Open access • Journal Article • DOI:10.1007/S10458-011-9179-0

Long-term information collection with energy harvesting wireless sensors: a multi-armed bandit based approach — Source link

Long Tran-Thanh, Alex Rogers, Nicholas R. Jennings

Institutions: University of Southampton

Published on: 01 Sep 2012 - Autonomous Agents and Multi-Agent Systems (Springer US)

Topics: Energy management, Wireless sensor network, Multi-armed bandit, Routing (electronic design automation) and Throughput

Noname manuscript No.
(will be inserted by the editor)

Long–Term Information Collection with Energy Harvesting Wireless Sensors: A Multi–Armed Bandit Based Approach

Long Tran-Thanh ·
Alex Rogers ·
Nicholas R. Jennings

Received: date / Accepted: date

Abstract This paper reports on the development of a multi–agent approach to long-term information collection in networks of energy harvesting wireless sensors. In particular, we focus on developing energy management and data routing policies that adapt their behaviour according to the energy that is harvested, in order to maximise the amount of information collected given the available energy budget. In so doing, we introduce a new energy management technique, based on multi–armed bandit learning, that allows each agent to adaptively allocate its energy budget across the tasks of data sampling, receiving and transmitting. By using this approach, each agent can learn the optimal energy budget settings that give it efficient information collection in the long run. Then, we propose two novel decentralised multi–hop algorithms for data routing. The first proveably maximises the information throughput in the network, but can sometimes involve high communication cost. The second algorithm provides near–optimal performance, but with reduced computational and communication costs. Finally, we demonstrate that, by using our approaches for energy management and routing, we can achieve a 120% improvement in long-term information collection against state–of–the–art benchmarks.

1 Introduction

Due to their flexibility, low cost and ease of deployment, networks of wireless sensors are being used in a wide variety of applications ranging from environmental, habitat and traffic monitoring, to object tracking and military field observations (Chong and Kumar, 2003; Merrett, 2008; Rogers *et al.*, 2009; Romer and Mattern, 2004). Specifically, in this context, a *wireless sensor network* (WSN) is viewed as a network of small, densely deployed, spatially decentralised autonomous sensor devices (referred to hereafter as "nodes") communicating through a wireless communication network, whose task is monitoring physical or environmental conditions including, but not limited to, temperature, sound, vibration, pressure, seismic, infrared, magnetic and motion information (Baldus *et al.*, 2004; Juang *et al.*, 2002; Simon *et al.*, 2004).

These networks are typically deployed for collecting information from a heterogeneous and dynamically changing environment (i.e. the characteristics may vary over both space and time), and are typically required to operate over an extended period of time (covering months or even years). Within this paper, we typically focus on environmental changes such as varying occurrence of the observed phenomena, varying distribution of collectable information (both in space and time), or network topology changes during operation time. The information collected is then typically forwarded to a base station (BS) (also referred to as a sink or gateway) [1]. Information that arrives at the BS is then processed or transmitted outside the sensor network to end-users for

Intelligence, Agents, Multimedia Group
School of Electronics and Computer Science
University of Southampton, Southampton, UK E-mail: {ltt08r,acr,nrj}@ecs.soton.ac.uk

[1] In most cases, the BS is a specialised node that has significantly more power than the ordinary ones and, depending on the application, there can be more than one of them in the network.

further analysis, fulfilling the specific goals of the WSN deployment. Note that some real–world WSNs requires newest data only, and thus, the value of information that is sampled in the past rapidly decays as time passes by. Such WSNs are typically deployed for real–time target tracking or real–time object localisation (He *et al.*, 2006; Simon *et al.*, 2004). Within these networks, a fundamental goal is to send collected data to the BS as fast as possible (i.e. the data has a strict delivery time constraint). On the other hand, other networks focus on collecting information within a non real–time manner. That is, the deployed network continuously collects information from the surrounding environment, without having the aforementioned strict delivery time constraint (i.e. the collected information can be delayed for a longer time before it is delivered to the BS). Since most of the WSN applications are deployed to fulfil the latter type of monitoring (Chong and Kumar, 2003; Merrett, 2008; Rogers *et al.*, 2009), here we focus on networks where the goal is to collect information over a period of time, in a non real–time manner.

Since the energy capacity of sensor nodes is typically limited, the total operation time of WSNs is often heavily restricted (Akyildiz *et al.*, 2002; Rogers *et al.*, 2009). In particular, limited energy capacity implies rapid depletion of the sensor batteries, which may lead to insufficient data collection from the network. Given this, in order to extend the operation time of sensor networks, a number of recent research works have proposed *energy harvesting sensors*, which have the capability of scavenging ambient energy from their surrounding environment, using solar, vibration, temperature, and radioactive sources (Beeby *et al.*, 2006; Torah *et al.*, 2008; Zhang *et al.*, 2004). Within such settings, sensor nodes typically seek to comply with the concept of *energy–neutrality*, in which the energy consumption of a node should be equal to its harvested energy (Kho *et al.*, 2009; Vigorito *et al.*, 2007). The advantage of such approaches is that sensors can indefinitely extend their life span, which is especially important when information collection has to operate over a prolonged period of time. Given this background, in this paper, we focus in particular on the challenges associated with *efficient long–term information collection* in networks of sensors with rechargeable batteries. That is, we aim to maximise the total information value collected and delivered to the BS in a given extended time interval of operation, or over the entire lifetime of the network, where the nodes follow the concept of energy–neutrality.

Now, to forward data to the BS, sensor nodes can use single-hop (i.e. sensors directly send data to the BS) or multi–hop (i.e. data is forwarded towards the BS via relay nodes) routing scenarios. The former needs no coordination and cooperation between the nodes. However, due the limited communication range of the sensor nodes, this scenario is only feasible when the nodes are deployed in a small area (since the BS needs to be within the communication range of all the nodes). Since WSNs typically cover a significantly larger area, the multi–hop routing scenario is more desirable in such networks. However, this demands an *efficient control mechanism* that coordinates the nodes to collect information. In particular, this mechanism dictates the information collecting and forwarding actions of the sensor nodes, so that the objective of the network deployment can be achieved (Horling *et al.*, 2006; Kho, 2009). Against this background, this paper focuses on the control side of the WSN; that is, we propose a sensory task control mechanism that sensor nodes can use in order to maximise the total amount of information collected within a prolonged period of time.

Such control mechanisms fall into two broad categories: *centralised* and *decentralised*. In the former, a single controller receives information from all the nodes, and then determines the actions of each node indicating how they should sample, receive, forward, and route data. Within this approach, however, the central controller is required to perform a large number of computations in order to find each node's optimal actions. Thus, it often represents a significant computational bottleneck, especially in large networks. Furthermore, since the controller needs to collect all the information from the nodes, there is typically a delay in time before it can start determining the nodes' actions. Given this, the centralised approach may not respond well to the environmental changes, and thus, it can suffer from a lack of ability to efficiently adapt to those changes. In contrast, in decentralised control regimes such a central controller does not exist. Instead, the nodes are autonomous and each decides its individual actions based on its own local state and observations, and those of its neighbouring nodes. Thus, the aforementioned disadvantages disappear within the decentralised control regime (Boukerche, 2008; Wagner and Wattenhofer, 2007). Therefore, in this work, we focus on decentralised control mechanism for WSNs. However, this approach raises several new challenges. Specifically, to achieve system–wide goals, the nodes must typically coordinate their actions with their neighbours (e.g. to forward data or to track objects). In addition, since the nodes typically operate in a dynamically changing environment, they must be able to autonomously adapt their

behaviour, without having global information about the system, in order to achieve the long-term global goals (e.g. maximal information collection or optimal coverage) (Bulka *et al.*, 2007). Such issues naturally lend themselves to a *multi–agent system* (MAS) perspective (Lesser *et al.*, 2003; Pchouek and Mak, 2008; Soh and Tsatsoulis, 2005), in which each sensor is represented by an agent, which autonomously and cooperatively acts, in order to achieve system-wide objectives (Jennings, 2001).

Against this background, the information collection problem that we address consists of a set of energy harvesting sensor agents (i.e. nodes), collecting information from a dynamic environment over an extended period of time, without the aid of a centralised controller. However, due to the limited energy harvesting capacity of the agents (Kansal and Srivastava, 2003), energy efficiency is perhaps the most important issue within the information collection problem (Chong and Kumar, 2003; Stankovic, 2004). In particular, since the agents follow the concept of energy–neutrality, their energy consumption cannot exceed the amount of energy that they can harvest. Given this, it is important to wisely *manage the energy consumption* of the agents, such that they can decide whether to allocate more of this scarce resource to the tasks of sampling, receiving, or transmitting data, in order to achieve maximal long–term information collection. In addition, we also need to develop routing techniques in order to deliver the data to the BS, and thus, to maximise the amount of information collected in the network. Given this, in this paper, we focus in particular on the challenges of energy management and data routing. However, tackling this joint problem of energy management and routing is hard. In particular, each agent has a number of options to allocate amounts of energy to its sensory tasks. In addition, it needs to decide which packet it has to send, and to whom among its neighbouring agents. These options together result in a large task combination space (i.e. the space of combined tasks of energy allocation and packet transmission/receiving), from which the agent has to determine an optimal one (i.e. the task combination that leads to the desired goal of the network). This task combination space is typically exponential, compared to the size of the network, so the joint problem quickly becomes infeasible in terms of complexity. Thus, to simplify the complexity of the original joint problem, we separate the energy management and data routing problems. However, as we will show, by using the solutions of the separated problems, efficient information collection can be still achieved.

In more detail, the decomposition of the original problem can be described as follows. It is based on the observation that by adaptively setting the value of the energy budgets allocated to the various sensory tasks, the agents should achieve better performance in dynamic environments than systems without the ability to adapt in this fashion. However, in order to determine which energy budget allocation combinations are optimal (*exploitation*), the agent first has to *learn* the performance of all the combinations (*exploration*). Thus, it has to balance between exploration and exploitation. Given this, within the energy management problem, we seek for an efficient learning method that finds a trade–off between exploring and exploiting the energy budget allocation combinations, in order to achieve optimal performance of long–term information collection. Now, suppose that all the agents have already set their energy budget value for sampling, receiving, and transmitting tasks. In this case, to maximise the value of the total collected information, it is obvious that we need to maximise the total information value of data sampled or relayed by agents that are one hop from the BS. The latter, however, is equal to data that is sampled or relayed by agents that are two hops from the BS, and so on. Thus, it is also important to maximise the *information throughput* (i.e. the total transmitted information value) between neighbouring *layers* of agents (i.e. the group of agents that are the same distance from the BS) by using efficient routing techniques. This forms the routing problem we aim to solve within our paper.

To solve the energy management problem, we propose a *multi–armed bandit learning* (MAB) based energy management model for each agent within the network. In particular, within the MAB model, each agent chooses from its action set (i.e. set of energy budget allocations) at each round, and, based on this choice, it gets a reward. The goal of the agent is simply to maximise its total reward over a given time period. For the routing problem, we propose two simple decentralised routing algorithms. The first is proveably optimal, but can sometimes use a large number of communication messages to coordinate the routing. The second algorithm is near–optimal, but its communication cost is significantly lower. By using one of the proposed routing algorithms, our approach can calculate the total amount of information throughput that the routing algorithm produces within that particular time step. This amount of information then forms the reward value that the MAB model gets by using the chosen energy budget allocation combination (see Section 4 for more details). With this reward value, the MAB model

gets the feedback about the efficiency of the chosen energy allocation combination, and thus, it can learn which combinations are more efficient ones.

Given this context, this work advances the-state-of-the-art in the following specific ways:

- We introduce the first integrated model for WSNs that considers energy neutrality, adaptability, information content valuation techniques and long-term efficiency.
- We devise the first multi–armed bandit learning based energy budget allocation approach, called MAB/EM. Based on this, we show how efficient energy management can be sustained in the long term, by using this approach.
- We propose two simple decentralised routing algorithms, MITRA and MITRA_τ. The former is the first to proveably maximise the total information throughput between layers of agents, whilst the latter has a near–optimal performance (it achieves, on average, 98% of the optimal solution), but with a reduced communication cost.
- We empirically evaluate the performance of these algorithms through extensive simulations and show that information collection is increased by up to 120%, by applying the proposed algorithms, compared to that of USAC, a state-of-the-art method (see Section 2 for more details of USAC). Furthermore, we show that the communication cost of our approaches are low, compared to the cost of real data transmission.

The remainder of this paper is organised as follows. First, we present related work in this area in Section 2, and detail why it does not meet all our requirements. Following this, we give the formal descriptions of our network model and research objectives in Sections 3. We then discuss our approach for efficient long-term data collection, which includes the MAB learning based energy management method (Section 4), and routing algorithms (Section 5), respectively. Our approach is then empirically evaluated in Section 6. Finally, Section 7 concludes.

2 Literature Review

Previous work on information collection in WSNs has primarily focused on energy efficiency, and thus, aims to lengthen the life span of the network. In particular, a number of energy efficient algorithms use clustering techniques to minimise energy consumption during routing through the rotation of cluster–heads such that the high energy consumption in communicating with the BS is spread across all nodes. These algorithms include *low energy adaptive clustering hierarchy* (LEACH), proposed by Heinzelman *et al.* (2000), and *power efficient gathering in sensor information systems* (PEGASIS), proposed by Lindsey and Raghavendra (2002). In general, these methods make a good attempt to balance the energy consumption by electing cluster–heads, each of which is responsible for relaying the data from a subset of nodes back to the BS in an intelligent way. However, these cluster–heads all need to be placed inside the BS's radio range as they communicate with it directly. Thus, this assumption limits the size of the monitoring environment, since the wireless radio range of the BS is limited. Moreover, these single cluster–heads can become a communication bottleneck of the network, since in each round they need to communicate with a large number of nodes within their cluster. Hence, this aspect contains many of the drawbacks of the centralised control regime.

In addition, energy efficiency can be maintained by reducing the total energy consumption needed to deliver the packets to the BS. From this perspective, Dekorsy *et al.* (2007) proposed an approach that jointly controls the routing and energy management, in order to achieve efficient data forwarding. In particular, their approach aims to minimise the total energy consumption of each node, whilst the collected data has to be delivered to the BS using multipath routing (i.e. there can be multiple routing paths between a node and the BS). In so doing, the approach considers each node's residual energy level, the transmission power level, and maximal communication bandwidth. This approach, however, assumes that the data is already sampled, and that future data is not taken into consideration when optimal routing paths are calculated. Given this, whenever the environment changes (e.g. node failure occurs, or the distribution of collected information changes), it has to recalculate the optimal paths, and thus, it requires significant computational resources.

Another way to lengthen the life span of the network is to perform *energy balancing* (Dinga *et al.*, 2004). That is, to maximise the residual energy level of the bottleneck node (i.e. the node with the lowest energy level) in the

network during the routing. In this vein, Ok *et al.* (2009) used a metric to take the energy cost of transmission, as well as the sensors' remaining energies into account. This metric gives rise to the design of the *distributed energy balanced routing* (DEBR) algorithm, to balance the data traffic of sensor networks in a decentralised manner. Furthermore, Li *et al.* (2007) proposed a global–energy balancing routing scheme (GEBR) for real–time traffic. Now, whilst both of these algorithms perform well in prolonging the lifetime of the WSN, they are not designed for adapting to dynamic environments, since they do not take the environmental changes into account.

More recently, Merrett (2008) developed the *information managed energy aware algorithm for sensor networks* (IDEALS) protocol, which aims to extend the network lifetime of WSNs. IDEALS is an application specific heuristic protocol as it requires that every sensor node decides its individual network involvement based on its own energy state and the importance of information contained in each message. Similarly, USAC (for utility based sensing and communication protocol) uses an *opportunity cost* of the energy used by each sensor to balance the energy consumption of the tasks of sampling and forwarding (Padhy *et al.*, 2010). That is, by evaluating its own opportunity cost, each sensor can decide whether it spends energy on sampling or forwarding, depending on which is the more preferable opportunity for the sensor. Moreover, USAC also considers the total energy consumption required to transmit a packet along a particular path as well. These methods, since they can vary the energy budgets allocated to the sensory tasks, are suitable for adapting to environmental changes. However, they are not designed for sensors with rechargeable batteries, and thus, they do not follow the concept of energy–neutrality. Given this, an increasing percentage of sensor batteries will deplete as time goes by, and thus, the global performance of the network will be decreased in the long term.

Beside the concept of energy–neutrality, another way to achieve efficient long–term information collection in WSNs is to distinguish important from less important data, so that the network can preferentially deliver important data to the BS, in order to maximise the amount of collected information. This is especially vital when communication cost is much higher than other costs, as is often the case with lower power devices (Merrett, 2008). To achieve this, *information content valuation* techniques are typically used to calculate the importance of data (Frieden, 2004; Kho, 2009; Krause *et al.*, 2006). In so doing, an information value metric, such as Fisher information or mutual information, is often used to determine the level of importance of information. Given this, in this paper, we also use the concept of information content valuation, in order to maximise the performance of the network in long–term information.

Within the literature, routing approaches that use the aforementioned information content valuation techniques are typically referred to as *information–centric routing* protocols. One of these algorithms, *directed diffusion* (DD), has been developed by Intanagonwiwat *et al.* (2003). In DD, the BS sends out a data collection query description by flooding the query to the entire network. That is, data collection happens only when the BS needs a certain type of data. However, since data collection applications (e.g environmental monitoring or area surveillance) typically require continuous data delivery to the BS, a significant number of queries will be sent to the network. In this case, the communication cost of DD caused by query floodings is high, meaning DD is not suitable for long–term information collection. To avoid flooding, the *rumour routing* (RR) protocol routes the queries to the nodes that have observed a particular event to retrieve information about the occurrence of the event, and thus, it reduces the total communication cost (Braginsky and Estri, 2002). However, both DD and RR perform well only when the number of events is small, and the events are predictable and periodic. Otherwise, if a large number of unexpected events occur in the future, the algorithms become infeasible, since they cannot predict these events a priori, and thus, the BS does not know where to send the query in order to collect information of the particular events that occur within the network. Moreover, all such information–centric protocols do not take into account the network's dynamism (i.e. the changes of network topology) in forwarding data packets. Given this, these algorithms are not suitable in our settings.

