Decentralized Reinforcement Learning for Wake-up Scheduling

Mihail Mihaylov[1], Yann-Aël Le Borgne[1], Ann Nowé[1], and Karl Tuyls[2]

[1]Vrije Universiteit Brussel and [2]Maastricht University

Abstract. Latency is a key issue in the application of delay-critical sensor nets. Reducing latency requires non trivial synchronization of the sensor nodes. This work presents a decentralized reinforcement learning algorithm allowing to decrease the end-to-end latency. Each node learns on-line the most beneficial period to stay awake, such that it avoids idle listening and overhearing. We illustrate this approach and using the OMNET++ simulator we show that it reduces end-to-end latency in a network while preserving its energy efficiency.

1 Decentralized Learning

The nodes of a WSN are limited in power, processing and communication capabilities, which requires that they optimize their activities. The focus of our work is on periodic data collection, where sensor nodes must periodically communicate their measurements to a centralized unit referred to as *sink*. We show that using a decentralized reinforcement learning approach, the WSN achieves lower end-to-end latency with (i.) no communication overhead, (ii.) no clock synchronization, (iii.) at no energy expense and (iv.) without the need for a central mediator.

In our learning environment, a data collection round (or epoch) is divided in S slots, and we assume that the duty-cycle is fixed. We use Q-learning as a Reinforcement Learning algorithm [Watkins C., Learning from delayed rewards, PhD thesis, University of Cambridge.]. Every node stores a "quality-value" (Q-value) for each slot within its epoch. This value for each slot indicates how beneficial it is for the node to stay awake during that slots for every frame, i.e. what is the most optimal 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 wake period (idle listening), that node updates the quality-value of the slot(s) when this event happened. Nodes update their quality values using the following update rule: $Q_s^i \leftarrow (1-\lambda) \cdot \hat{Q}_s^i + \lambda \cdot r_s^i$, where $Q_s^i \in \{-1, \ldots, 1\}$ is the quality of slot $s \in \{1, \ldots, S\}$ within the epoch of agent i. Intuitively, 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 Q-value $\hat{Q}_s^i$ of the previous epoch for that slot, the learning rate $\lambda \in \{0..1\}$, that decreases with time, and the newly obtained reward r_s^i for the event that (just) occurred in slot s.

When forwarding messages to the sink, over time, nodes acquire sufficient information on "slot quality", to determine the best period within the epoch to

stay awake. For example, if the required duty cycle of the nodes is set to 10%, each node will stay awake for those 10 consecutive slots within its epoch (of 100 slots) that have the highest sum of Q-values. This behaviour makes neighbouring nodes (de)synchronize their actions, resulting in faster message delivery and thus lower end-to-end latency.

2 Results and Conclusion

Our learning approach was carried out on a 3-hop random network topology with 8 nodes. We chose this type of random topology, since it is not as trivial as a grid topology, yet, the network is small enough to allow for the learning behaviour to be manually analyzed and validated. Epochs were divided in $S = 100$ slots, and we modelled four different events, namely overhearing ($r = -0.5$), idle listening ($r = -0.1$ for each slot), successful transmission ($r = 0.5$ when ACK received) and successful reception ($r = 0.5$). These constants were empirically chosen to maximize the impact of learning.

