Decentralized Learning in Wireless Sensor Networks

Mihail Mihaylov[1], Karl Tuyls[2], and Ann Nowé[1]

[1] Vrije Universiteit Brussel, Brussels, Belgium
{mike,ann}@como.vub.ac.be
[2] Maastricht University, Maastricht, The Netherlands
ktuyls@gmail.com

Abstract. In this work we present a reinforcement learning algorithm that aims to increase the autonomous lifetime of a Wireless Sensor Network (WSN) and decrease its latency in a decentralized manner. WSNs are collections of sensor nodes that gather environmental data, where the main challenges are the limited power supply of nodes and the need for decentralized control. To overcome these challenges, we make each sensor node adopt an algorithm to optimize the efficiency of a small group of surrounding nodes, so that in the end the performance of the whole system is improved. We compare our approach to conventional ad-hoc networks of different sizes and show that nodes in WSNs are able to develop an energy saving behaviour on their own and significantly reduce network latency, when using our reinforcement learning algorithm.

1 Introduction

An increasingly popular approach for environmental and habitat monitoring is the use of Wireless Sensor Networks (WSNs) [Car04, Rog06, Yic08]. The nodes in such a, WSN are limited in power, processing and communication capabilities, which requires that they optimize their activities, in order to extend the autonomous lifetime of the network and minimize latency. A complicating factor is communication, because some nodes can fall outside the transmission range of the base station, or can belong to different stakeholders, serving various purposes, thus rendering the common centralized approach inapplicable for large networks.

This article extends the work done in [Mih08] to a random network topology, reduces the communication overhead and significantly improves the results. In this work we use a reinforcement learning algorithm to optimize the energy efficiency of a WSN and reduce its latency in a decentralized manner. We achieve that by making nodes (hereby regarded as agents) develop energy-saving schemes by themselves without a central mediator. In many cases such a central authority is undesirable or simply impossible to implement [Rog06]. The idea behind our approach is that agents learn, by themselves, to reduce the negative effect of their actions on other agents in the system, based on a certain reward function. We investigate the performance of our algorithm in two networks of different

M.E. Taylor and K. Tuyls (Eds.): ALA 2009, LNAI 5924, pp. 60–73, 2010.

sizes. We show that when agents learn to optimize their behaviour, they can increase the energy efficiency of the system and significantly decrease its latency with minimal communication overhead.

The outline of this chapter is as follows: Section 2 presents the background of our approach by describing the basics of a wireless sensor network and the MAC communication protocol. Section 3 describes the idea behind our algorithm and its application to the energy efficiency optimization of nodes. In Section 4 we explain the experiments and discuss our findings. Lastly, Section 5 presents our conclusions from this research and suggests some areas for improvement in the future.

2 Background

In this section we describe the basics of a Wireless Sensor Network and the MAC communication protocol. Subsection 2.1 elaborates on WSNs and Subsections 2.2 and 2.3 explain the working of the MAC protocol and the way nodes communicate. Lastly, Subsection 2.4 presents some related work in this field.

2.1 Wireless Sensor Networks

A Wireless Sensor Network is a collection of densely deployed autonomous devices, called sensor nodes, that gather environmental data with the help of sensors. The untethered nodes use radio communication to transmit sensor measurements to a terminal node, called the sink. The sink is the access point of the observer, who is able to process the distributed measurements and obtain useful information about the monitored environment. Sensor nodes communicate over a wireless medium, by using a multi-hop communication protocol that allows data packets to be forwarded by neighbouring nodes to the sink. This concept is illustrated in Figure 1. The environmental or habitat monitoring is usually done over a long period of time, taking into account the latency requirements of the observer.

The WSN can vary in size and topology, according to the purpose it serves. The sensor network is assumed to be homogeneous where nodes share a common communication medium (e.g. air, water, etc.). We further assume that the communication range is equal in size and strength for all nodes. They have a single

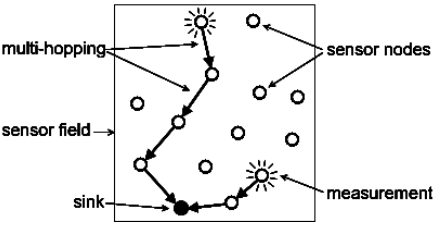


Fig. 1. Wireless Sensor Network

omnidirectional antenna that can only *broadcast* a message, delivering it to all nodes in range. In our network, sensor nodes can neither vary their transmission power, nor are they able to estimate their distance from the transmitting node by measuring the signal strength – such features are not generally available in sensor nodes and therefore are not considered here. The motivation to use such simple devices is to reduce the overall cost of nodes and to keep our solution applicable to the most general sensor network.

