Collective Intelligent Wireless Sensor Networks

Mihail Mihaylov [a] Ann Nowé [a] Karl Tuyls [b]

[a] *Vrije Universiteit Brussel, Pleinlaan 2, Brussel, BE,*
{*mihail.mihaylov; ann.nowe*} [AT] *vub.ac.be*
[b] *Technische Universiteit Eindhoven, Postbus 513, Eindhoven, NL, k.p.tuyls* [AT] *tue.nl*

Abstract

In this paper we apply the COllective INtelligence (COIN) framework of Wolpert et al. to Wireless Sensor Networks (WSNs) with the aim to increase the autonomous lifetime of the network in a decentralized manner. COIN describes how selfish agents can learn to optimize their own performance, so that the performance of the global system is increased. WSNs are collections of densely deployed 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 its own energy efficiency, so that the energy efficiency of the whole system is increased. We introduce a new private utility function that will measure the performance of each agent and we show that nodes in WSNs are able to develop an energy saving behaviour on their own, when using the COIN framework.

1 Introduction

An increasingly popular approach for environmental and habitat monitoring is the use of Wireless Sensor Networks (WSNs) [2]. The nodes in 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. A complicating factor is communication, because some nodes 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.

In this paper we use the MacCOIN approach, proposed by Beyens et al. [1], to optimize the energy efficiency of a WSN by making nodes (hereby regarded as agents) learn energy-saving schemes by themselves in a distributed manner. This is achieved by applying the COllective INtelligence (COIN) framework of Wolpert et al. [7, 8, 6]. The idea behind this approach is that agents use an algorithm to optimize their performance, based on a certain performance measure, defined by a utility function. Whereas the contribution of [1] remains conceptual, we apply the approach to simulated sensor networks in this paper. Moreover, we propose two different algorithms and a utility function that will help agents achieve energy efficiency and thus extend the autonomous lifetime of the WSN.

The outline of the paper 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 collective intelligence and its application to the energy efficiency optimization of nodes. Section 4 shows the results of our experiments and some discussion about its applicability in real life settings. 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 shows the challenges of WSN communication and the design criteria for an energy efficient network. Subsection 2.2 describes the MAC protocol used in our research and the operation of a single node in the WSN.

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.

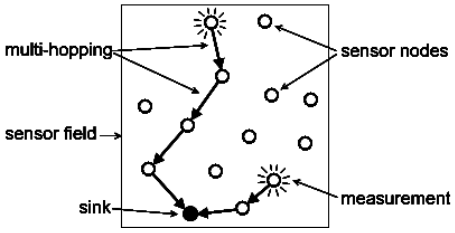

Figure 1: Wireless Sensor Network

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 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 paper we use the COIN framework to make the selfish and computationally bounded agents optimize their own performance in order to achieve the global objective of energy efficiency. The framework assumes that communication between the agents is limited and that central control is not possible [8]. 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 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.

In our research we use a Time Division Multiple Access (TDMA) protocol that divides the signal into frames, where each frame is further divided into time slots. Each node independently determines its schedule, i.e. the amount of time slots it will spend in one of its four modes of operation: transmit, listen, receive and sleep. Those modes are defined as follows:

- A node goes in **transmit** mode, when it starts to send a message through the channel. The omnidirectional antenna of the node broadcasts the message to *all* nodes in range, which we call neighbours (or neighbouring nodes), but the message itself is addressed to only *one* of them. Neighbouring nodes that receive a broadcasted message, which is not addressed to them, will simply discard it.

- When in **listen** mode, the sensor node is actively listening for broadcasts in the medium. When a signal is detected, the node goes in receiving mode.

- A sensor node is in **receive** mode when receiving a message from a neighbouring node. When the whole message is received, the node goes in one of the three other modes.

- When a node is in **sleep** mode, its radio transceiver is switched off and therefore no communication is possible. Nevertheless, the node continues its sensing and processing tasks.

The TDMA protocol allows nodes to synchronize their schedules prior to communication and thus avoid collisions and overhearing – typical sources of energy waste. The amount of wasted energy is very much dependent on the hardware of the node. Therefore we will provide only relative values for the energy consumption of the above four modes of operation (for exact values, see [5]):

- transmitting is two times more expensive (in energy consumption) than listening or receiving;
- listening and receiving consume nearly the same energy;
- sleeping is ten times cheaper than listening or receiving.

