action a_i at *t=0* is set to zero for every $a_i \in A_i$ except for $a_i = [a^0]_i$, whose probability is set to 1, since it should be certainly executed at *t=0* by i in terms of joint policy π in b_0.

The main part of the algorithm is a `while`-cycle **(#8-22)**, where the stop-criteria is that either t should reach t_{max}, or the latest set of possible beliefs of node i, B_i^t should be completely contained in the set of previously reached beliefs, B_i, for every i. If the stop-criteria is satisfied, then a directed graph $G_i = (V_i, E_i)$ of node i's policy is returned for every i **(#23)**. Otherwise the `while`-cycle starts by incrementing t **(#9)**, and then executes a belief and action probability update and an expansion of G_i for every i **(#10-21)**. For this first B_i^t is initialized to be empty, and B_i^{t-1} is united with B_i **(#11)**. This is followed by setting the probability $p(a_i^t = a_i)$ of node i executing action a_i at time-period t initially to zero for every $a_i \in A_i$ **(#12)**. Now the algorithm takes every previous belief $b \in B_i^{t-1}$ of node i, and figures out what beliefs are reachable from b, with what probability, and what action follows from that according to π **(#13-20)**. So first we identify the joint action a^{t-1} which is recommended by π in b, and which also specifies node i's action $[a^{t-1}]_i$. We create a vertex $v_{t-1} = (b, [a^{t-1}]_i)$ from this belief-action pair **(#14)**. We now calculate all the new beliefs b_i^t of node i possibly resulting from executing $[a^{t-1}]_i$ at time *t-1*, the corresponding actions of node i, and the probability of these actions, which depend on the probability of node i's observations and the observed value of its fully observable variables S_i. Thus between lines **(#15-19)** we are examining the case of every possible $o_i \in O_i$ and $s_i^t \in S_i$, where s_i^t is a possible value of node i's fully observable variables. Assuming a given $o_i \in O_i$ observation and $s_i^t \in S_i$ value, first we calculate the belief $b_i^t = \tau(b, [a^{t-1}]_i, s_i^t, o_i)$ resulting from taking action $[a^{t-1}]_i$, at time *t-1* in belief b, and then observing o_i and s_i^t. This possible new belief, b_i^t is then added to the set of possible beliefs of node i at time t, B_i^t, and also a joint action a^t is derived from b_i^t according to π **(#16)**. The calculation of belief b_i^t is omitted due to its complexity.

In line **(#17)** we continue by incrementing the hitherto accumulated probability $p(a_i^t = [a^t]_i)$ that node i does $[a^t]_i$ at time t with the probability of observing o_i and s_i^t (given b and $[a^{t-1}]_i$), since this is a probability that we get belief b_i^t according to **(#16)**, which then determines $[a^t]_i = [\pi(b_i^t)]_i$. For complexity reasons we omit the calculation of probability $p(o_i, s_i^t | b, [a^{t-1}]_i)$. It is straightforward to derive.

If the probability $p(o_i, s_i^t | b, [a^{t-1}]_i)$ is not zero, then we create a vertex $v_t = (b_i^t, [a^t]_i)$ representing the belief-action pair we just reached, and add it to the set of vertices V_i of graph G_i, and also add an edge from v_{t-1} to v_t labeled with s_i^t and o_i **(#18)**, which means that if i is in v_{t-1} and does $[a^{t-1}]_i$ and then observes s_i^t and o_i, then it should go to v_t and execute action $[a^t]_i$ according to π, and so on.

Algorithm 2 returns a simple mixed-observability policy graph G_i for every WSN node i **(#23)** at a cost of complexity $\mathcal{O}(|N||OS_{max}|^{t_{max}})$, where $|OS_{max}| = \max_{i\in N}|O_i \times S_i|$. The construction of the policy graphs is done *offline*, but they are used by the nodes in runtime. It should be noted, that the constructed graphs are not equivalent to original joint policy π, rather they are just a joint approximation of it, which is based on the assumption, that the solution method that constructed π improved its V value for *every* belief $b \in B$. Eventually Algorithm 2 approximates the DEC-POMDP problem [13] in general sacrificing the guarantee of optimality for smaller complexity.