In this work we show that the selfish and computationally bounded agents can optimize their own performance, in a decentralized manner, in order to reduce both their own energy consumption and the latency of the network. We assume that communication between the agents is limited and that central control is not possible. We further require that the communication protocol considers not only energy efficiency, but also scalability and fault tolerance, so that our approach is able to adapt to a dynamic topology, where nodes may move, fail or new nodes may be added to the system. The communication protocol, therefore, constitutes an important part of the WSN design.

2.2 The MAC Protocol

The Medium Access Control (MAC) protocol is a data communication protocol, concerned with sharing the wireless transmission medium among the network nodes. Typical MAC protocols, used by ad-hoc networks, cannot be applied to WSNs, due to a number of differences between the two types of networks. Some differences include the large number and density of sensor nodes in a WSN, compared to the nodes in ad-hoc networks; the frequently changing topology of sensor nodes and their power constraints, etc.

We use a simple asynchronous MAC protocol that divides the time into small discrete units, called frames. Each node independently determines its sleep duration (or schedule), i.e. the amount of time in a frame that the node's antenna will be turned off. During that time the agent is not able to communicate with other nodes and therefore saves energy. Nevertheless, the agent continues its sensing and processing tasks. Our protocol allows nodes to synchronize their schedules prior to communication and thus avoid collisions and overhearing – typical sources of energy waste.

Since communication is the most energy expensive action [Dam03], it is clear that in order to save more energy, a node should sleep more. However, when sleeping, the node is not able to send or receive any messages, therefore it increases the latency of the network, i.e., the time it takes for messages to reach the sink. On the other hand, a node does not need to listen to the channel when no messages are being sent, since it loses energy in vain. As a result, nodes should learn on their own the number of time slots they should spend sleeping within a frame. For example, nodes far away from the sink may learn to sleep more, since they will have fewer messages to forward, while nodes close to the sink should learn to listen more, because the workload near the sink is usually heavier. Learning to optimize nodes' own schedules will ensure good energy efficiency of the network, while minimizing the latency. The MAC protocol should therefore support the exchange of

additional information, necessary for the algorithm for optimization. It is clear that the amount of this information within message packets should be kept as little as possible, in order to minimize the energy waste by control packet overhead. A brief description of the communication protocol is presented next.

2.3 Communication and Routing

When the WSN is deployed, nodes first need to determine their hop distance to the sink, i.e. the minimum number of nodes that will have to forward their packets. This is achieved by broadcasting SYNchronization (SYN) packets in the following way: the sink broadcasts a SYN packet, containing a counter, initially set to 0; all receivers set their hop equal to the counter, increment it and broadcast the new SYN packet further on, with a small random delay to avoid collisions. For example, a node right next to the sink will receive a SYN packet with `hop=0` and will broadcast a new one with `hop=1`.

When a node has a message to send[1], it broadcasts a Request To Send (RTS) packet to all nodes within range, which we call neighbours (or neighbouring nodes). All neighbours at an equal or higher hop simply go to sleep, since they do not need to forward the sender's message. All lower-hop neighbours wait a small random amount of time before replying with a Clear To Send (CTS) packet. Once one node broadcasts a CTS packet, all its neighbours go to sleep, except the sender of the RTS, who in turn broadcasts the actual data. In other words, all immediate neighbours of the two communication partners are sleeping during the broadcast of the data, in order to avoid collisions and overhearing. Once the receiver obtains the data packet, it replies with an ACKnowledgment (ACK) and thus the communication is over.