Whilst in DD and RR, information is collected by sending explicit queries from the BS, other methods focus on continuous information collection. That is, they provide information collection, without the need of sending any queries, during the whole operation of the network. For instance, USAC considers the remaining battery power of the communicating nodes and the importance of the data being transmitted, in order to determine the appropriate routing path for the packet. In a similar vein, the *adaptive routing algorithm* (ARA) of Zhou and de Roure (2007) considers the link cost (assumed to be proportional to the distance) between the nodes into account as well. However, these protocols are not designed for solving the maximal information throughput

routing problem, since their main goal is not to maximise the information throughput between the neighbouring layers of sensor agents, but rather to identify optimal paths between each node and the BS, that can be used for forwarding data.

Despite their efficiency in energy consumption, none of the aforementioned approaches is designed to comply with the concept of energy–neutrality. To date, very few studies have focused particularly on data collection in networks of energy harvesting nodes. A notable exception, however, is the work of Kansal and Srivastava (2003). In their work, the authors used spectral estimation functions and prediction filters to estimate the expected amount of harvested energy of each sensor in a given future time interval. Based on these estimations, the sensors are able to schedule their tasks, in order to achieve long-term goals. However, their approach does not consider information content valuation, and thus, will not perform well in information collection. More recently, Kho *et al.* (2010) proposed an energy neutral information–centric data collection algorithm, that combines sampling and routing policies, in order to maximise the collected information in one time slot. That method, however, does not plan for long–term operation, since it does not consider dynamic changes of the environment.

To date, very few approaches have attempted to use learning in the WSN domain. Notable exceptions, however, typically focus on sleep–awake cycle scheduling, or adaptive routing. In the former, Mihaylov *et al.* (2009) proposed a reinforcement learning based technique to extend the life span of the network by learning the most efficient sleep–awake ratio. This topic is outside the scope of our interest, since we assume that our sensors are able to recharge their battery, and thus, they do not need to schedule their sleep–awake cycles. On the other hand, in the latter research area, a number of researchers proposed learning–based techniques to maintain a routing tree that efficiently handles the node failures, creating an efficient connectivity between the nodes to the BS in all circumstances (Zhang and Huang, 2006). In addition, Aghaeil *et al.* (2007) developed swarm intelligence–based algorithms which use learning for data packet routing in WSNs. Furthermore, Gelenbe and Lent (2004) proposed a concept called a *Cognitive Packet Network* (CPN) for intelligent packet forwarding in wireless ad hoc networks. In particular, CPN is an autonomous adaptive *quality of service* (QoS) driven network, which adaptively selects paths so as to offer best effort QoS to the end users based on user defined QoS. CPN uses neural network based reinforcement learning to make routing decisions separately at each node. However, these approaches do not focus on maximising the collected information in a long–term operation, and thus, will not perform well in the long run. More recently, Jain *et al.* (2009) proposed a reinforcement learning based technique in order to efficiently coordinate the sensors over a prolonged period of time. This work can be regarded as most related to our paper, since it also aims to maximise the total amount of rewards (e.g. information value) over the operation time. However, their approach exploits the fact that the environment is static, and thus, the reward matrix (i.e. an action–reward mapping) is static over time. Thus, this approach will not perform well in dynamic environments.

Finally, note that apart from the MAB approach, there exist other, more sophisticated, learning concepts to tackle the information collection problem. This includes, but is not limited to, the following: multi–state Markov decision processes (MDPs) (Sutton and Barto, 1998), partially observable MDPs (POMDPs) (Cassandra, 1998), and decentralised POMDPs (DecPOMDPs) (Seuken and Zilberstein, 2008). However, these learning methods require higher computational complexity, compared to that of the MAB learning. In particular, these learning concepts also take into account the state of the environment, which is modified by the actions of the agents. This implies that within the concepts above, an agent needs to consider a significantly larger space of options (i.e. space of state–action pairs), compared to that of the MAB model, where the space of options only contains the set of the agent's actions. In addition, as we will see later, the MAB approach produces remarkably good results in data collection (see Section 6 for more details).

3 System Models and Problem Definitions

Having described the research objective and related work of our paper, we now introduce a formalisation of the long–term information collection problem for WSNs in this section. To this end, we first provide a formal description of the WSN system in Section 3.1. In particular, we describe the models of adaptive sampling, information content valuation, data routing, and energy management policies that play fundamental roles in efficient information collection of WSNs. Here, we also discuss the assumptions, on which the model formalisation is based.

Following this, in Section 3.2, we formulate the main objective of our research: that is, to achieve efficient long-term information collection in WSNs. Finally, we decompose the information collection problem into the two separate sub–problems described in Section 1: (i) energy management; and (ii) maximal information throughput routing, which we introduce in Sections 3.3, and 3.4, respectively.

3.1 The Wireless Sensor Network Model

In order to formalise the long–term information collection challenge introduced earlier, we first need to introduce a suitable WSN model. Given this, we now present our WSN model, that covers the energy management, sampling, information content valuation, and routing components, respectively. Recall that for all the reasons outlined in Section 1, we pursue a multi–agent system model, whereby sensor nodes are represented as agents.

Now, since the main focus of the paper is on the control side of the WSN, we make the following assumptions about the physical world of the network, in order to simplify the complexity of the model:

- The network that we are studying is not a mobile network (i.e. the agents cannot change their location), however, link failures, node failures and node additions are taken into account. That is, the network can be *topologically dynamic, but not mobile*.
- In our model, the *energy consumption of memory management* (i.e. reading from memory and writing to memory) *is negligible* compared to the energy consumption of data sampling and forwarding. This assumption is reasonable according to the experimental studies reported in Mathur *et al.* (2006) and Anastasi *et al.* (2004).
- We also assume that once the communication channel is set between two nodes, data transmission between these nodes is *perfect* (i.e. no data loss occurs). This assumption is reasonable, espacially in networks where there is a demand of high quality of service (QoS) (Younis *et al.*, 2004). In particular, if the ratio of successful transmission of a communication channel is low (i.e. the QoS is low), then that communication channel cannot be established. In order to guarantee high QoS within WSNs, efficient techniques can be used, such as time synchronisation policies (Degesys and Nagpal, 2008; Elson and Estrin, 2001; Sundararaman *et al.*, 2005), or medium access control (MAC) protocols that control the data transmission of each node (Demirkol *et al.*, 2006; Wu and Biswas, 2007). By using the aforementioned techniques, we can guarantee that no data loss occurs during data transmission.
- Each node can *periodically recharge its battery*, making it independent of human intervention. Here, we use a realistic model taken from existing WSN applications. Such models can be found for example, in Beeby *et al.* (2006), Torah *et al.* (2008), Zhang *et al.* (2004), Roundy *et al.* (2004), or Merrett (2008).

Given this, we can formulate the WSN model as follows. Let $I = 1, 2, \ldots, N$ be the set of agents in the network, which contains one base station, denoted BS[2]. We assume that each agent knows its distance in hops from the BS. This can be achieved by using any of the standard shortest path algorithms (e.g. distributed breadth-first search or distributed Bellman–Ford). Furthermore, each agent can only communicate with those who are inside its communication range, and different agents may have different ranges.

Here, for the sake of simplicity, we split the time line into slots. That is, hereafter we assume that time is discrete, and can be denoted with the sequence of $t = 0, 1, 2, \ldots$. We consider three specific kinds of energy consumption for each agent in the network, namely: the energy required to (i) acquire (i.e. sample); (ii) receive; and (iii) transmit a single data packet (we assume that each packet has the same size in bytes). Given this, let e_i^{S}, e_i^{Rx}, and e_i^{Tx} denote the energy consumption that agent i has to spend for sampling, receiving, and transmitting a single data packet, respectively.

Let B_i denote the average amount of harvested energy of agent i over a single time period. Since the agents comply with the concept of energy–neutrality, the total energy budget agent i can use is equal to the amount of energy that agent i can harvest. Note that this assumption slightly simplifies the real–world models, since in reality the energy consumption limit does not need to perfectly match the amount of harvested energy as long as

[2] Our model can easily be extended to cover systems with multiple base stations.

the battery of the agents is not depleted. However, for the sake of simplicity, we assume that the energy harvesting is stable within this paper [3]. Given this, we assume that the amount of harvested energy per slot is constant, and we denote it with B_i. That is, for each time slot t, the energy consumption of agent i cannot exceed B_i in our settings. In addition, we disregard the energy required for other types of processing since it is negligible in comparison (Mathur *et al.*, 2006; Merrett, 2008).

For data sampling, since our goal is not to develop new sampling techniques, we use existing sampling techniques from the literature. Specifically, we focus on *adaptive data sampling* techniques. Such policies have been advocated as the way to achieve accurate estimates of the environmental conditions, whilst minimising redundant sampling of the environment. Relevant examples can be found in Kho *et al.* (2009), Willett *et al.* (2004), Jain and Chang (2004), or Cover and Thomas (2006). In particular, adaptive sampling techniques often include sets of rules that control a node's *sampling rate* (i.e. how often a node is required to collect data by sampling during a particular time interval) and *sampling scheduling* (i.e. when a node is required to sample). The advantage of adaptive sampling is that it can efficiently deal with the environmental changes by adaptively changing the sampling rate, and thus, is capable of achieving good performance in the long–term. Given this, we aim to use an efficient adaptive method for data sampling in this work.

To calculate the importance of sampled data, we use information content valuation methods. Similar to the sampling case, any existing technique from the literature can be used for this. Such techniques can be found, for example, in Guestrin *et al.* (2005), Osborne *et al.* (2008), or Kho *et al.* (2009). This information content valuation method should assign real values to each of the sampled data packets in the way that more important packets have higher values. Since the environmental characteristics of the network may vary over space and time, the value of data that agents can sample may vary as well. However, the agents within the network do not have *a priori* information about these environmental changes. Furthermore, we also assume that the information value of the collected data is *discounted* over time by a factor $\lambda \in (0, 1]$ (i.e. loses its value as time passes by), if it is not delivered to the BS yet. This assumption is justified by the fact that in many applications, more up to date information is preferable to older information. Since our main focus is on networks without real–time delivery constraints (see Section 1 for more details), we assume that the information discount factor is typically high (i.e. $\lambda > 0.5$). The intuition of this assumption is that with higher information discount factor, the collected information then can be delayed for a longer time, without losing much of its value, before it is delivered to the BS. Note that within our model, the information value of non–collected data (i.e. data that are not sampled yet by the agents) may also decay over time. However, we assume that the underlying sampling method can efficiently sample data so that important data can be collected earlier than less important data.

In existing routing protocols, agents typically forward data to other agents, which are closer to the BS, either in terms of physical distance or number of hops. Thus, following this concept, we assume that in our model, agents can send data to those which are closer to the BS in terms of number of hops. Finally, we assume that data sampled or received at each agent i at slot t can only be forwarded from slot $(t + 1)$. This assumption is also reasonable, since without it, newly sampled data could be delivered to the BS instantaneously.

3.2 The Long–Term Information Collection Problem

Given the model that considers adaptive sampling, routing, information valuation and energy management of WSNs, we now aim to give a formal description of the research objective. That is, to maximise the total collected information in WSNs, in a given finite time interval. In more detail, let $\mathbf{S}_i(t)$, $\mathbf{Rx}_i(t)$ and $\mathbf{Tx}_i(t)$ denote the set of sampled, received and transmitted data packets of agent i at time slot t. Let p denote a single data packet, whose information value at time slot t is $v(p, t)$. Furthermore, we assume that the WSN operates in the finite time interval $[0, T]$. Given this, our objective is formulated as follows:

$$\max \sum_{t=0}^{T} \left\{ \sum_{p \in Rx_{BS}(t)} v(p, t) \right\} \tag{1}$$

[3] The relaxation of this assumption, however, remains as future work (see discussion in Section 7).

Here, $Rx_{BS}(t)$ denotes the set of packets that the BS receives at time slot t. That is, we aim to maximise the total information value delivered to the BS over the time interval $[0, T]$, with respect to the following constraints:

$$\mathbf{Tx}_i(t) \subseteq \mathbf{Q}_i(t) \tag{2}$$

for each agent i and time slot t, where $\mathbf{Q}_i(t)$ is the set of total transmittable data packets in the memory. That is, the set of transmitted data is the subset of the total transmittable data (packets that were sampled or arrived until the previous time slot) of each agent i. Furthermore,

$$\mathbf{Q}_i(t+1) = (\mathbf{Q}_i(t) / \mathbf{Tx}_i(t)) \cup \mathbf{S}_i(t) \cup \mathbf{Rx}_i(t) \tag{3}$$

for each agent i. Note that $\mathbf{Q}_i(t) / \mathbf{Tx}_i(t)$ denotes the set of packets that is in $\mathbf{Q}_i(t)$ but not in $\mathbf{Tx}_i(t)$ (i.e. exclusion). That is, the set of transmittable data of agent i at time slot $(t+1)$ is the union of the sets of residual data (i.e. $(\mathbf{Q}_i(t) / \mathbf{Tx}_i(t))$), the received data and the sampled data at time slot t. For the concept of energy-neutrality, we have the following constraints:

$$e_i^{\mathrm{S}} |\mathbf{S}_i(t)| + e_i^{\mathrm{Rx}} |\mathbf{Rx}_i(t)| + e_i^{\mathrm{Tx}} |\mathbf{Tx}_i(t)| \leq B_i \tag{4}$$

for each agent i, where $|\{.\}|$ denotes the size of set $\{.\}$. Furthermore, e_i^{S}, e_i^{Rx}, and e_i^{Tx} are the costs of sampling, receiving, and transmitting a single data packet, as defined in Section 3.1. This constraint demonstrates that the energy consumption of each action taken by agent i cannot exceed the energy budget given in time slot t.

Furthermore, for each $p \in \mathbf{S}_i(k) \cup \mathbf{Rx}_i(t)$ (i.e. received data or sampled data of agent i at time slot t), that is not delivered to the BS before time slot t:

$$v(p, t+1) = \lambda v(p, t) \tag{5}$$

where $\lambda \in (0, 1]$ is the discount coefficient. That is, the information value of packet p is decayed with the discount factor λ, as time goes by.

As mentioned in Section 1, to efficiently solve the problem formulated in Equation 1, we separate the study of the energy management and routing of the WSN, whilst we assume that efficient sampling and information content valuation can be achieved by using existing techniques. Given this, Section 3.3 discusses the energy management problem in more detail, whilst Section 3.4 focuses on the routing problem.

3.3 The Energy Management Problem

As mentioned in Section 1, the definition of the energy management problem is based on the observation that since each agent can sample, receive or transmit data, it is necessary for the agents to vary the energy budget they associate with each of these action types, so that their overall performance can effectively adapt to environmental changes. That is, by adaptively setting the value of the energy budgets assigned to the sensory tasks, the agents can decide whether to put more effort on sampling (e.g. when significant events are occurring in the monitored area), receiving important data from the others (e.g. when they have collected high value information that has to be delivered to the BS), or transmitting data (e.g. when the delivery of data cannot be delayed too long). With such capabilities, our hypothesis is that the agents should achieve better performance than systems without the ability to adapt in this fashion. However, in order to find the optimal combination of budget allocation, the agents first have to learn the efficiency of each combination, which leads to the dilemma of exploration versus exploitation (see Section 1). In more detail, if the agent only focuses on learning the optimal combination (i.e. exploration), the total collected information of that agent over the operation time might not be maximal, since the agents has to try out all the combinations (including those with low efficiency). On the other hand, if the agent decides to focus on the best combination so far (i.e. exploitation), it may miss the chance to find a better combination that results in better overall performance (i.e. better collected information over a long term). Furthermore, since the environment is dynamic, the optimal combinations may vary over time. Thus, the learning method has to be able to adapt to these environmental changes as well. Given this, by using the notations of the WSN model and the information collection problem described above, the energy management problem can be described as follows:

Definition 1 The *energy management problem*, that we are facing with, is a sequential decision making problem where at each time slot t, each agent i has to choose a combination of energy budget allocations for sampling, receiving, and transmitting, respectively. Following this, agent i evaluates the efficiency of the chosen combination by measuring the amount of sampled, received, and transmitted information within that time slot, with respect to the chosen energy budgets. The goal of each agent i is to find a sequence of decisions (i.e. learning method) that efficiently tackles the trade–off between exploration and exploitation, and the dynamic behaviour of the environment, leading the overall system to achieve maximal long–term information collection.

This problem can be formalised as follows. Since energy harvesting is possible, we follow the concept of energy–neutrality, in order to guarantee the long–term operation of the network (see Section 1 for more details). Consequently, the total amount of energy that agent i can use at each slot is equal to the total harvested energy in that slot. Given this, recall that B_i denotes the average amount of agent i's harvested energy for each time slot t; that is, the amount of energy available for agent i in slot t is B_i.

