Decentralised reinforcement learning for energy-efficient scheduling in wireless sensor networks

Mihail Mihaylov* and Yann-Aël Le Borgne

Vrije Universiteit Brussel,
Pleinlaan 2, Brussels, Belgium
E-mail: mmihaylo@vub.ac.be
E-mail: yleborgn@vub.ac.be
*Corresponding author

Karl Tuyls

Maastricht University,
Sint Servaasklooster 39, Maastricht, The Netherlands
E-mail: k.tuyls@maastrichtuniversity.nl

Ann Nowé

Vrije Universiteit Brussel,
Pleinlaan 2, Brussels, Belgium
E-mail: ann.nowe@vub.ac.be

Abstract: We present a self-organising reinforcement learning (RL) approach for scheduling the wake-up cycles of nodes in a wireless sensor network. The approach is fully decentralised, and allows sensor nodes to schedule their active periods based only on their interactions with neighbouring nodes. Compared to standard scheduling mechanisms such as SMAC, the benefits of the proposed approach are twofold. First, the nodes do not need to synchronise explicitly, since synchronisation is achieved by the successful exchange of data messages in the data collection process. Second, the learning process allows nodes competing for the radio channel to desynchronise in such a way that radio interferences and therefore packet collisions are significantly reduced. This results in shorter communication schedules, allowing to not only reduce energy consumption by reducing the wake-up cycles of sensor nodes, but also to decrease the data retrieval latency. We implement this RL approach in the OMNET++ sensor network simulator, and illustrate how sensor nodes arranged in line, mesh and grid topologies autonomously uncover schedules that favour the successful delivery of messages along a routing tree while avoiding interferences.

Keywords: wireless sensor networks; WSN; reinforcement learning; RL; energy efficiency; wake-up scheduling; decentralised; synchronisation; desynchronisation; self-organisation; latency; duty cycle; routing tree; radio interference.

Reference to this paper should be made as follows: Mihaylov, M., Le Borgne, Y-A., Tuyls, K. and Nowé, A. (2012) 'Decentralised reinforcement learning for energy-efficient scheduling in wireless sensor networks', *Int. J. Communication Networks and Distributed Systems*, Vol. 9, Nos. 3/4, pp.207–224.

Biographical notes: Mihail Mihaylov received his BSc from University of Duisburg-Essen, Germany in 2007 and MSc from Maastricht University, The Netherlands in 2008. He is currently working on his PhD at the Computational Modelling Lab (CoMo) in Vrije Universiteit Brussel, Belgium. His main interests lie in the fields of distributed multi-agent systems and reinforcement learning. His work focuses on the development of decentralised algorithms for intelligent communication between energy-constrained devices.

Yann-Aël Le Borgne received his PhD in Computer Science in 2009 from the Machine Learning Group of the Université Libre de Bruxelles, Belgium. His main interests concern the design of distributed machine learning techniques for energy-efficient data processing in WSNs. He is currently a researcher at the Computational Modelling Lab and Informatics and Electronics Group (ETRO) of the Vrije Universiteit Brussel, Belgium.

Karl Tuyls received his MSc in 2000 and PhD in 2004 in Computer Science, both from the Vrije Universiteit Brussels, Belgium. In 2000, he has been awarded the Information Technology prize in Belgium and in 2007, he was elected as the best junior researcher (TOPDOG) of the Faculty of Humanities and Sciences, Maastricht University, The Netherlands. He is currently an Associate Professor at the Department of Knowledge Engineering, Maastricht University, where he leads a research group on swarm robotics and learning in multi-agent systems (Maastricht Swarmlab). He is also an Associate Editor of two journals and has published in absolute top journals in his research area such as *Artificial Intelligence*, *Journal of Artificial Intelligence Research*, *Theoretical Biology, Autonomous Agents and Multi-Agent Systems*, *Journal of Machine Learning Research*, etc.

Ann Nowé graduated from the University of Ghent, Belgium in 1987. Then she became a Research Assistant at the Vrije Universiteit Brussel, Belgium where she finished her PhD in 1994 in collaboration with Queen Mary and Westfield College, University of London. She is currently a Professor at the Computational Modelling Lab of the Vrije Universiteit Brussel.

1 Introduction

Wireless sensor networks (WSNs) form an emerging class of networks able to monitor our daily environment with a high spatiotemporal accuracy (Ilyas and Mahgoub, 2005; Akyildiz et al., 2002). WSNs are composed of small sensing devices, also known as wireless sensor nodes, endowed with sensing, processing and wireless communication capabilities. Given the current technological trend, WSNs are envisioned to be mass produced at low cost in the next decade, for applications in a wide variety of domains. These include, to name a few, ecology, industry, transportation, or defence.

A typical WSN scenario consists of a set of sensor nodes, scattered in an environment, which report their data periodically to a centralised entity called *base station* (Ilyas and Mahgoub, 2005; Akyildiz et al., 2002). The resources of the untethered sensor nodes are often strongly constrained, particularly in terms of energy and communication. The base station usually possesses much larger resources, comparable to those of a standard laptop or desktop computer.

The limited resources of the sensor nodes make the design of a WSN application challenging. Application requirements, in terms of lifetime, latency, or data throughput, often conflict with the network capacity and energy resources. The standard approach for addressing these tradeoffs is to rely on *wake-up scheduling* (Ilyas and Mahgoub, 2005; Schurgers, 2008), which consists in alternating the active and sleep states of sensor nodes. In the active state all the components (CPU, sensors, radio) of a node are active, allowing the node to collect, process and communicate information. During the sleep state all these components are switched off, allowing the node to run with an almost negligible amount of energy. However, nodes in sleep mode cannot communicate with others, since their radio transmitter is switched off. The fraction of time in which the node is in the active mode referred to as *duty cycle*.