2.4 Related Work

A good literature survey on various aspects of Wireless Sensor Networks is presented in [Yic08]. The authors of [Ai04] present a method that adapts the duty cycle scheme of nodes to the traffic pattern of the network in order to minimize latency and maximize throughput. They, however, use a synchronous protocol (AC-MAC). In [Bar05] a new MAC protocol is presented (μ-MAC) with high sleep ratios while keeping latency and reliability at acceptable levels. This protocol, however, uses synchronous schedules, relies on relatively static environment and requires prior knowledge of the traffic pattern. An asynchronous protocol (X-MAC) is presented in [Bue06] that optimizes the energy efficiency of nodes, but not the latency of the network. A decentralized coordination protocol is presented in [Far08], that maximizes the social welfare within a group of interacting agents through local message passing. This approach, however, comes at a higher communication cost. A somewhat different application of decentralized learning is presented in [Jai09], where the authors use distributed constraint optimization techniques to make mobile wireless sensors maximize the sum of signal strengths on all network links over time.

[1] We assume that all messages are forwarded toward the sink.

3 Learning Algorithm

Besides on its hardware, the energy consumption of a node is also dependent on its position in the WSN. Nodes, closer to the sink have to forward more messages and therefore need to listen more, while those far away from the sink could spend more time sleeping. For this reason, the behaviour of agents cannot be the same for all (e.g. all listen and sleep the same amount of time in a frame). Each node needs to *learn* what behaviour is energy efficient in the network. To achieve that, we make nodes adopt an algorithm for optimization in order to improve the performance of the whole system.

Each agent in the WSN uses a reinforcement learning (RL) algorithm to learn an optimal schedule (i.e. sleep duration in a frame) that will maximize the energy efficiency and minimize the latency of the system in a decentralized manner. The main challenge in such a decentralized approach is to define a suitable reward function for the individual agents that will lead to an effective emergent behaviour as a group. Another challenge is that agents in a WSN can obtain only local information from surrounding nodes, due to their small transmission range. To tackle these challenges, we proceed with the definition of the basic components of the reinforcement learning algorithm.

3.1 Actions

The actions of each agent are restricted to selecting a sleep duration for a frame. The action space consists of a discrete number of sleep durations at equal increments within one frame length. Defining the size of the increment constitutes a tradeoff, since a rather large value will result in only few actions for the agent to choose. On the other hand, a small increment will result in a large action set, which makes it difficult for the algorithm to converge [Len08]. Agents choose their actions according to a probability distribution and use that action for a certain number of frames, which we call a frame window. The reason for using an action for more than one frame is that the agent will thus have enough time to experience the effect of that action on the system. The size of the frame window and the discretization increment will be discussed in Section 4.1.

3.2 Rewards

Before proceeding with the formulation of the reward signal, we first need to define what Energy Efficiency (EE) of a single agent is.

Energy Efficiency. We consider an agent to be energy efficient when it minimizes most of the major sources of energy waste in WSN communication – idle listening, overhearing and unsuccessful transmissions, while quickly forwarding any packets in its queue to ensure low network latency. Formally, the energy efficiency for agent i in frame f is:

$$\begin{aligned} EE_{i,f} = \quad & \alpha(1 - IL_{i,f}) + \beta(1 - OH_{i,f}) + \\ & + \gamma(1 - UT_{i,f}) + \delta(1 - DQ_{i,f}) + \epsilon BL_i \end{aligned}$$

where:

- $IL_{i,f}$ is the duration of idle listening of agent i within frame f;
- $OH_{i,f}$ is the duration of overhearing of agent i within frame f;
- $UT_{i,f}$ is the amount of unsuccessful transmissions of agent i within frame f;
- $DQ_{i,f}$ is the sum of the durations that each packet spent in the queue of agent i within frame f;
- BL_i is the remaining battery life of agent i;
- the constants α, β, γ, δ and ϵ weight the different terms accordingly.

All values are in the unit interval.

It is easy to show that if agents try to increase simply their own energy efficiency, they will prefer to sleep until they obtain a measurement (thus minimizing energy waste) and then wake up only to broadcast it (to ensure low latency). That will not lead to high global efficiency, due to the high number of collisions and unsuccessful transmissions that nodes will experience. Therefore, individual agents should also consider other agents in the system when optimizing their own behaviour. A similar approach was undertaken by Wolpert and Tumer in [Wol08], where they apply their Collective Intelligence framework to align the selfish agents' goals with the system goal.

