

# An Adaptive Strategy for Energy-Efficient Data Collection in Sparse Wireless Sensor Networks

Mario Di Francesco<sup>1</sup>, Kunal Shah<sup>2</sup>, Mohan Kumar<sup>2</sup>, and Giuseppe Anastasi<sup>3</sup>

<sup>1</sup> Center for Research in Wireless Mobility and Networking (CReWMaN)  
University of Texas at Arlington  
`mariodf@uta.edu`

<sup>2</sup> Pervasive and Invisible Computing (PICO) Lab  
University of Texas at Arlington  
`kshah@cse.uta.edu`, `mkumar@uta.edu`

<sup>3</sup> Pervasive Computing and Networking Laboratory (PerLab)  
University of Pisa, Italy  
`giuseppe.anastasi@iet.unipi.it`

**Abstract.** Sparse wireless sensor networks (WSNs) are being effectively used in several applications, which include transportation, urban safety, environment monitoring, and many others. Sensor nodes typically transfer acquired data to other nodes and base stations. Such data transfer operations are critical, especially in sparse WSNs with mobile elements. In this paper, we investigate data collection in sparse WSNs by means of special nodes called Mobile Data Collectors (MDCs), which visit sensor nodes opportunistically to gather data. As contact times and other information are not known a priori, the discovery of an incoming MDC by the static sensor node becomes a critical task. Ideally, the discovery strategy should be able to correctly detect contacts while keeping a low energy consumption. In this paper, we propose an adaptive discovery strategy that exploits distributed independent reinforcement learning to meet these two necessary requirements. We carry out an extensive simulation analysis to demonstrate the energy efficiency and effectiveness of the proposed strategy. The obtained results show that our solution provides superior performance in terms of both discovery efficiency and energy conservation.

## 1 Introduction

Wireless sensor networks (WSNs) have become an enabling technology for a wide range of applications [1]. WSNs are based on sensor nodes which are constrained in terms of their resources: energy, computational power and radio bandwidth. They normally operate in uncertain and dynamic environments where the state of the system changes considerably over time. For example, in data collection applications, uncertainty exists due to intermittent links or traffic conditions. Moreover, the network itself is dynamic due to such events as node mobility and depleted battery. WSN applications need to cope with such dynamic and

uncertain conditions inherent in sensor networks, while simultaneously achieving application-specific QoS requirements. As a consequence, adaptive resource management is a key to any successful middleware solution enabling such applications [2].

In the context of environmental monitoring, a large number of sensor nodes are typically deployed over a geographical area to form a dense ad hoc network. Sensors use multi-hop communication to send data acquired from the external environment to a sink node or to an Access Point (AP) in the infrastructure. However, several environmental monitoring applications do not require fine-grained sensing. Examples of such applications include monitoring of weather conditions in large areas, air quality in urban scenarios, terrain conditions for agriculture, and so on. In this case, it is possible to consider a *sparse wireless sensor network*, i.e., a WSN where the density of nodes is so low that they cannot communicate each other through multi-hop paths, or even directly. In order to make communication feasible, data collection in sparse WSNs can be accomplished by means of *mobile data collectors* (MDCs). MDCs are special mobile nodes responsible for data gathering and/or dissemination. They are assumed to be powerful in terms of data storage and processing capabilities, and are not energy constrained, in the sense that their energy source can be replaced or recharged easily. An MDC can serve either as a Mobile Sink (MS), a mobile node which is also the endpoint of data collection, or as a Mobile Relay (MR), which carries data from sensors to the sink node or an infra-structured AP. In either role, the MDC moves throughout the WSN, and in most cases it is autonomous.

Sparse WSNs with MDCs have many advantages if compared to traditional dense WSNs. First, costs are reduced, since fewer nodes can be deployed, as there is no need for a connected network. Second, as data is collected directly by the MDC from sensor nodes, reliability is improved as a result of less congestion and collisions. Finally, data collection by MDC can extend the WSN lifetime, as the energy consumption is spread more uniformly in the network with respect to dense (static) WSN, where the nodes close to the sink are usually more loaded than the others. However, the data collection paradigm in sparse WSNs with MDCs is different, and introduces significant challenges. Among them, we can mention contact detection and mobility-aware energy conservation schemes.

Communication between an MDC and sensor nodes takes place in two phases. First, sensor nodes discover the presence of the MDC in their communication range. Then, they can transfer collected data to the MDC while satisfying certain reliability constraints, if required. Unlike MDCs, sensor nodes have a limited energy budget, so that the data-collection process has to be energy efficient in order to prolong network lifetime. In addition, such energy-conserving mechanisms should not compromise the timeliness of communication. This is critical especially when the MDC has only a short contact time with sensors, and also in the case when such contacts cannot be predicted accurately. In fact, a major problem in data collection is that sensor nodes usually do not have *a priori* knowledge of the MDC mobility pattern. Furthermore, even in cases where the arrivals can be predicted, there is a chance that the MDC contacts can be

affected by delays or can change their rate. Hence robust and flexible mechanisms have to be defined in order to adapt to operating conditions autonomously. To this end, mechanisms based on artificial intelligence are very appealing, since they are flexible and robust.