Wake-up scheduling offers an efficient way to significantly improve the lifetime of a WSN application, and is best illustrated by SMAC, a standard synchronised medium access control (MAC) protocol for WSN (Ye et al., 2004). In SMAC, the duty-cycle is fixed by the user, and all sensor nodes synchronise in such a way that their waking periods take place at the same time. This synchronised active period enables the forwarding of messages between any pair of nodes, regardless of the size of the network.

In this paper, we demonstrate how the performance of a WSN network can be further improved, if nodes not only synchronise, but also *desynchronise* with one another. More precisely, the wake-up schedules of nodes that need to communicate with one another (i.e., nodes on a given branch of the routing tree) are synchronised to improve message throughput. We say that those nodes belong to one *coalition*. At the same time, the schedules of neighbouring nodes which do not need to communicate (i.e., nodes on neighbouring branches) are desynchronised in order to avoid radio interferences and packet losses.

We show that coordinating the duty cycles of sensor nodes can successfully be done using the multi agent systems and the reinforcement learning (RL) frameworks by rewarding successful interactions (e.g., transmission of a message) and penalising the ones with a negative outcome (e.g., message loss or overhearing). This behaviour drives the agents (i.e., the nodes) to repeat actions that result in positive feedback more often and to decrease the probability of unsuccessful interactions. Agents that tend to select the same successful action naturally form a coalition. Within the RL framework, the scheduling of the sensor nodes emerges from simple and local interactions without the need of central mediator or any form of explicit coordination. We illustrate the benefits of the proposed RL approach by implementing it in the OMNET++ simulator, and study three different WSN topologies, namely line, mesh and grid. We show that nodes form coalitions which enable a quicker delivery of the data packets to the base station, allowing shorter active periods and therefore lower energy consumption.

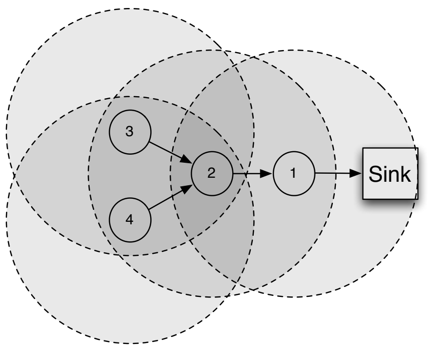
The paper is organised as follows. Section 2 presents the background of our study and outlines the energy challenges related to the communication and routing protocols in WSN. Our approach is exposed in Section 3, together with its application in WSNs on three different topologies. The results of this work are analysed in Section 4 and concluded in Section 5.

2 WSNs: energy-efficiency challenges

A WSN is a collection of densely deployed autonomous devices, called *sensor nodes*, which gather data with the help of sensors (Ilyas and Mahgoub, 2005; Akyildiz et al., 2002). The untethered nodes use radio communication to transmit sensor measurements to a terminal node, called the base station or *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.

When the WSN is deployed, the routing protocol requires that the nodes determine a routing path to the sink (Al-Karaki and Kamal, 2004; Ilyas and Mahgoub, 2005). This is achieved by letting nodes broadcast packets immediately after deployment in order to discover their neighbours. Nodes in communication range of the sink propagate this information to the the rest of the network. During the propagation process, each node chooses a *parent*, i.e., a node to which the data will be forwarded in order to reach the sink. The choice of a parent can be done using different metrics, the standard one being the hop distance, i.e., the minimum number of nodes that will have to forward their packets (Couto et al., 2005; Woo et al., 2003). An example of multi-hop shortest path routing structure is given in Figure 1, together with the radio communication ranges of sensor nodes.

Figure 1 Sensor nodes connected to a base station by means of a multi-hop routing tree



Note: Greyed circles indicate overlapping communication regions.

Since wireless sensor nodes operate in most cases on finite energy resource, low-power operation is one of the crucial design requirements in sensor networks (Akyildiz et al., 2002; Ilyas and Mahgoub, 2005). The challenge of energy-efficient operation must be tackled on all levels of the network stack, from hardware devices to protocols and applications. Although sensing and data processing may incur significant energy consumption, it is commonly admitted that most of the energy consumption is caused by the radio communication. A large amount of research has therefore been devoted in the recent years to the design of energy-efficient communication protocols (Ilyas and Mahgoub, 2005; Ye et al., 2004).

Figure 2 reports the radio characteristics of three representative and often used radio platforms: the CC1000 (used in MICA2), CC2420 (used in TelosB and IMote2) and the Xbee-802-15.4 (used in the Waspmote). An important observation is that for these typical radios, the transmit and receive power are comparable, and that the sleep power is at least two orders of magnitude lower. Therefore, the only way to significantly reduce power consumption is to have the radio switched off most of the time, and to turn it on only if messages must be received or sent. This problem is referred to as *wake-up scheduling*.

Figure 2 Typical wireless sensor hardware developed in the recent years, together with their main radio characteristics (see online version for colours)

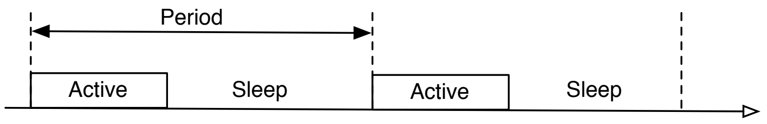
Mote	Mica-2	Tmote Sky	Imote2	Waspmote
Year	2002	2005	2007	2009
Radio	CC1000	CC2420		Xbee-802.15.4
Outdoor range	150 m	50/100m		500 m
Data rate	76 Kbps	250 Kbps		250 Kbps
Sleep power	100 μW sleep	60 μW sleep		<30 μW sleep
Receiving power	36 mW receive	63 mW receive		150 mW receive
Transmit power	75 mW xmit	57 mW xmit		135 mW xmit
Startup time	2 ms setup	1ms setup		2 ms setup

Wake-up scheduling in WSNs is an active research domain, and a good survey on wake-up strategies in WSNs is presented in Schurgers (2008). Three types of wake-up solutions can be identified, namely, on-demand paging, synchronous and asynchronous wake-up.