Effect Set. Our hypothesis is that if each agent "cares about others" that will improve the performance of the whole system. To achieve that, we introduce the concept of an Effect Set (ES) of a node, which is the subset of that node's neighbourhood, with which it communicates within a frame window. In other words, the ES of agent i is the set N_i of nodes, whose messages agent i (over)hears within a frame window. Thus, the energy efficiency of agent i is directly dependent on the actions of all agents in N_i and vice versa.

Effect Set Energy Efficiency. As a result of the influence of agents on each other's performance, we form our hypothesis: if each agent seeks to increase not only its own efficiency, but also the efficiency of its ES, this will lead to higher energy efficiency of the whole system. For this reason, we set the reward signal of each agent to be equal to its mean Effect Set Energy Efficiency (ESEE) over a frame window of size $|F|$. We define the ESEE of agent i in the frame window F as

$$\mathrm{ESEE}_{i,F} = \frac{1}{|F|} \cdot \sum_{f}^{F} \frac{\mathrm{EE}_{i,f} + \sum_j \mathrm{EE}_{j,f}}{|N_i| + 1} \qquad \forall j \in N_i$$

where $EE_{i,f}$ is the energy efficiency of agent i in frame f and $|N_i|$ is the number of agents in the effect set of agent i. In other words, the reward signal that each agent receives at the end of each frame window is the mean energy efficiency of its effect set and of itself, averaged over the size of the frame window. Thus, agents will try to increase the value of their ESEE by optimizing their own behaviour.

Challenge. One challenge in our reward signal is that nodes cannot compute their ESEE directly, because to do so, they would have to obtain the efficiency

of each agent in N_i. To achieve that, nodes simply include the value of their own EE in the three control packets – RTS, CTS and ACK, so that neighbouring agents can (over)hear these values and compute their ESEE. This is the only information that nodes need to exchange for our algorithm to work. Although including additional information in control packets is expensive, we will show that the network performs still better than one without learning. We will now show how each agent can learn to optimize its ESEE.

3.3 Update Rule

At the end of each frame window, agents compute the average ESEE from the past frames and use this value to learn the best sleep duration that will maximize efficiency and minimize latency. Agents use the update rules of a classical learning automaton to update their action probabilities. More specifically, after executing action x in every frame of F, its probability $p_i(x)$ is updated in the following way

$$p_i(x) \leftarrow p_i(x) + \lambda \cdot ESEE_{i,F} \cdot (1.0 - p_i(x))$$

where λ is a user-defined learning rate. The probability $p_i(y)$ for all other actions $y \neq x$ in the action set of agent i then becomes

$$p_i(y) \leftarrow p_i(y) - \lambda \cdot ESEE_{i,F} \cdot p_i(y) \qquad \forall y \neq x$$

At the beginning of each frame, agents select their actions according to the updated probabilities and execute them in that frame window. As a result, the learning process is done on-line – the algorithm adapts to the topology of the network and the traffic pattern, which typically cannot be known in advance in order to train nodes off-line.

3.4 Discussion

Although a comparison with other existing learning approaches would provide good insights into coping with decentralized learning, a comparative study is not the topic of this paper. Rather, our aim is to show whether an efficient decentralized learning *can* be achieved by selfish and computationally bounded agents. The motivation behind the selection of our learning approach is twofold. Firstly, learning automata has very appealing theoretical guarantees for convergence [Tha04] and has been successfully applied in different multi-agent settings [Ver04, Tuy06, Vra08]. In other words, the latter property of learning automata suggests that the policy of our learning agents, as a team, will converge to an equilibrium. Secondly, this policy iteration technique uses an implicit exploration strategy that is "built-in" the algorithm itself, i.e. it does not require an additional aspect that needs to be tuned.

4 Results

4.1 Experimental Setup

We applied our algorithm on two networks of random topology and different sizes – one small network with 10 nodes and a large one with 50 nodes. The density of

both networks was the same, i.e. on average each node had 4 neighbours, because we found out empirically that it influences the speed of learning. In this work we focus on how well learning scales in terms of the number of nodes, rather than in terms of the density. The reason for the slower learning in more dense networks is the higher degree of interdependence of the actions of neighbouring agents. In other words, agents in dense networks have to consider more neighbours in optimizing the performance of their ESEE and thus converge to an optimal action slower than agents in less dense networks. An in-depth study of the optimal density of sensor networks is presented in [Ess08].