IV. EXPERIMENTS

In this section we report on our preliminary experiments with a simple 3-node WSN network (c.f. Fig. 5) with properties according to Section III/A. Naturally our approach is not restricted to such simple networks (it can actually work for larger WSNs with arbitrary topology), but optimization of larger networks is computationally more intensive, and the examination of results would also be more complex and thus less illustrative. The below network is just enough to demonstrate the concept in a necessary level of detail.

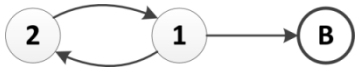

Figure 5. Simple WSN topology with $|N| = 3$ nodes (B is the sink node)

Experiments were conducted on a single PC with 4 GB of memory, a quad-core Intel i5 2.8 GHz CPU (but only 1 core was used for tests), and 32-bit Windows 7. We used the latest version (v0.95) of **APPL** (**A**pproximate **P**OMDP **PL**anning) open-source POMDP/MOMDP solver written in C++. It does an approximation of the optimal policy by sampling the optimally reachable beliefs. During this it maintains a lower $\underline{V}$, and upper $\overline{V}$ bound on the optimal value function V^*. The approximation stops when $\overline{V} - \underline{V}$ is below a given threshold. In our experiments this threshold was 10^{-3}.

APPL accepts factored POMDPs/MOMDPs described in XML (eXtensible Markup Language), and produces solution policies consisting of $\Gamma_Y(x)$ sets of α-vectors for each $x \in X$ and also policy graphs as described in Section II/C.

The MOMDP model of the WSN shown in Fig. 5 was created according to Section III/B. We defined 9 fully-observable $data_{i,j}$ variables (for representing the position of data generated at node 1, 2 and their aggregation 1+2 being at nodes 1, 2 and B), which are initially all *false* with a probability $p(data_{i,j} = false) = 1$, and 2 partially observable $event_i$ variables (representing event occurrence at sensors), which are *true* with a probability $p(event_i = true) = 0.1$. Since events are assumed to occur according to Poisson distribution, 0.1 is eventually the probability that $k \geq 1$ events occur during a time-period with $\lambda \approx 0.10536052$ events occurring expectedly per period. Correspondingly 2 observation variables O_i were defined for the 2 sensor nodes (to model their possible detection of event occurrence).

This was followed by 3 action variables A_i for each node with possible values according to Section III/B. The transition probabilities of state-variables ($data_{i,j}$ and $event_i$) were set in principle also according to Section III/B. Their concrete numerical values were mainly the following: probability p=1.0 was assigned by default to $data_{i,j}$ variables' value not changing except when sending, receiving, generating or aggregating. Successful sending had a probability of p=0.99. Receiving side could fail with p=0.01. Thus the probability of successful transmission was p=0.99^2=0.9801. The probability of the other 3 cases can be calculated similarly. The probability of successfully generating data in case of event occurrence was p=0.98. Processing and aggregation were successful with probability p=1.0 in case data was present at the node.

Observation function Z was as follows: the probability of not detecting an event when doing anything but listening was set to p=1.0. In case of listening the probability of detection was set to p=0.97 in case the event was occurring, and to p=0.05 in case it was not occurring (false-positive detection). The probabilities of the other 2 cases of not detecting the event can be calculated from these values.

Every node had its own reward function R_i as discussed in Section III/B. In particular for the sake of simplicity sending was assumed to draw 10 mA of energy, while receiving a bit more, 15 mA for a message of 512 bytes (we experimented with more realistic values also, e.g. 17.4 and 19.7 mA, but that didn't change the end results essentially). We assumed that the time needed for sending/receiving is proportionate with energy usage, and thus set the reward for successfully sending a 512 byte message to –10 and for receiving it, to –15. In case of the sink node we didn't consider the cost of receiving (c.f. Section III/B). The (negative) reward for listening and generating data was set relative to previous rewards, to –1 and –5 respectively. Aggregation had no cost, and thus no reward.