In this paper, we address the problem of the MDC discovery by exploiting reinforcement learning, a branch of artificial intelligence targeted to unsupervised learning. We define discovery and data transfer protocols for energy-efficient data collection in sparse WSNs with MDCs, and propose an adaptive strategy exploiting a middleware framework based on *Distributed Independent Reinforcement Learning* (DIRL). The proposed solution is specifically targeted to energy-aware resource allocation in sparse WSNs with MDCs. The remainder of the paper is organized as follows. Section 2 presents an overview of the related work, while Section 3 introduces the system model and the reference scenario. Section 4 describes the data collection application built on top of the DIRL framework. Section 5 outlines the simulation setup and introduces relevant metrics for evaluation. Section 6 discusses the obtained results. Finally, Section 7 concludes the paper.

## 2 Related Work

Solutions for energy-efficient data collection and adaptive resource management in sparse WSNs have already been proposed in the literature. However, in most cases these two issues have been considered separately. Accordingly, we will consider the relevant literature in different sections below.

Several researchers have investigated data collection in sparse WSNs. In [3,4,5] data collection is performed by autonomous MDCs. A data collection scheme is presented in [3], under the assumption that the MDC has a completely predictable mobility. The problem of data collection has been considered also in [4], where both discovery and data transfer are characterized, and the mobility pattern of the MDC is assumed to follow a Poisson distribution. An extensive characterization of data collection in sparse WSNs is provided in [5], where the impact of discovery on data transfer is evaluated. The major limitation of these solutions is that they assume static operating parameters. An adaptive data collection strategy has been proposed in [6], but the approach does not consider the problem of discovery. Solutions based on reinforcement learning have been proposed in [7,8] for the context of WSNs with mobile elements. However the focus of [7] is more on routing rather than discovery, since the WSN is assumed to be rather dense. On the other side, [8] actually exploits reinforcement learning for discovery, but in the different context of sparse WSNs where nodes operate as peers.

As for middleware solutions, while many papers such as [9,10,11] have addressed resource management in dense WSN, there are only a few works specifically targeted to the same problem in the different context of sparse WSNs. Among them, the Impala [12] middleware architecture has been proposed for application adaptation and update. However, Impala has been specifically targeted to scenarios where all nodes are mobile and act as peers. In addition, the

focus is more on application reconfiguration rather than on resource allocation. More recently, the TINYLIME [13] middleware has been proposed for the specific scenario of sparse WSNs. TINYLIME – which is based on a tuple space model – provides mechanisms to perform data aggregation and tune the activity of nodes in order to save energy. However, the focus of TINYLIME is on the proposed programming abstraction rather than on adaptation and resource management. On the contrary, in this paper we propose an adaptive middleware approach to resource allocation for energy-efficient data collection in sparse WSNs.

### 3 System Overview

Before introducing the adaptive data collection strategy, it is necessary to present the elements on top of which our proposed strategy is built. First we present two background elements necessary to describe our proposed strategy: (i) a reference scenario; and (ii) the DIRM middleware framework that will be employed for data collection applications.

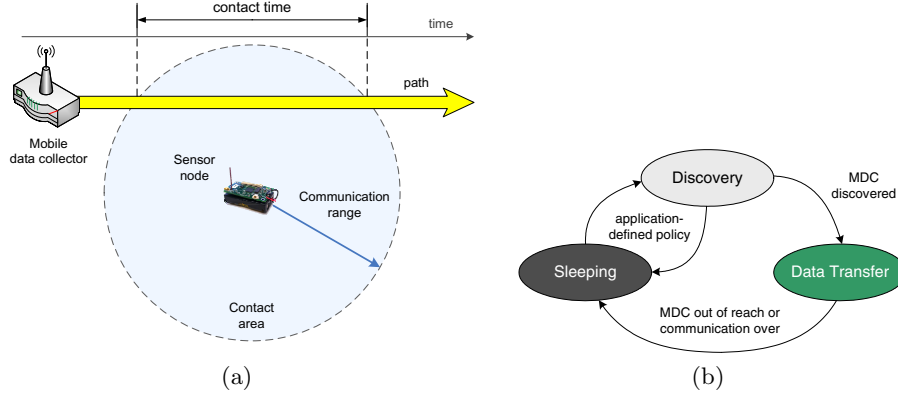
#### 3.1 Network Scenario

The reference network scenario is depicted in Figure 1(a). Specifically, we will consider a single MDC and assume that the network is sparse so that, at any time, the MDC can communicate with at most one static node.

Data collection takes place only during a *contact*, i.e., when the static node and the MDC can reach each other. Furthermore, the area within the communication range of the static node is called *contact area*, and the overall time spent by the MDC inside the contact area is called *contact time*. During a contact, messages exchanged between the MDC and the static node experience a certain message loss, denoted by  $p(t)$ . We also assume that the MDC mobility is not controllable, and we define as *tour* (and denote it with  $T$ ) the smallest time duration after which the mobility pattern repeats [7]. On the other side, we define as *inter-contact time* the actual period of time elapsed from the beginning of a contact to the beginning of the subsequent one.