In on-demand paging, the wake-up functionality is managed by a separate radio device, which consumes much less power in the idle state than the main radio. The main radio therefore remains in a sleeping state, until the secondary radio device signals that a message is to be received on the radio channel. This idea was first proposed with the PicoRadio and PicoNode projects (Guo et al., 2001) for extremely low power systems, and extended in Shih et al. (2002) and Agarwal et al. (2005) with hand-held devices. On-demand paging is the most flexible and energy-efficient solution, but adds non-negligible costs in the hardware design.

In synchronous wake-up approaches, nodes duty-cycle their radio in a coordinated fashion. Several MAC protocols have been proposed, allowing nodes to wake-up at predetermined periods in time at which communication between nodes becomes possible. A standard paper detailing this idea is that of sensor-MAC (S-MAC) (Ye et al., 2004). The basic scheme is that nodes rely on a fixed duty-cycle, specified by the user, where nodes periodically switched between the active and sleep states. This is illustrated in Figure 3. Several extensions to S-MAC have been proposed. In particular, authors in van Dam and Langendoen (2003) proposed T-MAC, which aims at improving the efficiency by making the active period adaptive. This is achieved by making the active period very small, e.g., only the time necessary to receive a packet, and by increasing it at runtime if more packets have to be received. The main concern with protocols based on synchronous wake-up is the overhead which can be caused by maintaining the nodes synchronised.

Figure 3 Structure of S-MAC with duty cycle and synchronous wake-up scheduling



Finally, in asynchronous wake-up solutions, the nodes are not aware of each other's schedules, and communication comes at an increase cost for either the sender or the receiver. In sender-based asynchronous wake-up, the sender continuously sends beacons until the receiver is awake. Once the receiver gets the beacon, it sends an acknowledgment to notify the sender that it is ready to receive a packet. This scheme is the basis for the low-power listening (Hill and Culler, 2002) and preamble sampling (El-Hoiydi, 2002) protocols. The receiver-based wake-up solution is the mirror image of sender-based, and was exposed in the etiquette protocol (Goel, 2005). Sender-based and receiver-based asynchronous protocols can achieve very low power consumption. Asynchronous wake-up solutions however require an overhead due to the signalling of wake-up events, which makes them inefficient when wake-up events are relatively frequent (Schurgers, 2008).

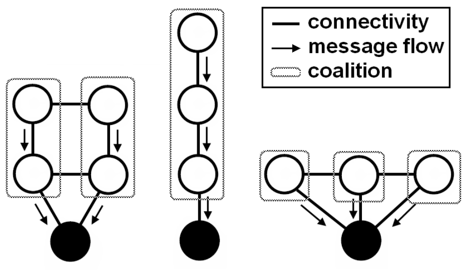
3 WSN wake-up scheduling with RL

3.1 Motivations and network model

The starting point of our work is to consider an approach similar to SMAC, but where the active periods are shifted in time. That is, instead of having all nodes communicate at the same moment, which causes interferences and packet losses, we aim at making the nodes wake up when the two following conditions are met. First, the parent of the node in the routing tree should be awake, so that the communication is successful. Second, the neighbouring nodes other than the parent should not be active.

Depending the routing protocol, coalitions (e.g., synchronised groups of nodes) logically emerge across the different hops, such that there is, if possible, only one agent from a certain hop within a coalition. Figure 4 illustrates this concept in three different topologies. It shows as an example how coalitions form as a result of the routing protocol. Intuitively, nodes from one coalition need to synchronise their wake-up schedules. As defined by the routing protocol, messages are not sent between nodes from the same hop, hence these nodes should desynchronise (or belong to separate coalitions) to avoid communication interference. The emergence of coalitions will be experimentally illustrated for different topologies in Section 4.

Figure 4 Examples of routing and coalition formation



The underlying scheduling mechanism resemble that of SMAC, i.e., that sensor nodes are duty-cycled, and the period of the duty-cycle is determined a priori by the user. We further divide each active period into time slots. The sensor nodes then rely on a standard duty cycle mechanism, in which the node is awake for a predetermined number of slots during each period. The active period is fixed by the user, while the wake-up slot is initialised randomly for each node. These slots will be shifted as a result of the learning (see Section 3.2), which will coordinate nodes' wake-up schedules in order to ensure high data throughput and longer battery life.

The routing protocol is not explicitly part of our learning algorithm and therefore any multi-hop routing scheme can be applied without loosing the properties of our approach. It is however noteworthy that the communication partners of a node (and thus the formation of coalitions) are influenced by the communication and routing protocols that are in use.

3.2 RL approach: methodology

In the following, we see a WSN as a multi-agent system (MAS), in which agents are the sensor nodes which will aim at improving their communication performances. Each agent in the WSN uses a RL (Sutton and Barto, 1998) algorithm to learn an efficient wake-up schedule (i.e., when to remain active within the frame) that will improve throughput and lifetime in a distributed manner. It is clear that learning in MASs of this type requires careful exploration in order to make the action-values of agents converge. We use a value iteration approach similar to single-state Q-learning (Watkins, 1989) with an implicit exploration strategy, as Subsection 3.5 will further elaborate on. However, our update scheme differs from that of traditional Q-learning (cf., Subsection 3.4). The main challenge in such a decentralised approach is to define a suitable reward function for the individual agents that will lead to an effective emergent behaviour as a group. To tackle this challenge, we proceed with the definition of the basic components of the proposed RL algorithm.