Given these energy consumption rules, 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 listening and 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 preserving the latency requirements of the observer. The MAC protocol should therefore support the exchange of additional information, necessary for the algorithms for optimization. It is clear that the amount of this information within message packets should be kept as little as possible, to minimize the energy waste by control packet overhead. Since it is not our aim here to provide detailed description of the protocol used, we refer the interested reader to [3].

3 Methodology

In this section we present the collective intelligence framework and then apply it to the energy conservation problem of WSNs. In Subsection 3.2 we define a performance measure function for the energy efficiency of nodes and let them use an algorithm to optimize their efficiency, based on this function.

3.1 COIN Framework

The emerging science of COllective INtelligence (COIN) considers how to design large multi-agent systems, where selfish agents learn to optimize a private utility function, so that the performance of a global utility is increased. The main difficulty in COIN is that there is little to no centralized control over the behaviour of the computationally bounded agents. Each agent is guided by its own *private utility*, while the performance of the collective is measured by the *global utility* (or *world utility*). Thus, the main challenge in the COIN framework is to define a suitable private utility function for the individual agents that will lead to an effective emergent behaviour. Several utility functions for different problems have been proposed in the past [7], but they cannot be applied to the energy conservation problem of wireless sensor networks, due to the limited information available to nodes. Agents in a WSN can obtain only local information from surrounding nodes, due to their small transmission range. In the next Subsection we propose a new utility function that will lead to a global energy efficient behaviour.

3.2 MacCOIN

To find a good private utility function that will measure the performance of agents, we first introduce what Energy Efficiency (EE) of a single agent is. Then we formulate the utility function of an agent and explain how it can be obtained from local information. Lastly, we present two algorithms for private utility optimization, which will make nodes find an energy efficient solution by themselves.

3.2.1 Energy Efficiency

An agent is energy efficient when it minimizes most of the major sources of energy waste in WSN communication – idle listening, overhearing, unsuccessful transmissions and control packet overhead, while maximizing sleep mode and taking into account the latency requirement of the observer. In this paper we will not consider the latency requirement of the user *directly*, because the latency of a packet is only known

when it arrives at the sink. Instead, we include a latency component in the function for EE of an agent. This component "penalizes" the agent for each frame that it holds a packet in its queue. Additionally, we specify the maximum number of frames during which a data packet is allowed to stay in the queue of the sensor node. More specifically, if a packet remains in the queue of a node for more than 3 frames, that packet is dropped and thus will not be delivered to the sink, because it has expired. This value depends on the application of the WSN. It has been experimentally chosen to stimulate faster forwarding by penalizing the node for dropped packets, while still maximizing the number of delivered messages in a network, designed for environmental monitoring. For a formal description of the energy efficiency of agents, we refer the interested reader to [1]. The motivation to consider latency in such an indirect manner is that we focus rather on investigating whether nodes *can* develop energy saving schemes on their own and to examine the trade-off between efficiency and latency for different algorithms for optimization. After introducing the energy efficiency of a single agent, we will motivate the need for a suitable private utility function and then will proceed with its formulation.

3.2.2 Private Utility Function

It is easy to show that if agents try to increase simply their *own* energy efficiency, that will not lead to high global efficiency, because agents will tend to sleep more (since that is the cheapest mode) and hence – higher latency of the network[1] and more unsuccessful transmissions of neighbouring nodes. Therefore, individual agents should also consider other agents in the system when optimizing their own behaviour. We believe that if each agent "cares about the rest" 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 set of all nodes that are influenced by the actions of that single node. In other words, the ES of agent i is the set N_i of nodes that lie on the possible message paths between agent i and the sink, since those agents have to forward the messages of agent i.[2] Thus, the energy efficiency of all agents in N_i is directly dependent on the actions of agent i. Therefore, if each agent seeks to increase not only its own efficiency, but also the efficiency of its ES, our hypothesis is that this will lead to higher energy efficiency of the whole system. For this reason, we set the private utility of each agent to its Effect Set Energy Efficiency (ESEE). We define the ESEE of agent i in frame f as

$$\text{ESEE}_{i,f} = \frac{\text{EE}_{i,f} + \sum_j \text{EE}_{j,f}}{|N_i| + 1} \qquad \forall j \in N_i$$

where $\text{EE}_{i,f}$ 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 ESEE of an agent is the average EE of all the agents in its effect set, together with its own efficiency. Thus, agents will try to increase the value of their ESEE by optimizing their own behaviour.