The reward of the sink node was zero except when it processed data (a motivation for the MOMDP solver to construct policies that eventually relay data to the sink). We experimented with several values for processing data. For example, when the reward for processing a single data at the sink was $\mathrm{R}(1) = 15.9$, a surprisingly simple policy graph emerged (with 1 vertex and 1 edge) suggesting to always be idle for all the nodes. The cause for this is that even in the best case the cost of node 1 getting its data to the sink would be (–1)+(–5)+(–10) = –16 (listening+generating+sending), while the reward for processing it would be 15.9, which would produce a negative total expected reward overall, which is worse than a reward of zero for being idle. This emphasizes that setting the reward values in the MOMDP model appropriately is of crucial importance. These rewards must reflect real investment and gain in the real WSN, and only then can the generated joint policy be considered really optimal. As a reference for finding these rewards we can e.g. approximate the investment of getting the data of the farthest sensor node in network to the sink on the least hops route.

On the other hand, a significantly more complex joint policy graph (with 36 vertices and 190 edges) emerged with $\mathrm{R}(1) = 100$ and $\mathrm{R}(2) = 1.5 \cdot \mathrm{R}(1) = 150$, where $\mathrm{R}(2)$ denotes the reward for processing aggregated data at the sink. This policy realized non-aggregating network behavior generating and routing data of node 1 and 2 to the sink nearly optimally.

With given reward configuration the near-optimal behavior of the network may include data aggregation. For this the reward for processing aggregated data must be above a given level. In our WSN scenario, if $2 \cdot \mathrm{R}(1) > \mathrm{R}(2)$ holds, that means that data accuracy is rewarded, while $2 \cdot \mathrm{R}(1) \leq \mathrm{R}(2)$ means that we neglect data accuracy. We did several test runs with the latter setting ($\mathrm{R}(1) = 100$ and $\mathrm{R}(2) = 200$). The result was that the generated joint policy now included aggregation (but only conditionally).

Fig. 6 shows a near-optimal solution of a scenario, where $R(1) = 47$ and $R(2) = 1.5 \cdot R(1) = 70.5$, which is enough to motivate the network to manifest node 1's data at the sink (for a best case reward of –16+47=**31**), but it is not enough to compensate for also manifesting node 2's data at the sink in any way (and thus node 2 is idle, reserving overall energy). If R(1) is set below a given level, then it becomes not worth for given nodes even to generate their data (or to listen), not speaking of routing it toward the sink. In our case node 1 goes idle around $R(1) \approx 16$ as discussed before.

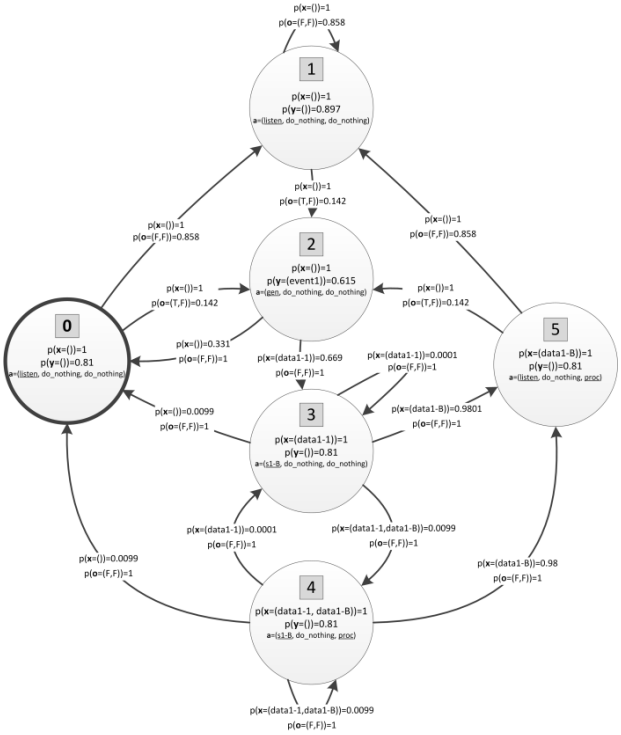

Figure 6. Policy graph G_π ($|V_\pi| = 6, |E_\pi| = 16$) of a near-optimal joint policy π in case of $R(1) = 47$ and $R(2) = 70{,}5$.