3.3 Actions and rewards

The actions of each agent are restricted to selecting a time window (or an active period) within a frame for staying awake. Since the size of these frames remains unchanged and they constantly repeat throughout the network lifetime, our agents use no notion of states, i.e., we say that our learning system is stateless. The duration of this active period is defined by the duty cycle, fixed by the user of the system. In other words, each node selects a slot within the frame when its radio will be switched on for the duration of the duty cycle. Thus, the size of the action space of each agent is determined by the number of slots within a frame. In general, the more actions agents have, the slower the RL algorithm will converge (Leng, 2008). On the other hand, a small action space might lead to suboptimal solutions and will impose an energy burden on the system. Setting the right amount of time slots within a frame requires a study on itself, that we will address in future work.

Every node stores a 'quality value' (or Q-value) for each slot within its frame. This value for each slot indicates how beneficial it is for the node to stay awake during these slots for every frame, i.e., what is an efficient wake-up pattern, given its duty cycle and considering its communication history. When a communication event occurs at a node (overheard, sent or received a packet) or if no event occurred during the active period (idle listening), that node updates the quality-value of the slot(s) when this event happened. The motivation behind this scheme is presented in Subsection 3.5.

3.4 Updates and action selection

The slots of agents are initiated with Q-values drawn from a uniform random distribution. Whenever events occur during node's active period, that node updates the quality values of the slots, at which the corresponding events occurred, using the following update rule:

$$Q_s^i \leftarrow (1-\alpha) \cdot \hat{Q}_s^i + \alpha \cdot r_{s,e}^i$$

where $Q_s^i \in [0,1]$ is the quality of slot s within the frame of agent i. A high Q_s^i value indicates that it is beneficial for agent i to stay awake during slot s. This quality value is updated using the previous Q-value ($\hat{Q}_s^i$) for that slot, the learning rate $\alpha \in [0,1]$, and the newly obtained reward $r_{s,e}^i \in [0,1]$ for the event e that (just) occurred in slot s. Thus, nodes will update as many Q-values as there are events during its active period. In other words, agent i will update the value Q_s^i for each slot s where an event e occurred. The latter update scheme differs from that of traditional Q-learning (Watkins, 1989), where only the Q-value of the selected action is updated. The motivation behind this update scheme is presented in Subsection 3.5. In addition, we set here the future discount parameter γ to 0, since our agents are stateless (or single-state).

Nodes will stay awake for those consecutive time slots that have the highest sum of Q-values. Put differently, each agent selects the action $a_{s'}$ (i.e., wake up at slot s') that maximises the sum of the Q-values for the D consecutive time slots, where D is the duty cycle, fixed by the user. Formally, agent i will wake up at slot s', where

$$s' = \arg\max_{s \in S} \sum_{j=0}^{D} Q_{s+j}^i$$

For example, if the required duty cycle of the nodes is set to 10% ($D = 10$ for a frame of $S = 100$ slots), each node will stay active for those 10 consecutive slots within its frame that have the highest sum of Q-values. Conversely, for all other slots the agent will remain asleep, since its Q-values indicate that it is less beneficial to stay active during that time. Nodes will update the Q-value of each slot for which an event occurrs within its duty cycle. Thus, when forwarding messages to the sink, over time, nodes acquire sufficient information on 'slot quality' to determine the best period within the frame to stay awake. This behaviour makes neighbouring nodes desynchronise their actions, resulting in faster message delivery and thus lower end-to-end latency.

3.5 Exploration

As explained in the above two Subsections, active time slots are updated individually, regardless of when the node wakes up. The reason for this choice is threefold. Firstly, this allows each slot to be explored and updated more frequently. For example, slot s will be updated when the node wakes up anywhere between slots $s-1$ and $s-D+1$, i.e., in D out of S possible actions. Secondly, updating individual Q-values makes it possible to alter the duty cycle of nodes at run time (as suggest some preliminary results, not displayed in this paper) without invalidating the Q-values of slots. In contrast, if a Q-value was computed for each start slot s, i.e., the reward was accumulated over the wake duration and stored at slot s only, changing the duty cycle at run-time will render the computed Q-values useless, since the reward was accumulated over a different duration. In addition, slot s will be updated only when the agent wakes up at that slot. A separate exploration strategy is therefore required to ensure that this action is explored sufficiently. Thirdly, our exploration scheme will continuously explore and update not only the wake-up slot, but all slots within the active period. Treating slots individually results in an implicit exploration scheme that requires no additional tuning.

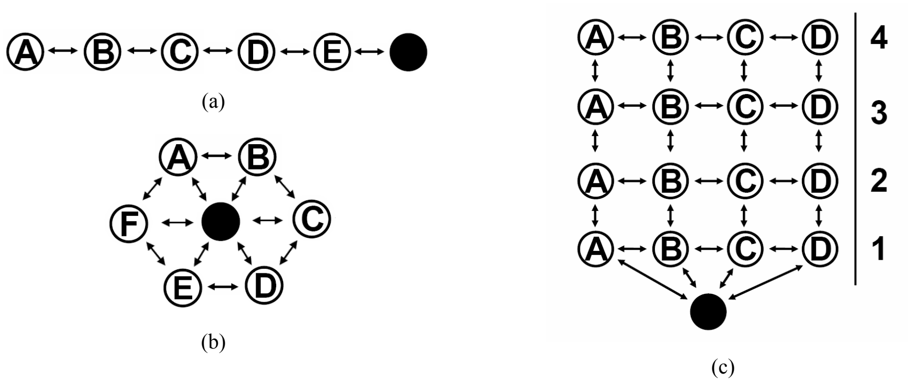
Even though agents employ a greedy policy (selecting the action that gives the highest sum of Q-values), this 'smooth' exploration strategy ensures that all slots are explored and updated regularly at the start of the application (since values are initiated randomly), until the sum of Q-values of one group of slots becomes strictly larger than the rest. In that case we say that the policy has converged and thus exploration has stopped. Due to the implicit exploration scheme explained above, our greedy policy is not vulnerable to local optima since slots are regularly updated. Once messages are not delivered properly, due to a failing node, the 'goodness' of the awake slots will decrease, until a different policy is found, causing nodes to re-learn their wake-up schedule (usually within 10–12 iterations). The speed of convergence and a possible re-learning is influenced by the duty cycle, fixed by the user, and the learning rate is typically in the range [0.1; 0.2]. A constant learning rate is in fact desirable in a non-stationary environment to ensure that policies will change with respect to the most recently received rewards (Sutton and Barto, 1998).