The idea behind the effect set of a node is based on the work of Wolpert et al. [7], who introduced this concept for their Wonderful Life private utility function. However, they use this concept in a different way, namely to "clamp" the actions of agents in an ES in order to determine whether the system performs better without them. The Wonderful Life utility is further explained in Subsection 4.3.

One challenge for our private utility function 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 and compute the average. Obtaining the energy efficiency of distant nodes is merely not possible due to the small transmission range of nodes. Instead, they should only *estimate* the efficiency of their ES. To achieve that, nodes have to send information about the efficiency of all nodes in range, i.e. their neighbourhood, to their neighbours. This information is easy to obtain for each node, because neighbouring nodes, by definition, lie within transmission range. Starting from the neighbours of the sink, each node takes the average of the efficiencies of its own neighbourhood and the ES of its neighbours[3] (which in turn do the same, while excluding already counted nodes), to obtain its own ESEE. Thus, the ESEE of agent i is the average energy efficiency of all the neighbourhoods between agent i and the sink, i.e. all the nodes in N_i. We will now show how each agent can learn to optimize its ESEE.

[1] unless the latency component "penalizes" nodes too much, in which case they will try to forward packets very often to decrease latency, leading to high control packet overhead and higher chance of collisions and overhearing.

[2] The set N_i could be different in size for each node i, depending on the topology of the network and the position of agent i in it.

[3] only those neighbours that lie on the possible message paths towards the sink, i.e. only those nodes in i's neighbourhood that are closer to the sink.

3.2.3 Algorithms for ESEE Optimization

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 both for itself and for its ES. To achieve that, we make nodes adopt an algorithm for optimization in order to increase the payoff of their private utility function, i.e. to optimize their ESEE. We tested the effect of two different algorithms on the performance of nodes. Each node compares the ESEE of the last frame to the one before that in order to determine whether its efficiency has dropped or increased. This learning is done on-line, since the algorithms adapt to the topology of the network and the traffic pattern, which typically cannot be known in advance in order to train them off-line. Based on their private utility function and the algorithm they implement, the nodes will take different actions.

The first algorithm lets nodes listen more, if their efficiency drops. The motivation behind it is to decrease the latency of the network and the energy waste of unsuccessful transmissions, caused by nodes, trying to send messages to sleeping neighbours. This algorithm is a simple decision rule, saying that if the ESEE of a node drops, regardless of the amount, that node should increase the time for listening by a fixed amount of time slots within a frame. On the other hand, if the ESEE of the node increases, the decision rule specifies that the node should sleep more, because nodes in its ES are "already doing well" and therefore it can safe more energy for itself by sleeping.

The second algorithm is a learning function, which aims to decrease the energy waste of idle listening and overhearing by letting nodes sleep more, if their ESEE drops. Nodes try to optimize this learning function by changing their listening and sleeping intervals according to the change in the payoff of their private utility. The faster this payoff decreases, the more time slots the node will sleep in order to minimize idle listening and overhearing. Increasing the ESEE will result in more listening, so that the latency is reduced.

Both algorithms have opposite effect on the nodes' behaviour, because one cannot know in advance what the *major* source of energy waste in a WSN is. Therefore, we seek to investigate this issue by designing two different algorithms, aimed at minimizing different sources of energy waste, and observing how the ESEE of nodes change. Additionally, those algorithms for optimization will have different effect on nodes in different parts of the WSN, depending on node density and message rate. In this way, nodes, close to the sink might learn to listen more, while distant nodes might find that more sleeping optimizes their performance, based on the payoff of their private utility function.