The overall data collection process can be split into three main phases. Figure 1(b) shows the state diagram of the static sensor node [14]. As MDC arrivals are generally unpredictable, the static node performs a discovery phase for the timely detection of the MDC. Upon detecting the MDC, the static node switches from the discovery state to the data transfer state, and starts transmitting data to the MDC. At the end of the data transfer phase, the static node may switch to the discovery state again in order to detect the next MDC passage. However, if the MDC has a (even partially) predictable mobility, the static node can exploit this knowledge to further reduce its energy consumption [14]. In this case, the static node can go to sleep until the next expected arrival of the MDC.

Similar to [5], we will use an asynchronous discovery protocol and an ARQ-based protocol for data transfer. In detail, the MDC periodically sends special messages called *beacons* to advertise its presence in the surrounding area. The



**Fig. 1.** Reference scenario (a) and state diagram for the static node (b)

duration of a beacon message is equal to  $T_{BD}$ , and subsequent beacons are spaced by a *beacon period*, indicated with  $T_B$ . In order to save energy during the discovery phase, the static node operates with a duty-cycle  $\delta$ , whose active time  $T_{ON} \geq T_B + T_{BD}$  so that a complete beacon can be received during the active time, provided that it wakes up when the MDC is in the contact area.

As soon as it receives a beacon from the MDC, the static node enters the data transfer state. While in this state, the static node remains always active to exploit the contact as much as possible. On the other hand, the MDC enters the data transfer phase as soon as it receives the first message sent by the static node, and stops beacon transmissions. The communication protocol adopted during the data transfer phase is *selective repeat* [15], i.e., a window-based ARQ protocol with selective retransmission, whose window size is assumed to be equal to  $W$  messages. Note that the acknowledgement messages in the ARQ scheme are used not only for implementing a retransmission strategy, but also as an indication of the MDC presence in the contact area.

The data transfer phase ends either when the static sensor has no more messages to transmit during a contact, or the MDC is not reachable any more. However, since the static node generally does not know when the MDC will leave the contact area, it assumes that the MDC has exited the contact area when it misses  $N_{ack}$  consecutive acknowledgments. Similarly, the MDC assumes that the communication is over when it does not receive any message in a given period of time.

### 3.2 Distributed Independent Reinforcement Learning (DIRL)

*Distributed Independent Reinforcement Learning* (DIRL) [16] is a framework that exploits Q-learning [17] to enable autonomous and adaptive applications with inherent support for efficient resource management. The main idea of DIRL is to allow each individual sensor node to self-schedule its tasks and allocate its

resources by learning their usefulness (i.e., their utility) in any given state while honouring application defined constraints and maximizing total amount of reward over time. Q-learning demands minimal computational resources and does not require a model of the environment in order to operate. Hence it is ideal for implementations running on resource-constrained sensor nodes. In addition, DIRL is based on independent learning where each agent applies the learning algorithm in a classic sense (like a single agent system) and ignores the presence of other agents. As a consequence, each sensor node can autonomously and dynamically self-configure in order to maximize its own reward. The main advantage of using independent learning in DIRL is that no coordination is required among sensor nodes, which is beneficial to scenarios where sensors are sparsely deployed.

DIRL uses a simple single step immediate reward and uses a simple weighted hamming distance between two states in order to reduce the state space, which otherwise would be unaffordable for constrained sensor nodes. In addition, DIRL uses the classic exploration and exploitation strategy used in most approaches based on reinforcement learning to get the utilities of the individual tasks. Instead of using the original DIRL exploration policy, we consider a mobility-aware exploration probability based on the number of contacts. More specifically, it is given by

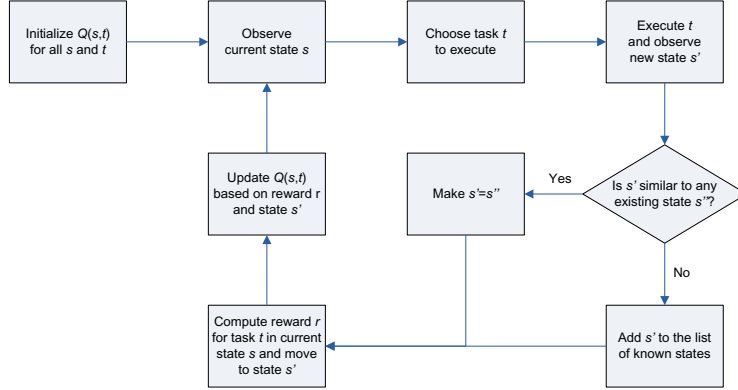
$$\epsilon = \epsilon_{min} + \max(0, k \cdot (c_{max} - c) / c_{max})$$

where  $\epsilon_{max}$  and  $\epsilon_{min}$  define upper and lower boundaries for the exploration factor, respectively;  $c_{max}$  represents the maximum number of contacts (as obtained from the application) after which a steady state condition is likely to be reached, while  $c$  represents current number of detected contacts; finally,  $k$  is a constant that can be tuned to control the descending rate to the minimum exploration probability. Therefore, the heuristic presented above allows initial exploration with a higher rate and gradually decreases over time as DIRL is able to detect up to  $c_{max}$  contacts. Note that some minimum exploration is always required, so as to allow a sensor node to dynamically reconfigure in case of environmental changes.