We considered networks of random topology, rather than organized in a grid structure (as in [Mih08]), so that the WSN can be deployed more freely (e.g. nodes can be scattered from a moving vehicle). The synchronization phase of the network was set to 20 seconds – this duration was enough for all nodes to find their hop distance to the sink in both networks. During this phase, agents do not learn to optimize their behaviour, since the resulting traffic pattern is independent of that from the actual data. We set the duration of a frame to 0.5 seconds and the message rate – to 1 sensor measurement in a frame on average. We chose this high message rate to make the effect of agents' actions more apparent and to give agents enough information in order to learn a good policy. A sufficient frame window size was found to be 4, i.e. agents repeat their selected action for 4 times, before obtaining a reward signal. The discretization coefficient (Subsection 3.1) was selected such that it results in 11 different actions (or sleep durations). The five terms in the computation of the EE (Subsection 3.2) are from a network perspective interdependent (yet not redundant) and their weighting coefficients are up to the network designer to set. We carried out an exhaustive brute-force search to determine a set of values that is most suitable for our network settings. Thus, the five weighting coefficients were empirically chosen in the following way: $\alpha = 0.2$, $\beta = 0.3$, $\gamma = 0.1$, $\delta = 0.3$ and $\epsilon = 0.1$. The same brute-force search was carried out to determine the best learning rate. Small coefficient results in a relatively slow learning process, while large learning rates make the network unstable. Thus, the best learning rate λ was found to be 0.280 for the small network and 0.299 for the large one, where in both cases the initial action probability was uniform. Finally, the networks were allowed to run for 500 seconds, i.e. 1000 frames, before the simulation was terminated. This duration was sufficient to produce valuable results.

4.2 Experiments

As stated above, we evaluated our algorithm on two random topology networks of the same density, but of different sizes. We compared the performance of each setting to a network of the same size where agents do not optimize their behaviour, but rather all sleep the same pre-defined amount of time, as it is usually done in practice [Mar04]. In each experiment we measured six performance criteria:

1. Average **remaining battery** at the end of the simulation (i.e. after 1000 frames). This value shows what the battery levels of nodes will be after 500 seconds of runtime with the selected settings.

2. **Standard deviation of the average remaining battery** – indicates the difference between the most and the least efficient nodes. Here a small deviation is desirable, since it signifies a rather equal dissipation of energy over time.
3. **Average latency** of the network over all packets delivered to the sink. This criterion measures the average time a message takes from the moment it was generated to the time it reaches the sink.
4. **Standard deviation of the average latency** of the network. Again, a small deviation is preferable, because it signifies consistent traffic latency.
5. **Maximum latency** of the network, i.e. the latency of the packet that took the most time to be delivered to the sink. This value indicates the worst case scenario for the latency that the user of the WSN can experience for a packet.
6. Number of **received packets** by the sink within 500 seconds. This is an inverse indication of latency and it shows how many messages actually reached the sink during the simulation runtime.

The sleep duration of the two non-learning networks was selected such that it maximizes the above six performance criteria. The same technique was used to select the best learning rate of the networks with optimization. In other words we compared the optimal "non-learning" system to the optimal one *with* learning. This comparison is displayed in Figure 2. The first column shows the above six performance criteria, where the last two rows indicate the average sleeping time of the agents and the standard deviation. The second column indicates the objective (*obj.*) of the corresponding performance criterion – whether it should be maximized (*max*) or minimized (*min*). The third and forth column display the results from our experiments when agents are not learning and when they are learning, respectively. The column labeled *improvement* displays the percentage increase of the six performance measures when agents adopt our learning algorithm.[2]

As it can be seen from Figure 2, in both cases our learning agents sleep on average less than those in the non-learning network. One would expect that less sleeping results in lower battery level, due to idle listening and overhearing, and higher latency, due to collisions. However, our learning algorithm aims to reduce precisely those sources of energy waste, by making nodes optimize their behaviour, based on the actions of neighbouring nodes. Thus, agents learn to avoid "harming" other agents by adapting to the traffic pattern and therefore learning the optimal sleep duration in their neighbourhood. In other words, agents learn to sleep when their neighbours communicate (so as to avoid overhearing); stay awake enough to forward messages quickly (and thus decrease latency); and yet sleep enough (to ensure longer network lifetime). Figure 3 shows agents' actions (sleep durations) (vertical axis) over time (horizontal axis). Each dot represents that agent's selected action at the corresponding time in the simulation. The graph indicates that in the small network agents learn, as the time progresses, to sleep less and listen more, so that they reduce the latency of the network,

[2] The concept of "improvement" is not applicable to the last two rows.