4 Experimental study

4.1 Experimental setup

We apply our approach on three networks of different size and topology. In particular, we investigate two extreme cases where nodes are arranged in a 5-hop line [Figure 5(a)] and a 6-node single-hop mesh topology [Figure 5(b)]. The former one requires nodes to synchronise in order to successfully forward messages to the sink. Intuitively, if any one node is awake while the others are asleep, that node would not be able to forward its messages to the sink. Conversely, in the mesh topology it is more beneficial for nodes to fully desynchronise to avoid communication interference with neighbouring nodes. Moreover, the sink is able to communicate with only one node at a time. The third topology is a four by four grid [Figure 5(c)] where sensing agents need to both synchronise with some nodes and at the same time desynchronise with others to maximise throughput and network lifetime. The latter topology clearly illustrates the importance of combining synchronisation and desynchronisation, as neither one of the two behaviours alone achieves the global system objectives. Subsection 4.2 will confirm these claims and will elaborate on the obtained results.

Figure 5 The three experimental topologies, (a) line topology (b) mesh topology (c) grid topology



Each network was ran for 3600 seconds in the OMNeT++ simulator (http://www.omnetpp.org/ – a C++ simulation library and framework). Results were averaged over 50 runs with the same topologies, but with different data sampling times. Table 1 summarises the differences between the experimental parameters of the three topologies.

Table 1 Parameters for experimental topologies

Topology type	*Num. of nodes*	*Num. of hops*	*Avg. num. of neighbours*	*Data rate (msg/sec)*
Line	5	5	1.8	0.10
Mesh	6	1	3.0	0.33
Grid	16	4	3.25	0.05

Frames were divided in $S = 100$ slots of ten milliseconds each, and we modelled five different events, namely overhearing ($r = 0$), idle listening ($r = 0$ for each idle slot), successful transmission ($r = 1$ if ACK received), unsuccessful transmission ($r = 0$ if no ACK received) and successful reception ($r = 1$). Maximising the throughput requires both proper transmission as well as proper reception. Therefore, we treat the two corresponding rewards equally. Furthermore, most radio chips require nearly the same energy for sending, receiving (or overhearing) and (idle) listening (cf., Table 2), making the three rewards equal. We consider these five events to be the most energy expensive or latency crucial in wireless communication. Additional events were also modelled, but they were either statistically insignificant (such as busy channel) or already covered (such as unsuccessful transmissions instead of collisions).

Due to the exponential smoothing nature of the reward update function (cf., Subsection 3.4) the Q-values of slots will be shifted towards the latest reward they receive. We would expect that the 'goodness' of slots will decrease for negative events, and will increase for successful communication. Therefore, the feedback agents receive is binary, i.e., $r^i_{s,e} \in \{0, 1\}$, since it carries the necessary information. Other reward signals were also evaluated, resulting in similar performance.

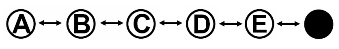
To better illustrate the importance and effect of combining synchronisation and desynchronisation in these topologies, we compare our approach to networks where all nodes wake up at the same time. All other components of the compared networks, such as the routing and communication protocols, remain the same. In other words, we compare our RL technique to networks with no coordination mechanism, but which employ some means of time synchronisation, the small overhead of which will be neglected for the sake of a clearer exposition. It is worth mentioning that both the learning and the synchronous protocols use a carrier sense multiple access (CSMA) scheme that is designed to minimise packet collisions. The synchronised approach ensures high network throughput and is already used by a number of state-of-the-art MAC protocols, such as RL-MAC (Liu and Elhanany, 2006) and S-MAC (Ye et al., 2004). However, as we will demonstrate in Subsection 4.2, synchronisation alone will in fact decrease system performance.

4.2 Evaluation

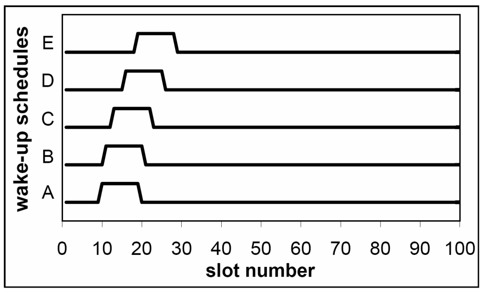
Figure 6(b) displays the resulting schedule of the line topology [Figure 6(a)] after the action of each agent converges. The results indicate that nodes have successfully learned to stay awake at the same time in order for messages to be properly forwarded to the sink. In other words, we observe that all nodes belong to the same coalition, as suggested in Figure 4. If any one node in the line topology had remained active during the sleep period of others, its messages, together with those of its higher hop neighbours would not have been delivered to the sink. Even though neighbouring nodes are awake at the same time (or have synchronised), one can see that schedules are slightly shifted in time. The reason for this desynchronisation is to reduce overhearing of messages from lower hop nodes and to increase throughput – a behaviour that nodes have learned by themselves. The size of this time shift depends on the time at which nodes send data within their active period. As mentioned in Subsection 2, messages are sent at a random slot within a frame. Therefore, the difference between the wake-up times is small enough to increase the chance of successful transmissions and large enough to ensure fast throughput, compensating for propagation delays.