The near-optimal joint policy shown in Fig. 6 has the following logic: it starts in vertex 0, where the fully observed state is **x**=(), meaning that no generated data is anywhere, while the belief over the partially observed state associates a probability of p=0.9^2=0.81 to **y**=(), which means that neither sensors' event is occurring with this probability. That is just according to the initial state of our MOMDP model. In this belief-state the joint policy prescribes joint action **a**=(*listen, do_nothing, do_nothing*) to node 1, 2 and B respectively. After the joint execution of **a** the new belief-state may be either vertex 1 or vertex 2 depending on which observation did node 1 receive after listening: F=False (with probability p=0.858) or T=True (with probability p=0. 142). The fully observable state does not change by executing joint action **a**.

If the observation of node 1 is F, then we should continue in vertex 1, where the same joint action is recommended again (and again in case of failure). In case of success (if node 1 finally detects the an event while listening) we go to vertex 2, where the state is still **x**=(), but the probability of the occurrence of an event at node 1 (while not occurring at node 2) has a probability of p=0.615, which is the highest achievable value. So in this case of this belief the policy recommends to node 1 to generate data while other nodes should still be idle (*do_nothing*). Data generation may fail with a probability p=0.331, so we would be routed back to vertex 0 from vertex 2. Otherwise it is successful (the data appears at node 1), and so we go to vertex 3, where node 1 should try to send its generated data to the sink node according to the policy. Four outcomes of sending this data are possible: ***(i)*** the transmission is successful with probability p=0.9801 [go to vertex 5]; ***(ii)*** data is really lost with probability p=0.0001 [stay in vertex 3]; ***(iii)*** data is not lost, but node 1 doesn't receive the ACK from node B, and so it isn't dropping the data, with probability p=0.0099 [go to vertex 4]; or ***(iv)*** data is lost because of some failure at node B about which node 1 is not notified, with probability p=0.0099 [go to vertex 0]. Staying in vertex 3 means a necessary re-transmission of data. Going to vertex 4 implies an unnecessary re-transmission of data (while the sink should already process the received data). Going to vertex 5 is the normal case with probability p=0.98: here node 1 should start to listen again, while the sink should process its received data. Overall this joint policy in Fig. 6 nearly maximizes the total expected reward of the WSN in Fig. 5.

To evaluate the effectiveness of the MOMDP policies generated e.g. for the above 3 highlighted cases, we made a comparison with Dynamic Source Routing (DSR) [16], which is equivalent to optimal static routing (best benchmark) in case of the simple WSN in Fig. 5, since there is only one acyclic route from each sensor to the sink. At MAC level nodes used CSMA/CA. They listened until event detection, and then tried to generate data. In case of no aggregation they tried to relay available (successfully received/generated) data instantly toward the sink according to CSMA/CA. In case of full aggregation node 1 waited for node 2's data to arrive, aggregated it with its own (if it was available, otherwise it listened and generated it), and then the aggregated data was sent to the sink, which processed it instantly. Table I summarizes the results of this evaluation.