DIRL needs the following as inputs from the application:

- A set of tasks to be executed, in some priority order. Note here that priority is important only until Q-values, i.e., the learned utilities for all actions in each state, are not established or if two tasks have similar Q-values.
- An applicability predicate associated with each task, incorporating both application-specific constraints and reward functions.
- A state representation consisting of both system and application variables, along with the corresponding weights for deriving the distance between states, and aggregating similar ones.
- The maximum number of states that DIRL should try to explore. This gives an upper bound on number of states in the system. In case of need, a state replacement policy might be introduced, or the hamming distance threshold might be tweaked to accommodate new states into existing (similar) ones.

After obtaining the input from the application, DIRL executes the following algorithm (depicted in Figure 2). Initially all Q-values are set to zero. At each



**Fig. 2.** State diagram for the static node during data collection

time-step DIRL selects a task to execute based on the exploration/exploitation strategy as described earlier. Exploration selects an available task randomly, while exploitation selects the best task according to the learned utilities, i.e., the Q-values. After the execution of a task, DIRL observes the new state  $s'$  and compares it with all existing states based on a hamming distance. Finally, DIRL computes the reward for the executed task in state  $s'$  and updates the corresponding Q-values.

#### 4 Adaptive Data Collection (ADC) Strategy

In this section we define an adaptive strategy based on DIRL for energy-efficient (and reliable) data collection in sparse WSNs. The goal of this strategy is to maximize the percentage of data successfully transferred during contacts, while minimizing the energy consumption of sensor nodes, even in scenarios where the mobility pattern of the MDC is not known in advance. To this end, we defined the different tasks to be used by DIRL with reference to the discovery phase only.

In the context of the reference scenario already introduced in Section 3.1, we have identified three major tasks, each corresponding to a different duty-cycle used for discovering the MDC. In order to make the derivation of tasks more general, we have defined the actual duty-cycles on the basis of a maximum allowed duty-cycle, denoted as  $\delta_{max}$ .

- *High Duty-Cycle (HDC)*. The static sensor is executing a HDC, equal to  $\delta_{max}$ . Ideally this task should be executed whenever the probability of MDC being in the contact area is high.
- *Low Duty-Cycle (LDC)*. The static sensor is executing a LDC, equal to  $0.5 \cdot \delta_{max}$ . Ideally this task should be executed whenever the probability of MDC being in the contact area is low, so that the correspondent energy consumption is very low as well.

- *Very Low Duty-Cycle* (VLDC). The static node executes a VLDC, equal to  $0.1 \cdot \delta_{max}$ . Ideally this task should be executed whenever the probability of MDC being in the contact area is very low, so that the correspondent energy consumption can be considered as almost negligible with respect to the maximum allowed duty-cycle.

As can be seen from the above-mentioned task definitions, the MDC discovery and the successful data transfer process can be maximized while minimizing the energy usage if we can adaptively schedule above tasks based on learned probability of MDC being in contact. DURL learns this probability in the form of utilities built by using local rewards. In order to implement the adaptive data collection strategy, DURL executes the discovery tasks according to the algorithm depicted in Figure 2 and already presented in Section 3.2.

The data transfer is executed as a different process, in the sense that it is not a DURL task. In order to manage this, we have introduced a state variable  $i_c$  which is true when the MDC is assumed to be in contact with the static sensor. In detail,  $i_c$  is set to one when the discovery phase ends with success, i.e., a beacon is successfully received by the static sensor. Moreover,  $i_c$  is set to zero when the static sensor has lost a number  $N_{ack}$  of consecutive acknowledgement messages as a result of the data transfer phase, thus assuming that the MDC has exited the contact area. Hence, the data transfer phase can be entered only after the MDC has been detected (i.e.,  $i_c = 1$ ), under the constraint that messages in transmission<sup>1</sup> are enough to fill a complete window. On the other hand, discovery tasks can always be executed.

For all tasks scheduled by static node, the reward is defined as  $r_t = i_c \cdot e_p - e_s$ , where  $i_c$  is the contact state variable,  $e_p$  is the expected price, and  $e_s$  the energy spent. Note that the expected price is chosen as a multiple of the energy spent for that task, so as to allow a symmetric evaluation of the reward function. Thus, for each task, the reward is equal to the expected price  $e_p$  minus the energy spent  $e_s$  if the MDC has been successfully detected, otherwise it is equal to minus  $e_s$ .

In order to map the presence of the MDC to the specific instants where it is in contact, we have introduced a temporal characterization in the state representation of the static sensor nodes. On the basis of the concept of tour, we split the time (as perceived by a static sensor) into a number of intervals called *time domains*, whose duration is denoted as  $T_d$ . More specifically, each task is scheduled for one time domain, at the end of which utilities are updated and the sensor node evaluates the new state. The granularity of time domain length  $T_d$  represents a trade-off between the storage and computational requirements at the static sensor, and the efficiency of DURL. The lower the value of  $T_d$ , the higher is the accuracy of DURL to schedule the duty-cycle tasks with a fine granularity. On the other hand, a higher number of states increases the overall computation requirements of the learning algorithm.

As all statistics regarding the mobility pattern of the MDC are estimated by the static sensors, we added some filtering techniques to avoid misinterpretation

<sup>1</sup> Messages in transmission can be either buffered messages or messages which have been already transmitted but not yet acknowledged.

of context. For instance, the static sensor might consider a single actual MDC contact as multiple observed contacts. To this end, we implemented a simple timeout technique, so that the static sensor considers the successful reception of a beacon message as a new contact only when a certain time has elapsed since the preceding contact detection. We set this timeout value to 30 s. Similarly, it is required to handle missed contacts, i.e., an actual contact not being detected by the static sensor, as this would result in incorrect learning of the MDC mobility pattern. For this reason, we maintained a short history of contacts (in terms of the time domains where they occur), and adjusted the state evaluation accordingly.

