A Node Discovery Service for Partially Mobile Sensor Networks

Vladimir Dyo
Department of Computer Science
University College London
UK London WC1E6BT
v.dyo@cs.ucl.ac.uk

Cecilia Mascolo
Department of Computer Science
University College London
UK London WC1E6BT
c.mascolo@cs.ucl.ac.uk

ABSTRACT

Wireless Sensor Networks (WSNs) are challenging types of networks where resources can be scarce. In particular, battery is often a very limited resource and the radio interface is the culprit for most of the energy consumption. This makes any discovery of other sensors a difficult task, less cumbersome if sensors are fixed but crucial if (some) sensors start being mobile (such as in wildlife monitoring projects with tagged animals). In this paper we propose a middleware offering node discovery for partially mobile wireless sensor networks, where fixed nodes (sinks), deployed in the environment to monitor the movement of entities, detect those patterns with low power consumption. The approach is based on various machine learning techniques which allows for learning and adapting the wake up strategy of the sinks dynamically. We also report on the evaluation of the approach through simulation and use of real movement traces.

1. INTRODUCTION

Energy saving in wireless sensor networks is an issue of paramount importance. This is because in most scenarios the location of the sensors forbids both the use of constant power sources and the frequent replacement of the batteries. Energy scavenging solutions are also sometimes not sufficient, due, for example, to limited solar exposure or to the footprint of a possible solar panel. Most sensor networks therefore use some *duty cycling* to control the awake time of the sensor (or of its radio interface, which often is the most considerable source of power consumption).

Duty cycling however may jeopardize the ability to successfully discover neighbouring nodes: this is less of a problem with a fixed sensor network where nodes can discover once in a while and then synchronize but becomes an issue in mixed sensor networks where some nodes start being mobile. In these networks nodes arrival times is often uncertain. To obtain a full discovery, a node would have to be awake all the time, waiting for a contact with other nodes which would limit the useful lifetime of a typical mote to just 80 hours[1].

In this paper we present a node discovery approach for partially mobile sensor networks, where sink nodes, aiming at registering the presence of the moving nodes around them, optimize their duty cycle through the use of machine learning based techniques which learn movement patterns and adapt to changes dynamically. We formulate the problem in terms of reinforcement learning and suggest a control strategy for a sink. The control strategy adjusts the sink's duty cycle depending on the mobile node arrival patterns, and attempts to maximize the number of registered encounters within a given limited energy budget, while investing enough energy into exploration of alternative options.

Applications of these ideas are mainly, but not only, in the area of wildlife monitoring, where sink nodes, strategically positioned in crucial places, register movement and encounters of animals tagged with sensors. However other applications of human movement in other environments could offer other examples.

In this paper we also report about the evaluation we have conducted through the use of real human connectivity traces using R statistical package [20]. Our evaluation compares different machine learning techniques and reports of the strengths and limitations in various conditions.

The paper is organized as follows: Section 2 describes our approach. Section 3 illustrates a reinforcement learning framework and explains why it is suitable for our problem. Section 4 presents our results and Section 6 indicates possible future work.

2. SYSTEM OVERVIEW

In this section we give an overview of our approach and of its basic assumptions.

2.1 Assumptions

We consider our sensor network to consist of numerous wireless mobile sensor nodes and static sinks located in strategic locations (Figure 1). The network is sparsely connected and used for tracking and data collection applications.

We assume asynchronous operation. Global time synchronization requires precise clocks or GPS which is expensive and not always available. The use of time synchronization protocols [10, 18] is of course possible but limited to connected areas of the network only. This means the nodes need to adopt asynchronous operation and rely on synchronization only where possible. The precision of the clocks is

[1]TMote Sky 30mA @3V, with AA 2500mA battery

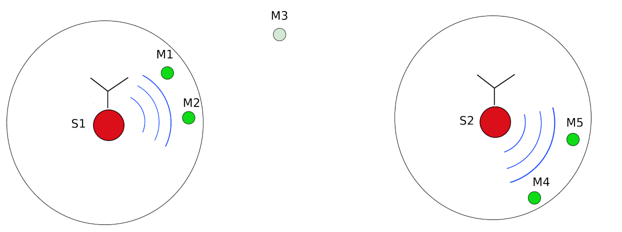

Figure 1: S1, S2 - static sinks, M1-M5 - mobile nodes. R - discovery range. M3 is unreachable by both sinks.

limited by cost of a crystal and is about 50 parts per million for inexpensive watch crystals.