Now, let $B_i^{\mathrm{S}}(t)$, $B_i^{\mathrm{Rx}}(t)$, and $B_i^{\mathrm{Tx}}(t)$ denote the energy budgets that agent i allocates to sampling, receiving and transmitting at time slot t, respectively. That is, at each time step, the agent makes a decision of choosing values for $B_i^{\mathrm{S}}(t)$, $B_i^{\mathrm{Rx}}(t)$, and $B_i^{\mathrm{Tx}}(t)$. In so doing, it has to take into account the following constraints:

$$\begin{aligned} e_i^{\mathrm{S}} \left|\mathbf{S}_i(t)\right| &\leq B_i^{\mathrm{S}}(t) \\ e_i^{\mathrm{Rx}} \left|\mathbf{Rx}_i(t)\right| &\leq B_i^{\mathrm{Rx}}(t) \\ e_i^{\mathrm{Tx}} \left|\mathbf{Tx}_i(t)\right| &\leq B_i^{\mathrm{Tx}}(t) \end{aligned} \tag{6}$$

for each agent i. These constraints demonstrate that the energy consumption of each action made by agent i cannot exceed the energy budget of each task given in time slot t. Furthermore, we have:

$$B_i^{\mathrm{S}}(t) + B_i^{\mathrm{Rx}}(t) + B_i^{\mathrm{Tx}}(t) \leq B_i$$

Now, it is obvious that by allocating more energy to a sensory task, each agent i can improve the performance of that task (i.e. it can sample, receive, or transmit more data if a higher energy budget is allocated to the tasks of sampling, receiving, or transmitting). Given this, we assume that at each time slot t, each agent i fully allocates its energy budget to its sensory tasks. That is, we have the following modified constraint:

$$B_i^{\mathrm{S}}(t) + B_i^{\mathrm{Rx}}(t) + B_i^{\mathrm{Tx}}(t) = B_i \tag{7}$$

Thus, the set of combinations, from which agent i has to choose one, is the set of energy budget allocations that satisfy Equation 7. Given all this, our goal is to maximise the objective given in Equation 1 by providing an efficient decision making policy to the energy management problem described in Definition 1. Therefore, in Section 4, we propose a MAB learning based approach, in order to efficiently tackle this problem.

3.4 The Maximal Information Throughput Routing Problem

Having described the energy management problem, we now discuss the maximal information throughput routing problem, which aims to maximise the total information that can be forwarded between neighbouring layers (i.e. the group of agents that are the same distance from the BS) of agents. Given this, we group the agents within the network into layers, such that $\mathfrak{L}_l$ denotes the set of agents that are l hops from the BS. Let L denote the number of layers in the network. Note that the BS itself is layer 0. Thus, we have the following:

Definition 2 The *maximal information throughput problem* is the optimisation problem where agents in layer $\mathfrak{L}_l$ have to perform the maximal total information throughput to layer $\mathfrak{L}_{l-1}$ in time slot t, with respect to the energy budgets of each agent.

The formulation of the problem can be described as follows:

$$\max \left\{ \sum_{i \in \mathfrak{L}_l} \sum_{p \in Tx_i(k)} v(p,t) \right\} \tag{8}$$

with respect to the following constraints:

$$E_i^{\mathrm{Tx}} \left|\mathbf{Tx}_i\left(t\right)\right| \leq B_i^{\mathrm{Tx}}\left(t\right) \tag{9}$$

for each $i \in \mathfrak{L}_l$, where $\mathbf{Tx}_i\left(t\right)$ is the set of transmitted data of node i at time slot t, and $v\left(p,t\right)$ is the information value of packet p at t. That is, each sender agent cannot exceed its transmitting energy budget during its data transmission operation. Furthermore,

$$E_j^{\mathrm{Rx}} \left|\mathbf{Rx}_j\left(t\right)\right| \leq B_j^{\mathrm{Rx}}\left(t\right) \tag{10}$$

for each $j \in \mathfrak{L}_{(l-1)}$, where $\mathbf{Rx}_j\left(t\right)$ is the set of received data of node i at time slot t. Thus, each receiver agent cannot exceed its receiving budget during data receiving. Finally, constraints described in Equations 2, 3, and 5, that express the conservation of information within our setting, have to be taken into account as well.

In order to solve this problem, we propose two decentralised algorithms, one is optimal, but with significant communication costs, whilst the other is near–optimal, but with reduced costs. We describe these algorithms in more details in Section 5.

4 Multi–Armed Bandit Based Energy Management

Given the problem definitions described above, we now concentrate on the energy management problem presented in Definition 1. Therefore, we first introduce the foundation of the method used for energy management, namely the multi–armed bandit (MAB) problem, in Section 4.1. Following this, we describe the MAB learning based energy management approach in Section 4.2. Then we analyse the computational complexity of this approach in Section 4.3. In particular, we show that our approach has linear running time, and linear memory usage, compared to the number of each agent's available options of energy budget allocation.

4.1 The Multi–Armed Bandit Problem

The standard *multi–armed bandit* (MAB) problem was originally proposed by Robbins, (1952). In the MAB problem, there is a machine with K arms, each of which delivers rewards, that are independently drawn from an unknown distribution, when the machine's arm is pulled. Given this, a gambler must choose which of these arms to play. At each time slot, he pulls one arm of the machine and receives a reward or payoff. The gambler's purpose is to maximise his return; that is, the sum of the rewards he receives over a sequence of pulls. As the reward distributions differ from arm to arm, the goal is to find the arm with the highest expected payoff as early as possible, and then to keep gambling using that best arm.

Using the terminology of multi–agent systems hereafter, we refer to the gambler as an agent, and refer to each arm pulling action of the gambler as an action of that particular agent. Thus, we can formulate the MAB problem as follows. Let K denote the number of actions that the agent can make. At each time slot t, the agent takes action a_t, which delivers the reward $r_{a_t}\left(t\right)$. Finally, let $T > 0$ denote the time horizon in which the agent operates. Thus, we have the following optimisation problem:

$$\max \sum_{t=1}^{T} r_{a_t}\left(t\right) \tag{11}$$

Thus, the agent has to choose a policy (i.e. a sequence of actions), that may deliver the maximal reward at each time slot t in order to achieve the maximum of Equation 11.

A fundamental dilemma in the MAB problem is the trade-off between exploration and exploitation outlined in Section 1. However, in applications with a dynamic environment, such as WSNs, beside the aforementioned trade-off, we face the challenge that the expected value of the arms may vary over time. In addition, the time (i.e. when) and the magnitude (i.e. how) of the change are unknown to the agent. Specifically, in our settings, these changes can also be due to changes in the behaviour of other agents, that also use learning methods in order to determine their best actions. This indicates that the agent has to repeatedly re–learn the current optimal

Algorithm 1 Algorithm Exp3.Sub

1: **Initialisation:** Let $\gamma \in (0, 1]$, and $w_k(1) = 1$ for $k = 1, 2, ..., K$;
2: **for all** $t = 1, 2, \ldots$ **do**
3: Set $p_k(t) = (1-\gamma)\frac{w_k(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K}$ for $k = 1, 2, \ldots, K$;
4: Draw a_t randomly accordingly to the probabilities $p_1(t), p_2(t), \ldots, p_K(t)$;
5: Receive reward $r_{a_t}(t) \in [A, B]$;
6: **for all** $i = 1, 2, ..., K$ **do**
7: **if** $(i == a_t)$ **then**
8: $\hat{r}_i(t) = \frac{r_i(t) - A}{(B-A)p_i(t)}$;
9: **else**
10: $\hat{r}_i(t) = 0$;
11: **end if**
12: $w_i(t+1) = w_i(t) \exp\left(\frac{\gamma \hat{r}_i(t)}{K}\right)$;
13: **end for**
14: **end for**

arm, since this may also vary as well. Furthermore, this re-learning must be done without knowing when and how the values have changed. To address this complex problem, Auer *et al.* (2003) proposed a simple learning technique, Exp3, that has been shown to achieve efficient performance in dynamic environments. Other MAB techniques, such as Adapt-EvE (for adaptive exploration versus exploitation) (Hartland *et al.*, 2006), and D-MAB (for dynamic multi–armed bandit) (DaCosta *et al.*, 2008), are also suitable for dealing with the dynamic environmental changes. However, these techniques rely on the assumption that the reward value a single agent receives is piece-wise stationary (i.e. the distribution of the reward value remains the same in intervals of operation time), which does not hold in our case. In particular, in our settings, the reward value that an agent receives depends on the environment and the behaviour of other agents as well. Thus, piece-wise stationarity cannot be guaranteed (see Section 4.2 for more details). On the other hand, Exp3 does not have any assumptions about the environment of the agent, and thus, it is suitable for our settings as well. Given this, in this paper, we focus on the Exp3 algorithm.

Now, before describing Exp3, let us introduce the basic algorithm *Exp3.Sub* that is used as a subroutine in Exp3. This subroutine is shown in Algorithm 1. At each time slot t, Exp3.Sub randomly chooses an action i ($i \leq K$) with probability $p_i(t)$. Then, for all the actions (including the chosen one), it updates the probability $p_i(t)$ for the next slot, proportionally to the current estimate of the expected reward value of the action (i.e. the higher the current estimate is, the higher the probability it chooses that action). In particular, suppose that the reward values are taken from the interval $[A, B]$, where $A < B$ are arbitrary real numbers (Algorithm 1, step 5). Given this, Exp3.Sub maintains a weight value $w_i(t)$ for each action i. The update of these weights is shown in steps $6 - 13$. Using the new value of the weights, Exp3.Sub adaptively updates each probability $p_i(t)$ as shown in step 3. This indicates that the higher the current estimate becomes, Exp3.Sub will increase the value of $p_i(t)$, and thus, will choose action i with higher probability, and *vice versa*. Consequently, the algorithm always focuses on the actions with highest current estimates; that is, on those actions which are more likely to be the current best choice. In effect, this update policy guarantees that the agent can efficiently adapt to environmental changes.

The efficiency of Exp3.Sub, however, depends on the value of parameter γ, since it uses this parameter to calculate the probabilities (see step 3 in Algorithm 1). Since this γ has to be given *a priori* (Algorithm 1, step 1), it may happen that the chosen value is not efficient for the current MAB model. However, we can overcome this shortcoming by adaptively modifying the value of γ. This modification leads to the Exp3 algorithm, described in Algorithm 2. In particular, Exp3 divides the time line into rounds. At each round r, a new γ_r is chosen (see step 4 in Algorithm 2), and at each time slot t, Exp3 calls subroutine Exp3.Sub (step 6). Exp3 changes round when the maximal cumulative reward of an arm (i.e. the total amount of rewards Exp3 achieves when that arm is pulled) exceeds a given threshold (step 5). In this case, it restarts the subroutine Exp3.Sub as well.

Algorithm 2 Algorithm Exp3

Initialisation: Let $t = 1$, and $G_k(1) = 0$ for $k = 1, 2, ..., K$;
for all $r = 0, 1, 2, \ldots$ **do**
  $g_r = \frac{(K \ln K) 4^r}{e-1}$;
  Restart Exp3.Sub choosing $\gamma_r = \frac{1}{4^r}$
  while $\max_k G_k(t) \leq g_r - \frac{K}{\gamma_r}$ **do**
    Let a_t be the random action chosen by Exp3.Sub and $r_{a_t}(t)$ the corresponding reward;
    $G_{a_t}(t+1) = G_{a_t}(t) + r_{a_t}(t)$;
    $t := t + 1$;
  end while
end for

4.2 Using Multi–Armed Bandits for Energy Management

Given the description of the MAB model above, we now apply it to the energy management problem described in Section 3.3. In so doing, consider the formal model we introduced in Section 3. Recall that within this model, each agent i has an energy budget B_i for each time slot t, which is constant over time. Furthermore, agent i has to allocate budgets $B_i^{\mathrm{S}}(t)$, $B_i^{\mathrm{Rx}}(t)$, and $B_i^{\mathrm{Tx}}(t)$ to sampling, receiving and transmitting, respectively. The energy budget allocation, however, has to satisfy Equation 7.

Given this, we can formulate the energy management problem of a single agent as a MAB as follows. We first define the action set of each agent. Then we determine the reward function of each action. The latter is the mechanism that assigns reward values to the action of the agent at each time slot. Finally, we show how each agent uses the Exp3 algorithm to efficiently tackle its MAB problem.

In so doing, let us consider a decision that agent i can make at time slot t. Since the decision making task for an agent i consists of setting the values of the sampling, receiving and transmitting budgets of that agent at time slot t, we have the following definition:

Definition 3 Let $n_i^{\mathrm{S}}(t) = \lfloor \frac{B_i^{\mathrm{S}}(t)}{e_i^{\mathrm{S}}} \rfloor$, $n_i^{\mathrm{Rx}}(t) = \lfloor \frac{B_i^{\mathrm{Rx}}(t)}{e_i^{\mathrm{Rx}}} \rfloor$, and $n_i^{\mathrm{Tx}}(t) = \lfloor \frac{B_i^{\mathrm{Tx}}(t)}{e_i^{\mathrm{Tx}}} \rfloor$ denote the sampling, receiving and transmitting capacities (i.e. the maximal number of packets that the agent can sample, receive, or transmit) of agent i at time slot t, respectively. At each action, agent i chooses a combination of the values of those capacities, with respect to the constraint described in Equation 7. Thus, the *3-tuple* $a_i(t) = \left\langle n_i^{\mathrm{S}}(t), n_i^{\mathrm{Rx}}(t), n_i^{\mathrm{Tx}}(t) \right\rangle$ *denotes an action* of agent i at time slot t.

That is, an action of agent i at time slot t is a combination of $n_i^{\mathrm{S}}(t)$, $n_i^{\mathrm{Rx}}(t)$, and $n_i^{\mathrm{Tx}}(t)$, where the corresponding $B_i^{\mathrm{S}}(t)$, $B_i^{\mathrm{Rx}}(t)$, and $B_i^{\mathrm{Tx}}(t)$ satisfy Equation 7. Given this, the *action set* of agent i at time slot t is determined as follows:

$$\mathfrak{A}_i := \left\{ a_i(t) = \left\langle n_i^{\mathrm{S}}(t), n_i^{\mathrm{Rx}}(t), n_i^{\mathrm{Tx}}(t) \right\rangle \right\} \tag{12}$$

where $a_i(t)$ is presented in Definition 3. That is, $\mathfrak{A}_i$ is the set of 3-tuple of capacities where the total energy consumption does not exceed the energy limit given at each time slot t. Since the energy consumption limit (i.e. B_i) is fixed for each time slot t, the number of options for $a_i(t)$ (i.e. the number of combinations of the capacities) is constant over time as well. Given this, $\mathfrak{A}_i$ is fixed over time, and thus, it can be regarded as agent i's action set (since in MAB models, the action set cannot vary over time).

In contrast with the action set above, the definition of a single agent's reward function is not obvious. In particular, the reward function has to satisfy the following requirement: By maximising the total rewards that each agent can receive, the agents together maximise the total information collected in the network. However, in so doing, each agent has to take into account the behaviour of other agents within the network as well. Thus, the reward function has to capture the affect of other agents' behaviour on the performance of a single agent. Given this, we develop a reward function for each agent i as follows. Recall that $\mathbf{S}_i(t)$, $\mathbf{Rx}_i(t)$ and $\mathbf{Tx}_i(t)$ are the set of sampled, received and transmitted data packets of agent i at time slot t. Furthermore, $\mathbf{Q}_i(t)$ is the set of total

transmittable data packets in the memory (see Section 3.2 for more details). Let $\mathbf{Re}_i(t)$ denote agent i's set of residual packets from slot $(t-1)$ that are not transmitted until slot t. That is,

$$\mathbf{Re}_i(t) = \mathbf{Q}_i(t) / \mathbf{Tx}_i(t) \tag{13}$$

Given this, before we determine the reward function, let us consider the following. Assume that $\lambda = 1$; that is, there is no information decay as time passes by. Given this, throughout the operational time T of the network, the total information that is delivered to the BS is equal to the difference in the total information sampled by the agents in the network until time slot $(T-1)$, and the total amount of information that remains in the memory of the agents in the network at time slot T. In particular, since we assume that there is no data loss in our model, sampled data until time slot $(T-1)$ is either successfully delivered to the BS or still remains as residual data in the network at time slot T. Note that data sampled in time slot T is not considered here, since we assume that it cannot be delivered immediately to the BS, and as defined in Equations 13 and 3, $\mathbf{Re}_i(T)$ does not contain data that are sampled in time slot T. Thus, for each $t \in [1,T]$, let $r(t)$ denote the following function:

$$r_i(t) = \sum_{p \in S_i(t-1)} v(p,t-1) - \sum_{p \in Re_i(t)} v(p,t) + \sum_{p \in Re_i(t-1)} v(p,t-1) \tag{14}$$

Note that the first term on the right hand side of this equation is the total amount of sampled information of agent i at time slot $(t-1)$. The second term is the total information value of the residual data on agent i at time slot t, whilst the third term is the total information value of the residual data on agent i at time slot $(t-1)$. The intuition behind Equation 15 can be explained as follows. From the definitions given in Equations 3 and 13, the sum of the first and the third terms form the total amount of information that agent i can transmit in time slot t. In more detail, as we mentioned in Section 3.1, data sampled in time slot $(t-1)$ can only be transmitted from time slot t, and not earlier. Thus, first term represents the total amount of this sampled data. The third term represents the amount of information that is not transmitted until time slot $(t-1)$. Both the sampled data and residual data, however, is available at time slot t for transmission. On the other hand, the second term represents the information value of data that is not sent by the end of time slot t, and thus, by subtracting it from the set of transmittable data (i.e. sum of previously sampled data and residual data from $(t-1)$), we get the throughput of agent i within time slot t. Given this, by using $r_i(t)$ as the reward function within the case of $\lambda = 1$, each part of agent i's chosen action (i.e. the chosen energy budgets) will effect the value of $r_i(t)$. In particular, the size of $n_i^{\mathrm{S}}(t)$ affects the total amount of sampled information, while $n_i^{\mathrm{Rx}}(t)$ and $n_i^{\mathrm{Tx}}(t)$ affect the size of residual data.

Now, we show that by maximising the sum of $r_i(t)$ over all t and i indeed leads to the maximisation of the total amount of collected information within the network, in the case of $\lambda = 1$. In so doing, recall that $\sum_{p \in Re_i(t-1)} v(p,0) = 0$ for each agent i, since there is no residual data at all at the beginning. Given this, it is easy to see that if we sum up $r_i(t)$ by t from 1 to T, what we get as a result is exactly the difference of the total information collected by the network and the total amount of information that remains in the memory of the agents in the network. More precisely, we have

$$\begin{aligned}
\sum_{t=1}^{T} r_i(t) &= \sum_{t=0}^{T-1} \sum_{p \in S_i(t)} v(p,t) - \sum_{t=1}^{T} \sum_{p \in Re_i(t)} v(p,t) + \sum_{t=0}^{T-1} \sum_{p \in Re_i(t)} v(p,t) \\
&= \sum_{t=0}^{T-1} \sum_{p \in S_i(t)} v(p,t) - \sum_{p \in Re_i(T)} v(p,T) + \sum_{p \in Re_i(0)} v(p,0) \\
&= \sum_{t=0}^{T-1} \sum_{p \in S_i(t)} v(p,t) - \sum_{p \in Re_i(T)} v(p,T)
\end{aligned}$$

Recall that this value is equal to the total information that is delivered to the BS throughout the operation time of the network. Thus, $r_i(t)$ could be a possible reward function for agent i, since by maximising the total reward on interval $[0,T]$, the agents together also maximise the total amount of collected information value that is delivered to the BS as well.