In the line topology this time shift is, however, marginal to the active period and therefore the performance of the learning nodes is comparable to that of the fully synchronised network. This effect can be observed in Figure 6(c), which displays the average end-to-end latency over 50 runs for the learning and the synchronised nodes respectively. We can conclude from the graph that in a line topology, where nodes have no neighbours on the same hop, the improvements of our learning algorithm are marginal to that of a synchronised network. As mentioned above, the reason for this comparable performance lies in the fact that a successful message forwarding in the line topology requires synchronisation. Nevertheless, our agents are able to independently achieve this behaviour without any communication overhead.

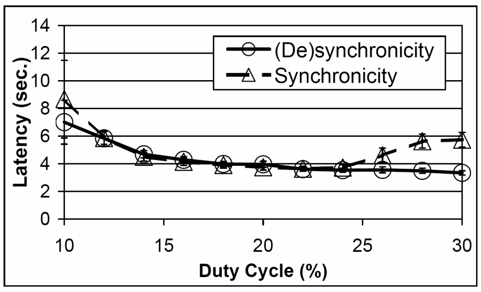
Figure 6 Experimental results for the line topology, (a) line topology (b) resulting wake-up schedule after convergence of one simulation run for duty cycle of 10% (c) average end-to-end latency together with standard deviation bars for different duty cycles



(a)



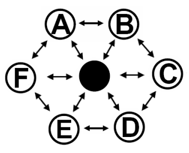
(b)



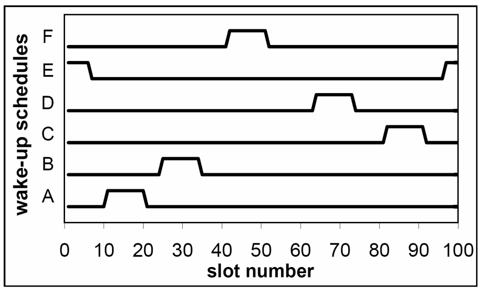
(c)

In contrast to the previous topology, our second set of experiments investigate the performance of the network where all nodes lie on the same hop from the sink. This setup presents agents with the opposite challenge, namely to find an active period where no other node is awake. The latter behaviour will eliminate communication interference with neighbouring nodes and will ensure proper reception of messages at the sink. Figure 7(b) displays the wake-up schedule of the learning nodes for a duty cycle of 10% after the actions of agents converge. One can observe that desynchronisation has been successfully achieved where each node is active at a different time within a frame. Put differently, each node has chosen a different wake-up slot and therefore belongs to different coalition. The benefit of this desynchronised pattern is clearly evident in Figure 7(c) where we compare the latency between the desynchronised and synchronised approaches. The latter figure represents the average end-to-end latency over 50 runs together with the standard deviation bars. For duty cycles lower than 25% the reduction in latency is significant as compared to a synchronised network of the same topology. It is worth noting that for most values the standard deviation is negligible (i.e., smaller than 0.5), which signifies that the reported averages are consistent across runs.

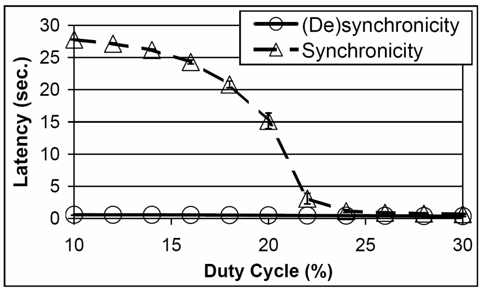
Figure 7 Experimental results for the mesh topology, (a) mesh topology (b) resulting wake-up schedule after convergence of one simulation run for duty cycle of 10% (c) average end-to-end latency together with standard deviation bars for different duty cycles



(a)



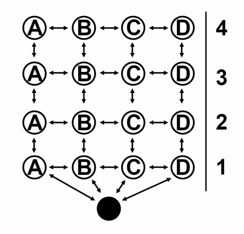
(b)



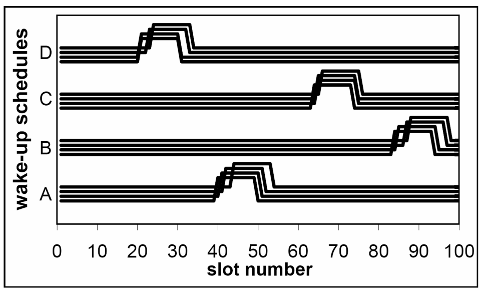
(c)

Lastly, we investigate a ‘mix’ of the above two topologies, namely a grid shown in Figure 8(a). Nodes here need to synchronise with those that lie on the same branch of the routing tree to ensure high throughput, while at the same time desynchronise with neighbouring branches to avoid communication interference. The resulting schedule of the learning nodes after convergence is displayed in Figure 8(b). As expected, the four columns of nodes belong to four different coalitions, where nodes in one coalition are synchronised with each other (being active nearly at the same time) and desynchronised with the other coalitions (sleeping while others are active). Nodes in one coalition exhibit comparable behaviour to those in a line topology, i.e., they have synchronised with each other, while still slightly shifted in time. At the same time nodes on one hop have learned to desynchronise their active times similar to the mesh topology.