TABLE I. QUANTITATIVE COMPARISON OF NETWORK PERFORMANCE

		CSMA/CA + Optimal routing (DSR)						MOMDP		
		No aggregation			Full aggregation					
		ΣE	Bpp	ΣR	ΣE	Bpp	ΣR	ΣE	Bpp	ΣR
R1=47, R2=70.5	0.5 KB	58425 (±2151)	69.71 (±3.54)	**5560** (±1768)	49895 (±1697)	23.40 (±1.88)	**-17675** (±1885)	26612 (±819)	39.52 (±3.41)	**9660** (±2353)
	2 KB	187990 (±8071)	277.76 (±14.34)	**66971** (±7366)	153440 (±7781)	89.09 (±6.71)	**-23748** (±9181)	63254 (±2850)	158.09 (±11.17)	**81861** (±7838)
R1=100, R2=150	0.5 KB	58427 (±1745)	69.65 (±3.41)	**77597** (±5198)	50068 (±1332)	23.52 (±2.18)	**18844** (±5978)	63468 (±1926)	71.94 (±3.92)	**77026** (±6272)
	2 KB	188460 (±9897)	278.39 (±19.2)	**355260** (±28806)	153700 (±5935)	94.24 (±7.89)	**122380** (±20282)	190590 (±8071)	288.05 (±16.49)	**371940** (±27174)
R1=100, R2=200	0.5 KB	58372 (±1972)	69.66 (±3.28)	**77670** (±5363)	49936 (±1440)	23.33 (±1.85)	**41174** (±6532)	62653 (±1889)	66.43 (±3.15)	**78212** (±4965)
	2 KB	188280 (±7462)	278.38 (±12.38)	**355360** (±19121)	153210 (±7354)	93.82 (±8.32)	**213210** (±27084)	187560 (±6920)	266.05 (±12.21)	**377380** (±21155)

Table I shows the average of total energy use (∑E [mA]), *throughput* (Bytes per period: **Bpp**, e.g. Bps) and *total reward* (∑**R**, corresponding to network performance metric) of different policies in different cases, each value gained from 100 runs each 10000 time-periods long. Mainly DSR with no/full-aggregation was compared against near-optimal MOMDP policies generated for the 3 above highlighted cases, in each case with a data size of 0.5 and 2 KB (in the 2 KB case energy for sending/receiving and thus reward R(1) and R(2) were 4-times higher proportionally). This means 3*3*2=18 cases, each with 3 numeric values (∑E, Bpp, ∑R). The standard deviation of each value is also included in the table (as a result of the stochastic environment). Simulations were implemented in MATLAB R2011a, and show that MOMDP performed best in 5 out of 6 cases (3*2=6 rows) in terms of performance metric (∑R) which we were seeking to maximize. The best value for every case in each category is underlined. The only case, where MOMDP's ∑R was worse than optimal routing (by 0.7%: effectively equivalent) was due to local information exchange overhead (Hello messages). Without overhead it would be better even then. This overhead becomes less significant for larger data (e.g. 2 KB) and in larger networks the near-optimality of MOMDP should compensate this even more compared to e.g. DSR.

The time for generating MOMDP policies can be divided in 3 parts: ***(i)*** parsing the MOMDP model took around 14s in average; ***(ii)*** the initialization of the algorithm took 30 minutes in average; and finally ***(iii)*** the approximation took only around 1-1.5 seconds. Thus the initialization phase of the algorithm consumed most of the time. Nonetheless these results are currently state-of-the-art considering that the problem solved has 2^{11} states, indicating that with further optimization, by exploiting the parallelization/distribution possibilities of MOMDP solution algorithms (e.g. even by taking advantage of special hardware), by specializing the algorithm for the WSN domain, by allowing longer runtimes (on stronger, dedicated hardware) and/or by integrating recently successful online solution principles [11] WSNs larger by orders of magnitude can be optimized in practice.

V. CONCLUSIONS

A generic framework was presented for overall network performance optimization in WSNs based on offline approximation of optimal joint MOMDP policies. A realistic MOMDP model of WSNs was given, and two algorithms to distribute its solution among individual nodes depending on whether local information exchange is allowed. Resulting node-policies optimize overall WSN behavior according to a user-defined performance metric (e.g. finding the best tradeoff between total energy usage, time delay and data accuracy). The framework was demonstrated in case of a smaller WSN in detail by generating and evaluating its near-optimal joint policies in several cases and comparing them against CSMA/CA with optimal routing (DSR). MOMDP performed best in 5 out of 6 cases by increasing network performance with more than 20% in average.