Note that the definition of $r_i(t)$ in Equation 14 guarantees that in order to maximise the total amount of collected information, agent i cannot either ignore sampling, receiving or transmitting. In particular, for example,

suppose that agent i ignores transmitting, and only focus on sampling/or receiving. In this case, the set of residual data at the end of time slot t is equal to the accumulated set of sampled data and residual data at time slot $(t-1)$, and thus, the value of the reward is 0. Now, it is easy to see that if the transmitting capacity is greater than 0 (i.e. $n_i^{\mathrm{Tx}}(t) > 0$), the reward value is definitely higher than 0 as well. In similar veins, we can easily see that agent i cannot get high reward values on the long term if it ignores the other sensory tasks as well.

Now, to generalise Equation 14 to the case of $\lambda \neq 1$, consider the following:

$$R_i(t) = \lambda^{d_i-1}\left\{ \sum_{p\in S_i(t-1)} v(p,t-1) - \sum_{p\in Re_i(t)} v(p,t) + \lambda \sum_{p\in Re_i(t-1)} v(p,t-1) \right\} \tag{15}$$

where d_i is the distance of agent i from the BS (in hops), and λ is the information discount coefficient. This equation differs from Equation 14 in two places. First, it is weighted by the factor λ^{d_i-1}. The intuition behind using this factor is that since agent i is d_i hops away from the BS, the information value that agent i transmits is discounted with a factor λ^{d_i-1} when the BS receives that data. The second difference is that the third term of Equation 15 is weighted with λ. The reason here is that since the third term represents the set of packets that are not sent by the end of time slot $(t-1)$, the information value of those packets is discounted in the next time slot. Note that in the case of $\lambda = 1$, this equation is reduced to Equation 14. To show that this reward function is suitable for maximising the total collected information of the network in the long term, we state the following:

Theorem 1 (Main result 1) *Using the reward function defined in Equation 15, the total reward value that the agents in the WSN achieve together over the interval $[0,T]$ is equal to the total information content value delivered to the BS over that time interval.*

That is, Theorem 1 states that by maximising each agent's total reward over interval $[0,T]$, where the reward function is defined as in Equation 15, we can achieve the maximal information collected and delivered to the BS. We prove the theorem as follows:

Proof (Theorem 1) For the sake of simplicity, let $\mathfrak{L}_j$ denote the set of agents that are j hops from the BS. That is,

$$d_i = j, \forall i \in \mathfrak{L}_j \tag{16}$$

Now, consider Equation 1 in Section 3.2. Let us note that since no data can be sampled and forwarded, or received and forwarded at the same time slot (see Section 3.1), no data packets are transmitted or received at time slot 0 in the whole WSN. Thus, using the notation of Section 3, the main objective can be rewritten as follows.

$$\max \sum_{t=1}^{T} \left\{ \sum_{p\in Rx_{BS}(t)} v(p,t) \right\} \tag{17}$$

Let us consider a particular member of Equation 17, which is $\sum_{p\in Rx_{BS}(1)} v(p,1)$. This equation determines the total information value that arrives to the BS at time slot 1. According to our assumptions in Section 3.1, no data loss occurs during any transmission. Thus, the amount of received information at the BS is equal to the total amount of information that is transmitted from agents that are 1–hop from the BS at time slot 1. That is,

$$\sum_{p\in Rx_{BS}(1)} v(p,1) = \sum_{j\in\mathfrak{L}_1} \sum_{p\in Tx_j(1)} v(p,1) \tag{18}$$

Note that the set of transmitted data of $\mathfrak{L}_1$ at time slot 1 is equal to the set of sampled data at time slot 0, excluding the set of residual data at time slot 1 (since there is no received data and the residual set is still empty at time slot 0). Since newly sampled data does not suffer from information value discounting, the right side of Equation 18 can be rewritten as the following:

$$\sum_{j\in\mathfrak{L}_1} \sum_{p\in Tx_j(1)} v(p,1) = \sum_{i\in\mathfrak{L}_1} \sum_{p\in S_i(0)} v(p,0) - \sum_{i\in\mathfrak{L}_1} \sum_{p\in Re_i(1)} v(p,1) \tag{19}$$

Now, let us consider the second member of Equation 17, which is $\sum_{p\in Rx_{BS}(2)} v(p,2)$. Similarly, this can be rewritten as follows.

$$\sum_{p\in Rx_{BS}(2)} v(p,2) = \sum_{j\in\mathfrak{L}_1} \sum_{p\in Tx_j(2)} v(p,2) \tag{20}$$

However, this is equal to the union of the set of received data, the set of sampled data, and the set of residual data at time slot 1, excluding the set of residual data of layer 1 at time slot 2. Furthermore, any of these sets may not be empty. The packets in the sets of received and residual data suffer from value discounting, thus, Equation 20 is equal to the following:

$$\begin{aligned}\sum_{p\in Rx_{BS}(2)} v\left(p,2\right) &= \sum_{j\in\mathfrak{L}_1}\sum_{p\in Tx_j(2)} v\left(p,2\right) = \\ &= \sum_{i\in\mathfrak{L}_1}\sum_{p\in S_i(1)} v\left(p,1\right) + \lambda\sum_{i\in\mathfrak{L}_1}\sum_{p\in Re_i(1)} v\left(p,1\right) + \\ &+ \lambda\sum_{i\in\mathfrak{L}_1}\sum_{p\in Rx_i(1)} v\left(p,1\right) - \sum_{i\in\mathfrak{L}_1}\sum_{p\in Re_i(2)} v\left(p,2\right) \end{aligned} \tag{21}$$

where λ is the discount coefficient of the network. Now let us consider $\sum_{i\in\mathfrak{L}_1}\sum_{p\in Rx_i(1)} v\left(p,1\right)$. Similar to Equation 18, this can be written as:

$$\lambda\sum_{i\in\mathfrak{L}_1}\sum_{p\in Rx_i(1)} v\left(p,1\right) = \lambda\sum_{i\in\mathfrak{L}_2}\sum_{p\in Tx_i(1)} v\left(p,1\right) \tag{22}$$

Using Equations 21 and 22, and replacing $\mathfrak{L}_1$ with $\mathfrak{L}_2$ in Equation 19, we obtain the following:

$$\begin{aligned}\sum_{p\in Rx_{BS}(2)} v\left(p,2\right) &= \sum_{i\in\mathfrak{L}_1}\sum_{p\in S_i(1)} v\left(p,1\right) - \\ &- \sum_{i\in\mathfrak{L}_1}\sum_{p\in Re_i(2)} v\left(p,2\right) + \lambda\sum_{i\in\mathfrak{L}_1}\sum_{p\in Re_i(1)} v\left(p,1\right) + \\ &+ \lambda\sum_{i\in\mathfrak{L}_2}\sum_{p\in S_i(0)} v\left(p,0\right) - \lambda\sum_{i\in\mathfrak{L}_2}\sum_{p\in Re_i(1)} v\left(p,1\right) \end{aligned} \tag{23}$$

In general, if we take the t^{th} member of Equation 17, then it can be decomposed as follows. If $t \leq L$, where L is the number of the layers in the network, then:

$$\begin{aligned}\sum_{p\in Rx_{BS}(t)} v\left(p,t\right) &= \sum_{j=1}^{t}\lambda^{j-1}\sum_{i\in\mathfrak{L}_j}\sum_{p\in S_i(t-j)} v\left(p,t-j\right) - \\ &- \sum_{j=1}^{t}\lambda^{j-1}\sum_{i\in\mathfrak{L}_{j+1}}\sum_{p\in Re_i(t-j+1)} v\left(p,t-j+1\right) + \\ &+ \sum_{j=1}^{t}\lambda^{j}\sum_{i\in\mathfrak{L}_j}\sum_{p\in Re_i(t-j)} v\left(p,t-j\right) \end{aligned} \tag{24}$$

Let us note that here $\sum_{i\in\mathfrak{L}_j}\sum_{p\in Re_i(0)} v\left(p,0\right) = 0$ for any layer j. That is, we can say that the amount of information that arrives to the BS at time slot t can be decomposed into the sum of data on layer 1 at time slot $(t-1)$, on layer 2 at time slot $(t-2)$, and so on. If $t > L$, however, the equation for this case is slightly different, since the decomposition stops at the last layer of agents. Thus, we have:

$$\begin{aligned}\sum_{p\in Rx_{BS}(k)} v\left(p,k\right) &= \sum_{j=1}^{L}\lambda^{j-1}\sum_{i\in\mathfrak{L}_j}\sum_{p\in S_i(t-j)} v\left(p,t-j\right) - \\ &- \sum_{j=1}^{L}\lambda^{j-1}\sum_{i\in\mathfrak{L}_{j+1}}\sum_{p\in Re_i(t-j+1)} v\left(p,t-j+1\right) + \\ &+ \sum_{j=1}^{L}\lambda^{j}\sum_{i\in\mathfrak{L}_j}\sum_{p\in Re_i(t-j)} v\left(p,t-j\right) \end{aligned} \tag{25}$$

Given this, combining Equations 24 and 25, and taking each t into account, we can reformulate our main objective to the following:

$$\begin{aligned}&\sum_{t=1}^{T}\left\{\sum_{p\in Rx_{BS}(t)} v\left(p,t\right)\right\}\\&=\\&=\sum_{t=1}^{T}\sum_{j=1}^{\min(t,L)}\lambda^{j-1}\sum_{i\in\mathfrak{L}_j}\left\{\sum_{p\in S_i(t-j)} v\left(p,t-j\right)-\sum_{p\in Re_i(k-j+1)} v\left(p,t-j+1\right)+\lambda\sum_{p\in Re_i(k-j)} v\left(p,t-j\right)\right\}\end{aligned} \tag{26}$$

Consider the core part of Equation 26 in the braces. Now, using the definition of the reward function in Equation 15 to replace that part, and recall that the distance of agent i is defined in Equation 16, we can reformulate 26 as follows:

$$\max \sum_{j=1}^{\min(T,L)}\sum_{t=0}^{T-j}\sum_{i\in\mathfrak{L}_j} R_i\left(t\right) \tag{27}$$

That is, the original objective can be decomposed to the sum of reward functions of agents on each layer j, from time slot 0 to time slot $T-j$. □

Now, using the aforementioned reward function and the action set, the energy management problem of each agent i can be reduced to a MAB problem. Thus, the multi–armed bandit based energy management algorithm works as follows. Each agent i runs the Exp3 algorithm, in order to determine the energy budget allocation combination for each time slot t. In particular, agent i first assigns a probability value p_i^j to each of the its possible energy budget allocation combinations (i.e. arm) $j \in \mathfrak{A}_i$, such that $\sum_j p_i^j = 1$. According to Algorithm 1 (step 3), these probability values are initially set to be uniform (e.g. it is the same for all the combinations). Based on these probability values, each agent i randomly chooses an energy budget allocation combination $a_i\left(t\right)$ at time slot t, and allocates energy budgets to each of the tasks of sampling, receiving and transmitting. This energy budget allocation combination $a_i\left(t\right)$ is chosen as shown in Algorithm 2. Then, as mentioned earlier, to control the sampling task, the agent can use an existing adaptive sampling technique. On the other hand, the tasks of receiving and transmitting are controlled by the routing algorithm, which we will describe later in Section 5. Following this, after agent i finishes the sampling, receiving and transmitting tasks, the reward value of the chosen energy budget allocation combination is evaluated. Then, Exp3 updates the value of probabilities p_i^j, such that higher probability values will be assigned to energy budget allocation combinations with higher average reward (see algorithm 2 for more details). Let us hereafter refer to this approach (i.e. using Exp3 for allocating energy budgets) as the *multi-armed bandit based energy management* (MAB/EM).

Now, by using MAB/EM, the agents do not explicitly coordinate with each other (i.e. they do not use coordination messages). In more detail, our approach uses explicit communication messages within the routing part (for more details, see Section 5), but not within the energy budget allocation phase. However, these communication messages are only for evaluating the reward value of the chosen action (i.e. the chosen combination of energy budget allocations). Given this, the agents do not need to coordinate when they take an action. Despite the lack of explicit coordination within MAB/EM, the agents can still achieve coordination by only observing the reward value they get. In more detail, consider the definition of the reward function (Equation 15). Note that this reward function is affected by the agent's current chosen action (i.e. the energy amounts allocated to sampling, receiving and transmission). In particular, according to Equations 3 and 13, $\mathbf{Re}_i\left(t\right)$ (i.e. the list of residual packets) depends on the lists of sent and received packets, respectively. Thus, in order to achieve higher rewards, each agent aims to find actions that result in better reward values. However, the effectiveness of a chosen action also depends on other agents' action as well. Indeed, the effectiveness of data receiving (or transmitting) depends on the allocated budget to transmitting (or receiving) of other neighbouring agents. For example, it is not efficient for agent i to allocate a large amount of energy to receiving if its neighbours are only willing to send a small amount of data, and *vice versa*. Given this, by only observing which actions result in higher rewards, the agents also learn to cooperate with the others as well.

4.3 Computational Complexity Analysis

Since WSNs are heavily resource constrained (i.e. the low energy capacity, small size and tight computational constraints), algorithms that are implemented for such networks need to take into consideration the limited computational capacity and memory space (Akyildiz *et al.*, 2002; Rogers *et al.*, 2009). Thus, in order to ensure that MAB/EM is suitable for WSNs (i.e. it can be installed to real sensors), we have to guarantee that it has low computational complexity and low memory demand. Given this, we study the performance of the MAB/EM in terms of computational complexity in this section. More precisely, we investigate the number of computational steps (i.e. running time cost) and the memory usage that MAB/EM uses at each time slot.

From the aspect of computational cost, each agent i has to update the probability value p_i^j for each of its actions j. In so doing, agent i needs to maintain a weight value w_i^j, which also needs to be updated (see Algorithms 1 and 2 for more details). Given this, the number of computational steps of agent i at each time slot is $O\left(2\left|\mathfrak{A}_i\right|\right)$, where $\left|\mathfrak{A}_i\right|$ is the size of agent i's action set $\mathfrak{A}_i$. That is, the running time of MAB/EM is linear to the size of each agent's action set.

In terms of memory usage, MAB/EM is also efficient. In particular, recall that each agent i maintains the arrays of w_i^j, and of p_i^j, respectively. Furthermore, each action is represented as a 3–tuple of integers (see Definition 3). Given this, the memory usage of MAB/EM is $O\left(5\left|\mathfrak{A}_i\right|\right)$. To demonstrate that the memory usage is indeed low, compared to the size of data packets, consider the following example. Note that the action set typically has the size of few hundreds. This can be easily calculated by using the typical sensory parameter values, which can be found, for example, in Kansal and Srivastava (2003). Now, suppose that to store a number, each agent uses 4 bytes of memory. Given this, the total memory usage (i.e. to store the arrays of probability and weight parameters) is typically a few kilobytes. This is small, compared to the total size of real data that the agents typically have to forward in many applications (e.g. in wireless visual sensor networks) the average size of a single data packet is likely to be $10 - 100$ kBytes (Kho *et al.*, 2010).

5 Optimal Data Routing

Given the energy management approach described in the previous section, we now focus on the maximal information throughput problem presented in Section 3.4. Thus, this section outlines the work undertaken towards addressing this routing problem. Specifically, here we describe two decentralised algorithms that allow agents to achieve maximal information throughput between neighbouring layers, with respect to their energy constraints. In particular, the first algorithm, called MITRA (for maximal information throughput routing algorithm), achieves optimal performance in terms of solving the maximal information throughput problem. However, it may have significant computational and communication costs. On the other hand, the second algorithm, called MITRA_τ, produces near–optimal performance (approximately 98% of the optimal performance), but with reduced communication and computational costs. To this end, we first introduce MITRA in more detail in Section 5.1. Following this, we show that this approach is optimal in terms of maximising the information throughput in Section 5.2. Furthermore, we provide a theoretical upper bound for the computational and communication costs of MITRA in Section 5.3. Finally, we propose MITRA_τ, a modified version of MITRA with reduced communication and computational costs in Section 5.4.

5.1 The Maximal Information Throughput Routing Algorithm

Recall that at each time slot t, all the agents within the system run the MAB/EM in order to set up the energy budgets for that current time slot. Then, their next step is to maximise the amount of forwarded information value conditional on the budgets in that given time slot. That is, the agents aim to maximise the total information value forwarded between neighbouring layers of agents (see definition 2 for more details). Now, let $\mathfrak{L}_l$ and $\mathfrak{L}_{l-1}$ denote the corresponding layers. The pseudocode of the MITRA run by the agents within these layers is depicted in Algorithm 3. In more detail, we refer to the agents in layers $\mathfrak{L}_l$ and $\mathfrak{L}_{l-1}$ as senders, and receivers, respectively. The algorithm can be outlined as follows:

Algorithm 3 MITRA

1: **for all** pair of layers $\mathfrak{L}_l$ and $\mathfrak{L}_{l-1}$ **do**
2: agents in layer $\mathfrak{L}_l \leftarrow$ senders, agents in layer $\mathfrak{L}_{l-1} \leftarrow$ receivers;
3: $\forall i$ sender s_i broadcasts list of information values;
4: **while** data transmission is feasible **do**
5: $\forall j$: when receiver r_j receives all the broadcast information (or time threshold expires), it identifies best packets it can receive;
6: $\forall j$ receiver r_j sends REQUEST messages to senders;
7: $\forall i$ when sender s_i receives all the REQUEST messages (or time threshold expires), it sends data to receiver with best offer;
8: **if** $\exists$ sender s_i has not exceed transmission budget **then**
9: sender s_i broadcasts a SEND message to receivers;
10: **end if**
11: **end while**
12: **end for**

– **Step 3**: First, each sender s_i broadcasts a message that contains the list of 2–tuples to each of its neighbouring receivers. The first element of the tuple contains the packet ID, whilst the second element contains the information value of sender s_i's transmittable packets (i.e. the list of $\mathbf{Q}_{s_i}(t)$, see Section 3.2 for more details). Then, whilst data transmission is still feasible, the algorithm repeatedly executes steps $5-10$ as follows.