4 Results

4.1 Experimental Setup

The wireless networks, used in the experiments, were composed of nodes, aligned in a rectangular grid structure, rather than being randomly distributed in the environment. Although this is rarely applied in practice, we use such a simple structure to test the *theoretical application* of the COIN framework, rather than to propose an "out-of-the-box" practical solution. We believe that a grid structure is suitable for the purposes of this research. The transmission range of all nodes is 1.5 grid units in radius, therefore, a node can have at least 3 neighbours, when at the "corner" of the grid, and at most 8, when in the inner part of the grid. In other words, all immediate neighbouring nodes in the grid are also neighbours in the network. Initially, all nodes start synchronously to listen and sleep for equal amount of time slots in a frame. One frame consists of 10 time slots, which means that the periodic listening occurs in the first 5 slots and the sleeping – in the last 5 slots of each frame for all nodes. Agents forward messages only to lower hop neighbours, i.e. to nodes that are closer to the sink, than they are. If more than one such nodes exist, the sender will choose the one that has processed fewer messages in the past frames. The information about the amount of processed messages is shared between neighbouring nodes, so that each node keeps track of the number of processed messages of its neighbours. We assume that this simple routing algorithm performs better than sending a message to a random neighbour. Lastly, we perform our experiments on a network consisting of 100 nodes, for the duration of 5000 frames, in which each node makes 4 measurements in the environment. We assume that measurements may have variable length (e.g. due to compression or information fusion with measurements of neighbouring nodes [4]) and therefore one measurement may take up more than one message packets.

4.2 Experiments

We evaluated three experimental settings, designed to test the performance of nodes:

0 Nodes use **no algorithm** to optimize their private utility function. They are simply following their fixed periodic schedules with equal number of time slots for listening and sleeping.[4] This scenario serves only as the base line for the two algorithms for optimization.

1. Nodes use a **decision rule** to determine the amount of time slots to sleep and to listen in their periodic schedules. A decreasing payoff of their private utility function (irrelevant of the magnitude) results in more listening, while an increasing payoff makes the node sleep more. The motivation behind this algorithm is to decrease the latency of the network and the energy waste of unsuccessful transmissions.

2. Nodes use a **learning function** to find a periodic schedule that will ensure high values of energy efficiency. The more the ESEE of a node is decreasing, the more the node will tend to sleep, while increasing ESEE results in more listening. The purpose of this algorithm aims to decrease the energy waste of idle listening and overhearing.

In each experiment we measured the minimum, average and maximum battery life (i.e. mean and standard deviation) among all nodes and the latency of the network. The average battery life indicates how fast nodes dissipate their energy across time. The steeper this value drops over time, the less energy efficient the network is. The minimum and maximum battery life show the difference of energy levels between the least and the most efficient nodes respectively. The closer these values are around the average battery life, the better the efficiency of the algorithm is, because that would indicate a uniform dissipation of energy across the network. The latency of the network is expressed in terms of received messages, which signify the number of messages that the sink has received throughout the simulation runtime (5000 frames) as a part of all generated messages within that time. Here the intuition of latency is reversed, because higher number of received messages means low latency. When this value is 100%, it means that all generated messages are received by the sink within those 5000 frames.

The results of each experiment are averaged over 1000 runs with the same settings, but with a different random seed and position of the sink in the network. Figure 2 shows the performance of the network when nodes do not optimize their behaviour. All nodes simply listen and sleep for equal amount of time slots in each frame. However, when nodes use the above mentioned decision rule to improve the efficiency of their ES, the sensor network increases its performance significantly, as shown in Figure 3. In the latter case, the average remaining battery in the end of the simulation is 83%, while with no optimization, nodes waste more energy and therefore have only 60% remaining battery after the 5000 time frames. Recall that this decision rule aims to decrease the energy waste of unsuccessful transmissions by letting nodes *listen* more if their efficiency drops. However, we observed that nodes nevertheless tend to sleep more, since sleeping is the cheapest mode. This behaviour improves their efficiency and therefore, according to the decision rule, leads to even more sleeping (cf. subsection 3.2.3). These results imply that unsuccessful transmissions do not constitute a big source of energy waste, compared to the energy savings during sleep mode, and therefore nodes improve their efficiency more by sleeping than by trying to minimize unsuccessful transmissions. The drawback of this algorithm is that it causes nodes to sleep much and therefore increases the latency of the whole network. This can be seen in Figure 4, which shows the percent of received messages by the sink within 5000 frames for the three experiments. Although the decision rule is able to make nodes save more energy, it decreases the response time of the network significantly, because the results showed that almost all nodes increase their sleeping duration, rather than appropriately adapting their periodic schedules to the message flow in the network.