Our approach targets applications which demand low energy consumption, such applications where battery replacement is not easy and energy scavenging is impossible or limited. Applications do not need to be necessarily long running, but in general the framework is suitable for those where battery size needs to be limited, e.g., a mote attached to a small animal or to a patient might be required to operate on a coin cell battery, which demands for a very low duty cycle.

The approach is based on the assumption that sensor discovery is an expensive service. This applies to a variety of low-power wake-up scheduling protocols, where, in the absence of global synchronization, a node has to use long continuous transmissions to advertise its presence to low duty cycled neighbours.

2.2 The Middleware

The node discovery service we propose allows application developers to specify a daily energy budget for potentially expensive node discovery function. The daily energy budget is equivalent to an average daily duty cycle and provides a convenient way to budget and plan the node's battery power. Our middleware learns the arrival pattern of the mobile nodes and provides a feedback to an application by suggesting how to optimally spread the duty cycle throughout a day.

Because learning is expensive, the middleware has to balance exploitation and exploration costs. High resolution data allow to take better decisions, but are expensive to obtain, thus negating the benefit of learning, i.e., frequent beaconing will guarantee fast discovery of potential neighbours but will quickly deplete the nodes battery. Low resolution data lead to less optimal decisions, but is relatively cheap to obtain in terms of energy.

We have investigated machine learning techniques to optimize the discovery algorithm of the sinks. The techniques try to learn and exploit the temporal patterns in mobile node arrivals and wake-up nodes only when the arrival is expected. The static sinks located in various locations are responsible for waking up and then communicating with mobile sensors. The mobile nodes only perform periodic carrier sensing and therefore work at a very low duty cycle.

The rest of the discussion will assume that the middleware will work on top of low power listening (LPL) based MAC protocols. The low power listening technique, also known as carrier sensing has been first proposed by [9] for the WiseMAC protocol. In LPL, the nodes periodically

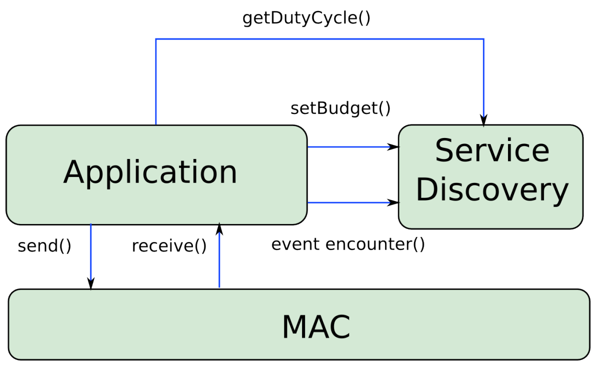

Figure 2: Middleware Architecture.

wake-up and sample for energy in the channel for very short amount of time. Carrier sensing takes a very short amount of time, much shorter than receiving an entire packet. When a node (say node B) needs to send data, it prepends each data packet with a preamble, which is long enough to wait for the time needed for its neighbour(s) to wake up.

In our middleware, the role of node discovery and beaconing is performed by static sinks, whereas mobile nodes do only carrier sampling and work at a very low duty cycle. We assume that the mobile nodes have fixed check interval and sinks have fixed preamble length, so the energy consumption of sinks depends on frequency of service advertisements (packets with long preambles).

In the following section we describe the learning problem and how we balance exploitation and exploration costs.

3. WAKE-UP ALGORITHM

We formulate the node discovery problem as follows: *Given a limited daily energy budget P, what is the best wake-up strategy to maximize the number of encounters, if the temporal distribution of node arrivals is not known?* [2].