– **Step 5**: Based on the received information lists from the neighbouring senders, each receiver r_j chooses the best packets (i.e. packets with the highest information value) it can receive, with respect to its residual receiving capacity (i.e. the maximal number of packets it can still receive without exceeding its total receiving capacity $n_{r_j}^{\mathrm{Rx}}(t)$). Note that $n_{r_j}^{\mathrm{Rx}}(t)$ is set by the MAB/EM (see definition 3 for more details). In so doing, it needs to wait until it receives all the broadcast information from its neighbouring senders. However, since node failures may occur, agent r_j does not exactly know which of its neighbours is available within the current time slot t, and thus, will send to r_j a broadcast message. In such cases, r_j does not know when to stop waiting for the broadcast messages, and thus, it cannot move on to the next step of MITRA. In order to avoid this situation, we set a time threshold, so that if this threshold expires, the sender stops waiting for further broadcast messages. Following this, r_j chooses the best packets it can receive as follows. It first sorts the received lists of 2–tuples in decreasing order of the value of information, then it merges these lists into a joint list, also with the decreasing order of the information value. From this joint list, it chooses the best packets it can receive.

– **Step 6**: Following this, receiver r_j propagates REQUEST messages to each of its neighbouring senders. In particular, each REQUEST message contains the number of packets that r_j requests from that sender. This number is calculated in step 5 of the algorithm.

– **Step 7**: When s_i receives all the REQUEST messages from its neighbouring receivers, it chooses the best offer; that is, the one with the highest number of requested packets. However, similarly to step 5 of the algorithm, it may occur that s_i does not know when to stop waiting for all the REQUEST messages, due to node failure. Thus, to prevent it from waiting indefinitely for the messages, we also use a time threshold here. Given this, after all the REQUEST messages arrive to s_i, or the time threshold expires, s_i sends the requested packets to the receiver with the best offer. If the receiver with the best offer is not unique, then s_i randomly chooses one among them.

– **Steps 8–10**: After data transmission in the previous step, if sender s_i still has the capacity to transmit data (i.e. $n_{s_i}^{\mathrm{Tx}}(t)$ is not exceeded), then it broadcasts a SEND message to each of its neighbouring receivers. This message contains the number of packets that it transmitted in step 7. Based on this message, all the receivers can update the list of packets they can request from s_i (i.e. they update the joint list described in step 3). Furthermore, they also update the value of their remaining receiving capacity.

Now, to detect whether data transmission is still feasible, the participating agents do the following. From the sender side, when sender s_i does not receive any REQUEST messages in step 7, it considers data transmission as not feasible. From the receiver side, when receiver r_j does not receive any broadcast messages (e.g. the list of information value, or the SEND messages) in step 5, then it also considers data transmission as not feasible.

Given this, if an agent sees that it cannot receive and transmit data anymore (i.e. receiving and transmission is not feasible), it stops running MITRA for that time slot. That is, the agents rerun MITRA at each time slot t. Note that the time thresholds in steps 5 and 7 are for only communication messages (i.e. `REQUEST` and broadcast messages). Once the agent receives one of these messages from its corresponding neighbour, it sets up a communication channel, in which data packets are assumed to be successfully forwarded, without any loss.

5.2 Performance Analysis

Given the description of MITRA above, we now show this algorithm provides the optimal solution to the maximal information throughput routing problem presented in Definition 2. In so doing, we state the following:

Theorem 2 (Main result 2) *Assuming that the communication between senders and receivers is perfect, that is, none of the messages arrive after the timeout, the MITRA algorithm results in an optimal solution for the maximal information throughput routing problem (i.e. the solution that gives the maximal throughput of information value between the sender and receiver layers).*

Proof (Theorem 2) . Here we use the contradiction technique. Let us assume that the MITRA algorithm given in the previous section is not optimal. That is, the output solution does not maximise the total transmitted information value between the two layers. Let $\mathfrak{O}$ denote the output solution of the MITRA algorithm and $\mathfrak{O}_{OPT}$ be one of the optimal solutions. Since we assume that $\mathfrak{O}$ is not optimal, there should be p_1 and p_2 packets such that only one of them is allocated in $\mathfrak{O}$ and the other one is allocated in $\mathfrak{O}_{OPT}$. Without loss of generality, we can assume that p_1 is allocated in $\mathfrak{O}$ and p_2 is allocated in $\mathfrak{O}_{OPT}$. We can also assume that both p_1 and p_2 are sent to the same receiver r_j. It is easy to prove that if $\mathfrak{O} \neq \mathfrak{O}_{OPT}$ then there exist two packets such that these assumptions hold.

In particular, there are two cases to investigate. In the first, both p_1 and p_2 are from the same sender. Note that it is easy to show that $v(p_1, k) \geq v(p_2, k)$. That is, p_1 has a higher information value than p_2, since the corollary states that those data which are sent from the sender must be the packets with the highest values in the set of packets of that sender.

In the second case, p_1 and p_2 are from different senders. Since in MITRA, the receiver uses a greedy approach to allocate possible arriving packets, when p_1 is accepted and p_2 is not at r_J, the only explanation is that $v(p_1, k) \geq v(p_2, k)$.

One can see that in both cases p_1 has a higher, or at least the same value, as p_2. If p_1 has a higher value than that of p_2, then by replacing p_2 in $\mathfrak{O}_{OPT}$ with p_1, we would have a better solution than $\mathfrak{O}_{OPT}$. However, this is a contradiction, since $\mathfrak{O}_{OPT}$ is assumed to be optimal. If p_1 has the same value as p_2, then by replacing all the possible p_i-s that are in $\mathfrak{O}$ but not in $\mathfrak{O}_{OPT}$ (since they all have the same value, otherwise we would be faced with the former case), we would have that $\mathfrak{O}$ is also an optimal solution, which would also contradict our assumption at the beginning. Therefore one can see that the original assumption, that is, $\mathfrak{O}$ is not optimal, is not true. □

5.3 Computational and Communication Cost of MITRA

In the previous section, we showed that MITRA achieves an optimal solution for the maximal information throughput problem. Given this, here we continue the analysis of MITRA by studying its computational and communication cost. In particular, similarly to the case of MAB/EM, we need to analyse whether MITRA is efficient in terms of computational and communication complexity. In so doing, recall that at each time slot t, each agent i within the network repeatedly runs steps $4-11$ of Algorithm 3 until data transmission is not feasible at that time slot. For the sake of simplicity, hereafter we refer to this cycle as the *communication round* of MITRA (since the agents communicate with each other during this cycle in order to find the maximal information throughput). Note that since MITRA is rerun at every time slot, each time slot t contains a number of communication rounds. Thus, the number of communication rounds that MITRA uses within a particular time slot cannot be larger in

time, compared to the length of a single time slot. Given this, here we aim to analyse whether we can upper bound the number of communication rounds. Furthermore, note that both the computational and communication costs of agent i depend on the number of communication rounds that the agent needs to run. Thus, in order to guarantee low computational and communication costs of a single agent, we also need to ensure that the number of communication rounds that an agent uses within the MITRA is also low. In more detail, each receiver determines the best packets (i.e. packets with highest information value) it can receive by sorting the list of receivable packets at each communication round (step 5 of Algorithm 3). Since this list typically has a size at most of few thousands, sorting it is simple and fast (e.g. by quicksort). However, since the sorting is repeatedly executed at each communication round, if the number of those rounds is high, then the total computational cost can be significant. Now, note that the communication cost of a single agent consists of the cost of sending REQUEST messages and the cost of sending a SEND broadcast message at each communication round. Thus, again, if the number of communication rounds is high, then the total communication cost can also be significant.

Against this background, we provide a worst–case upper bound (i.e. an upper bound that holds for all the cases) for the number of communication rounds that MITRA uses. More precisely, we state the following:

Theorem 3 (Main result 3) *Consider neighbouring layers $\mathfrak{L}_l$ and $\mathfrak{L}_{l-1}$. At each time slot t, let $T_{\mathrm{com}}(t)$ denote the total number of communication rounds, that MITRA needs to run until data transmission is not feasible between layers $\mathfrak{L}_l$ and $\mathfrak{L}_{l-1}$ within time slot t. Given this, we have:*

$$T_{\mathrm{com}}(t) \leq \frac{\ln\left(\sum_{r_j \in \mathfrak{L}_{l-1}} n_{r_j}^{\mathrm{Rx}}(t)\right)}{\ln|\mathfrak{L}_{l-1}| - \ln\left(|\mathfrak{L}_{l-1}| - 1\right)}$$

where $|\mathfrak{L}_{l-1}|$ denote the size of layer $\mathfrak{L}_{l-1}$ (i.e. layer of receivers).

Proof Recall that, at each communication round, each receiver r_j chooses the best packets it can receive, conditional to the value of its residual receiving capacity (see step 5 of algorithm 3). Let $D_{r_j}(\tau)$ denote the maximal number of packets r_j can receive from its neighbouring senders at communication round τ. It is easy to see that for each r_j, $D_{r_j}(\tau)$ is monotone decreasing function of τ, within time slot t. In more detail, recall that the senders cannot forward information that are sampled or received at time slot t. Given this, $D_{r_j}(\tau)$ only contains data that are sampled/or received until time slot $(t-1)$. This set of data, however, is already given at the beginning of time slot t, and thus, during the communication rounds, the size of these data cannot be increased. Furthermore, at each communication round (within time slot t), receiver r_j receives a non–negative number of packets. Given this, the value of $D_{r_j}(\tau)$ is monotone decreasing.

Given this, we first show that at each communication round τ, the total number of successfully received packets within MITRA is at least $D_{\max}(\tau)$, where

$$D_{\max}(\tau) = \max_{r_j} D_{r_j}(\tau)$$

Indeed, according to algorithm 3, each receiver r_j send REQUEST messages to its neighbours at each communication round τ, requesting $D_{r_j}(\tau)$ packets in total. Some of these requests will be accepted by the senders, whilst the others will be rejected. However, a sender only rejects a request, if it gets a better request (or a same request) of total amount of information value from another receiver. This implies that the number of packets of the better request is not lower than the number of packets r_j requests from that sender. Given this, it is easy to see that the total amount of transmitted (received) packets is at least $D_{r_j}(\tau)$ for any r_j (i.e. it is also at least $D_{\max}(\tau)$). Therefore, we have the following inequality:

$$\sum_{r_j} D_{r_j}(\tau+1) \leq \sum_{r_j} D_{r_j}(\tau) - D_{\max}(\tau) \tag{28}$$

Now, note that at each communication round τ, we have:

$$D_{\max}(\tau) \geq \frac{\sum_{r_j} D_{r_j}(\tau)}{|\mathfrak{L}_{l-1}|} \tag{29}$$

That is, $D_{\max}(\tau)$ is not lower than the average value of $D_{r_j}(\tau)$. Using Equations 28 and 29, we get:

$$\sum_{r_j} D_{r_j}(\tau+1) \leq \frac{|\mathfrak{L}_{l-1}|-1}{|\mathfrak{L}_{l-1}|} \sum_{r_j} D_{r_j}(\tau)$$

That is, we can show by induction that the following holds for each τ:

$$\sum_{r_j} D_{r_j}(\tau+1) \leq \left(\frac{|\mathfrak{L}_{l-1}|-1}{|\mathfrak{L}_{l-1}|}\right)^{\tau} \sum_{r_j} D_{r_j}(1) \tag{30}$$

Note that $D_{r_j}(1) \leq n_{r_j}^{\mathrm{Rx}}(t)$; that is, the maximal number of packets that r_j can receive at the first communication round is not greater than the receiving capacity of r_j. Given this, from Equation 30 we get:

$$\sum_{r_j} D_{r_j}(\tau+1) \leq \left(\frac{|\mathfrak{L}_{l-1}|-1}{|\mathfrak{L}_{l-1}|}\right)^{\tau} \sum_{r_j} n_{r_j}^{\mathrm{Rx}}(t) \tag{31}$$

Now, note that MITRA stops after τ communication rounds if and only if

$$\sum_{r_j} D_{r_j}(\tau+1) < 1$$

That is, no more packets can be sent to the receivers. Given this, MITRA still runs after τ communication rounds if

$$\left(\frac{|\mathfrak{L}_{l-1}|-1}{|\mathfrak{L}_{l-1}|}\right)^{\tau} \sum_{r_j} n_{r_j}^{\mathrm{Rx}}(t) \geq 1 \tag{32}$$

This can be reformulated as:

$$\sum_{r_j} n_{r_j}^{\mathrm{Rx}}(t) \geq \left(\frac{|\mathfrak{L}_{l-1}|}{|\mathfrak{L}_{l-1}|-1}\right)^{\tau} \tag{33}$$

Taking the logarithmic function of both sides, we get:

$$\ln\left(\sum_{r_j} n_{r_j}^{\mathrm{Rx}}(t)\right) \geq \tau\left(\ln|\mathfrak{L}_{l-1}| - \ln\left(|\mathfrak{L}_{l-1}|-1\right)\right) \tag{34}$$

Substituting $T_{\mathrm{com}}(t)$ into this inequality concludes the proof. □

Note that from the proof, it is easy to show that this upper bound is tight. Thus, $T_{\mathrm{com}}(t) = O\left(\ln\left(\sum_{r_j \in \mathfrak{L}_{l-1}} n_{r_j}^{\mathrm{Rx}}(t)\right)\right)$; that is, the upper bound of T_{com} is the logarithm of the total number of packets that need to be forwarded within each time slot t.

5.4 Communication Round Limited MITRA

In the previous section, we provided an upper bound for the number of communication rounds that MITRA uses. In particular, we demonstrated that the number of these communication rounds is low, compared to the total size of data to be forwarded at a single time slot. However, since this upper bound is tight, the total number of communication rounds that MITRA uses in the worst case scenario (i.e. when the bound is tight) is still significant in terms of total time length. For example, consider a WSN, where each layer has 10 agents on average, and each agent can receive 100 packets per time slot. Given this, according to Theorem 3, the upper bound of the number of communication rounds is around 66. Note that each communication round consumes a certain amount of time, and thus, 66 communication rounds together may not fit into the length of a single time slot (since MITRA has to terminate within the same time slot).

In order to address this shortcoming, we can either shorten the time length of a communication round, risking the higher rate of data loss in WSNs (i.e. not all of the `SEND` and `REQUEST` messages arrive on time), or limit the number of communication rounds that MITRA can use. We show that by using the latter solution, we can significantly reduce the number of communication rounds, whilst the reduction in the performance of the algorithm is not significant. We denote the communication round limited MITRA with MITRA_τ, where τ is the threshold value of the number of communication rounds. Given this, the algorithm for MITRA_τ is similar to that of MITRA, except that it stops executing steps $4-11$ after exactly τ rounds (see Algorithm 3 for more details). In Section 6.4, we will demonstrate that with low τ values (e.g. $\tau = 8$), MITRA_τ can still achieve 98% of MITRA's performance.

6 Performance Evaluation

Having calculated the computational and communication complexity of MAB/EM and MITRA in the previous sections, we now demonstrate that by using MAB/EM for energy management and MITRA_τ for data routing, our proposed algorithms together significantly outperform the state–of–the–art. In so doing, we present empirical results against state–of–the–art algorithms in long–term information collection in the WSN domain. The reason we choose MITRA_τ instead of MITRA to route data is that the communication cost of MITRA_τ is guaranteed to be low (see Section 5.4 for more details). However, as we will show later, it achieves, on average, 98% of the performance of MITRA. Now, to show the efficiency of our proposed approach compared to that of the state–of–the–art, we need to choose a benchmark algorithm that has to fulfil the following requirements:

- It must be capable of using efficient adaptive sampling methods for collecting data from the environment.
- It must use information content valuation, in order to distinguish important data from unimportant data.
- It must contain an energy management policy, which allocates energy budgets to different sensory tasks of sampling, receiving, and transmitting.

In particular, as we discussed in Section 2, algorithms that guarantee these requirements may perform well in WSNs with dynamic environments for efficient long–term information collection. On the other hand, those which fail to fulfil the aforementioned requirements are not suitable for long–term information collection in our settings (see Section 2 for more details). Since it has been shown that USAC achieves significant performance improvement in long–term data collection, compared to that of other state–of–the–art algorithms, especially in dynamic environments (Padhy *et al.*, 2010). On the other hand, as we demonstrated within Section 2, other state–of–the–art methods typically fail to fulfil these criteria. Given this, we choose USAC as a benchmark for our performance evaluation. In more detail, we compare the performance of our approach to USAC through extensive simulations, and we show that our approach typically outperforms USAC on average by around 120%. Furthermore, we also benchmark the performance of our approach against a non–learning approach, that solely uses MITRA for routing. In particular, within this benchmark approach, each agent randomly chooses an energy budget allocation combination, that it uses throughout its operating time (i.e. the budgets are fixed over time). Here, MITRA with fixed budgets represents a benchmark algorithm that does not intelligently set the budgets of the sensory tasks to adapt to the environmental changes. With this comparison, we demonstrate that by using adaptive learning (i.e. the MAB/EM), we can also double the amount of collected information in the long term.