Small Network (10 nodes)

performance criteria	obj	non-learning	learning	improvement
End battery - mean (%)	max	23.283	25.706	10.4% (increased)
End battery - std. dev. (%)	min	4.514	2.220	50.8% (decreased)
Latency - mean (sec.)	min	11.413	3.937	65.5% (decreased)
Latency - std. dev. (sec.)	min	8.455	3.348	60.4% (decreased)
Latency - max (sec.)	min	62.359	18.975	69.6% (decreased)
Packets arrived at Sink	max	2007	2167	8.0% (increased)
Sleeping time - mean (sec.)	n/a	0.120	0.094	n/a
Sleeping time - std. dev. (sec.)	n/a	0.000	0.136	n/a

Large Network (50 nodes)

performance criteria	obj	non-learning	learning	improvement
End battery - mean (%)	max	22.375	22.789	1.9% (increased)
End battery - std. dev. (%)	min	4.362	5.251	20.4% (increased)
Latency - mean (sec.)	min	20.552	5.823	71.7% (decreased)
Latency - std. dev. (sec.)	min	14.768	5.850	60.4% (decreased)
Latency - max (sec.)	min	88.669	50.892	42.6% (decreased)
Packets arrived at Sink	max	544	2296	322.1% (increased)
Sleeping time - mean (sec.)	n/a	0.220	0.166	n/a
Sleeping time - std. dev. (sec.)	n/a	0.000	0.176	n/a

Fig. 2. Comparison between non-learning and learning in the small and large networks

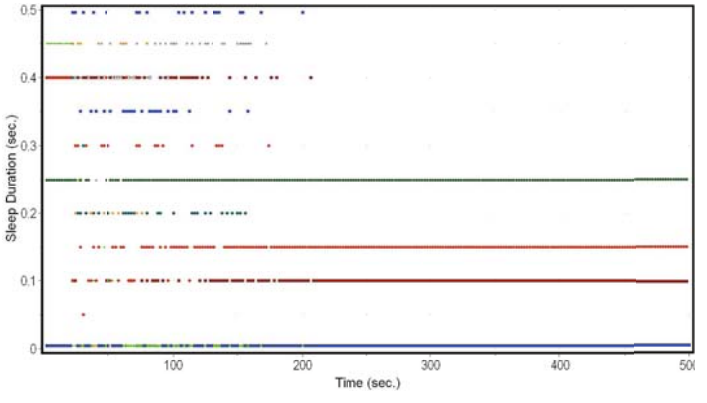
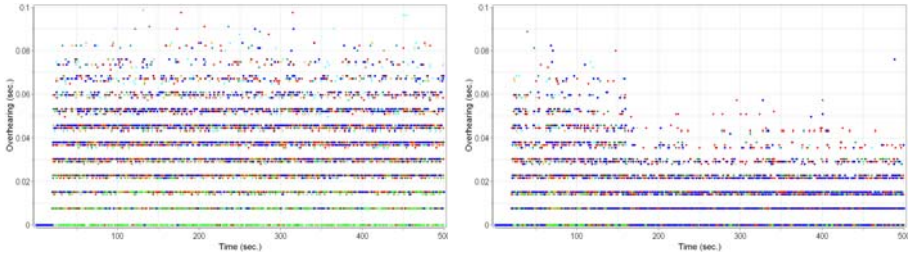


Fig. 3. Sleep Duration over Time when learning, Small Network (10 nodes)