## 5 Simulation Setup

In this section we will evaluate the performance of the DIRM-based adaptive strategy introduced in the previous section. To this end, we will consider the following performance metrics.

- *Activity ratio*, defined as the ratio between the active time and the total time spent during discovery<sup>2</sup>.
- *Discovery ratio*, defined as the average of the ratio between the number of contacts correctly detected by the static sensor and the total number of contacts.
- *Energy efficiency*, defined as the mean energy spent by the static sensor per each message (or byte) correctly transferred to the MDC.

As for the energy expenditure, we implemented a simple model that characterizes the radio, while we do not address the energy expenditure of the CPU, since it is almost negligible. Specifically, the energy expenditure of the radio is calculated as  $P_{state} \cdot T_{state}$ , where  $P_{state}$  and  $T_{state}$  denote respectively, the power consumption of the radio and the amount of time spent in a given state, i.e., receive, transmit and sleep. We assume that the energy consumption of the radio during idle periods, i.e., when it is monitoring the channel, is the same as in the receive state. As for message loss, we used the model considered in [6,5] and based on experimental data measured in a real testbed in the same scenario [18].

In order to compare the performance of DIRM with other approaches, we also considered the following schemes.

- *Random*. At each time step a static node executes a task chosen at random between available ones.
- *SORA*. Static nodes use the heuristic-based reinforcement learning defined in [11].

---

<sup>2</sup> In this metric we do not consider the activity due to data transfer. Hence, the activity ratio is a measure of the average duty-cycle which derives from the executions of the different discovery tasks.

**Table 1.** Parameters used for simulation

Parameter	Value	Parameter	Value
Minimum exploration ( $\epsilon_{min}$ )	0.1	Beacon duration ( $T_{BD}$ )	10 ms
Maximum exploration ( $\epsilon_{max}$ )	0.3	Window size ( $W$ )	16
Descending rate ( $k$ )	0.2	Consecutive lost acks ( $N_{ack}$ )	5
Maximum contacts ( $c_{max}$ )	10	Message payload size	24 bytes
Time domain duration ( $T_d$ )	100 s	Frame size	36 bytes
Message generation interval	10 s	Radio transmit power (0 dBm)	49.5 mW
Expected price ( $e_p$ ) multiplier	10	Radio receive/idle power	28.8 mW
Beacon period ( $T_B$ )	100 ms	Radio sleep power	0.6 $\mu$ W

- *Oracle*. Static nodes have perfect knowledge on MDC contacts, so they do not perform discovery at all. They start transmitting data as soon as the MDC is in the contact area and stop transmitting when there are no more data or the MDC is out of contact.

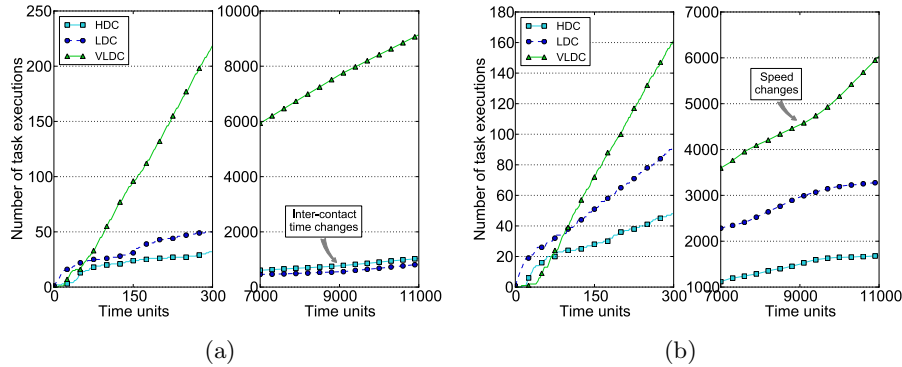
As for the mobility pattern of the MDC, we considered three different scenarios.

- *Deterministic mobility*. The MDC arrivals are periodic, the inter-contact time is fixed. This mobility pattern corresponds to the case where the arrivals of the MDC are known in advance, e.g. when the MDC is a shuttle [3].
- *Gaussian mobility*. The MDC arrivals are periodic, the inter-contact time follows a normal distribution with given mean and variance. This mobility pattern corresponds to the case where the MDC arrivals are rather predictable, but suffer from a certain spread [14]. This can be the case of cars which are affected by traffic conditions.
- *Poisson mobility*. The MDC arrivals are periodic, the inter-contact time is exponential. This mobility pattern corresponds to the case where MDC arrivals are rather unpredictable [4].

We carried out a performance evaluation by using a discrete event simulator written in Java. To derive confidence intervals we used the replication method with a 95% confidence level. In all experiments we performed 10 replicas, each consisting of at least 1000 MDC passages. In the following, we will assume a MICA2 series mote [19] as the static node, and use the related parameters for power consumption. We will assume that the radio is operating at a link speed of 19.6 kbps bitrate. All other simulation parameters, chosen according to the methodology used in [5], are summarized in Table 1.