In addition, we also benchmark the performance of our approach against the theoretical optimal performance of the network (i.e. the maximal value of collected information that the network can achieve). This benchmark aims to provide the theoretical upper bound of the performance that we can achieve within long–term information collection in WSNs. In particular, in order to determine the optimal performance of the network, we need global information about each agent's sampled information values at each time slot. However, to gather this global information, a centralised control mechanism is needed, which is not feasible in our settings (as outlined in Section 1). Thus this is a benchmark algorithm only, not a feasible solution to our information collection problem.

Finally, we demonstrate that by using MITRA_τ with small values of τ, we can still achieve near–optimal routing performance, while the number of communication rounds needed is significantly reduced (compared to that of the MITRA).

To this end, we first set the parameters, that will be used throughout our simulations, in Section 6.1. Following this, to demonstrate the efficiency of MAB/EM combined with MITRA_τ, we analyse simulation results in detail in Section 6.2. Here, we compare the performance of our approach to that of USAC, and the centralised optimal algorithm. We then study the behaviour of each agent within the network in more detail in Section 6.3. Finally, in Section 6.4, we show that by using a small value of τ, MITRA_τ achieves near optimal performance (e.g. 98% of the optimal solution can be achieved with $\tau = 8$).

6.1 Parameter Settings

To compare the performance of the algorithms, we measure the overall amount of information collected by each algorithm over time. To this end, we run each algorithm on several networks with different topologies and environmental characteristics (e.g. the occurrence frequency of the events, or the expected value of information of each event). Then, we take the average of the specific results of the networks. In order to do this, we have to create a number of networks that may differ from each other in both topology and environmental characteristics. Given this, we now describe the parameter settings, that are used throughout our simulations, in order to create these networks and their environments.

In our simulation model, the dynamic behaviour of the network is captured in both the changing environmental characteristics and the varying network topology. In order to demonstrate the dynamic nature of the former, we set up our simulation environment as follows. We first set three test environments, namely: (i) static (i.e. the environmental characteristics do not change); (ii) moderately dynamic (i.e. the environmental characteristics slowly change); and (iii) extremely dynamic (i.e. the environmental characteristics rapidly change). Note that in real world applications, the environment is typically piece–wise stationary. That is, there are time intervals whereby the environmental characteristics remain the same. This, however, does not indicate that from the perspective of one single agent, the environment is also piece–wise stationary. In particular, as outlined in Section 4.2, the performance of each agent also depends on the behaviour of the others. Thus, the reward value it can get cannot be guaranteed to be piece–wise stationary, since different combinations of the behaviour of the others may provide different reward distributions. This verifies our choice of Exp3, instead of D–MAB or Adapt–EvE, as the learning method within MAB/EM (see Section 4.2 for more details).

Now, in order to capture the piece–wise stationarity of the WSN environment, we divide the simulation time period into intervals called epochs (which are obviously not known to the agents). In each epoch, the environment has different characteristics, which affect the information value of the collected data. Here, we assume that within each epoch, the information value is randomly generated from a normal distribution, and that the mean and variance of this normal distribution changes at the transition from one epoch to the next. For example, in one epoch, agent i can collect data with information value generated from the distribution $\mathcal{N}(5, 3)$ (i.e. normal distribution with mean 5 and variance 3). Whilst in the next epoch, the distribution may change to $\mathcal{N}(45, 10)$. Furthermore, since information content valuation techniques typically assign higher information value to packets collected when extreme phenomena occur within the environment (e.g. sudden temperature increment due to fire, or unidentified vehicles enter the area) (Guestrin *et al.*, 2005; Kho *et al.*, 2009; Osborne *et al.*, 2008), the agents can sample data with higher information value in epochs that contain more extreme phenomena. Given

this, we define 5 types of environmental characteristics, each of which represents epochs which contain different environmental phenomena, as follows:

1. In this environment, packets are sampled with the information value in the range of 0 and 10, with distribution $\mathcal{N}(5,3)$ (i.e. the distribution is truncated at 0 and 10).
2. Here, packets are sampled with the information value in the range of 10 and 20, with distribution $\mathcal{N}(15,3)$.
3. The information value of each packet is in the range of 20 and 40, with distribution $\mathcal{N}(30,6)$.
4. The information value of each packet is in the range of 30 and 60, with distribution $\mathcal{N}(45,10)$.
5. The information value of each packet is in the range of 60 and 100, with distribution $\mathcal{N}(80,10)$.

Note that these numerical values are chosen such that they represent the differences between the characteristics types. Other settings with different values also show the same broad patterns in the result of the simulations. Thus, in order to capture the dynamic nature of the environment (i.e. how often it changes its characteristics), we set the length of each epoch to be $1,000$ time slots for the moderately dynamic case, 200 for the dynamic case, and 50 for the extremely dynamic case. Whilst in the static case, there is only one epoch (i.e. there is no change), with type 1 of environmental characteristics. When the environment changes its epoch, it randomly chooses one of the aforementioned characteristics types, with the probabilities of 0.5, 0.25, 0.1, 0.1 and 0.05, respectively. This represents the common observation that more extreme environmental phenomena occur less frequently.

To capture the dynamic behaviour of the network topology, we allow node failures during the operation of the WSN. In so doing, we again divide the time line into epochs of 20 (i.e. each epoch lasts for 20 time steps). At each epoch, each agent node may stop functioning for the whole epoch with probability 0.2, independently from other nodes. Nodes with failures may be functioning again in the next epoch. Note that in our settings, epochs of node failures are independent from the epochs of environmental changes.

Now, we set the energy settings of each agent node as follows. Each sensor's transmission, receiving and sampling energy consumption is uniformly and randomly chosen from intervals of $30-42$, $20-34$, and $15-25$ per packet, respectively, and the solar energy harvesting energy budget of each node varies between 500 and 1500 [4]. Given this, in our simulations, we use these values to set the parameters, such as e_i^{S}, e_i^{Tx}, e_i^{Rx}, and B_i, of the agents. In addition, the network contains 100 agents, forming a 10–layer topology, with 10 nodes in each layer. The communication edges of the network are randomly generated with probability 0.5 (i.e. two nodes within neighbouring layers can communicate with each other with probability 0.5).

Now, note that within our paper, we focus on the long-term information collection, and thus, we do not have strict constraints on the delivery time of each collected information (see Section 1 for more details). Given this, the information discount factor that we consider here is typically high (see Section 3.1). However, it would be also interesting to study the performance of our approach in systems where real–time information collection is desired. Within these systems, the real–time monitoring typically requires newest data only, and thus, the value of sampled information rapidly decreases as time passes by. This indicates that the discount factor is significantly low within such systems. Now, note that MITRA does not have any guarantee that it will deliver the sampled data to the BS within a certain time threshold (which is a key requirement in real–time monitoring systems). Given this, our hypothesis is that our approach may not perform well in systems that demand low discount factors. In order to evaluate this hypothesis in more detail, we vary the value of λ during our simulations. In particular, we set the information discount coefficient $\lambda = 0.9$, 0.5, and 0.2, respectively. The former represents the discount factor of non real–time systems, while the latter two are a typical values for real–time WSNs.

6.2 Overall Performance Evaluation

Given the parameter settings above, we now discuss the numerical results of the simulations in more detail. In particular, we study the performance of MAB/EM combined with MITRA_8 (i.e. $\tau = 8$). As we will show later in Section 6.4, the choice of $\tau = 8$ results in both low performance loss and low number of communication

[4] These values are proportional to real world sensor values as reported in Kansal and Srivastava (2003).

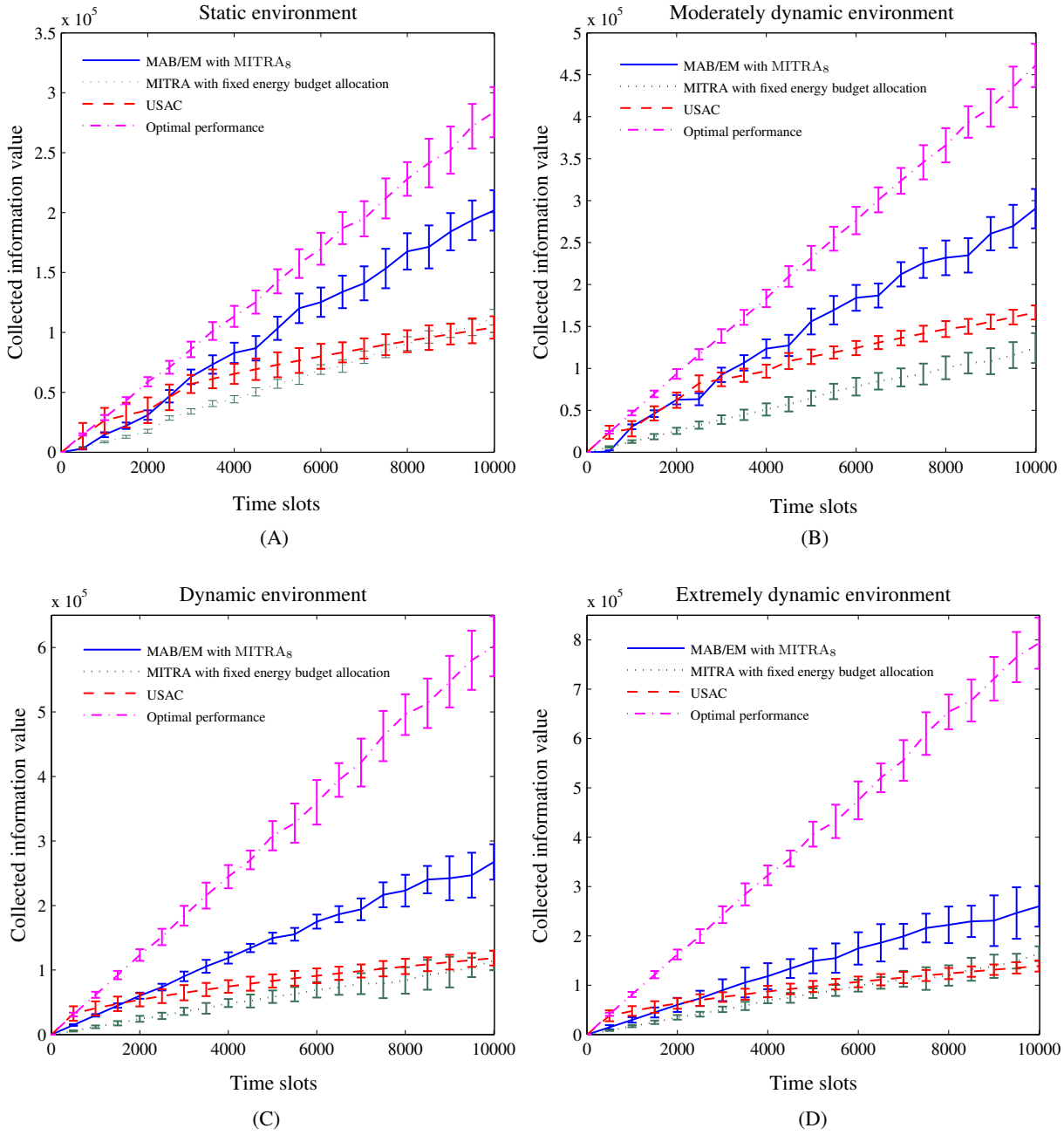

Fig. 1 Information collection in a 100–agent network with static topology and $\lambda = 0.9$, in static, moderately dynamic, dynamic, and extremely dynamic environments.

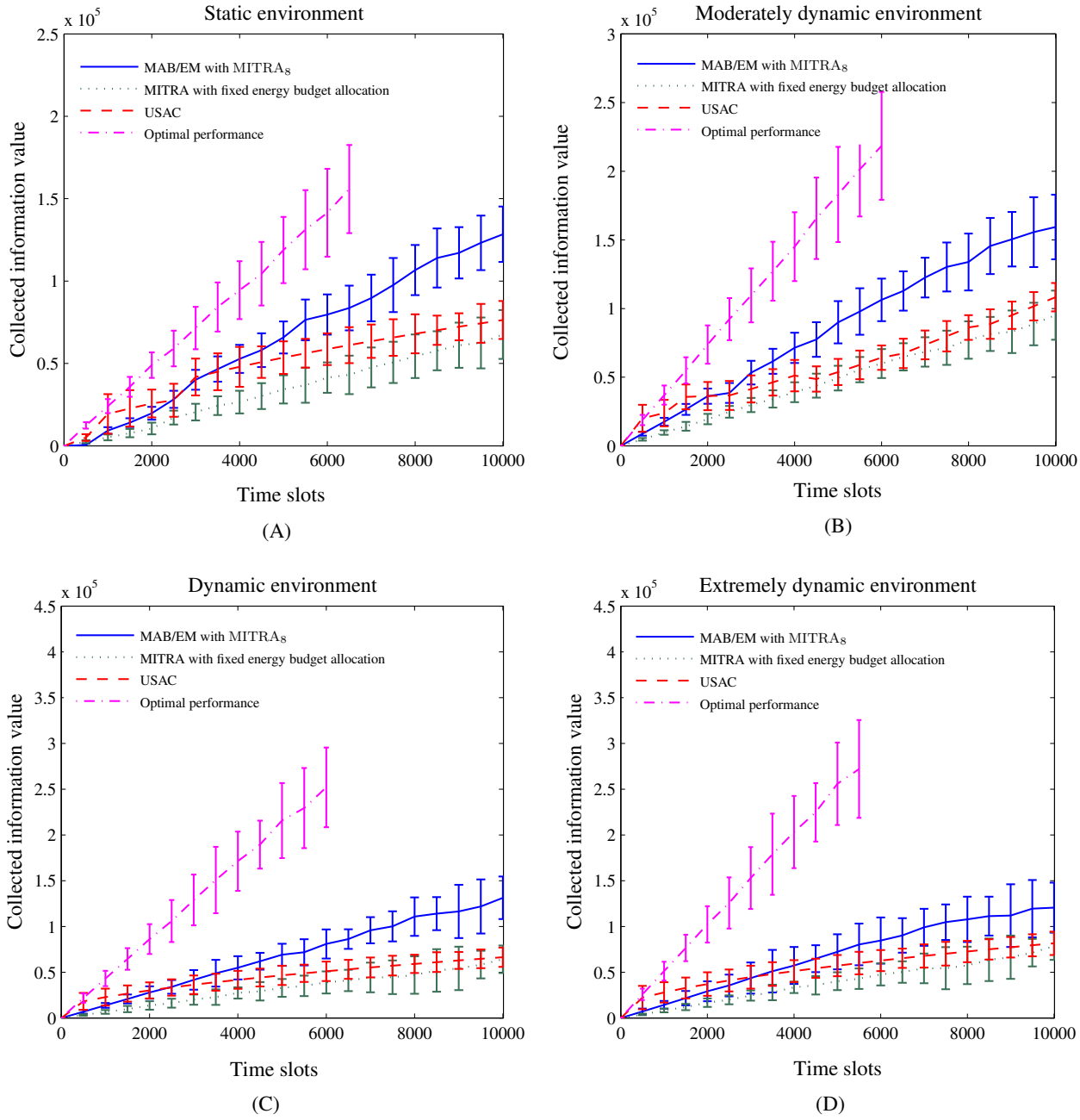

Fig. 2 Information collection in a 100–agent network with dynamic topology (probability of node failures $= 0.2$) and $\lambda = 0.9$, in static, moderately dynamic, dynamic, and extremely dynamic environments.

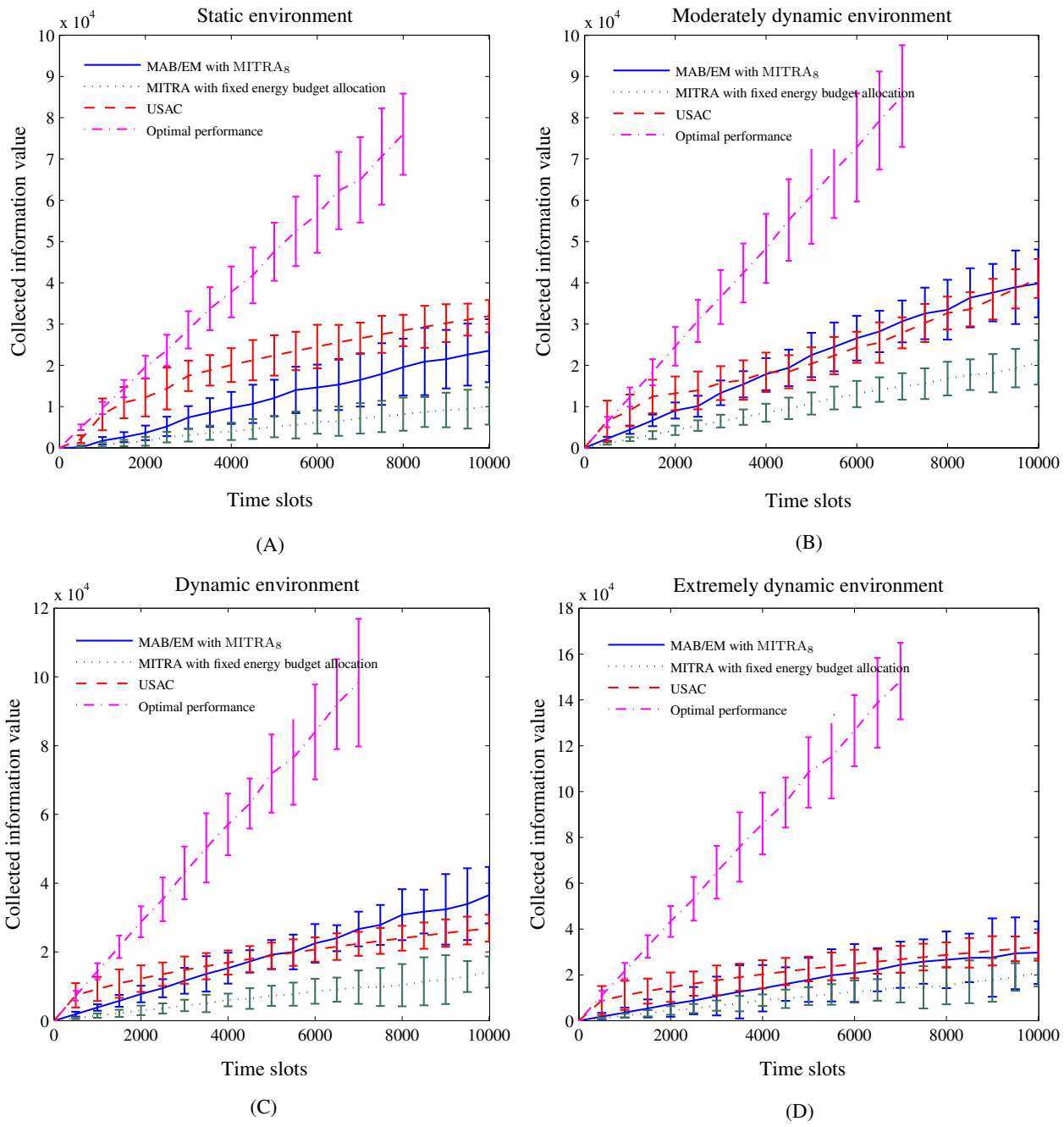

Fig. 3 Information collection in a 100–agent network with dynamic topology (probability of node failures $= 0.1$) and $\lambda = 0.5$, in static, moderately dynamic, dynamic, and extremely dynamic environments.

rounds within MITRA. Note that within our simulations, both USAC and our approach are run with the same environmental characteristics.