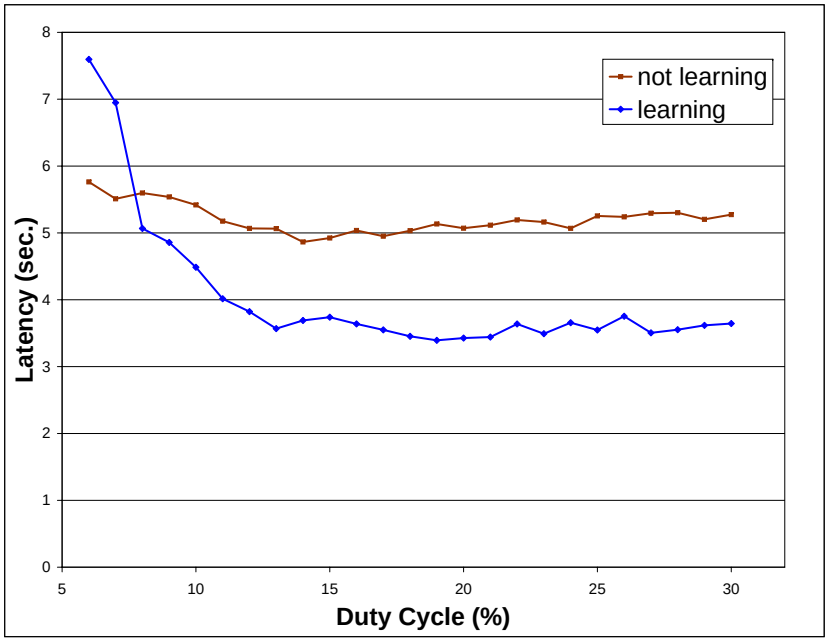

Fig. 1. Average end-to-end latency for different duty cycles with learning [blue line] and without learning [brown line].

The gains in latency resulting from our learning approach are displayed in Figure 1. While for duty cycles less than 8% our learning approach fails to improve on latency, the decrease in latency becomes more apparent as the duty cycle increases. The lower bound on the network latency is dependent on parameters, such as bitrate, frame length, packet size, etc. and was estimated at

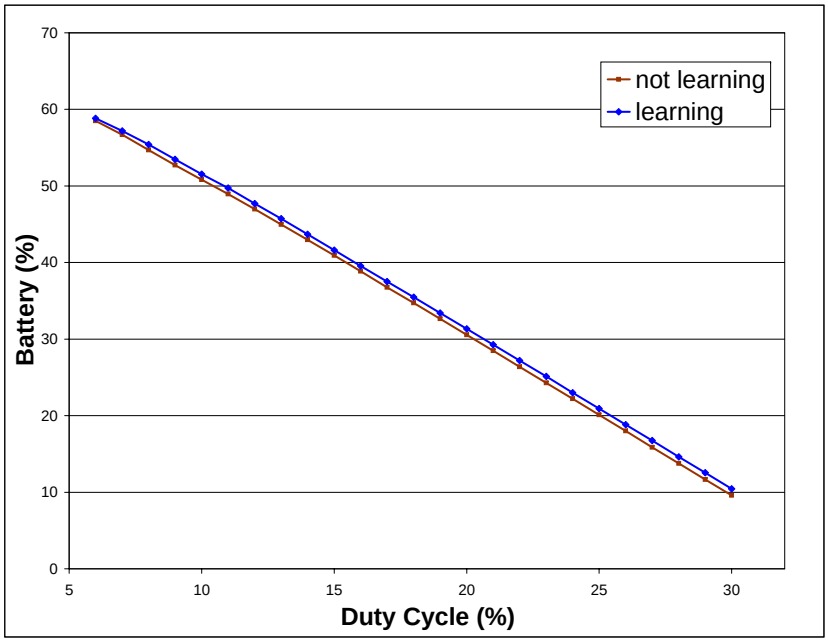

Fig. 2. Average remaining battery of nodes after 3000 seconds of runtime for different duty cycles with and without learning.

3.0343, which our learning approach closely follows. The reason for the decreased performance of the learning algorithm in very low duty cycles is that the nodes spend very little time being awake and thus are not able to learn a good schedule quickly enough.

The learning behaviour of the nodes comes at no energy expense, as it is apparent in Figure 2. The latter figure shows that learning does not result in lower battery life than when not learning. On the contrary, we were able to detect 1% increase in battery life when nodes use our reinforcement learning algorithm. Computational and memory requirements for running the algorithm are minimal and therefore – neglected in the final results.

Using the learning approach we were able to achieve lower end-to-end latency at no energy cost. When forwarding sensor data to the sink, nodes determine the most beneficial period for staying awake and thus adapt to the network topology and data rate at no additional communication cost. The reinforcement learning rule indeed only depends on the activity of the node's radio, and therefore the proposed strategy does not incur any communication overhead. In addition, our algorithm requires no clock synchronization and no central mediator, since each agent learns independently. This makes the proposed approach fully decentralized, and therefore scalable to large networks. Our future work will address the extension of our algorithm to adaptive duty cycles.