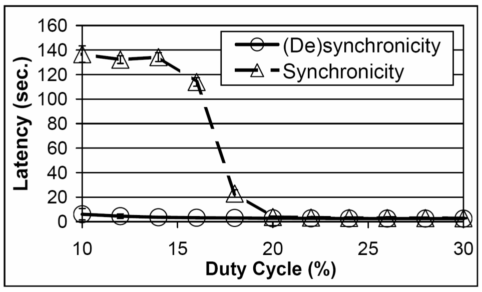
Figure 8 Experimental results for the grid topology, (a) grid topology (b) resulting wake-up schedule after convergence of one simulation run for duty cycle of 10% (c) average end-to-end latency together with standard deviation bars for different duty cycles



(a)



(b)



(c)

The result of applying our learning approach in a grid topology for various duty cycles can be observed in Figure 8(c). It displays the average end-to-end latency of the network together with standard deviation bars when using synchronisation alone and learning. For duty cycles lower than 20% the improvements of our learning nodes over a synchronised network are outstanding. Due to the high data rate, the synchronised nodes are incapable of delivering all packets within the short active period, resulting in increased latency and high packet loss. The high amount of dropped packets at low duty cycles is simply intolerable for real-time applications. This reduced performance at low duty cycles is due to the large number of collisions and re-transmissions necessary when all nodes wake up at the same time. The learning approach on the other hand drives nodes to coordinate their wake-up cycles and shift them in time, such that nodes at neighbouring coalitions desynchronise their active periods. When nodes coordinate their actions, they effectively reduce communication interference with neighbouring nodes. This behaviour results in lower amount of overheard packets, less collisions and therefore fewer retries to forward a message, as compared to the fully synchronised network. Since the duration of the duty cycle directly influences energy consumption, using our learning scheme one can increase the energy-efficiency (and therefore the lifetime) of the WSN by lowering the active period of nodes. As our experiments indicate, this reduction in the duty cycle is possible without sacrificing throughput.

Finally, we would like to mention the convergence rate of the learning agents. The implicit exploration scheme, described in Subsection 3.5 makes nodes select different actions in the beginning of the simulation in order to determine their quality. As time progresses, the Q-values of slots are updated sufficiently enough to make the policy of the agents converge. We measured that after 500 iterations at most the actions of agents in the four by four topology do not change and thus they successfully have both synchronised and desynchronised their actions. In other words, after 500 seconds each node in the grid topology finds the wake-up schedule that improves message throughput and minimises communication interference. This duration is sufficiently small compared to the lifetime of the system for a static WSN, which is in the order of several days up to a year depending on the duty cycle and the hardware characteristics (Ilyas and Mahgoub, 2005). The convergence time for the mesh and line topologies was empirically measured to be at most 200 and 100 seconds respectively.

For illustrative purposes, we assess the benefits of the proposed approach in terms of energy and lifetime by relying on the following standard settings. Sensor nodes are usually powered by a pair of 1.5 V AA batteries (Ilyas and Mahgoub, 2005), whose combined capacity amounts to about $C =$ 7,500 mWh. In terms of energy, this amounts to $E =$ 7.5 * 3,600 = 27,000 J. The energy consumed by sensor nodes is largely dominated by the radio. For example, a node using the CC2420 radio consumes on average 60 mW, i.e., 60 mJ/s (cf., Figure 2). Without relying on a duty-cycle, the lifetime of such a sensor node is therefore about $\frac{27{,}000}{0.06} = 450,000$ s, i.e., slightly more than five days.

Duty cycling allows to reduce the energy consumption by a factor proportional to the duty cycle. For a duty cycle of $D =$ 20% (approximately the duty cycle below which performances of the synchronised approach degrades), the lifetime is significantly extended to $\frac{450{,}000}{0.2} = 26$ days. The proposed desynchronised approach allows to further reduce the duty cycle to 10% without sacrificing latency and doubles the lifetime to 52 days. The convergence time overhead of 500 seconds for the learning agents is therefore marginal to the gains obtained in terms of network lifetime.

4.3 Discussion

Some WSN applications require longer service duration due to the costly deployment and maintenance of nodes. This can be achieved by lowering the duty cycle of nodes and therefore preserving their battery. However, for low duty cycles, nodes are required to forward more messages during their short active times. The above experiments demonstrate that both synchronisation and desynchronisation are crucial in such settings for applications that require real-time data. Our approach drives agents to coordinate their wake-up cycles without any communication overhead. This behaviour ensures that neighbouring nodes avoid communication interference at no energy expense. The synchronised approach, on the other hand, lets all nodes be awake at the same time.

Thus, for low duty cycles nodes increase the chance of collisions and therefore re-transmissions. Based on our results we can conclude that synchronising as well as desynchronising the actions of agents increases the system performance in mesh and grid topologies for low duty cycles as compared to a standard, fully synchronised approach, where all nodes wake-up at the same time. Although our learning agents in the line topology perform similarly to synchronised nodes, we achieve this synchronisation with no communication overhead.

When duty cycles of (synchronised) agents become larger, nodes have less chance of collisions and hence re-transmissions, leading to decreased latency and no packet loss. As mentioned in Subsection 2, packets are sent at random slots within the active period. Thus, the negative effect of being awake at the same time becomes less pronounced as the duty cycle increases. Similarly, as the number of messages per active period decreases, the learning agents receive less reward signals, leading to slower convergence and poorer adaptive behaviour. In such cases it is less beneficial to apply our RL approach, as it performs similar to the fully synchronised one. Nevertheless, synchronised nodes still overhear packets of neighbours, resulting in higher battery consumption as compared to nodes that use our learning algorithm. Moreover, we achieve both synchronisation and desynchronisation in a decentralised manner with no explicit form of coordination, whereas full synchronisation requires periodical calibration packets.