Now, since USAC does not follow the concept of energy neutrality, it would be unfair to compare it directly, since the agents may deplete their batteries during operation. In more detail, if USAC does not use the concept of energy neutrality, in the long term, a large number of nodes will be depleted, and thus, they cannot contribute to the overall performance of the network (see Section 2 for more details). Therefore, in our simulations, we modify USAC so that each agent i, analogously to MAB/EM, cannot exceed the energy budget B_i at each time slot. Note that with respect to this total energy budget limit, USAC can still intelligently allocate each agent's budget to the tasks it thinks to be important (see Padhy *et al.* (2010) for more details).

In our simulations, we first focus on the performance evaluation in systems where only the surrounding environment is dynamic. We then run simulations on systems with a dynamic environment and a changing topology as well. Furthermore, we evaluate the performance of our approach in real–time monitoring systems, where the data has to be delivered very quickly to the BS (i.e. the discount factor is low). The reason we choose these scenarios is that we aim to analyse the impact of different dynamic behaviours that can effect the performance of our algorithm. In more detail, within the first scenario, MAB/EM has to deal with the environmental changes only, while in the second scenario, it has to take the varying topology into account as well. In addition, within the third scenario, the system is forced to deliver the packets to the BS as fast as possible, since the information value of the packets rapidly converges to 0. Given this, we group our simulations into three scenarios, namely: (i) network with no node failures (i.e. the topology is fixed); (ii) network with node failures (i.e. with changing topology); and (iii) network with node failures and low λ.

The results of the first scenario (i.e. network with no node failures) are depicted in Figures 1a, 1b, 1c, and 1d, respectively. More precisely, these figures show the performance of the algorithms in static, moderately dynamic, dynamic, and extremely dynamic environments, where the topology is fixed. In addition, we set $\lambda = 0.9$, that is, we assume that the network is not a real–time monitoring system. Here, the error bars demonstrate the 95% confidence interval. From Figure 1a, we can see that our approach achieves up to 70% of the optimal solution in the case of a static environment. Since the environmental characteristics do not change over time, by using our approach, the agents learn the energy budget allocation combinations that best fit these characteristics. In particular, the agents achieve the best performance after $2,000$ time slots, compared to that of the others. In contrast, USAC has the best performance at the beginning, due to its efficiently combined sampling and routing behaviour (e.g. it outperforms our approach by around 150%). In more detail, USAC increases the sensors' sampling rate in unknown environments or when changes occur, and will decrease this rate as time goes by (for more details see Padhy *et al.*, 2010). However, within USAC, when a packet is chosen to be delivered to the BS, it is assumed to be delivered to the BS, regardless of possible future events. That is, USAC does not consider situations when future events may be more important, and thus, the delivery of that future information has a higher priority. Rather, the resources are already occupied to deliver the current data. This is not the case in our approach, where such delivery guarantees do not hold, and thus, USAC is outperformed by our proposed approach up to 90% in the long run. However, from Figure 1, we can also see that without the adaptive learning part (i.e. MAB/EM), our approach cannot outperform USAC. In particular, the non–learning approach (denoted as "MITRA with fixed energy budget allocation" in the figures) does not outperform USAC, and is the worst of the benchmarks. Thus, MITRA itself cannot efficiently collect the information from the environment, compared to state–of–the–art algorithms, such as USAC.

Now, as the environment becomes more dynamic, the performance of our approach is decreased, compared to that of the optimal solution (see Figures 1b,1c, and 1d, respectively). The reason is that whenever changes occur, each agent has to find a new, efficient energy budget allocation combination that fits the best to the new environmental characteristics. That is, it has to learn how to adapt to the new environment characteristics. This learning period decreases the performance of the algorithm, since less efficient energy budget allocations have to be explored before good ones can be exploited. In particular, our approach achieves, on average, up to 60% of the optimal solution in the moderately dynamic environment (see Figure 1b), and it only achieves around 40% of the optimal method in the dynamic environment (see Figure 1c). Furthermore, this value decreases down to 25% in the case of the extremely dynamic environment. However, our approach is better in all three cases. In particular, it outperforms USAC with around 100% in moderately dynamic environments, 120% in the case of

dynamic environments, and up to 83% in extremely dynamic environments. Thus, within the first scenario, our approach is more efficient in adapting to the environmental changes than USAC.

The simulation results for the scenario of networks with node failures are depicted in Figures 2a, 2b, 2c, and 2d, respectively. Similarly to Figures 1a–d, these figures show the performance of the algorithms in static, moderately dynamic, dynamic, and extremely dynamic environments, but within networks with node failures. Here, we also set $\lambda = 0.9$, and thus, we assume that there is no significant need to deliver information to the BS as quick as possible. Note that within this scenario, due to the high number of dynamic parameters to be taken into account, the centralised optimal solution, that we use as a theoretical benchmark, becomes intractable, typically after 6000 time slots [5]. Now, we can see from Figures 2a–d that the performance of our approach is decreased in this scenario, compared to that of the case of networks with no node failures. In more detail, by comparing our approach's performance to that of the optimal solution, the ratio we get in the case of networks with node failures is significantly lower than that of the case of networks without node failures. The main reason of this performance decrement is that when node failures are allowed to be occur, the system is more dynamic from the view of each single agent, since node failures may occur besides the environmental changes. That is, our approach has to adapt to significantly more dynamic behaviours here, compared to the case of networks with no node failures. Given this, since there is less time to learn the changes, MAB/EM achieves a worse performance within the case of having node failures. This fact can be easily verified by comparing the performance of our approach to that of the non–learning approach, that only uses MITRA_8. Specifically, by combining MAB/EM with MITRA_8, the overall performance in the case of networks without node failures is typically doubled (see Figures 1a–d). However, it is not the case within networks with node failures. In fact, the performance improvement that we get here is typically lower, especially in the cases where the environment is highly dynamic (see Figures 2b–d). However, our approach still outperforms USAC in all environmental settings. The main reason is that within USAC, when a packet is chosen to be delivered towards the BS, an optimal routing path is chosen, and is fixed over the time of delivery (see Padhy et al., 2010). This technique brings up the following issue: a data packet, which is already on the way towards the BS, may not be physically delivered, due to possible node failures within the routing path of that packet. Thus, USAC may waste the budgets of the agents by occupying them with delivering the current data. This is not the case in our approach, where such delivery guarantees do not hold. Note that these delivery guarantees are also the reasons why USAC is outperformed by our approach in the case of networks without node failures. However, here (i.e. within networks with node failures) our approach achieves less significant improvement, compared to the improvement it can achieve in networks without node failures. This indicates that our approach is more sensitive to the topological changes than USAC.

We now focus on the simulation results of the third scenario, where both the environment and the topology of the networks are dynamic, and the discount factor is low. In particular, we set $\lambda = 0.5$. The results are depicted in Figures 3a–d. Again, these figures show the performance of the algorithms in static, moderately dynamic, dynamic, and extremely dynamic environments, respectively. Here, since the high probability of node failures may significantly modify the performance of the algorithms (see the numerical results of the previous scenario), we set the probability of node failures to 0.1, in order to show a clear effect of low λ values on the performance of the approaches. Given this, we can clearly see that within this scenario, the performance of MITRA is significantly decreased, compared to that of USAC (recall that in the previous scenarios, these two approaches shows similar performances). The reason here is that while USAC can guarantee the delivery of packets towards the BS within a time threshold by choosing a full routing path, MITRA does not. Therefore, with our approach, a large portion of collected packets are delayed within the network, and thus, their information value is typically close to 0 when the BS receives them. This performance loss is compensated when we combine MAB/EM with MITRA, especially when environmental changes are slow enough so that MAB/EM can adapt to the changes (see Figures 3a–c). However, in all the cases, our approach cannot outperform USAC, since it is not designed for fast packet delivery. In more detail, we can clearly see this in Figure 3a, where the environment is static. In this case, since USAC does not have to deal with the challenges of environmental changes, it significantly outperforms our approach. As the environmental changes become more frequent, but still slow enough so that MAB/EM can adapt to these changes, the performance of USAC is decreased, while our approach shows improvement in its performance (see Figures 3b–c). However, when the environment becomes extremely dynamic, the learning approach cannot adapt to the

[5] During our simulations, we used a Intel® Core 2^{TM} Quad q9400 computer with 4GB memory, and Java$^{\mathrm{TM}}$ 1.6.0_18 RE.

		End node	**Middle node**	**Start node**
	B^{Rx}	6.6%	35.3%	39.7%
$\lambda = 0.9$	B^{Tx}	56.3%	46.9%	43.1%
	B^{S}	37.1%	17.8%	17.2%
		End node	**Middle node**	**Start node**
	B^{Rx}	4.9%	30.7%	26.9%
$\lambda = 0.5$	B^{Tx}	53.3%	42.4%	41.3%
	B^{S}	41.8%	26.9%	31.8%
		End node	**Middle node**	**Start node**
	B^{Rx}	5.1%	17.4%	9.1%
$\lambda = 0.2$	B^{Tx}	25.7%	42.5%	49.5%
	B^{S}	69.2%	40.1%	41.4%

Table 1 Detailed agent behaviour within network with random topology.

		End node	**Middle node**	**Start node**
	B^{Rx}	5.2%	40.3%	37.2%
$\lambda = 0.9$	B^{Tx}	57.4%	51.6%	44.5%
	B^{S}	37.4%	8.1%	18.3%
		End node	**Middle node**	**Start node**
	B^{Rx}	4.8%	31.7%	30.3%
$\lambda = 0.5$	B^{Tx}	56.5%	46.8%	45.1%
	B^{S}	38.7%	21.5%	24.6%
		End node	**Middle node**	**Start node**
	B^{Rx}	4.7%	18.4%	6.7%
$\lambda = 0.2$	B^{Tx}	22.1%	31.3%	53.7%
	B^{S}	73.2%	50.3%	39.6%

Table 2 Detailed agent behaviour within network with star topology.

changes anymore, and thus, both USAC and MAB/EM show significantly lower performance (see Figure 3d). Note that we also run the simulations with $\lambda = 0.2$. However, due to the very low discount factor, both USAC and our approach show very poor performance in information collection, and there is no significant differences between their performance. Against this background, we can clearly state that our approach is not suitable for systems where information rapidly decays.

To conclude the performance evaluation of our approach using MAB/EM in conjunction with MITRA_8, we can state that it significantly outperforms USAC when the information discount factor is high, while it is not suitable for systems with rapidly decaying information. We also demonstrated that the learning efficiency of MAB/EM is decreased when the changes are rapid (e.g. the environment is extremely dynamic, or environmental changes are combined with node failures). Furthermore, we showed that MITRA is sensitive to the value of λ, and thus, its inefficiency in systems with low values of λ significantly decreases the overall performance of our approach as well. This can be verified by comparing the performance of our approach with that of the theoretical optimal approach. However, apart from real–time systems, our approach typically outperforms USAC. Given this, our approach is suitable for system with non real–time requirements.

6.3 Agent Behaviours within Different Network Settings

To better understand the behaviour of our approach, we now focus on a more detailed study of MAB/EM in this section. In particular, we aim to determine the typical energy budgets that each agent allocates to sampling, receiving and transmitting. However, as the numerical results within the previous section indicate, the behaviour of MAB/EM depends on the value of the discount factor, and the environmental changes. Thus, one of our objectives is to investigate these dependencies. Furthermore, the behaviour of the agents may also depend on the

		End node	**Middle node**	**Start node**
	B^{Rx}	4.1%	34.2%	40.5%
$\lambda = 0.9$	B^{Tx}	55.7%	46.9%	54.2%
	B^{S}	40.2%	18.9%	5.3%
		End node	**Middle node**	**Start node**
	B^{Rx}	3.5%	30.9%	25.8%
$\lambda = 0.5$	B^{Tx}	54.7%	44.3%	52.6%
	B^{S}	41.8%	24.8%	21.6%
		End node	**Middle node**	**Start node**
	B^{Rx}	4.3%	13.2%	8.7%
$\lambda = 0.2$	B^{Tx}	36.1%	35.1%	48.5%
	B^{S}	59.6%	51.7%	42.8%

Table 3 Detailed agent behaviour within network with chain topology.

network topology as well. In particular, the network topology restricts the set of agents that a particular agent node can interact with, while using MITRA. In addition, the distance between the agents and the BS may also have a significant impact on the agents' behaviour, since the closer an agent is to the BS, the more it needs to be able to handle a larger amount of traffic throughput that it has to forward to the BS. Given this, beside the network topology we described in Section 6.1, which we will refer to as the "random network", we run our simulations on two additional networks, namely: (i) network with "star topology"; and (ii) network with "chain topology". The former has 21 nodes, that together form 5 layers, where the middle layer contains exactly one node (i.e. the star node), while the other layers have 5 nodes each. Each node can communicate with any of the nodes within its neighbouring layers. On the other hand, the latter topology has 10 nodes, forming a 10–layer long chain, with exactly one node in each layer. Note that here, the random topology represents a regular network (i.e. the network topology is arbitrary), the star topology represents networks with some bottle–neck nodes (i.e. all the traffic of the network has to go through these nodes), while the chain topology represents networks where all the traffic has to follow a certain routing path.

We first study the impact of the network topology and the discount factor to the behaviour of MAB/EM in more detail. In particular, we investigate the average budget percentage (compared to the total energy budget) that a single agent allocates to data sampling, receiving and transmitting. In so doing, we calculate these budget allocation values with different λ, network topology, and distance of the agent from the BS. For varying the latter parameter, we focus on three different type of distances. The first type is the nodes that are farthest from the BS (i.e. the node lies in the last layer). We refer to this node as the "end node". The second type of node that we investigate is the one that is in the middle layer, which we refer to as the "middle node". Note that the middle node is in fact the star node in the case of the star topology. Furthermore, since the random topology contains 10 layers, it does not have the middle layer. Given this, within this topology, the middle node is chosen from layer 5. The third type is the one that is next to the BS (i.e. it can directly communicate with the BS), which is referred to as the "start node".

The results are depicted in Tables 1, 2, and 3, respectively. In more detail, Table 1 depicts the average budget allocation values of the end node, middle node, and start node in the random topology, with $\lambda = 0.9, 0.5$, and 0.2. For example, an end node within the random topology typically allocates 6.6% of its budget to receiving, 56.3% to transmitting, and 37.1% to sampling, when $\lambda = 0.9$. Similarly, Tables 2 and 3 contains these values for star and chain topologies, respectively. Here we set the environmental changes to be static, with no node failures, since within this type of environment, we can clearly observe the impact of the abovementioned factors. Note that we also evaluated our simulations with more dynamic environmental settings (i.e. changing information value distributions, and occurrence of node failures). But due to the dynamic environmental changes, the behaviour of the agents becomes more diverse, and thus, the patterns are less likely to be observed.

Now, from the tables, we can clearly see that as λ decreases, the nodes becomes more selfish, that is, each node puts more effort on sampling its own data. The reason for this is that with low λ values, the information value of packets, that are received/or sampled in the past, decays quickly, so that the information value of newly sampled data will dominate the reward that the agent can get when it evaluates its chosen action. In addition, we

can observe that the end node typically focuses on sampling and transmitting only, since they can learn that there is no data to receive. However, when the agent is closer to the BS, it changes its focus from sampling its own data to relaying others'. This pattern can be observed most clearly in the case of $\lambda = 0.9$. Furthermore, we can see that the pattern of the nodes' behaviour does not depend much on the type of the underlying network topology. In particular, it allocates similar budget ratios to each of the sensory tasks in different topologies. However, there are notable exceptions. For example, in the case of the star topology and $\lambda = 0.9$, the middle node behaves differently than in the other cases. Specifically, it allocates significantly lower budget to sampling, compared to the other cases. The reason here is that it has to take into account the others' collected data as well, as the information value of this data does not decay rapidly. In a similar vein, the start node also allocates significantly less budget to sampling within the case of chain topology and $\lambda = 0.9$, compared to other cases. To conclude, we can say that the behaviour of the nodes typically does not depend on the topology. Furthermore, the closer the agents are to the BS, the more they are willing to relay other agents' collected data. However, the cooperativeness between the agents decreases, and thus, they allocate more budget to sampling their own data, as the information discount factor decrease.

6.4 Performance Evaluation of MITRA_τ

Given the simulation results in the previous section, we can see that MITRA_τ, together with MAB/EM, performs well with $\tau = 8$. As mentioned in Secton 5.4, the advantage of using MITRA_τ instead of MITRA is that the former has limited communication cost. This limitation implies that the performance of MITRA_τ is decreased, compared to that of MITRA, which is proveably optimal. However, we shall now show that MITRA_τ still achieves near–optimal performance, even with small values of τ, by studying the performance of MITRA_τ with different values of τ. The performance of these MITRA_τ algorithms is compared to that of MITRA with an unlimited number of communication rounds. Note that MITRA may use tens of rounds in order to achieve optimal routing performance (as outlined in Section 5.4).