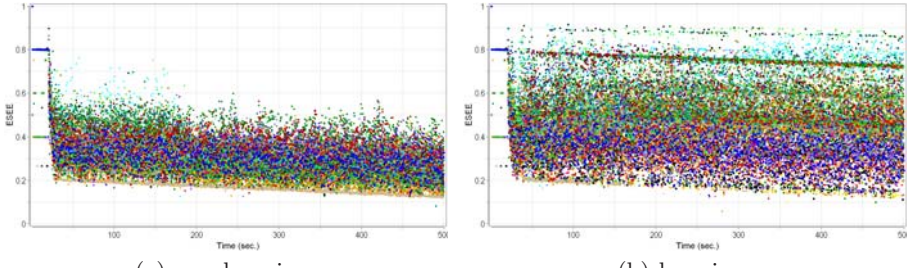
while increasing its lifetime.[3] The figure also shows that in the beginning of the simulation agents explore their action set and after approximately 200 seconds, the policy of all agents converges to an optimal action. In other words, after 400 frames, each agent finds the sleep duration that maximizes its ESEE and then sticks to it. The effect of adapting to the traffic pattern is even more

[3] Due to the discrete values in this graph, some values overlap and thus not all of them can be displayed at the same time.



(a) non-learning (b) learning

Fig. 4. Overhearing duration over Time, Small Network (10 nodes)



(a) non-learning (b) learning

Fig. 5. Effect Set Energy Efficiency over Time, Large Network (50 nodes)

apparent in the large network, where agents are able to decrease the average latency with over 70%, resulting in three times more packets delivered to the sink (cf. Figure 2).

Figure 4 compares the overhearing duration of nodes over time in the small network when all agents sleep the same amount of time (4(a)) and when they learn their optimal sleep duration (4(b)). Each dot represents that agent's overhearing duration within a frame at the corresponding time in the simulation. It is evident that when learning, agents reduce this source of energy waste, resulting in higher end battery level.[4] In other words, as the time progresses, agents learn to sleep when their neighbours are communicating, in order to reduce the amount of packets they overhear. This is evident from the fewer dots in Figure 4(b). As a consequence of the convergence to an optimal policy (explained above), one can see a large reduction in overhearing duration after approximately 200 seconds of network runtime. However, we did not measure significant decrease in the overhearing duration of the large network, as it can be predicted from Figure 2. The end battery level of the large network increased with only 2%. This was a result of the large number of nodes and consequently

[4] The discrete steps in the graph are a result of the fixed control and data packet lengths that nodes overhear.

the time they need to find an optimal action. Nevertheless, our learning agents had higher overall efficiency, due to the lower amount of unsuccessful transmission and the shorter stay of packets in the queues of the nodes.

The improved ESEE of agents in the large network can be seen in Figure 5(b), as compared to their non-learning counterparts (5(a)). Each dot represents that agent's ESEE within a frame window at the corresponding time in the simulation. In other words, the graph shows the relative energy efficiency of each node's neighbourhood over time. Although the efficiency of the worst performing nodes is comparable, the average ESEE of the learning agents is higher, than that of the non-learning nodes. This means that when using our algorithm for optimization, on average agents are more energy efficient than when they are not learning. The mean ESEE of both graphs, however, is constantly decreasing, since the remaining battery level of nodes is included in this reward signal (cf. Subsection 3.2). In other words, since battery level is inevitably decreasing, so is the ESEE of both networks.

5 Conclusion

In this article we used a reinforcement learning algorithm to improve the performance of Wireless Sensor Networks (WSN) in a decentralized manner, in order to prolong the autonomous lifetime of the network and reduce its latency. We were able to show that when agents in a WSN use an algorithm for optimization, they can learn to reduce the negative effect of their actions on other agents in the system, without a central mediator. Our results indicate that both in a small and large network, agents can learn to optimize their behaviour in order to increase the energy efficiency of the system and significantly decrease its latency with minimal communication overhead. Our results outperformed a conventional ad-hoc network, where all agents equally listen and sleep for a pre-defined amount of time. Thus, based on our experiments we can conclude that it is more beneficial for the sensor network when nodes *learn* what actions to take, rather than follow a pre-defined schedule. In our algorithm each node seeks to improve not only its own efficiency, but also the efficiency of its neighbourhood, which ensures that the agents' goal is aligned with the system goal of higher energy efficiency and lower latency.

Based on our empirical data, we can also generalize that a crucial point in achieving global efficiency in decentralized learning is aligning the agents' goals with the system goal. Letting each agent selfishly pursue its own objectives may simply lead to a suboptimal solution. In contrast, we were able to achieve global efficiency by making each agent consider a small group of surrounding agents. Thus, the objectives of each agent become identical with the goals of the "team" and therefore – of the whole system.