We would like to note that frames or slots of nodes need not be aligned. We have overlaid all wake-up schedules of the corresponding topology on the same graph for a clearer comparison. Individual nodes have no information on the schedules of neighbours, nor need they model other agents in order to coordinate their actions. Nodes learn to cooperate by simply observing their own reward signals while forwarding messages. The learning behaviour of agents therefore ensures that

1 nodes forward messages quickly to the sink and therefore increase throughput

2 they improve their energy efficiency by avoiding collisions and overhearing.

Finally, the convergence times reported in Section 4.2 were obtained experimentally. A deeper analysis of the convergence properties should be carried out, in order to better establish the convergence behaviour of the proposed approach in a wider range of settings. The main factors which are likely to affect the convergence duration are the number of slots in a frame, the number of agents and the radio connectivity. Although we cannot yet offer analytical results regarding the convergence properties, it is worth mentioning that for all our simulations, the proposed approach converged after a few hundred iterations. As noted in Section 4.2, this duration was in our settings negligible in comparison to the gains obtained in terms of energy and lifetime.

5 Conclusions

This paper presented a decentralised RL approach for energy-efficient wake-up scheduling in WSNs. Our approach drives nodes to coordinate their wake-up cycles based only on local interactions. In doing so, sensor nodes independently learn both to synchronise their active periods with some nodes, so that message throughput is improved, and at the same time to desynchronise with others in order to reduce communication interference. As a result, our learning protocol enables users, based on the WSN application requirements, to reduce the duty cycle of the system and increase its lifetime without sacrificing its functionality.

We applied this self-organising RL approach to nodes in a WSN and compared its performance to a standard, fully synchronised network. We investigated three different topologies and showed that agents are able to independently adapt their duty cycles to the routing tree of the network. For high data rates this adaptive behaviour improves both the throughput and lifetime of the system, as compared to a fully synchronised approach where all nodes wake up at the same time. We demonstrated how initially randomised wake-up schedules successfully converge to being both synchronised and desynchronised without any form of explicit coordination.

Several optimisations can be brought to the proposed approach. Our current focus is put on minimising the number of active time slots within a frame, and on relaxing the constraint of having the active time of nodes in one contiguous period. Future work will investigate the convergence properties of the proposed algorithm, as well as the analysis of its behaviour on real-world testbeds.

References

Agarwal, Y., Gupta, R. and Schurgers, C. (2005) 'Dynamic power management using on demand paging for networked embedded systems', in *Proceedings of the Asia and South Pacific Design Automation Conference*, Vol. 2, pp.755–759.

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'A survey on sensor networks', *IEEE Communications Magazine*, Vol. 40, No. 8, pp.102–114.

Al-Karaki, J.N. and Kamal, A.E. (2004) 'Routing techniques in wireless sensor networks: a survey', *IEEE Wireless Communications*, Vol. 11, No. 6, pp.6–28.

Couto, D.S.J.D., Aguayo, D., Bicket, J. and Morris, R. (2005) 'A high-throughput path metric for multi-hop wireless routing', *Wireless Networks*, Vol. 11, No. 4, pp.419–434.

El-Hoiydi, A. (2002) 'Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks', in *IEEE International Conference on Communications*, Vol. 5, pp.3418–3423.

Goel, S. (2005) 'Etiquette protocol for ultra low power operation in energy constrained sensor networks', PhD thesis, New Brunswick, NJ, USA.

Guo, C., Zhong, L.C. and Rabaey, J.M. (2001) 'Low power distributed MAC for ad hoc sensor radio networks', *GLOBECOM*, Vol. 5, pp.2944–2948.

Hill, J.L. and Culler, D.E. (2002) 'Mica: A wireless platform for deeply embedded networks', *IEEE Micro*, Vol. 22, No. 6, pp.12–24.

http://www.omnetpp.org/ – a C++ simulation library and framework.

Ilyas, M. and Mahgoub, I. (2005) *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, CRC.

Leng, J. (2008) 'Reinforcement learning and convergence analysis with applications to agent-based systems', PhD thesis, University of South Australia.

Liu, Z. and Elhanany, I. (2006) 'RL-MAC: a reinforcement learning based MAC protocol for wireless sensor networks', *Int. J. Sen. Netw.*, Vol. 1, Nos. 3/4, pp.117–124, ISSN 1748-1279.

Shih, E., Bahl, P. and Sinclair, M.J. (2002) 'Wake on wireless: an event driven energy saving strategy for battery operated devices', in *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, pp.160–171.

Sutton, R.S. and Barto, A.G. (1998) *Reinforcement Learning: An Introduction*, MIT Press Cambridge, MA, USA.

van Dam, T. and Langendoen, K. (2003) ‘An adaptive energy-efficient MAC protocol for wireless sensor networks’, in *Proceedings of the First International Conference on Embedded Networked Sensor Systems*, Los Angeles, California, USA, pp.171–180.

Watkins, C.J.C.H. (1989) ‘Learning from delayed rewards’, PhD thesis, University of Cambridge, England.

Woo, A., Tong, T. and Culler, D. (2003) ‘Taming the underlying challenges of reliable multihop routing in sensor networks’, in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ACM, pp.14–27.

Schurgers, C. (2008) ’Chapter wakeup strategies in wireless sensor networks’, in Li, Y., Thai, M.T. and Wu, W. (Eds.): *Wireless Sensor Networks and Applications*, pp.195–217, Springer.

Ye, W., Heidemann, J. and Estrin, D. (2004) ‘Medium access control with coordinated adaptive sleeping for wireless sensor networks’, *IEEE/ACM Trans. Netw.*, Vol. 12, No. 3, pp.493–506, ISSN 1063-6692, doi: http://dx.doi.org/10.1109/TNET.2004.828953.