## 6 Simulation Results

In order to evaluate the performance of the DURL-based ADC strategy, we split simulations in two parts. In the first one, we investigated how the proposed approach reacts to dynamic (transient) conditions, while in the second one we focused on the performance in steady-state conditions. For the sake of clarity, in the following we will consider a single MDC which collects data from a single static sensor node.



**Fig. 3.** Number of task executions over time for variations in inter-contact time (a) and speed (b) of the MDC

### 6.1 Analysis in Dynamic Conditions

As for the analysis in dynamic conditions, we considered two different kinds of variation.

- *Variation in the inter-contact time.* The MDC moves at 20 km/h and starts visiting the sensor node every 1800 s. Then, after some time, the inter-contact time changes to 900 s. This scenario has been considered as representative when the MDC increases the rate of visits, for instance, because there is a need to get fresh data more frequently from the environment.
- *Variation in the speed of the MDC.* The MDC completes a tour in 1800 s and starts visiting the sensor node at a speed of 3.6 km/h. Then, after some time, it changes its speed to 40 km/h while keeping constant the inter-contact time. This scenario has been considered to stress the ability of system to discover the MDC when the contact duration suddenly decreases significantly.

In both cases we set the duration of the simulation to 18000 time units<sup>3</sup> (corresponding to 500 hours of simulated time), and assumed that the variation takes place after 9000 time units (corresponding to 250 hours from the beginning of the experiment). We analyze the ability of our approach to adapt to the operating conditions by means of the (relative) number of task executions as a function of time. We show one representative simulation run for each kind of variation in Figures 3(a) and 3(b) (we have verified that the trend is almost the same also when additional replicas are performed). To highlight the initial learning phase and the reaction to the variation we split the horizontal axis into two parts.

We start considering the variation in the inter-contact time, as shown in Figure 3(a). During the initial phase, where exploration is performed more than

<sup>3</sup> For convenience, we denoted as a *time unit* the interval corresponding to the duration of a time domain, which is equal to 100 s in our experiments.

exploitation, the LDC task is executed more than the other two. However, in steady state conditions (e.g., after 7000 time units), the VLDC task gets the highest number of executions, followed by the HDC and the LDC tasks, respectively. The VLDC task has a high slope, different from the other two tasks. In addition, the number of executions in a HDC task is always higher than those in a LDC task. This happens because the contact time is relatively short, so that it is more convenient (in terms of rewards) to execute the HDC task during the time domains where the MDC is actually in contact with the static node.

In Figure 3(a), it can be noticed that the ADC strategy actually adapts to the new parameters after 9000 time units, when the inter-contact time reduces. More specifically, the VLDC task executions are decreased while the other two tasks are performed more often. This is related to the fact that the inter-contact time is shorter, so that the distance (in terms of time domains) between executions of task deriving from exploitation is shorter. This changes the relative frequency of executions of all tasks.

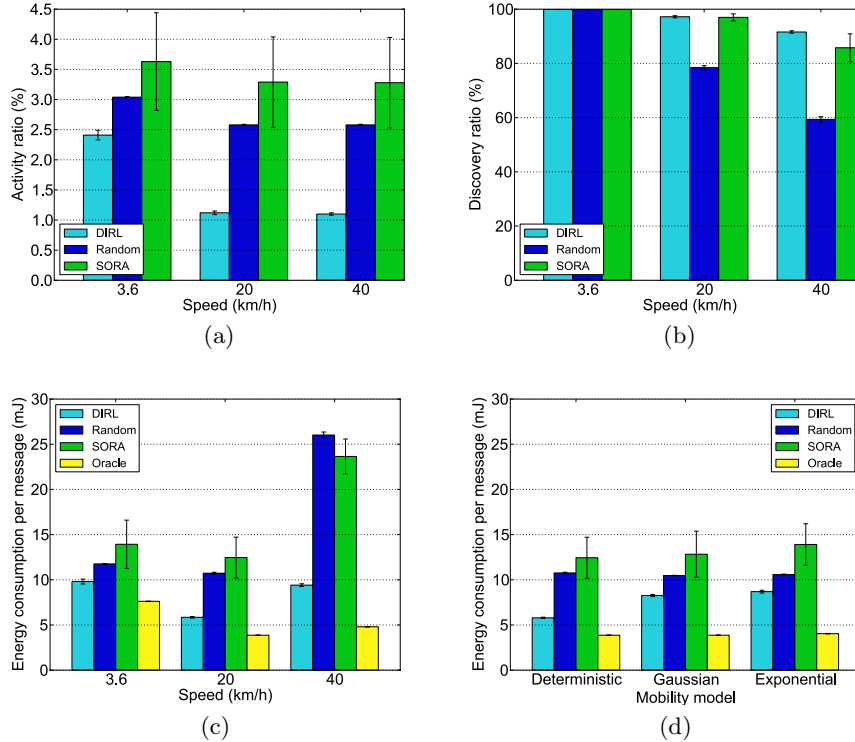
On the other side, the variation in the MDC speed is shown in Figure 3(b). During the initial phase, the LDC task gets the highest number of executions. After a while, similar to the previous case, the VLDC task increases its executions significantly. In stationary conditions, the LDC task is executed more often than the HDC task. In addition, the three plots are closer with respect to the previous case. This is because, when the contact time is longer (when the speed is 3.6 km/h), actually there is little difference between the different tasks. In fact, almost all contacts are detected irrespective of the duty-cycle used for discovery.