The second algorithm for optimization, i.e. the learning function, shows a comparable improvement in energy efficiency to the first (see Figure 5). This learning function aims to decrease the energy waste of idle listening and overhearing by letting nodes *sleep* more, if their ESEE drops. The difference to the above mentioned decision rule is that if here the efficiency of the nodes increases, they will start to listen more in order to decrease the latency of the network. The result of the latter behaviour can be seen in Figure 4. Despite the comparable high energy savings to the first algorithm, the learning function performs significantly better in terms of latency. This is because nodes near the sink usually have to process more messages and thus, according to the algorithm, learn to increase their payoff by listening more, while nodes

[4] Although we say there is "no actual optimization", nodes still save nearly 50% more energy, compared to conventional networks, because in our network they sleep already half of their lifetime.

far from the sink learn to avoid idle listening by spending more time sleeping, than they do with the decision rule. Therefore, on average, the energy savings with the second algorithm are comparable to the first, but the latency of the system using the learning function is far less, because nodes learn to adapt better to the message flow in the network, i.e. to minimize not only efficiency, but latency as well. In Figure 4 one can see that 78% of all generated messages are received by the sink within 5000 frames, when nodes optimize their behaviour using the second algorithm. Thus, the latency with the learning function is three times less than with the decision rule, while both algorithms show similar performance in terms of energy efficiency. This behaviour also explains the lower minimum battery level of the network (see Figure 5). This lower minimum is caused by all nodes near the sink, because they dissipate more energy by trying to minimize latency, i.e. by spending more time slots on listening and thus forwarding all the messages of other nodes.

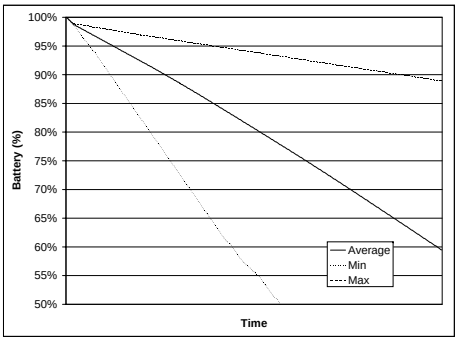

Figure 2: Battery life throughout simulation; 100 nodes, no algorithm for optimization

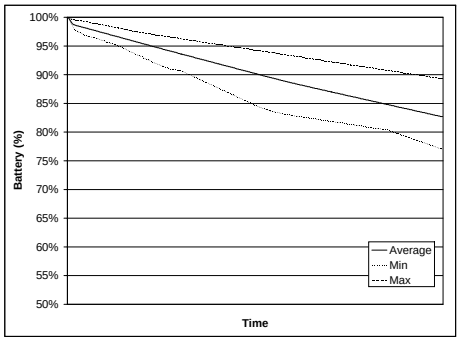

Figure 3: Battery life throughout simulation; 100 nodes, optimization using a decision rule

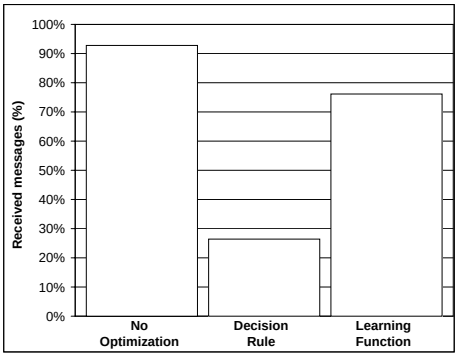

Figure 4: Amount of received messages by the sink within the duration of the simulation (5000 frames)

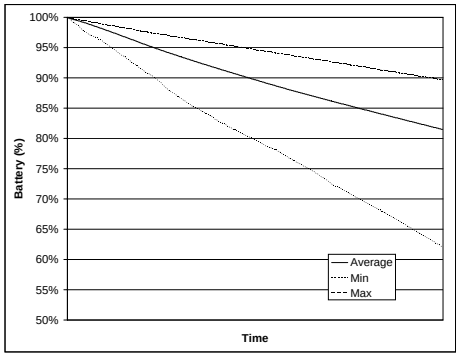

Figure 5: Battery life throughout simulation; 100 nodes, optimization using a learning function

4.3 Discussion

These experiments demonstrated the challenge of the COIN framework in finding an appropriate private utility function and an algorithm for optimization to make nodes achieve the designer's goal on their own in a decentralized fashion. We were able to show that when the algorithm is chosen carefully, it leads to a desirable outcome, as it was the case with the learning function. It was able to achieve high energy efficiency, while not sacrificing latency a lot.