Given this, the numerical results are depicted in Figure 4. In particular, the figure contains the performance of MITRA_τ with $\tau = \{1, 2, 4, 8\}$. Note that we also have evaluated the performance of MITRA_τ with higher values of τ, but their improvement is not significant, compared to that of MITRA_8. From Figure 4, we can see that MITRA_1 achieves the lowest performance (it performs 60% less well than the optimal solution in the case of networks with 100 agents). With $\tau = 2$, and $\tau = 4$, MITRA_τ achieves better results, but their performance loss (i.e. the difference between their performance and that of the optimal solution) is still significant. In particular, MITRA_2 performs, on average, 60%, whilst MITRA_4 achieves around 80% of the optimal solution in the case of 100 agents. In contrast, we can see that with $\tau = 8$, even in the case of networks with 100 agents, the performance of MITRA_τ is around 98% of the optimal unlimited MITRA. That is, by limiting the number of communication rounds that MITRA can use to $\tau = 8$, our approach still achieves near–optimal solution with around 2% performance loss. On the other hand, according to Theorem 3, MITRA without communication round limit may use up to 66 rounds in order to achieve optimal routing performance. That is, by limiting the communication rounds to $\tau = 8$, we can reduce the number of communication rounds by 87.5%. Given this, by using MITRA_8, the number of used communication rounds is small enough so that the total time needed for coordination will not exceed the size of the time slot. This, in particular, verifies our choice of MITRA_8 in Section 6.2.

7 Conclusions and Future Work

In this paper, we studied long-term information collection in the WSN domain. In particular, first we looked at previous on work energy management and data routing in WSNs. These are the perspectives from which efficient information collection can be achieved. Following this, we introduced our model for WSNs that includes the aforementioned perspectives, and we formalised our main objective of maximising the total amount of collected information at the base station over a given finite time interval.

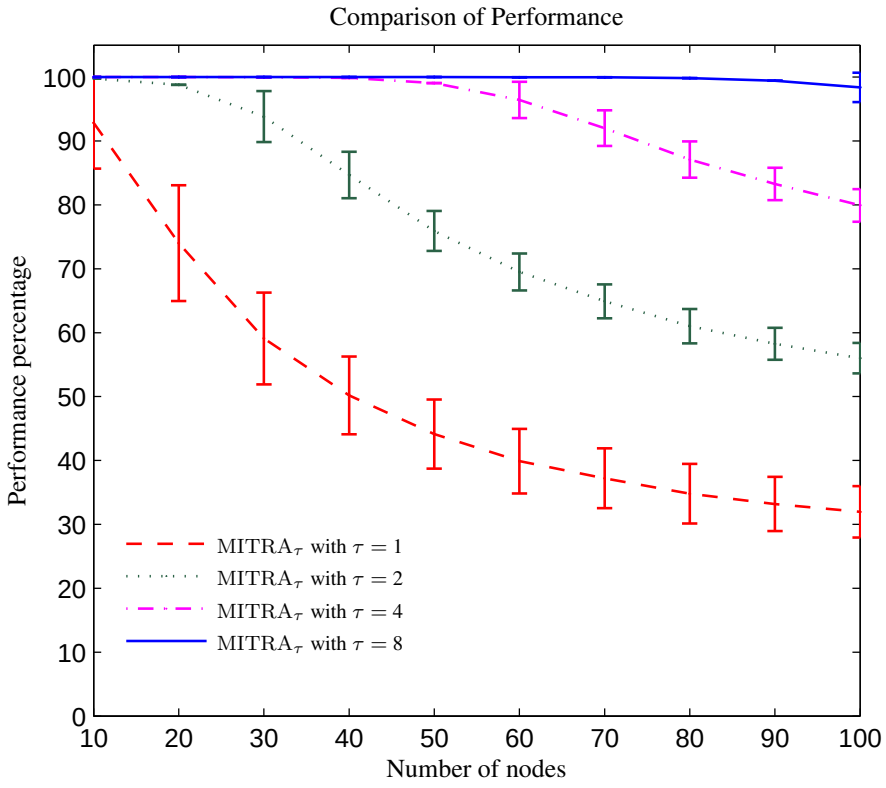

Fig. 4 Performance comparison of MITRA_τ with that of the unlimited MITRA. The optimal performance achieved by MITRA is 100%.

In a review of the relevant literature, we demonstrated that none of the state-of-the-art algorithms in the aforementioned perspectives meets all of our research requirements for long-term information collection. Given this, we developed a novel model that enables learning and adapting to the environmental changes. In particular, we focused on adaptive energy management and routing, whilst we assumed that the data sampling and information content valuation parts of the model use state-of-the-art techniques. In so doing, we decomposed the original problem into two sub–problems, namely: (i) energy management; and (ii) maximal information throughput routing.

Against this background, we proposed a multi–armed bandit based approach called MAB/EM for the energy management problem. In particular, we first introduced the MAB model, which forms the foundation of our approach. Then we reduced the energy management problem into a MAB problem, by defining the actions and the reward functions for the agents. Following this, we described in more detail a state-of-the-art MAB algorithm, Exp3, that can be used to efficiently deal with the energy management problem.

For the maximal information throughput routing problem, we devised two decentralised routing algorithms, MITRA (for maximal information throughput routing algorithm), and MITRA_τ, respectively. We proved that MITRA provides the optimal solution for the maximal information throughput routing problem. Furthermore, we also provided an upper bound for the number of communication rounds that MITRA needs to use within a time slot. Since the total number of communication rounds that MITRA uses may be large, we modified MITRA so that the number of communication rounds is reduced. The modification resulted in the introduction of MITRA_τ.

Following this, we demonstrated the efficiency of our approach (i.e. MAB/EM combined with MITRA_τ, where $\tau = 8$) through extensive simulation results. In particular, to measure the efficiency of our approach, we

compared with a state–of–the–art non–learning information collection algorithm, USAC. Moreover, to measure the performance surplus that MAB/EM adds to our approach, we also used a non–learning algorithm, that solely uses MITRA, as a benchmark method. Both comparisons showed that MAB/EM with MITRA_τ together are efficient in terms of long–term information collection, since it can adapt to the environmental changes. In particular, we demonstrated that, within systems with high values of information discount factor, our approach outperforms USAC. However, we also showed that as the discount factor is decreased, the performance of our approach also decreases. In addition, we also empirically showed that by choosing small values of τ, near–optimal routing performance can still be achieved, whilst the number of communication rounds is significantly reduced. Given this, the integrated model and the proposed algorithms are particularly useful for non–real time monitoring systems (i.e. the information discount factor is high), in which the environment has to be monitored over a prolonged time interval, and unpredicted, important events should be distinguished from the other events.

In our WSN model, we assumed that agents can harvest energy over the whole period of their operation time, and thus, they can follow the concept of energy–neutrality. In particular, we assumed that the energy budget that each agent can use at each time slot is fixed over time. This indicates that the agent's action set does not change over time either. However, there are systems in which this assumption does not hold. In particular, for example, sensor agents with solar energy collectors may not perform well in energy harvesting if they are deployed in environments with cloudy weather. In these cases, agents may fail to comply with the concept of energy–neutrality. Given this, we aim to extend our work to systems in which energy harvesting is either not efficient or not feasible. More precisely, within these systems, agents need to decide whether to follow the concept of energy–neutrality, and thus, they should be able to save energy for time slots when energy harvesting is not possible, in order to achieve efficient long–term information collection. However, this implies that the current MAB model is not suitable for such systems, since the action set of each agent is not fixed over time. Given this, we need to modify the MAB concept so that this extended model can be efficiently tackled as well.

References

Aghaeil, R. G., Rahman, M. A., Gueaieb, W., and Saddik, A. E. (2007). Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks. *In Proceedings of the Instrumentation and Measurement Technology Conference*, pages 1–6.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, **38**, 393–422.

Anastasi, G., Conti, M., Falchi, A., Gregori, E., and Passarella, A. (2004). Performance measurements of mote sensor networks. *Proceedings of the ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile System*, pages 174–181.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2003). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, **32(1)**, 48–77.

Baldus, H., Klabunde, K., and Muesch, G. (2004). Reliable set-up of medical body-sensor networks. *In Proceedings of the First European Workshop on Wireless Sensor Networks*, pages 353–363.

Beeby, S. P., Tudor, M. J., and White, N. M. (2006). Energy harvesting vibration sources for microsystems applications. *Measurement Science and Technology*, **9**, 175–195.

Boukerche, A. (2008). *Algorithms and Protocols for Wireless Sensor Networks*. WileyBlackwell.

Braginsky, D. and Estri, D. (2002). Rumor routing algorithm for sensor networks. *In Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, pages 22–31.

Bulka, B., Gaston, M., and desJardins, M. (2007). Local strategy learning in networked multi-agent team formation. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, **15(1)**, 29–45.

Cassandra, A. R. (1998). *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. Ph.D. thesis, Brown University, Department of Computer Science, Providence, RI, USA.

Chong, C.-Y. and Kumar, S. P. (2003). Sensor networks: Evolution, opportunities and challenges. *Proceedings of IEEE*, **91(8)**, 1247–1256.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. John Wiley and Sons.

DaCosta, L., Fialho, A., Schoenauer, M., and Sebag, M. (2008). Adaptive operator selection with dynamic multi-armed bandits. *In Proceedings of the Tenth Annual Conference on Genetic and Evolutionary Computation*, pages 913–920.

Degesys, J. and Nagpal, R. (2008). Towards desynchronization of multi-hop topologies. *In Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 129–138.

Dekorsy, A., Fliege, J., and Sollner, M. (2007). Optimal distributed routing and power control decomposition for wireless networks. *In Proceedings of the Fiftieth IEEE Global Telecommunications Conference*, pages 4920–4924.

Demirkol, I., Ersoy, C., and Alagoz, F. (2006). Mac protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, **44**(4), 115–121.

Dinga, W., Iyengara, S., Kannana, R., and Rummler, W. (2004). Energy equivalence routing in wireless sensor networks. *Microprocessors and Microsystems*, **28**, 467–475.

Elson, J. and Estrin, D. (2001). Time synchronization forwireless sensor networks. *Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*.

Frieden, B. R. (2004). *Science from Fisher Information: A Unification*. Cambridge University Press.

Gelenbe, E. and Lent, R. (2004). Power-aware ad hoc cognitive packet networks. *Ad Hoc Networks Journal*, **2(3)**, 205–216.

Guestrin, C., Krause, A., and Singh, A. P. (2005). Near-optimal sensor placements in gaussian processes. *In Proceedings of the Twenty-second International Conference on Machine Learning*, pages 265–272.

Hartland, C., Gelly, S., Baskiotis, N., Teytaud, O., and Sebag, M. (2006). Multiarmed bandit, dynamic environments and meta-bandits. *Online Trading of Exploration and Exploitation Workshop, NIPS*.

He, T., Vicaire, P., Yan, T., Luo, L., Gu, L., Zhou, G., Stoleru, R., Cao, Q., Stankovic, J., and Abdelzaher, T. (2006). Achieving real-time target tracking using wireless sensor networks. *In Proceedings of the Twelfth IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 37–48.

Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. *In Proceedings of the Hawaii International Conference on System Sciences*, pages 1–10.

Horling, B., Lesser, V., Vincent, R., and Wagner, T. (2006). The soft real-time agent control architecture. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, **12(1)**, 35–91.

Intanagonwiwat, C., Govindan, R., and Estrin, D. (2003). Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, **11(1)**, 2–16.

Jain, A. and Chang, E. Y. (2004). Adaptive sampling for sensor networks. *In Proceedings of the First Workshop on Data Management for Sensor Networks*, pages 10–16.

Jain, M., Taylor, M. E., Tambe, M., and Yokoo, M. (2009). Dcops meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In *In Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 181–186.

Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Communications of the ACM*, **44(4)**, 35–41.

Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L. S., and Rubenstein, D. (2002). Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *In Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 96–107.

Kansal, A. and Srivastava, M. B. (2003). An environmental energy harvesting framework for sensor networks. *In Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, pages 481–486.

Kho, J. (2009). *Decentralised Control of Wireless Sensor Networks*. Ph.D. thesis, University of Southampton, School of Electronics and Computer Science, Southampton UK.

Kho, J., Rogers, A., and Jennings, N. (2009). Decentralised adaptive sampling of wireless sensor networks. *ACM Transactions on Sensor Networks*, **5(3)**, 19–53.

Kho, J., L. Tran-Thanh, L., Rogers, A., and Jennings, N. R. (2010). An agent-based distributed coordination mechanism for wireless visual sensor nodes using dynamic programming. *The Computer Journal*, **53(8)**, 1277–1290.

Krause, A., Guestrin, C., Gupta, A., and Kleinberg, J. (2006). Near-optimal sensor placements:maximizing information while minimizing communication cost. *In Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, pages 2–10.

Lesser, V., Ortiz, C., and Tambe, M., editors (2003). *Distributed sensor networks: a multiagent perspective*. Kluwer Publishing.

Li, P., Gu, Y., and Zhao, B. (2007). A global-energy-balancing real-time routing in wireless sensor networks. *In Proceedings of the IEEE Asia-Pacific Services Computing Conference*, pages 89–93.

Lindsey, S. and Raghavendra, C. S. (2002). Pegasis: Power efficient gathering in sensor information systems. *In Proceedings of the IEEE Aerospace Conference*, **3**, 3.1125–3.1130.

Mathur, G., Desnoyers, P., Ganesan, D., and Shenoy, P. (2006). Ultra-low power data storage for sensor networks. *Proceedings of the Fifth international conference on Information processing in sensor networks*, pages 374–381.

Merrett, G. V. (2008). *Energy- and Information-Managed Wireless Sensor Networks: Modelling and Simulation*. Ph.D. thesis, University of Southampton, School of Electronics and Computer Science, Southampton UK.

Mihaylov, M., Tuyls, K., and Now, A. (2009). Decentralized learning in wireless sensor networks. pages 60–73.

Ok, C., Lee, S., Mitra, P., and Kumara, S. (2009). Distributed energy balanced routing for wireless sensor networks. *Computers & Industrial Engineering*, **57**, 125–135.

Osborne, M. A., Rogers, A., Ramchurn, S., Roberts, S. J., and Jennings, N. R. (2008). Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. *In Proceedings of the Seventh International Conference on Information Processing in Sensor Networks*, pages 109–120.

Padhy, P., Dash, R. K., Martinez, K., and Jennings, N. R. (2010). A utility-based adaptive sensing and multihop communication protocol for wireless sensor networks. *ACM Transactions on Sensor Networks*, **6(3)**, 1–39.

Pchouek, M. and Mak, V. (2008). Industrial deployment of multi-agent technologies: review and selected case studies. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, **17(3)**, 397–431.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society*, **55**, 527–535.

Rogers, A., Corkill, D. D., and Jennings, N. R. (2009). Agent technologies for sensor networks. *IEEE Intelligent Systems*, **24**(2), 13–17.

Romer, K. and Mattern, F. (2004). The design space of wireless sensor networks. *IEEE Wireless Communications*, **11(6)**, 54–61.

Roundy, S., Steingart, D., Wright, L. F. P., and Rabaey, J. (2004). Power sources for wireless sensor networks. *Proceedings of the First European workshop on wireless sensor networks*, **2920**, 1–17.

Seuken, S. and Zilberstein, S. (2008). Formal models and algorithms for decentralized decision making under uncertainty. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, **17(2)**, 190–250.

Simon, G., Ledeczi, A., and Maroti, M. (2004). Sensor network-based countersniper system. *In Proceedings of the Second International Conference on Embedded Networked Sensor Systems*, pages 1–12.

Soh, L.-K. and Tsatsoulis, C. (2005). A real-time negotiation model and a multi-agent sensor network implementation. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, **11(3)**, 215–271.

Stankovic, J. A. (2004). Research challenges for wireless sensor networks. *ACM SIGBED Review*, **1(2)**, 9–12.

Sundararaman, B., Buy, U., and Kshemkalyani, A. D. (2005). Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, **3(3)**, 281–323.

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

Torah, R., Glynne-Jones, P., Tudor, M., O'Donnell, T., Roy, S., and Beeby, S. (2008). Self-powered autonomous wireless sensor node using vibration energy harvesting. *Measurement Science and Technology*, pages 125202.1–125202.8.

Vigorito, C., Ganesan, D., and Barto, A. (2007). Adaptive control of duty-cycling in energy-harvesting wireless sensor networks. *In Proceedings of the Fourth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 21–30.

Wagner, D. and Wattenhofer, R. (2007). *Algorithms for Sensor and Ad Hoc Networks: Advanced Lectures*. Springer.

Willett, R., Martin, A., and Nowak, R. (2004). Backcasting: Adaptive sampling for sensor networks. *In Proceedings of the Third International Symposium on Information Processing in Sensor Networks*, pages 124–156.

Wu, T. and Biswas, S. (2007). Minimizing inter-cluster interference by self-reorganizing mac allocation in sensor networks. *Wireless Networks*, **13**(5), 691–703.

Younis, M., Akkaya, K., Eltoweissy, M., and Wadaa, A. (2004). On handling qos traffic in wireless sensor networks. *Hawaii International Conference on System Sciences*, **9**, 90292a.

Zhang, P., Sadler, C., Lyon, S., and Martonosi, M. (2004). Hardware design experiences in zebranet. *In Proceedings of the Second international conference on Embedded networked sensor systems*, **19**, 227–238.

Zhang, Y. and Huang, Q. (2006). A learning-based adaptive routing tree for wireless sensor networks. *Journal of Communications*, **1(2)**, 12–21.

Zhou, J. and de Roure, D. (2007). Floodnet: Coupling adaptive sampling with energy aware routing in a flood warning system. *Journal of Computer Science and Technology*, **22(1)**, 121–130.