The adaptation is also apparent when the MDC speed changes after 9000 time units. In contrast to the previous case, the executions of the VLDC task are increased, while the (higher) duty-cycle tasks are performed less frequently.

## 6.2 Analysis in Stationary Conditions

In this section we evaluate the performance of the DIRL-based ADC strategy in stationary conditions. This gives an indication of how the proposed solution is able to perform when the network is stationary, for instance when it has reached its steady state conditions. Whenever not specified otherwise, we assume that the mobility pattern is deterministic with a 1800 s inter-contact time. All other parameters are as specified in Table 1.

In the first set of experiments we compared our approach – in terms of activity ratio, discovery ratio and energy efficiency – to the other middleware schemes already presented in Section 5 for different speeds of the MDC. As for the activity ratio, from Figure 4(a) we can see that DIRL performs much better than Random and SORA for all MDC speeds. Actually, we can see that the highest activity ratio is obtained for the lowest MDC speed in all cases. That is because when the speed of the MDC is 3.6 km/h, the contact time is long enough so that the discovery tasks gets rewarded in a larger number of time domains, resulting in a higher discovery task being executed more times. When contacts get detected in a lower number of time domains, which happens at the higher speeds, the



**Fig. 4.** Activity ratio (a), discovery ratio (b) and energy consumption (c) as a function of the MDC speed for different middleware schemes. Energy consumption as a function on the mobility pattern for different middleware schemes (d).

activity ratio is actually lower. Apart from this, the results show that DIRL can get a very low activity ratio, in the order of a few percent, and outperforms other approaches such as Random and SORA. More specifically, it seems that SORA cannot efficiently exploit the limited number of rewards in the considered scenario.

The activity ratio alone is not a measure of discovery efficiency, since contacts may be missed as a result of sensors being asleep for most of the time. To this end we considered the discovery ratio, which is given in Figure 4(b). We can see that when the mobility is low, almost all contacts are detected, independent from the adopted middleware scheme. The situation is different, however, when the speed is high (i.e., 20 or 40 km/h). In this case the two middleware schemes based on reinforcement learning (i.e., both DIRL and SORA) clearly get better results than the Random approach. There also is a slight improvement of DIRL over SORA, since it obtains a discovery ratio always over 90%.

The most important metric, indeed is energy efficiency, which can characterize the joint effect of discovery and data transfer. In fact, the discovery efficiency does not give an indication of how effective the discovery is for the data transfer

(in other terms, if contacts are efficiently exploited to transfer buffered data). The results are provided in Figure 4(c) where the energy consumption per correctly transferred message is shown (we also show here the Oracle scheme as a reference). We can see that, as a general trend, the energy expenditure is higher for speeds of 3.6 km/h and 40 km/h, compared to the intermediate speed of 20 km/h. This is against the expected increase in energy expenditure with MDC speed due to the reduction in contact times and also the probability of successful discover. In the 3.6 km/h scenario under the evaluated conditions, the message generation rate is very low with respect to the contact time. As a consequence, since all middleware approaches are performing discovery tasks all the time, they end up completing transfers prematurely, by performing unnecessary discovery. Besides this consideration, the figure clearly shows the advantages of DIRL over the other approaches. DIRL performs much better than SORA because it can exploit the contact more efficiently, in terms of the time actually available for data transfer (and also as throughput per detected contact).

In the second set of experiments, we fixed the speed to the intermediate value of 20 km/h and evaluated the performance of the middleware schemes on the basis of different mobility patterns. For the sake of space, we will focus only on the energy expenditure per transmitted message, which is depicted in Figure 4(d). The goal of this set of experiments is to evaluate how the uncertainty related to the MDC mobility affects the energy consumption. To this end, we ordered the mobility patterns in increasing level of uncertainty: deterministic, Gaussian with a 30 s spread over the mean, and exponential. All mobility patterns use a 1800 s (average) inter-contact time.

As expected, the activity ratio increases when the uncertainty on the mobility pattern of the MDC increases for all middleware schemes. In all cases DIRL performs better than other approaches. This is because DIRL can tune discovery to the actual demand better than the other schemes. In any case, the variance in the energy expenditure is lower for DIRL rather than for SORA, as the former tracks contacts more accurately and efficiently.

In conclusion, proposed solutions using MDC can be effectively used in a wide range of scenarios, even when the contact time is short and the uncertainty on MDC arrivals is high. Thus DIRL results in effective resource allocation while, at the same time, exhibiting very good performance.

## 7 Conclusions

In this paper, a novel Adaptive Data Collection (ADC) strategy for sparse Wireless Sensor Networks (WSNs) with Mobile Data Collectors (MDCs) is proposed. The problem of energy-efficient MDC discovery by exploiting the Distributed Independent Reinforcement Learning (DIRL) framework has been addressed. Our results show that the ADC strategy results in efficient resource allocation, in terms of both low activity needed for discovery and a high data transfer efficiency. Compared to existing solutions, the proposed approach not only performs better, but also can adapt to different operating conditions and mobility patterns

characterized by high uncertainty. As a result, it can be effectively used in the context of sparse WSNs where rewards may be very limited.