Further research should focus mainly on scaling up the method to cope with larger networks e.g. by exploiting parallelization/distribution; by specializing the method for the WSN domain; and/or by integrating it with online MOMDP solution principles. The overhead of information exchange could be reduced by narrowing its horizon from the whole network to nodes' relevant neighborhood. The WSN model could be improved by allowing action duration; by adding observations of neighboring nodes' actions; or by merging listening and data generation into one action. Clock synchronization issues and a possible change of model-level assumptions in runtime could be considered. Also a deeper investigation of decentralization of joint policies could be performed and the approach should be tested in real WSNs.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Computer Networks, vol. 38, Mar. 2002, pp. 393-422.

[2] S. D. Glaser, "Some real-world applications of wireless sensor nodes," Proc. SPIE Symposium on Smart Structures & Materials (NDE 2004), SPIE Press, Mar. 2004, pp. 14-18, doi:10.1.1.129.2109.

[3] P. Sun, X. Zhang, Z. Dong, and Y. Zhang, "A novel energy efficient wireless sensor MAC protocol," Proc. Fourth International Conference on Networked Computing and Advanced Information Management, NCM, Sep. 2008, pp. 68-72.

[4] B. Yin, H. Shi and Y. Shang, "A two-level strategy for topology control in wireless sensor networks," Int. Journal of Wireless and Mobile Computing, vol. 4, no. 1, 2010, pp. 41-49.

[5] A. Sharif, V. M. Potdar, and A. J. D. Rathnayaka, "LCART: Lightweight Congestion Aware Reliable Transport protocol for WSN targeting heterogeneous traffic," Australian Journal of Intelligent Information Processing Systems, vol. 12, Nov. 2010, pp. 1-9.

[6] T. Anker, D. Bickson, D. Dolev and B. Hod, "Efficient clustering for improving network performance in wireless sensor networks," Proc. 5th European Conference on Wireless sensor networks (EWSN'08), Springer-Verlag, 2008, pp. 221-236.

[7] R. Srivastava and C. E. Koksal, "Energy optimal transmission scheduling in wireless sensor networks," IEEE Transactions on Wireless Communications, vol. 9, May. 2010, pp. 1550-1560.

[8] W. Li, M. Bandai, and T. Watanabe, "Tradeoffs among delay, energy and accuracy of partial data aggregation in wireless sensor networks," Proc. International Conference on Advanced Information Networking and Applications (AINA2010), IEEE Press, Apr. 2010, pp. 917-924.

[9] X. Fei, A. Boukerche and F.R. Yu, "A POMDP based K-coverage dynamic scheduling protocol for wireless sensor networks," Proc. IEEE Global Telecommunications Conf., IEEE, Dec. 2010, pp. 1-5.

[10] S. Chobsri, W. Sumalai, and W. Usaha, "A parametric POMDP framework for efficient data acquisition in error prone wireless sensor networks," Proc. 4th International Symposium on Wireless Pervasive Computing (ISWPC 2009), IEEE Press, Feb. 2009, pp. 1-5.

[11] J. Veness, K. S. Ng, M. Hutter, W. Uther and D. Silver, "A Monte-Carlo AIXI approximation," Journal of Artificial Intelligence Research, vol. 40, Jan. 2011, pp. 95-142.

[12] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under uncertainty for robotic tasks with mixed observability," Int. Journal of Robotics Research, vol. 29, Jul. 2010, pp. 1053-1068.

[13] D. S. Bernstein, R. Givan, N. Immerman and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," Mathematics of Operations Research, vol. 27, 2002, pp. 819-840.

[14] M. L. Puterman, Markov decision processes. John Wiley, 1994.

[15] J. Pineau, G. Gordon and S. Thrun, "Anytime point-based approximations for large POMDPs," Journal of Artificial Intelligence Research, vol. 27, Nov. 2006, pp. 335-380.

[16] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in Mobile Computing, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996, pp. 153–181.