We are currently focusing on comparing the performance of our algorithm to the X-MAC protocol [Bue06], which aims to increase energy efficiency in a decentralized way without any communication overhead. Additionally, we aim to extend our approach, presented in this work, to make it suitable for a larger

set of WSN applications, where the network will adapt to the latency requirement of the user directly. We are also aiming towards more sophisticated parameter studies (e.g. genetic algorithms) to ensure a (nearly) optimal parameter setting.

Future work involves computing the energy requirements of the algorithm itself and experimenting with different network topologies and reward functions to obtain a yet bigger improvement in energy efficiency and latency.

References

[Car04] Carle, J., Simplot-Ryl, D.: Energy-Efficient Area Monitoring for Sensor Networks. IEEE Computer Society 47, 40–46 (2004)

[Rog06] Rogers, A., Dash, R.K., Jennings, N.R., Reece, S., Roberts, S.: Computational Mechanism Design for Information Fusion within Sensor Networks. In: 9th FUSION (2006)

[Mih08] Mihaylov, M., Nowé, A., Tuyls, K.: Collective IntelligentWireless Sensor Networks. In: Proc. of the 20th BNAIC, pp. 169–176 (2008)

[Dam03] van Dam, T., Langendoen, K.: An Adaptive Energy-Efficient Mac Protocol For Wireless Sensor Networks. In: Proceedings of The 1st SenSys, pp. 171–180 (2003)

[Yic08] Yick, J., Mukherjee, B., Ghosal, D.: Wireless Sensor Network Survey. Computer Networks 52, 2292–2330 (2008)

[Ai04] Ai, J., Kong, J., Turgut, D.: An adaptive coordinated medium access control for wireless sensor networks. In: Proceedings of 9th ISCC, vol. 2, pp. 214–219 (2004)

[Bar05] Barroso, A., Roedig, U., Sreenan, C.J.: μ-MAC: An Energy-Efficient Medium Access Control for Wireless Sensor Networks. In: Proceedings of the 2nd EWSN (2005)

[Bue06] Buettner, M., Yee, G., Anderson, E., Han, R.: X-MAC: A Short Preamble MAC Protocol For Duty-Cycled Wireless Sensor Networks. University of Colorado at Boulder (2006)

[Far08] Farinelli, A., Rogers, A., Petcu, A., Jennings, N.R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: Proceedings of the 7th AAMAS, pp. 639–646 (2008)

[Jai09] Jain, M., Taylor, M., Yokoo, M., Tambe, M.: DCOPs Meet the Real World: Exploring Unknown Reward Matrices with Applications to Mobile Sensor Networks. In: Proceedings of the 21st IJCAI (2009)

[Len08] Leng, J.: Reinforcement learning and convergence analysis with applications to agent-based systems. University of South Australia (2008)

[Wol08] Wolpert, D.H., Tumer, K.: An Introduction To Collective Intelligence. NASA Ames Research Center (2008)

[Mar04] Martinez, K., Ong, R., Hart, J.: Glacsweb: a sensor network for hostile environments. In: The 1st IEEE Secon (2004)

[Ess08] Esseghir, M., Bouabdallah, N.: Node density control for maximizing wireless sensor network lifetime. Int. J. Netw. Manag. 18, 159–170 (2008)

[Ver04] Verbeeck, K.: Coordinated Exploration in Multi-Agent Reinforcement Learning. Ph.D Thesis, Computational Modeling Lab, Vrije Universiteit Brussel (2004)

[Vra08] Vrancx, P., Verbeeck, K., Nowe, A.: Decentralized Learning in Markov Games. IEEE Transactions on Systems, Man and Cybernetics 38, 976–981 (2008)

[Tha04] Thathachar, M.A.L., Sastry, P.S.: Networks of learning automata: Techniques for online stochastic optimization. Kluwer Academic Publishers, Dordrecht (2004)

[Tuy06] Tuyls, K., Hoen, P.J., Vanschoenwinkel, B.: An Evolutionary Dynamical Analysis of Multi-Agent Learning in Iterated Games. JAAMAS 12(1), 115–153 (2006)