In the above experiments we used the ESEE as a private utility function for each node. One advantage of this function is that it makes each agent consider the effect of its own behaviour on the other agents in the system. In this way, when agents learn to optimize their ESEE, they in fact learn to reduce the negative effect of their actions on the system. In other words, by using this private utility function, we align the agents' goal of improving one's own energy efficiency with the system goal of increasing the autonomous lifetime of the whole network. Additionally, the above two algorithms adapt well to adding or removing nodes from the system when the necessary information for the update propagates through the network. The

disadvantage is that updating the value of ESEE happens rather slow for large networks, due to the large number of messages that need to be exchanged.

Alternatively, we have tested a similar private utility function, proposed in the work of Wolpert et al. [7], namely the Wonderful Life Utility (WLU). This function lets each node estimate "how the world would look, if that node had never existed". In other words, agents try to determine their effect on the whole system by computing the performance of the network if they are removed from it. One disadvantage of the WLU is that each node needs to know the energy efficiency of the *whole* network and of the nodes that are *not* part of its ES, in order to determine whether the system will perform better without it. These requirements increase not only the delay of the payoff, but also the amount of information that needs to be transmitted across the network, so that nodes can compute the WLU. We, therefore, found this private utility function less suitable for this research.

5 Conclusion

In this paper we applied the COllective INtelligence (COIN) framework to the energy conservation problem of Wireless Sensor Networks (WSN) in order to prolong the autonomous lifetime of the network, while taking into account the latency requirement of the user. We let individual nodes adopt an algorithm to optimize their own behaviour, so that the energy efficiency of the whole system is increased. Each node seeks to improve not only its own energy efficiency, but also the efficiency of the nodes it affects, which will ensure that the agents' goal is aligned with the system goal of higher energy efficiency. The two algorithms, presented in this paper, are compared to a setting where nodes do not optimize their behaviour at all. The results show that when sensor nodes learn to optimize their actions, their battery consumption is reduced by more than 70%, compared to conventional ad-hoc networks. One drawback to this approach is that it comes at the cost of increased latency. However, in most WSN applications for environmental or habitat monitoring, latency is not that crucial and therefore it can be sacrificed to some extent in order to prolong the lifetime of the network to allow for longer observations.

We are currently focusing on extending the approach, presented in this paper, to make it suitable for a larger set of WSN applications, where the network will adapt to the latency requirement of the user directly. Future work involves improving the routing protocol, experimenting with different network topologies and testing additional algorithms for optimization to obtain a better trade-off between energy efficiency and latency.

References

[1] Pieter Beyens, Karl Tuyls, Kris Steenhaut, and Ann Nowe. Maccoin: Medium acces control for wireless sensor networks based on collective intelligence. In *Proceedings of GTDT'04*, pages 1–12, 2004.

[2] Jean Carle and David Simplot-Ryl. Energy-efficient area monitoring for sensor networks. *IEEE Computer Society*, 47(2):40–46, 2004.

[3] Mihail Mihaylov. Computational Mechanism Design for Wireless Sensor Networks. Master's thesis, Maastricht University, The Netherlands, March 2008.

[4] Alex Rogers, Rajdeep K. Dash, N. R. Jennings, Steve Reece, and Stephen Roberts. Computational mechanism design for information fusion within sensor networks. In *Ninth International Conference on Information Fusion*, 2006.

[5] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings Of The First International Conference On Embedded Networked Sensor Systems*, pages 171–180, Los Angeles, California, USA,, 2003.

[6] David Wolpert and Kagan Tumer. *Collectives and the Design of Complex Systems*, chapter A Survey of Collective Intelligence, pages 1–42. Springer-Verlag, 2004.

[7] David H. Wolpert. Collective intelligence. Technical report, NASA Ames Research Center, 2003.

[8] David H. Wolpert and Kagan Tumer. Collective intelligence, data routing and braess paradox. *Journal of Artificial Intelligence Research*, 16:359–387, 2002.