3.1 Example scenario

Let us consider a daily distribution of node arrivals presented on Figure 3 extracted from real human connectivity traces taken from Dartmouth College Campus [16], which we used as a possible example of mobility patterns of hypothetical entities. The traces contain log entries of wifi users registering with wifi access points installed around the campus.

To maximize the number of detected encounters a sink could be trained to wake up between 11am and 12am, when the probability of contact is the highest and occasionally try other timeslots to detect any potential changes (*ε-Greedy strategy*). Alternatively, the sink can pick a timeslot with probability proportional to the height of the peak (*Boltzmann strategy*). Thus, the highest peak would be more likely to be chosen, but if there are several peaks, they will be all equally likely to be chosen. Finally, it could be trained to spread the load to all time slots, but adjusting a duty cycle proportionally to the height of the peak (*Balanced strategy*). Thus spreading risk evenly throughout a day.

[2]Note that specifying a daily energy budget is equivalent to specifying an average daily duty cycle

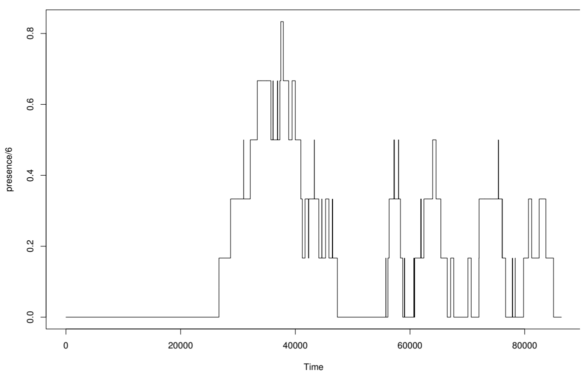

Figure 3: The probability of contact depends on a time of day. The middleware learns the temporal pattern in arrival of nodes and saves power during quiet periods.

Table 1: CC2420 power consumption

State	Power Consumption
transmit, P = 0dBm (max)	17.4mA
transmit, P = -25dBm (min)	8.5mA
receive	19.7mA
idle mode	426μA
power down	20μA

3.2 Reinforcement Learning Approach

Reinforcement learning [15] provides a formal framework for describing and solving our problem. Reinforcement learning algorithms try to learn by themselves in a trial-and-error fashion by iterating through all possible options and remembering the best ones. Reinforcement learning framework is particularly suitable to our problem because it naturally considers a trade-off between exploration and exploitation phases. It has been successfully applied to problems in operations research, control and robotics [15].

To illustrate the problem of exploration versus exploitation let us consider a classical multi-armed bandit problem from machine learning. Each arm of the machine when pulled, produces a certain amount of reward, and pulling an arm costs a coin. The goal of the gambler is to identify the best strategy of trying different arms to maximize the total reward.

The crux of the multi-armed bandit problem is that the probability distributions of rewards for each arm have unknown distribution. If a gambler finds an arm which gives a certain (positive) reward, shall it stick to current arm or try other options at extra cost?

While there are many theoretical solutions to this problem, the objective of this work is to apply the most basic ones and evaluate their performance on real human mobility traces.

In the following section we describe the problem in terms of reinforcement learning framework and compare various strategies.

3.3 Strategies

An agent interacts with an environment through perception and action. At each step an agent perceives a state $s_t \in S$ of an environment and responds with an action $a \in A(S_t)$. The action results in a certain reward $R : S \times A \rightarrow R$. The goal of an agent is to maximize a long-term reward based on the interactions with an environment. Specifically, the goal is to learn a policy mapping from states to actions that maximizes the long-term agent reward.

A day is modeled as N timeslots. A node has the following set of actions: 1. sleep 2. wake-up 3. set duty cycle (1-100%). A high duty cycle might or might not increase the chances of detecting more contacts. For example, it might be sufficient for a node to work from 11am to 12am, but with a 10% duty cycle (as opposed to 100%). A reward r is the number of successful encounters. The goal of an agent is to detect the maximum number of successful encounters within a given energy budget.

After taking each action a node observes the outcome and updates the payoff for a given timeslot. The payoff estimation is done using an exponential weighted moving average (EWMA) filter [2]. The filter estimates the current payoff value by taking into account the past measurements. $r_n = r_{measured} * \alpha + r_{n-1} * (1-\alpha)$. Where r_n and r_{n-1} are the estimated and previous payoff values, $r_{measured}$ is the measured payoff over the last time slot. The weight assigned to past measurements $(1-\alpha)$ depends on how responsive the node has to be to changing environment.

We compare four strategies: random, greedy and Boltzmann exploration and balanced strategies. The idea is to use the most basic strategies, and see their behaviour with real mobility traces.

3.3.1 *Random*

In a basic strategy the node spreads its energy budget evenly throughout a day, i.e., it sends service advertisement with a certain fixed interval. The strategy is equivalent to normal asynchronous wake-up scheduling with fixed duty cycle, so would not require additional implementation. This is the most basic strategy and will be used as a baseline. The obvious problem with random strategy is that a node will waste resources when there are no nodes around.

3.3.2 *ε-Greedy*

In a pure greedy strategy, a node always selects the best *known* timeslot. During this timeslot a node spends all its daily budget. The problem with a greedy strategy is that it can converge too early on suboptimal solution [15] without finding for an optimal solution.

In ε-Greedy, a node selects the greedy action, but also explores other choices with a probability ε. During exploration a node selects a random action. This guarantees that the node always invests a certain amount of resources in learning about the environment.

ε-Greedy is intuitively simple to understand and to implement and will be compared with other strategies.

3.3.3 *Boltzmann exploration*

In Boltzmann exploration the selection of a timeslot a depends on its expected reward *ER(a)*. The advantage is that better timeslots have higher chances of being explored. The reward has the following distribution:

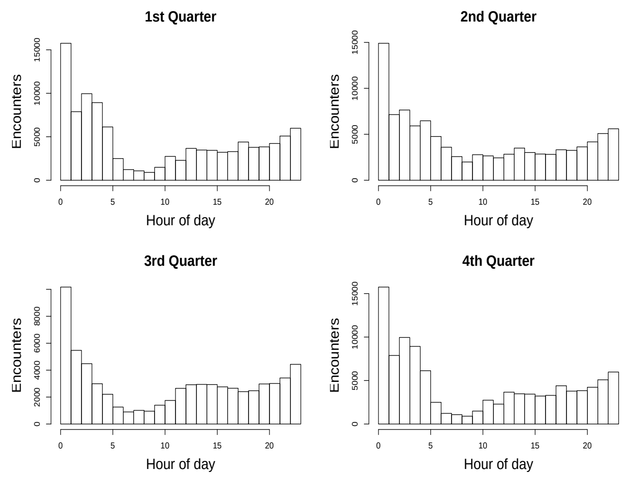

Figure 4: Seasonal arrival patterns in the residential building. The arrival patterns are relatively similar throughout a year.

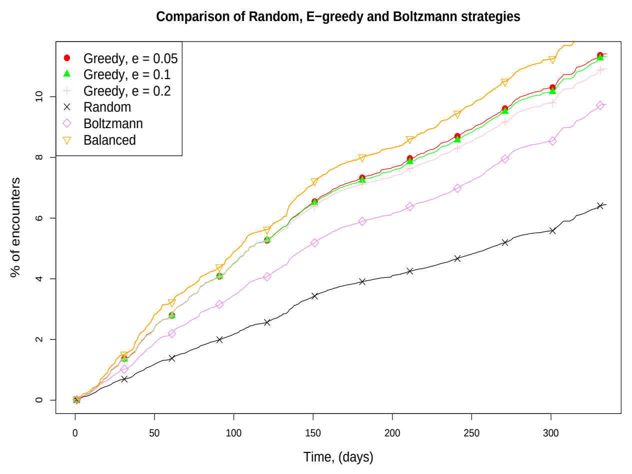

Figure 5: Comparison of various wake-up strategies.

$$P(a) = \frac{e^{ER(a)/T}}{\sum_{a\prime \in A} e^{ER(a\prime)/T}} \quad (1)$$

The parameter T is referred to as temperature of the system (in a metaphorical way). The temperature parameter sets a trade-off between exploitation and exploration. The low temperature values will encourage greedy behaviour whereas high temperature will force more exploration. Also known as softmax, the probability of choosing will depend on the past history.

3.3.4 Balanced

In a balanced strategy we propose to dynamically adjust node's duty cycle proportionally to an expected reward. Therefore the node *does not commit* to any timeslot, but spreads its energy proportionally to expected reward. The node sets its duty cycle according the following rule:

$$D(a) = \frac{ER(a)}{\sum_{a\prime \in A} ER(a)} \quad (2)$$

For example, if there are several peak hours during a day, the budget will be spread evenly among all peaks. During quiet times the node continues to sample the environment but with lower intensity.

4. EVALUATION

We evaluated our approach through simulations with real human connectivity traces and implemented a prototype for Tmote Sky motes. Obviously it is well known that human traces have clearly identifiable patterns, however also animals do have patterns (as already observed by various biological studies [8] and even by the collected Zebranet traces [14] . Human traces are readily available in abundance and are sufficient for the purposes of the evaluation of our work, which could however be applied much more generally.

4.1 Simulation setup

In order to evaluate our approach we used mobility traces collected and used them to simulate movement of entities around our sinks and evaluated how well the sinks were detecting the movement patterns, by optimizing on their awake times. We used the 802.11 mobility traces from Dartmouth campus [16]. We selected the busiest access point and analyzed the traces of a one year period. The system log was generated by Cisco wireless access points and contained association/disassociation entries. Each association was considered as an encounter with a fixed duration.

We compared four strategies: random, ε-greedy, Boltzmann exploration and a balanced strategies. ε-greedy strategy was tried with different exploration rates. The sinks were given a daily budget of 600s, and needed to distribute their energy according to the load.

The simulation was done by using the R [20] statistical package. The duration of a timeslot was set to one hour. Shorter time slots could allow more fine-grained decisions but would have very high variance. Longer time slots will have low variance, but very coarse-grained decisions. In the extreme case a system would have two time slots (day/night modes). The hourly distributions are stored in a vector $r[1..N_{slots}]$. Each value is updated using EWMA filter as: $r_n = r_{measured} * \alpha + r_{n-1} * (1-\alpha)$, with $\alpha = 0.75$. The values are updated only for the active timeslot. Inactive timeslots are not updated or decayed.

4.2 Results

In the first experiment we compared all four strategies for an access point (AP) located in residential building (Figure 5). The balanced strategy shows the best performance throughout an entire experiment. The performance of ε-greedy turned out to be very sensitive to exploration rates. Low exploration rates showed better performance; the node learned to select the good time slots and invested some resources into exploring other options. As the node spent more resources into exploration the performance was decreasing approaching that of a random strategy, as expected.

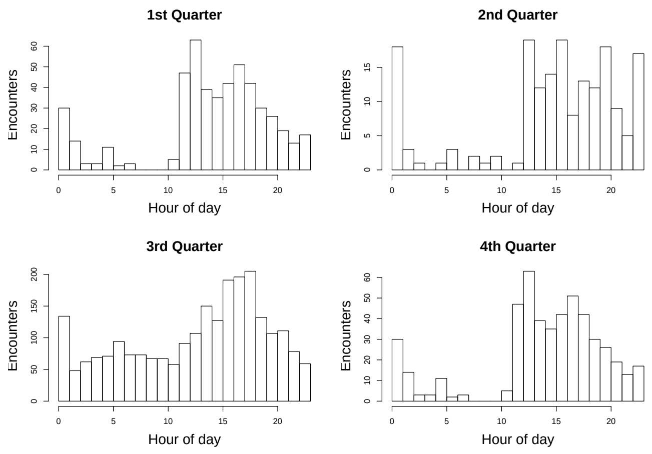

Figure 6: Seasonal arrival patterns from an AP in an academic building. The arrival pattern changes with the season.

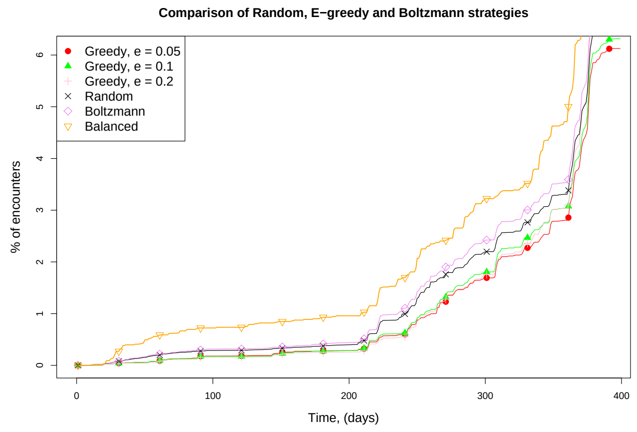

Figure 7: Comparison of various wake-up strategies.

To evaluate how well all strategies adapt to changing arrival patterns we experimented with the busiest AP located in an Academic building. It has different and more dynamic arrival patterns throughout the year depending a lot on the season (Figure 6). As we see from Figure 7, the balanced strategy again showed the best performance. It adapted quite well to changing traffic patterns. The ε-greedy strategy performed worse. We see that high exploration rates the system adapted well to changes and showed better results, while at low values, the system was very slow to adapt to seasonal changes and sometime performed worse than random. The rapid increase in the performance of all four in the second half of the experiment can be explained by much higher load during that time, leading all four strategies to detect more encounters.

To summarize, the balanced strategy showed the best overall result for both scenarios. It allocated energy dynamically depending on the expected number of arrivals and adapted well to changing arrival patterns. The ε-greedy performance was very sensitive to exploration rates. The optimal value of exploration rate depends on how dynamic system is, which can be difficult to know in advance. Random and Boltzman performed robustly in all scenarios.

4.3 Implementation

A prototype of the middleware has been implemented as a component for TinyOS 2.0 [17], which we have tested on a TMote Sky [5]. The implementation is very compact and takes 2958 bytes of ROM and 114 bytes of RAM.

The component provides an application programming interface (API) described in Table 2. It provides an interface allowing applications to set an energy budget through `setBudget()`. Applications use `getDutyCycle()` to inquire about recommended duty cycle depending on a time of day using a balanced strategy. Whenever a new encounter is discovered, an application registers it through the `encounter()` function. The component keeps an estimate of successful encounters in a current timeslot and updates its model every hour.

5. RELATED WORK

Existing service discovery protocols such as Universal Plug and Play (UPnP) [4], Service Location Protocol (SLP) [12] and Salutation [3] focused on autoconfiguration issues allowing devices to automatically join the network and learn about its capabilities. However such approaches were designed for resource rich networks and are not suitable for resource constrained systems. Our proposed mechanism complements existing work in this area and allows to discover mobile devices in the resource constrained networks. The specific feature of our middleware is that it helps sensors to optimize node's discovery decisions under uncertainty.

Energy efficient service discovery can be done using power efficient wake-up scheduling protocols, such as [21][19][22][6][11]. These protocols allow for very energy efficient communication in static wireless sensor networks. They do not however deal with the uncertainty of node contact times. Advertising services when there are no nodes around would be a waste of energy. Our approach is supposed to work on top of existing wake-up scheduling protocols, allowing them to make better decisions as to when to and how frequently perform service discovery.

Existing middleware solutions for wireless sensor networks usually have plug-in system [13] for service discovery protocols or use their own mechanisms [7]. TeenyLime [7] is a data-sharing middleware designed for mobile sensors. It uses an operational setting where sensors are fixed and relatively powerful mobile sinks are used to collect data from sensors. The base station uses long continuous packet trans-

Table 2: Service Discovery API

Function	Description
`void reset`	reset model
`void setBudget(uint16_t)`	sets daily budget
`uint16_t getBudget()`	retrieves a daily budget
`void encounter()`	registers a successful encounter
`uint16_t getDutyCycle()`	get recommended duty cycle
based on the model	

missions to wake-up low duty cycled sensors. Our setting is similar in that the base station take the burden of the node discovery. The difference is that in our setting, the sensors are mobile and their arrival time is uncertain. Therefore the basestation will always need to check the environment for potential neighbours, which is energy expensive.

Existing technologies such as 802.11 and Bluetooth protocols provide special power save modes for the equipment. 802.11 has a power-save mode where radio is active only a fraction of time. In the infrastructure mode an access point (AP) monitors the state of each client. The clients periodically wake up and checks if an access point has any buffered packets. This mechanism requires the clients to be synchronized with the access point and a static non-changing topology. BlueTooth [1] has a special power-save mode, and also relies on time synchronization between master and slave nodes.

6. CONCLUSIONS AND FUTURE WORK

We have presented a middleware for power efficient node discovery service in partially mobile sensor networks. Its main features are separation of a discovery function from mobile nodes to sinks and a novel adaptive node discovery algorithm. We showed how simple optimisations techniques from machine learning could significantly improve the energy efficiency of node discoveries. A prototype of the middleware has been implemented for Tmote Sky motes.

The future work is to extend the algorithm to support fully mobile wireless sensor networks. In particular, we would like the nodes to dynamically assign their daily energy budgets according to past encounters, remaining energy and policy. Another area of interest is evaluation of an impact of the adaptive duty cycling on the performance of message dissemination algorithms in multi-hop networks. We also intend to perform further evaluation on other type of traces, possibly in less predictable environments or with non-human entities.

7. ACKNOWLEDGEMENTS

We would like to acknowledge the support of EPSRC through projects CREAM, WILDSENSING and DHPA award.

8. REFERENCES

[1] Bluetooth specification.
[2] Moving average. http://mathworld.wolfram.com/movingaverage.html.
[3] Salutation consortium: Salutation architecture specification.
[4] Universal plug and play. http://www. upnp.org/.
[5] Tmote sky datasheet. moteiv corporation, 2006.
[6] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, New York, NY, USA, 2006. ACM Press.
[7] C. Curino, M. Giani, M. Giorgetta, and A. Giusti. Tiny lime: Bridging mobile and sensor networks through middleware. In *Tiny Lime: Bridging Mobile and Sensor Networks through Middleware.*, March 8-12 2005.
[8] Ebert, K. E. D., T. Rechlin, and W. Kaschka. Biological rhythms and behavior, advances in biological psychiatry. issn 0378-7354.
[9] A. El-Hoiydi and J. D. Decotignie. Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, 1:244–251 Vol.1, 2004.
[10] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.
[11] A. Giusti, A. L. Murphy, and G. P. Picco. Decentralized scattering of wake-up times in wireless sensor networks. In K. Langendoen and T. Voigt, editors, *EWSN*, volume 4373 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 2007.
[12] E. Guttman. Service location protocol: Automatic discovery of IP network services. *IEEE Internet Computing*, 3(4):71–80, 1999.
[13] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications, 2004.
[14] P. Juang, H. Oki, and Y. Wang. Energy efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. ASPLOS, Oct 2002.
[15] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning: A survey. *Jurnal of Artificial Intelligence Research*, 4:237–285, 1996.
[16] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. *Wireless Networks*, 11(1-2):115–133, January 2005.
[17] P. A. Levis. Tinyos: An open operating system for wireless sensor networks (invited seminar). In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)*, page 63, Washington, DC, USA, 2006. IEEE Computer Society.
[18] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM Press.
[19] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM Press.
[20] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3-900051-07-0.
[21] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(3):493–506, 2004.
[22] W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle mac with scheduled channel polling. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 321–334, New York, NY, USA, 2006. ACM Press.