Our work can be extended along different directions: (i) define a better characterization of the MDC mobility pattern so that time-domains are automatically derived or tuned; (ii) incorporate information resulting from the data transfer phase into the reward to allow exploitation of the feedback from data collection phase to improve discovery. In addition, we will implement the ADC strategy on real sensor hardware and perform experiments in a WSN testbed.

## Acknowledgments

This work has been carried out while Mario Di Francesco was with the Department of Information Engineering, University of Pisa, Italy.

The research was funded partially by the US National Science Foundation awards CSR 0834493 and CNS 0721951, and partially by the Italian Ministry for Education and Scientific Research (MIUR) under the FIRB ArtDeco and PRIN WiSe DeMon projects.

## References

1. Anastasi, G., Conti, M., Di Francesco, M., Passarella, A.: Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks* 7(3), 537–568 (May 2009)
2. Hadim, S., Mohamed, N.: Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online* 7(3) (March 2006)
3. Chakrabarti, A., Sabharwal, A., Aazhang, B.: Using predictable observer mobility for power efficient design of sensor networks. In: Zhao, F., Guibas, L.J. (eds.) *IPSN 2003*. LNCS, vol. 2634, pp. 129–145. Springer, Heidelberg (2003)
4. Jain, S., Shah, R., Brunette, W., Borriello, G., Roy, S.: Exploiting mobility for energy efficient data collection in wireless sensor networks. *ACM/Springer Mobile Networks and Applications* 11(3), 327–339 (2006)
5. Anastasi, G., Conti, M., Di Francesco, M.: Reliable and energy-efficient data collection in sparse sensor networks with mobile elements. *Performance Evaluation* 66(12), 791–810 (December 2009)
6. Anastasi, G., Conti, M., Monaldi, E., Passarella, A.: An adaptive data-transfer protocol for sensor networks with Data Mules. In: *WoWMoM 2007: Proceedings of the 8<sup>th</sup> IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1–8 (2007)
7. Baruah, P., Urgaonkar, R., Krishnamachari, B.: Learning-enforced time domain routing to mobile sinks in wireless sensor fields. In: *LCN 2004: Proceedings of the 29<sup>th</sup> Annual IEEE International Conference on Local Computer Networks*, pp. 525–532 (2004)
8. Dyo, V., Mascolo, C.: Efficient node discovery in mobile wireless sensor networks. In: Nikolettseas, S.E., Chlebus, B.S., Johnson, D.B., Krishnamachari, B. (eds.) *DCOSS 2008*. LNCS, vol. 5067, pp. 478–485. Springer, Heidelberg (2008)
9. Heinzelman, W., Murphy, A., Carvalho, H., Perillo, M.: Middleware to support sensor network applications. *IEEE Network* 18(1), 6–14 (January 2004)

10. Marron, P.J., Lachenmann, A., Minder, D., Hahner, J., Sauter, R., Rothermel, K.: TinyCubus: a flexible and adaptive framework sensor networks. In: EWSN 2005: Proceedings of the 2<sup>nd</sup> European Workshop on Wireless Sensor Networks, pp. 278–289 (2005)
11. Mainland, G., Parkes, D.C., Welsh, M.: Decentralized, adaptive resource allocation for sensor networks. In: NSDI 2005: Proceedings of the 2<sup>nd</sup> Symposium on Networked Systems Design & Implementation, pp. 315–328 (2005)
12. Liu, T., Martonosi, M.: Impala: a middleware system for managing autonomic, parallel sensor systems. In: PPOPP 2003: Proceedings of the 9<sup>th</sup> ACM SIGPLAN symposium on Principles and practice of parallel programming, pp. 107–118 (2003)
13. Curino, C., Giani, M., Giorgetta, M., Giusti, A., Murphy, A.L., Picco, G.P.: Mobile data collection in sensor networks: The TINYLIME Middleware. Elsevier Pervasive and Mobile Computing Journal 4(1), 446–469 (December 2005)
14. Jun, H., Ammar, M., Zegura, E.: Power management in delay tolerant networks: A framework and knowledge-based mechanisms. In: SECON 2005: Proceedings of the 2<sup>nd</sup> IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, pp. 418–429 (2005)
15. Kurose, J.F., Ross, K.W.: Computer Networking – A Top-Down Approach Featuring the Internet, 5<sup>th</sup> edn. Addison-Wesley Professional, Reading (2009)
16. Shah, K., Kumar, M.: Distributed independent reinforcement learning (DIRL) approach to resource management in wireless sensor networks. In: MASS 2007: Proceedings of the 4<sup>th</sup> IEEE International Conference on Mobile Adhoc and Sensor Systems), pp. 1–9 (2007)
17. Watkins, C.J., Dayan, P.: Q-learning. Machine Learning 8(3), 279–292 (1992)
18. Anastasi, G., Conti, M., Gregori, E., Spagoni, C., Valente, G.: Motes sensor networks in dynamic scenarios. International Journal of Ubiquitous Computing and Intelligence 1(1) (April 2007)
19. Crossbow Technology: Mica2 wireless measurement system, [http://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/MICA2\\_Datasheet.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf)