Evolution and Sustainability of a Wildlife Monitoring Sensor Network

Vladimir Dyo[◇], Stephen A. Ellwood[†], David W. Macdonald[†], Andrew Markham[‡]
Cecilia Mascolo[§], Bence Pásztor[§], Salvatore Scellato[§], Niki Trigoni[‡], Ricklef Wohlers[‡], Kharsim Yousef[§]

[◇]Dept. of Computer Science and Technology, University of Bedfordshire, UK [*][§]Computer Lab, University of Cambridge, UK
[†]Wildlife Conservation Research Unit, Dept. of Zoology and [‡]Computing Laboratory, University of Oxford, UK

Abstract

As sensor network technologies become more mature, they are increasingly being applied to a wide variety of applications, ranging from agricultural sensing to cattle, oceanic and volcanic monitoring. Significant efforts have been made in deploying and testing sensor networks resulting in unprecedented sensing capabilities. A key challenge has become how to make these emerging wireless sensor networks more sustainable and easier to maintain over increasingly prolonged deployments.

In this paper, we report the findings from a one year deployment of an automated wildlife monitoring system for analyzing the social co-location patterns of European badgers (*Meles meles*) residing in a dense woodland environment.

We describe the stages of its evolution cycle, from implementation, deployment and testing, to various iterations of software optimization, followed by hardware enhancements, which in turn triggered the need for further software optimization. We report preliminary descriptive analyses of a subset of the data collected, demonstrating the significant potential our system has to generate new insights into badger behavior. The main lessons learned were: the need to factor in the maintenance costs while designing the system; to look carefully at software and hardware interactions; the importance of a rapid initial prototype deployment (this was key to our success); and the need for continuous interaction with domain scientists which allows for unexpected optimizations.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and Embedded Systems

---

[*] This work was partially carried out when Vladimir Dyo was at Computer Lab, University of Cambridge, UK

General Terms

Algorithms, Design

Keywords

Wireless Sensor Networks, RFID Technology, In-Network Storage, Duty Cycling

1 Introduction

The deployment of sensor networks in a variety of real-world applications is gradually turning from scientific vision to reality. A multitude of systems have already been deployed, ranging from glacier monitoring [4] to real time environmental and wildlife tracking [30] [20]. Such systems have enabled the collection of spatio-temporal data at unprecedented granularities, and have revolutionized the way in which scientists perform field experiments. At the same time, with the outset of new sensor deployments, the need has come to maintain sensor networks over prolonged deployment periods. Low effort maintenance and self-reconfiguration have in fact been the idealistic selling points of wireless sensor networks. Network maintenance may involve a number of tasks, such as changing batteries, replacing faulty nodes and collecting data from special-purpose storage or gateway nodes. When the maintenance costs exceed user expectations and budget, there is a need to develop the system and make it sustainable. In this paper, we describe one such system and present our experience in building and developing a sustainable wireless sensor network. Our system consists of a distributed wireless sensor network designed to monitor wildlife and environmental conditions in a dense woodland environment, in Wytham Woods, Oxfordshire, UK. The system is made up of three components. The first consists of active RFID transmitters that are attached directly to European badgers (*Meles meles*) as wearable collars. They are monitored by a second component consisting of a collection of fixed detection nodes that are distributed throughout the woods at key locations close to known badger setts and latrines. The third component further complements the assembly by providing a bed of fixed sensor nodes that are deployed within badger foraging areas to monitor micro-climatic conditions and their effect on species migration and mobility patterns.

We first describe the initial 'exploratory' field-deployable prototype designed to understand the domain requirements

and the usage patterns. We then describe gradual alterations to initial design based on feedback from the zoologists. In particular, we evaluate each iteration in terms of maintenance cost and find that the initial commercial off-the-shelf based design resulted in ten-fold improvement in maintenance costs, while enabling zoologists to collect unprecedented amount of high resolution data on wildlife badger behavior.

In the first phase, we optimize the system at the software level. We propose a novel sampling approach for the power hungry animal detection nodes, based on reinforcement learning. The idea is to exploit the behavior patterns of observed animals in order to more efficiently control energy consumption. We also propose a novel storage management scheme that takes into account data urgency and sink mobility to allocate sensor data to carefully selected storage nodes. We observed that these proposed software optimizations had a noticeable effect on the maintenance costs, but the network still required long hours of hands-on human intervention.

In the second phase, we proceeded to enhance the hardware of the most power-hungry nodes to reduce their energy consumption. Here we provide details of the new platform, and how it drastically reduced the need for labor-intensive field trips to replace depleting batteries. This optimization led to a dramatic improvement in terms of maintenance costs. At the same time it triggered another round of software optimizations - we revisited sampling and in-network storage in the light of the new hardware capabilities. We validate the hypothesis that evolving hardware significantly impacted the performance of algorithms running on the nodes. This prompted us to introduce a more energy-efficient sampling algorithm for detecting badgers, which was not applicable in the old platform. It furthermore impacted the performance of our storage management scheme by altering the patterns of sink mobility. The running costs of the resulting system were reduced to such extent that it made it realistic for zoologists to envision network expansion. The data collected throughout our deployment have the potential to offer zoologists a deep insight into the social life of badgers and on the correlation of their activities with weather and micro-climatic variations.

The lessons learned in this paper highlight the impact of maintenance costs on system design and the evolution, as well as the interplay between hardware and software optimizations. They also point out the need to take into account domain knowledge and application requirements to enable successful long-term deployments. The remainder of this paper is organized as follows: Section 2 introduces the characteristics and requirements of the badger monitoring application. Section 3 presents the architecture, design and deployment of our initial monitoring system. Sections 4 and 5 present the two stages of network evolution. Section 6 illustrates the costs incurred by our various stages and existing monitoring techniques. Section 7 analyzes the data collected and presents our main observations of badger behavior. We discuss related work in Section 8 while Section 9 summarizes our findings and concludes the paper.

2 Wildlife Monitoring Application

In this section we describe the challenges and requirements of our badger monitoring application. Badgers are nocturnal mammals, spending their days in subterranean multi-entranced burrow systems (so called 'setts'), and foraging at night. In the UK, their habitat is typically mixed wood and farmland landscapes. Their active nocturnal period commences when they emerge above ground around dusk. These emergence times thus vary seasonally, therefore correlating with temperature and day length. During their active period, badgers visit specific places, such as 'latrines', which are thought to have an important role in their social behavior (see [22] for an introduction to badgers biology).

After foraging, they return to their setts, usually around dawn. Separate, spatially distinct, setts are arranged into social groups. It is thought that badgers move readily between setts within a social group: typically, they would return to their setts of origin each night. Zoologists' understanding of the social bonds between the individuals in a social group remains incomplete. Badgers are apparently territorial, but to what extent they actively or passively establish a home-range is poorly understood. Movements are difficult to observe on a fine temporal scale, but systematic trapping up to four times per year has indicated that movement between social groups, at the population level, appears to be minimal [17].

Zoologists would like to know more about the movements and social interactions of these animals: where and for how long they may meet is especially important. Since GPS receivers function poorly in densely wooded areas, such information is usually gathered by on-site, night time observation, VHF radio telemetry, and more recently by remote video surveillance. All these methods are labor intensive and expensive. E.g. VHF tracking requires at least two people to get accurate location information on the animal, and it is not often practical to track multiple animals simultaneously. Despite intensive study of these animals, answers to fundamental questions regarding socio-spatial dynamics, and foraging-patch use remain elusive. The degree of social-group interaction is only superficially known using current technologies. Given the role badgers are considered to play in the epidemiology of bovine tuberculosis, and the full economic implications of this disease, understanding the temporal distribution of potential disease-carrying contacts, at key resource focal points, such as burrows and food patches (especially where these are shared with domestic animals) [18], is critical. Badgers are also a protected species, vulnerable to persecution, and emblematic of various conservation organisations. These types of conservation issues are typical for many species, and these risk factors are equally hard to monitor.

Given these requirements we have devised an integrated system for badger monitoring that could further help zoologists understand the social and behavioral implications of badger movements. Fig. 1 shows the heterogeneous nodes and devices comprising our system. In this wildlife tracking installation we monitor badgers equipped with active RFID tags embedded within a small light-weight collar designed to have minimal impact on badger behavior. RFID receivers,
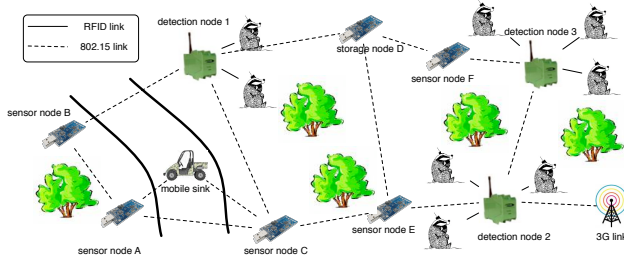
Figure 1. Heterogeneous network consisting of badgers (equipped with RFID collars), detection nodes (fixed RFID receivers), environmental sensor nodes, zoologists (mobile sinks) and a fixed gateway.

Figure 2. The badger detection node (left) and the active RFID tag, potted in epoxy and mounted on a collar (right).

referred to as *detection nodes*, were placed in key locations throughout the woods. In addition, we deployed a number of *sensor nodes* to monitor temperature and humidity in the same area. Sensor nodes and detection nodes were all connected through the same network. Our network also included a single solar powered gateway with cellular connectivity, which was located conveniently for 3G coverage and for its own maintenance. As it had cellular connectivity, it could relay data instantaneously to the end users.

Zoologists also contribute an element of the system as they perform regular trips to the study site to carry out routine observations and equipment maintenance. Thus, they can act as *mobile sinks* and assist in the task of data collection, relieving the network from part of its communication load.

The data generated by our network fall into three categories: 1) RFID readings that reflect badger observations and are captured by detection nodes, 2) environmental (humidity and temperature) data monitored at regular intervals by fixed sensor nodes; and 3) network health data indicating battery levels, memory usage and any sensor errors.

Zoologists and network engineers can assign priorities to different data types; a priority value reflects the tolerable delay between generating sensor data and delivering them to the user. For example, the detection of badgers dispersing from their natal burrows may be considered very important as it represents potential fission and fusion within badger society, while also generating the potential for the transmission of social, genetic, and disease 'information'. Zoologists require prompt notification of these dispersal events as soon as they happen, whereas they are able to wait days for summaries of badger activity data, and weeks for raw badger observations and environmental data.

3 Initial System Design

This section discusses the initial design of our animal monitoring system, whose focus was on strong modularity and portability.

3.1 Sensing

Environmental monitoring: To investigate the potential impact of microclimate on individual badger behavior, we equipped Tmote Sky nodes with two external SHT-71 digital temperature and humidity sensors. One of the sensors was buried 30 cm underground (where it only measured temperature), and the other was mounted at a 1 m height. Ten of these
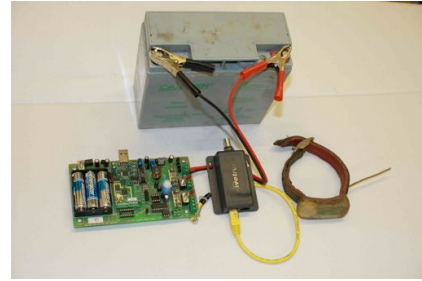
nodes were deployed in the woods and made a measurement every five minutes. Suitable sensor housing was developed by trial and error to protect the sensor and also to allow it to record accurate humidity measurements. Our early packaging resulted in the saturation of the humidity sensor due to local condensation within the enclosure. We found sealing the digital sensor within hot-melt glue and shaping heat-shrink to act as a shield to restrict wind chill resulted in the best solution. These devices were configured to either act as standalone data-loggers (which have very low average current consumption - approximately 30 μA) or as normal network nodes.

Badger Monitoring: The wildlife tracking presents unique challenges, requiring animal borne tags to be simultaneously small, very reliable, and inexpensive. This influenced a number of design decisions including the use of a commercial 433 MHz Active RFID tag [1] over the alternative of designing a custom miniature mote platform. The selected tags satisfied most design requirements including much lower cost, miniature size and long lifetime. The small size of the tags was crucial as it allows tracking of much smaller animals. Overall, the selection of commercial low cost tags also allowed the team to capitalize on the advantages of the tested component and focus on ground sensor network design, measurements, data collection and analysis. The tag measured 40x20x3mm in size (without a 56mm external whip antenna), and was powered by an on-board 3V CR2450 coin cell battery with an expected minimum lifespan of 2 years at 0.4s transmit interval. Each RFID tag was hermetically sealed ('potted') in waterproof epoxy resin to protect the tag from environmental and mechanical damage (e.g. chewing by an animal). The collars with potted tags (see Fig. 2) were attached to badgers during routine trapping sessions, approved by institutional ethical review [16] (UK Home Office Licence 30/2138; Natural England Licence 200001537). After full recovery from sedation badgers were released at their point of capture.

The presence of tagged animals was registered by 26 RFID detection nodes placed at setts and latrines, covering all main setts in the core study area (see Fig. 3). The detection range of a tagged animal was 0-30m, with the selected 433 MHz frequency providing longer communication range and lower obstacle fading through dense vegetation.

Each detection node consisted of an active RFID reader, a Tmote Sky mote and a custom designed mote extension
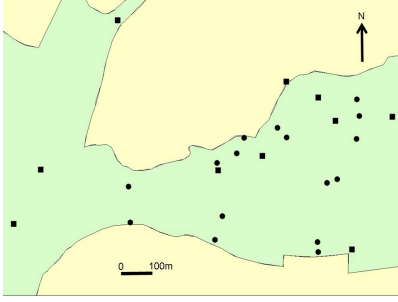
Figure 3. Map of the study area showing RFID detection nodes: square = setts; circle = latrines.

board. For each detected tag the reader provided the following information: tag ID, reader ID, serial counter number, received signal strength (RSSI) and a checksum. The serial counter number facilitated an estimation of the tag age and could be used for localization purposes, when the tag was simultaneously registered by several readers. The extension board allowed the interconnection of the mote, RFID reader and peripheral devices to an RS232-TTL converter, MOS-FET switches and the voltage regulators. The output voltage ranged from 6V to 12V and was configurable either through potentiometers and switches on board or from the mote via a standard I2C interface. The power management software on the mote duty cycled the peripheral devices including the reader, and monitored both mote and reader voltages to shut down the system should the voltage become low.

It should be noted that the tracking and ground communication currently have different requirements in terms of communication range and antenna configuration, so decoupling the two communication systems is desirable. In particular, ground communication requires extended communication range with preferably high bandwidth, whereas detection requires a biologically meaningful communication range, with a high degree of consistency, which requires consistent antenna orientation and receiver sensitivity of all detection nodes.

3.2 Data Collection

In our initial system design, we distinguish between two types of data - high-volume data, which consisted of raw badger observations, and low-volume data, which consisted of environmental readings, summaries of badger visits and network status reports.

Compression and local storage: As a result of the large data volumes generated by the network (typically in excess of 400 000 observations per week) , we implemented a simple delta based compression technique to allow more data to be stored in the 1Mbyte flash memory of the Tmote Sky. This approach, which is application-specific and computationally lightweight, achieves a 25% higher compression factor than standard compression methods, like gzip. This technique takes advantage of the large degree of similarity between successive RFID readings from the same RFID tag. In essence, we simply encode the difference between a base RFID observation and subsequent readings. Each raw reading, which consists of a timestamp, tag age and received

signal strength occupies 10 bytes in its uncompressed form. The difference between an observation and the base record can be stored using only 3 bytes of information. Using this simple scheme, raw data is typically compressed by an average factor of 2.7x. This compares favorably with the resource hungry gzip (LZ77) algorithm which only achieves a compression factor of 2.0x on the same dataset. Thus, by reducing the volumes of data that need to be buffered within the network, we were able to extend the memory lifetime of the reader node almost threefold. This data could be compressed further using dictionary type compression algorithms such as S-LWZ [24], but the gains would only be marginal and would require additional node resources.

Routing: Low-volume data (such as network status messages) are forwarded to the fixed 3G gateway node using a proactive shortest path routing algorithm. Every node maintains a routing table containing its distance to the gateway node. Initially the gateway advertises beacons with distance 0 to itself, and with increasing sequence (freshness) numbers. The distance from a node to the gateway is evaluated taking into account the link qualities along the route. Each node maintains a neighborhood table that shows statistics of outgoing traffic. The expected transmissions *etx* per message from the current node to a neighbor node N is computed as follows $etx(N) = attemptedtx(N)/successfultx(N)$. The *distance* to the gateway node is defined as the sum of expected transmissions on all links along the route. Note that if all the links along a route have an $etx = 1$, the distance is equal to the number of hops along the route.

Every node broadcasts its distance to the gateway every 30 minutes. Upon receiving an advertisement from a neighbor N, a node compares the advertised distance ($advDist$) to the distance in its local routing table ($rtDist$). If $advDist + etx(N) < rtDist$ then it sets $rtDist := advDist + etx(N)$ and sets neighbor N as its next hop. If the route quality deteriorates significantly, a node will simply select the next best available route.

Once routes have been identified, data are transmitted using the uIP (micro-IP) IPv6 networking stack [8], which in turn uses the X-MAC protocol [6], both being distributed with Contiki OS.

uIP: The choice of using Contiki's uIP networking stack was strongly influenced by the positive findings of Hui et al. [12]. The added flexibility of using the IPv6 standard will allow us to adapt the network to other tasks easily during further network deployment, for example accessing and maintaining individual nodes or allowing near real-time data streaming from specific nodes within the network. Although the overhead incurred in terms of code size is considerable for the T-Mote Sky platform (approximately 16KBytes of additional flash usage for our implementation), the added modularity and flexibility of the IPv6 network allowed us to easily maintain and extend the network with new functions. We made minor revisions to the original code in order to accommodate the low-power needs of our network.

Data are disseminated towards storage nodes on a local hop-by-hop basis, instead of an end-to-end basis. The usual end-to-end connection used in IPv6 networks is TCP/IP which requires an additional overhead for establishing a con-

nection and requires the end nodes to negotiate retransmissions. To avoid this overhead we use UDP connections to transmit data along each hop towards the storage node. Upon receiving a packet from a child node, the parent node returns a UDP ACK message to confirm reliable data transfer. Messages are stored in onboard flash and they are only marked for deletion once an ACK message is received from the parent indicating successful custody transfer. A background garbage collection routine periodically cleans up the flash memory, formatting pages where all messages have been successfully uploaded. In addition, we reduced the frequency of neighbor advertisement and solicitation messages such that they had a validity of 24 hours. This helped us to dramatically reduce network overhead.

MAC layer: We decided to use X-MAC [6] at the MAC layer, a preamble based protocol in which senders indicate their intent to send data by frequently transmitting short wake up messages. Nodes periodically wakeup, and if they hear a preamble which indicates a packet is addressed to them, respond with an acknowledgement. This terminates the wakeup phase and the packet is sent. Nodes are configured to wake up every 500 ms and listen for 5.8 ms. This results in effective basic duty cycle of 1.1%.

4 Evolution Stage 1: Improving Sensing and Data Collection

In this section we discuss how we started evolving our initial system design by introducing algorithmic improvements. The main weaknesses of our initial design were the high energy consumption of the badger detection nodes (RFID readers), and the heavy communication load around the fixed gateway. As shown in Sec. 6, about a visit a week was necessary to change batteries and keep the system running.

4.1 Adaptive Sensing

RFID readers are the major source of power consumption on detection nodes. Despite being powered by a 12V 18Ah battery, without a duty cycling they only last for one week. Increasing the lifetime of readers is therefore critical for large-scale long-lived deployments.

An obvious way to save energy is to duty cycle the RFID reader by periodically turning it on for a fixed duration of T_{on} seconds every $T_{interval}$ seconds. Nevertheless, setting optimal parameters is not straightforward: a high frequency sampling may be too wasteful, whereas low frequency sampling may lose important contacts. Tuning also requires knowledge of badger activity, which may not be known in advance.

We thus devised an adaptive duty cycling approach, that dynamically adapts the parameters T_{on} and $T_{interval}$ taking into account the badger activity. We formulate the problem in terms of reinforcement learning [14], and suggest a control strategy that adjusts node duty cycles based on animal arrival patterns [9]. The initial values of T_{on} and $T_{interval}$ are set to reflect the target duty cycle and the hardware capabilities of the detection nodes. For example, to achieve a target duty cycle of about 9%, T_{on} is set to 30s and the initial value of $T_{interval}$ is 330s. For efficiency reasons, T_{on} is chosen to be significantly longer than reader boot time T_{boot}, which was 10s.

The approach is composed of two main components: the *short-term* adaptation component and the *long-term* adaptation component. Short-term adaptation extends the awake time T_{on} of the reader by a fixed short period of T_{ext} seconds each time badger activity is detected (i.e., a tag is in range). The short-term adaptation exploits the temporal burstiness of badger arrivals, as detection of a beacon is usually a good predictor of activity. The drawback of the periodic sampling technique, even in the presence of short-term adaptation, is that it assumes uniform badger activity throughout the day. However, it is rare that animals or humans remain continuously active throughout a day but rather follow a 24-hour circadian rhythm, which may vary depending on the environmental conditions [2]. Badgers, for instance, are nocturnal animals who are inactive during the day, which means that sampling during the day may be wasteful.

The long-term adaptation component learns daily patterns of badger activity and adapts the interval $T_{interval}$ accordingly. We define a target daily budget B as the amount of seconds that a badger detection node should spend in active state per day. Each day is divided in N equal time slots. Then, each node computes the expected number of sightings $E(d,t)$ during a day d for timeslot t and assigns a budget $B(d,t)$, proportional to $E(d,t)$, to each timeslot:

$$B(d,t) = B \frac{E(d,t)}{\sum_{i=1}^{N} E(d,i)}. \tag{1}$$

This is the equivalent of 'bidding' more resources in what has been a productive timeslot in previous days. We constrain $B(d,t)$ in the range $[B_{min}, B_{max}]$ in order to still explore all timeslots, even if they have not recently experienced any sighting, and to constrain the maximum number of times the node wakes up within a given time slot. Since in a timeslot of length T the reader is to be active only for $B(d,t)$ seconds, we have that $B(d,t)/T = T_{on}/T_{interval}$ and the node can adjust the duty cycle in each timeslot by setting the interval $T_{interval} = T \frac{T_{on}}{B(d,t)}$ between successive wake ups. On the first day, the budget is spread uniformly throughout all N timeslots, since there is no information about sightings. Then, the expected number of sightings $E(d,t)$ in timeslot t of a particular day d is evaluated as follows:

$$E(d,t) = \alpha \times O(d-1,t) + (1-\alpha) \times E(d-1,t) \tag{2}$$

where $O(d-1,t)$ is the actual number of sightings that were observed in the same timeslot on the previous day and α is a weight in the range $[0,1]$ which controls how rapidly new information is incorporated into the filter. Small values of α will give more weight to past history, but will make the adaptation process slow and unable to capture sudden changes, whereas large values will make it very reactive to short term changes and less able to capture long term patterns.

Simulation-based evaluation: The evaluation of the adaptive duty cycling technique has been performed both through simulation and real deployment. Throughout our evaluation, we used $N = 24$ 1-hour timeslots, that is $T = 3600s$. Within each timeslot, the detection node turns the reader on and off to achieve the target duty cycle using Eq. 1. The on-time T_{on} was selected to be 30s, the ini-
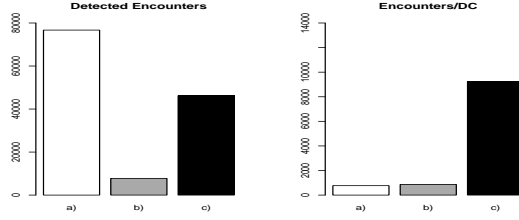
Figure 4. Simulation results. Comparison of detected encounters and encounters per effective duty cycle for a) always-on b) fixed c) adaptive algorithms
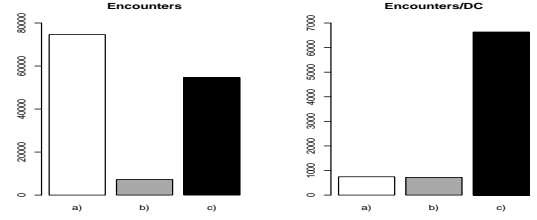

Figure 5. Deployment results. Comparison of detected sightings, effective duty cycle and the sightings per effective duty cycle for a) always-on b) fixed c) adaptive nodes.

tial interval $T_{interval} = 330s$ (which corresponded to a budget $B = 7854s$ and a duty cycle of about 9%), and the extension time $T_{ext} = 300s$. The $[B_{min}, B_{max}]$ range was set to $[B/120, B/24] \approx [65, 327]$. We used a fixed duty cycling algorithm, where a node wakes up and goes to sleep at fixed intervals of time, as a baseline. The algorithms have been implemented in Tossim 2.0.2 simulator and evaluated by replaying the real data recorded by always-on node. We run 10 simulation runs for each algorithm with random node offsets.

Fig. 4 shows the performance of always-on, fixed duty cycling and adaptive algorithms respectively. The always-on node detects all 76707 encounters at 100% duty cycle. The fixed duty cycling node detected 7773 encounters at 9% duty cycle. The adaptive node detected 46214 (60%) encounters at 5% duty cycle, resulting in much higher encounters per duty cycle than always-on and fixed nodes.

Deployment-based evaluation: In order to evaluate our duty cycling technique in a real deployment, we placed two detection nodes with the same hardware and antenna orientation next to each other. One of the nodes was always on, whereas the other executed our adaptive duty cycling technique. In addition, we processed data from the always-on node to simulate a fixed schedule. The adaptive node was configured to work at 9% duty cycle.

The evaluation was based on 833 hours of summer (July) deployment data from both nodes. The data were periodically retrieved from both nodes by a zoologist. The results are summarised in Fig. 5. The fixed duty cycling node captured 7201 sightings while using 10% of the power of the always-on node. The adaptive duty cycled node detected 54568 (73%) of all sightings, while consuming approximately 8.2% of the energy.

4.2 Delay-tolerant data collection

The initial design of the data collection algorithm was based on the principle that raw RFID data which are high-volume and low-priority data, are stored locally at sensor nodes. The remaining data had higher priority and were forwarded to the 3G gateway using a tree-based routing algorithm. This initial approach is similar to related work on prioritizing data traffic and taking into account routing costs to determine whether to discard data, store it locally, or forward it to the gateway [29].

Here we add a further step and propose a delay-tolerant data collection approach, which leverages the movement of

zoologists and other environmental scientists to efficiently collect sensor data. Not only do we prioritize data based on their urgency, but we also prioritize nodes based on the frequency in which mobile sinks visit them. In this way, we forward data to carefully selected storage nodes, purely based on data and node priorities.

Data priorities: When data were generated, they were assigned a *data priority* class that represents the latency allowed until they had to be delivered to the end-user. Our network generated observations of tagged badgers captured by the detection nodes, and environmental sensor data (temperature and humidity). Nodes also created heartbeat messages that reflected their current operational status. This included information such as remaining battery level, memory usage and network statistics. The motivation behind our delay tolerant networking approach is the fact that the majority of the generated data do not have strict latency constraints. It was imperative however, that all data are eventually collected. In order to maximize the battery lifetime of nodes in the network, we use a distributed storage and delivery method, where messages are directed to different destinations based on their tolerable delay. In our system, we have three offered priority classes, but this can be extended to an arbitrary number. The three priority classes are as follows:

Priority class 1 represents data with urgent latency requirements (maximum of a few hours delay). This data are forwarded to the 3G-router node for direct access by the researchers. Data of this class could either represent an unusual event or a network status report to ensure the network can function correctly throughout the deployment.

Priority class 2 represents data with medium latency requirements (maximum of a few days delay). This data are forwarded to frequently visited storage nodes for opportunistic collection. Data of this class could be summaries of badger visits.

Priority class 3 represents data with no latency constraints (delays of weeks are acceptable). All that is required is that it is eventually collected. Data of this class, such as raw sensor data, will remain in memory until collected through a direct download.

Priorities can be assigned not only to raw sensor data, but also to composite events or aggregated data. For example, raw badger information may have priority 3, but when unusually high activity is observed around a certain setts this composite event can be assigned priority 1, and will be for-
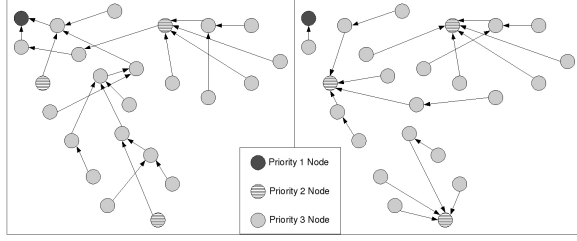
Figure 6. Example routing trees as found in our deployment: (left) Routing tree for priority 1 data, (right) Routing trees for priority 2 data.

warded to the fixed gateway for immediate delivery. Data priorities can either be fixed or dynamic, for example, they could vary depending on the zoologists' needs and the data collected from the sensor network.

Node priorities: Our priority based in-network storage management approach is very simple and effective. Initially, each node is assigned a priority class P_N based on the frequency it is expected to be visited by mobile sinks for data collection. Some nodes (such as those close to roads and paths) are regularly in contact with a mobile sink and thus contribute a small delay. Other nodes that are placed in rarely visited remote locations will be subject to a large delay.

The more frequently visited a node is, the lower the expected data delivery time, and the lower the assigned node priority class. In our system, the 3G gateway is assigned a priority class 1 as it can offer the lowest data delivery latency. Nodes that are visited at least every three days by mobile sinks act as temporary data storage nodes of priority class 2. The remaining nodes in the network have priority class 3.

In our storage management scheme, a data item of priority P_D is stored at the closest node with priority P_N, where $P_N \leq P_D$. Messages with the data priority class of 1 are directed towards the 3G enabled gateway, which allows users to access them with little delay. Data of priority class 2 is stored at the closest node that has priority 1 or 2. Data of priority class 3 is stored locally at the node where it is generated. Note that node priorities can change dynamically in response to changes in sink mobility. If a node becomes visited less often, some of the messages that it used to store may need to migrate to another node depending on their priorities.

By asking domain experts to classify data into priority groups, we can map data to suitable storage nodes, and in this way we can ensure that they are delivered on time and with the lowest communication cost. As a data item remains stored at a node, it gradually ages, and its remaining tolerable delay decreases. As a result, it can dynamically change priority and be forwarded to another suitable storage node

Priority- and mobility-aware routing: Once data are assigned a priority and are compressed, they are forwarded to the appropriate destination node, namely 3G gateway nodes of priority 1 or storage nodes of priority 2. Every node maintains a routing table containing the following information for each of the available priority classes:

priority	next hop	seq. no.	dest. node	distance
1	N_A	30	N_E	3
2	N_B	34	N_F	1

The *next hop* simply represents the neighbor to which the data of a certain priority will be forwarded. The *sequence number* (seq. no.) and *destination node* (dest. node) fields are used to deal with loops occurring in the network. The sequence number is issued by the destination node and represents the freshness of routing information concerning that node, as in DSDV [23].

We evaluated the *distance* to a destination node, taking into account the link qualities along the route, in exactly the same way as we evaluated distance to the gateway in Sec. 3.2. Every node periodically broadcasts its routing table information for each priority class. In our network, we set this broadcast period to 30 minutes. Note that a single advertisement contains routing information for all priority classes. The size of advertisements does not increase with the number of destination nodes, but only in proportion to the number of priority classes. Therefore, the routing overhead of building multiple trees, instead of one, was negligible. Fig. 6 shows the routing trees that were formed in our real deployment for priority 1 and 2 data.

Evaluation: In this section we present results from a 20 day network deployment period with a total of 24 RFID readers. For half of the time, data was collected using the previously described distributed storage approach, and for the other half using a centralised storage approach, as in the initial design. The centralised approach simply forwarded all data to the 3G node; the distributed approach used three additional priority-2 storage nodes at which data was temporarily stored for opportunistic pickup.

In order to have comparable results we utilised a fixed data generation rate for the network evaluation period. Priority 1 data consists of network status messages generated at each node every 30 minutes, which had to be delivered to the end user within two hours. Priority 2 data consists of badger activity summaries generated at each node every 15 minutes with a delivery latency of three days. In the centralised approach this data is forwarded to the fixed 3G gateway, whereas in the distributed approach, it is delivered to the nearest storage node that satisfies latency constraints [1].

In both centralised and distributed approaches, a very high delivery ratio was achieved (99.9% of the data was correctly transferred to the appropriate storage or 3G nodes). Furthermore, this was achieved with an average latency of 14.1 seconds per hop – thus data can be sent over five hops in under 75 seconds on average.

The network status messages, which contain the radio on-time at each node, allowed us to derive the average radio duty cycle of each node over the test period. Fig. 7 shows the distribution of radio duty cycles across the different nodes in the network, with the two storage management schemes. The centralised approach exhibits 46% higher duty cycle than the proposed distributed approach in the average case, and 57% in the worst case at routing hotspots. This shows that

---

[1]In our regular network operation, nodes also generate raw badger readings of priority 3, which are stored locally for both approaches, and thus do not incur any network overhead.
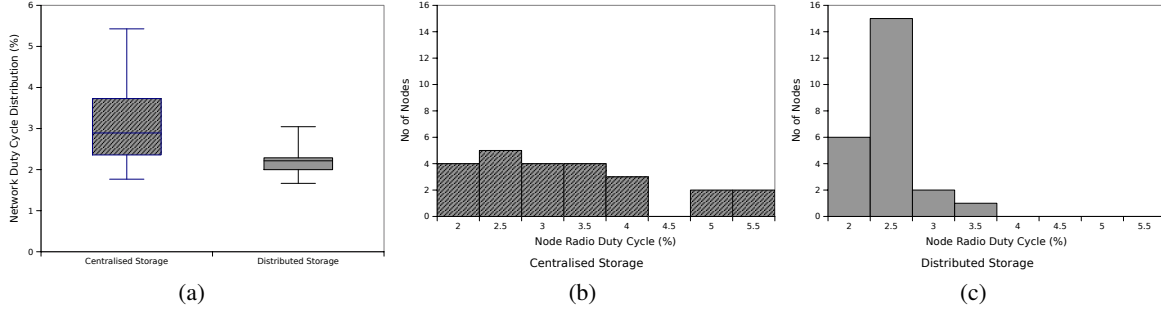
Figure 7. Results from the network test: (a) Box plot showing the distribution of radio duty cycles for the centralised (left) and distributed storage (right) approaches. (b) Distribution of node duty cycles for the centralised storage approach. (c) Distribution of nodes duty cycles for the distributed storage approach.

Table 1. Comparison between the two RFID reader versions

	Version 1 Reader	Version 2 Reader
Node	Tmote Sky	Zigbit Amp
Processor	MSP430	AVR atmega1281V
Node RAM	10 kbyte	8 kbyte
Node Flash	48 kbyte	128 kbyte
External Flash	1 Mbyte	Up to 2 Gb SD
RFID reader power	900 mW	96 mW
Reader turn on time	10 s	0.1 s
Radio range	50 m	1 km
Cost per unit	$590	$320
Mote battery	3 AA	none
Reader battery	18 Ah SLA	18 Ah SLA

by carefully forwarding data of different priorities to suitable storage nodes, we not only reduce the average energy consumption, but also balance the load more evenly in the network. Our benefits would be much more pronounced if we had forwarded priority-3 data to the gateway in the centralised approach.

5 Evolution Stage 2: Hardware Improvements

Although the algorithms proposed in Sec. 4 improved the usability of our initial design, our approach was limited by hardware - i.e. the RFID detection node. Experience dictates that field deployment and data gathering are imperative to a system's successful deployment. The detection node was built using off-the-shelf components enabling quick deployment, however these components turned out too general for our specific needs.

5.1 Design of the new node

We incorporated feedback from the users of the system (i.e. the zoologists) in order to make the system more useful. A summary of the major design changes made is shown in Table 1, and a photograph of the new node can be seen in Fig. 8

Although the ubiquitous Tmote Sky had enabled us to deploy a prototype system rapidly, its limitations in terms of radio range and usable memory were major constraints. We did not want to design a new custom node from scratch however, rather we wanted to incorporate a more modern and flexible module into the design. The salient criteria were that it should be low cost, power efficient and preferably

hand solderable. The cost and power requirements ruled out an advanced node such as the Imote2. Instead, we investigated small, wireless enabled modules that could act as the heart of a generic sensing platform. There were two modules that were a good fit to the application requirements: the Jennic JN5148 and the Meshnetics (now Atmel) Zigbit AMP (ATZB-A24-UFLR). Both of these modules were low power, inexpensive (less than $35 in single quantity) and had an external power amplifier which increased transmission power by +20dBm. They also came in small form-factor packages with numerous peripheral pins that could be used to interface with additional components. Although the Jennic module had a number of advantages, such as a low power 32 bit processor as opposed to the 8 bit Atmega1281V in the Zigbit AMP, we used the latter as it had better community support, especially in Contiki. This allowed us to port our existing code rapidly from the Tmote SKY platform to the AVR platform, with minor modifications to the existing RF230 radio driver. The radio range of the new modules was improved to be in excess of 1 km in woodland at maximum power, a great improvement that increases the span of the network considerably (note that this is the transmission range of the radio, and not the detection range of the RFID reader, which is unchanged). The drawback of transmitting at the highest power level is that this increases the current consumption from 17 mA to 50 mA.

As the Zigbit AMP is essentially a microcontroller with an embedded radio, we needed to add additional components in order to satisfy application requirements. Firstly, we added external memory to the board in order to remove the constraints present in the initial system. The board is equipped with a 4Mbyte serial dataflash chip and also a removable mini-SD memory card. At present, this allows the addition of up to 2 Gbytes of SD based flash, but larger capacities could be supported with modifications to the SD driver software, allowing high capacity cards to be used. We also added an RTC with battery backup to allow nodes to maintain their time when batteries were changed. Currently, nodes are unsynchronized – this is an issue that will be addressed in subsequent firmware iterations. One problem with the Tmote Sky is that the onboard sensors are not removable. This is not a problem in an indoors laboratory setting, but in a real deployment, sensors must be placed externally to the protec-

Figure 8. The second version of the node.

tive housing. Thus, we incorporated light and temperature sensors, which could be detached from the main board.

A major change in this version was the switch from the RS-485 version of the RFID reader to an OEM board. The RS-485 version was a suitable choice for the initial deployment, as it allowed us great flexibility in daisy chaining multiple readers together and had a simple serial interface. However, the power consumption and slow turn-on time were issues. These high power requirements necessitated the use of a separate reader and mote batteries, so that the mote would remain powered even if the reader exhausted its supply. Switching to the OEM version of the RFID reader negated these problems. It has a simple synchronous serial TTL interface and a pin that could be used to trigger an interrupt on the microcontroller when a tag was read. This allowed us to power down the microcontroller while the reader was active, whereas in the previous version, we had to maintain the clock for the UART. Furthermore, in the Tmote Sky, the radio and the UART were multiplexed, which led to a lot of problems with hardware locking to prevent concurrent access to the peripherals. In the new version, the RFID reader has its own dedicated pins. The turn-on time for the OEM reader is under 100ms, and it uses 96 mW when active.

Lastly, we used a simpler power distribution system, with 3V as a common rail. A small charge pump was used to generate the 5V required for the OEM RFID board. The nodes can be powered either from a 3V battery or from a 12V battery using a switching regulator. We also included a small prototyping area on the board, as our prior experience had shown us that there were often instances where we would want to connect an additional device (such as a moisture sensor) to a node.

In summary, the new version of the detection node has dropped the power consumption by nearly an order of magnitude. The storage space has been increased to such an extent that it allows for 40 years of storage at the current generation rates, as opposed to one week. This will allow us to gather more information and sample environmental sensors at a much higher resolution. The communication range has also been increased greatly, which allows the network to cover a much larger area with fewer devices. However it must be stated that it was our experience garnered from the prototype deployment that allowed us to design a well optimized successor.

5.2 Duty Cycling Revisited

Given that the RFID reader on the new node can be powered up in 0.1 s, as opposed to the 10s for the prior version,
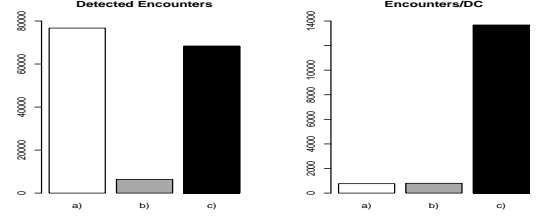

Figure 9. Simulation results. Comparison of detected encounters and encounters per effective duty cycle for a) always-on b) fixed c) adaptive algorithms.
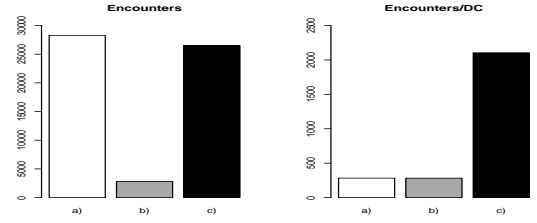

Figure 10. Deployment results with new hardware. Comparison of detected encounters and encounters per effective duty cycle for a) always-on b) fixed c) adaptive algorithms.

the parameters for the learning algorithm presented in Sec. 4.1 could be modified. The original $T_{interval}$ was set to 330s, with a duty cycle of 9%. Although this saved a large amount of power, allowing the node to operate for longer, it had the drawback of not being able to react to the presence of animals outside of the normal predicted times, as the off time could be quite long (up to an hour). In order to address this, we modified T_{on} to be 1s, with $T_{interval}$ to be 11s. This still resulted in a 9% duty cycle, but the short term adaptivity could react to the presence of unusual events, for example a badger emerging during the day. The longest time for which the reader was off was reduced to less than a minute, which increased the chances of detecting animals, while still accounting for their nocturnal behavior.

Fig 9 shows the simulation results for the same set of data as in Section 4.1, with new parameters. The shorter wake up interval resulted in both higher encounters and higher efficiency. The adaptive algorithm detected 89% of all encounters while working at 5% duty cycle. The deployment results conducted with the same parameters are shown in Fig 10.

5.3 Data Collection Revisited

The hardware improvements introduced in Stage 2 had a dual effect on the data collection process.

Change in sink mobility patterns: Recall that mobile sinks are domain scientists that roam through the woods and opportunistically collect data from storage nodes. Some of them are zoologists visiting the network for maintenance purposes, whereas others are from other disciplines visiting the woods for their own purposes unrelated to our sensor network. The visits of the former were reduced because hardware optimizations made the change of batteries less frequent. With fewer mobile sink visits, one would expect an

increase in the data propagated over multiple hops through the fixed network, and thus an increase in the average and worst-case communication cost.

Change in communication range: the effect of reduced sink mobility was, however, offset by the significant increase in the communication range of fixed nodes. Recall that the hardware optimizations introduced in the second stage dramatically increased the communication range of sensor and badger detection nodes from 50m to 1km. As a result, all nodes now have one-hop connectivity to the fixed 3G gateway, and no longer need to make use of mobile sinks.

Hence, in the second evolution stage, the hierarchy of nodes (based on priorities) collapsed and the use of mobile sinks to collect data efficiently and relieve the fixed network, became less relevant. This shows that the benefits of software-level optimizations, such as the priority-based delay-tolerant data collection, are tightly dependent on the hardware used. The priority-based delay-tolerant scheme proposed in Section 4.2 yielded significant benefits in the first version of the system, but proved of little use to the second one. However, we expect it to become relevant again in the near future, when we proceed to the third evolution stage of the system. Our short-term plan is to scale-up the network to cover a larger area. The extended network of badger and environmental sensor nodes will again become a multi-hop network, and the data collection scheme will be reinstated.

6 Network Maintenance Costs

In this section, we will describe the evolution of our system in terms of the costs involved. As a baseline, we will also show the approximate cost of conventional VHF tracking [15]. This involves tagging animals with VHF tags that emit periodic radio signals. VHF tags are analogue devices achieving individual identification by frequency separation, and limiting the number of IDs available. On the other hand, our active RFIDs are digitally encoded allowing more IDs in a given band without the need for a receiver to scan multiple channels. The VHF tags can be picked up by receivers carried by field-workers at a range of tens to thousands of meters depending on environmental conditions. Using triangulation (requiring at least two people on the ground), the approximate location of the animal can be found. Our RFIDs transmit at much lower power than VHF tags increasing battery life, while limiting range (30m), so giving a more precise location estimate for tagged animals. VHF tracking has been a popular method since the late 1960s because it was, and still is in many circumstances, the only way of tracking wild animals.

Note, although we are comparing the costs between VHF and our system, the data collected by the two methods are rather different - although they collect the same information (i.e. the location of a specific animal), our system logs an animal about twice a second when it is nearby a detection node, while this is not the case for VHF tracking. The more animals tracked by VHF, the more human trackers are required on the ground up to the point where the number of trackers disturbs the animals being tracked. Our system instead offers continuous automatic detection (presence/absence) of the animals at specific locations with minimal interference.

From previously tracking studies of badgers, using VHF, we know that at least one person is needed to work for about 10 hours a night to track one animal. If we assume we have enough people to work for 28 days, this would result in 280 hours per person, costing 2,030 USD using a 7.25 USD/h wage. It is easy to see how this is not feasible in the long run, especially, because one person can only track one animal at a time. It is also not possible to provide continuous tracking (i.e. 24/7) without considerable costs and man-hour overhead, not to mention the fact that the more people there are in the woods, the more the animals are disturbed.

Importantly, there are several other methods of animal tracking, such as the GPS and ARGOS satellite-based systems that we do not include in the direct comparison. They are inappropriate because of inferior spatial resolution (ARGOS) and reliability (GPS performs poorly in woodland). Furthermore, ARGOS tags can cost over 1500 USD each and a badger-sized GPS tag lasts for only a few months, whereas our RFID tags cost in the order of 60 USD each and last for ca. 2 years. Our RFID readers and sensor motes also contribute to the total cost of our system, however the price of each detection node is around 300 USD, thus still less than a comparable ARGOS system. We deployed 74 RFID tags and 26 detection nodes, summing to 4440 + 7800 = 12 240 USD, while buying 74 ARGOS tags would have cost us approximately 111 000 USD.

Table 2 shows the summary of the costs involved in maintaining our system. We consider the number of man-hours needed, as well as the battery costs for each stage. The total cost includes the price of monthly up-keep of the system. We also include how many animal detections we had recorded in a month and how much each of these recordings cost. In our deployment we had only two main sources of costs, maintenance visits to the woods by the zoologists and the costs involved in battery consumption and charging. Developing the new software and hardware for each stage also adds to the total cost, however this was excluded from our evaluation. There were two PhD students and two post-doctoral researchers working on the project for 3 years, however it is difficult to estimate accurately the amount of working hours spent developing the system.

Stage 1 is our initial hardware node deployed. We have logs of how much money we spent on batteries and how much time we spent in the woods. Each detection node is made up of an RFID reader and a Tmote Sky. The Tmotes are powered by AA batteries, while the readers are powered by an 18Ah 12V batteries. We spent about 147 USD on AA batteries and about 8.9 USD (4 times a month, using 0.4 kWh for 20c/kWh) for recharging the reader batteries on all 26 detection nodes. From the logs, we also see that about 30 hours per month were spent in the woods, summing to 372 USD (again, using 7.25 USD hourly wage). From our database, we collated the total number of active tags per month during the deployment, as well as the number of detections per month; thus on average, one animal generated 56,107 records per month, giving a single detection cost of around 0.6 cents.

At this point, the bottleneck becomes the 1 MB storage on the detection node - without compression, this fills up (de-

Table 2. Breakdown of the average cost incurred to maintain each stage of the system for 4 weeks. Costs are normalized with respect to the number of animals being monitored.

	Visits (man-hours)	Battery cost [USD]	Total cost [USD]	detection per animal	Cost per detection per animal
Stage 1 (HW only)	29.7	156.76	372.5	56107	0.006
Stage 1 (HW & SW)	10.8	52.9	131.4	40958	0.002
Stage 2 (HW only)	2.7	1.04	20.615	56107	0.0003
Stage 2 (HW & SW)	1.3	0.56	10.3	56107	0.0001

pending on activity) within a week, however, using our data compression technique, we were able to extend this to double the lifetime of the nodes, requiring only two field visit per month, totalling 10 hours. The adaptive duty cycling approach allowed the battery costs to be reduced to about 53 USD, or 131 USD per month. Slightly fewer records were generated, but a single record still cost less than in the previous stage i.e. 0.2 cent.

In stage 2 we introduced new hardware that radically increased the lifetime of the detection node, while yielding the same number of sightings as in stage 1. In our first stage 2 deployment, we put the hardware out for testing, without any software enhancement (such as duty cycling either the radio or the reader). The node lasted for 2 months on the same battery, and due to its extensive memory capacity, did not require data download. Since our new hardware used one large, rechargeable, car battery, this negated the need to buy AA batteries for the motes. The charging costs of the car batteries amounted to 0.2 c/kWh x 0.4 kWh x 26 x 0.5 (once in two months) = 1.04 USD per month. On average, one visit lasted for about 5.4 hours, so one visit for two months resulted in 2.7 hours per month. Since we needed to visit the nodes once in 2 months, our monthly cost was 2.7 hr x 7.25 USD + 1.04 = 20.615 USD. The cost of a single detection was reduced to 0.03 cent. It is worth noting, however, that at this point, the cost of getting to the woods or tagging the animals is actually higher than the maintenance cost.

The introduction of the enhanced software in Stage 2 (described in Section 5) further extended the lifetime of our new hardware. We obtained a 2-fold increase in the lifetime of the node, hence only one visit in every 4 months became necessary. This resulted in a maintenance cost of 10.3 USD per month, and the cost of a single detection thus became negligible.

7 Data Analysis

We collected over 29 million records since the system became fully operational in March 2009. This section analyses a subsection of these data (from 14 March 2009 to 19 September 2009) for illustration only, to demonstrate the utility of the system in generating biologically useful data. In doing so it is important to note that we do not attempt to infer biological significance from any of our observations, instead our analyses are purely descriptive. The full dataset, including microclimatic correlates gathered from sensor nodes, will be subject to zoological analysis elsewhere.

7.1 Data Gathered

Badgers are trapped up to four times a year for a concomitant research project [16]. This provides an opportunity to put RFID tags on the animals. There have been 9 trapping sessions since June 2008, during which 74 animals

were tagged.

Animals were able to remove 12 tags of these tags (collars), which were found on the ground. More tags were similarly lost, but not found. Whenever possible, these animals were retagged. Over the year, a lot of attention was given to keeping the system running uninterrupted, i.e., always replacing the batteries and downloading the data before the nodes stopped functioning. We set up a database where all the sightings were uploaded: as of June, 2010, we have about 28 million valid detections.

7.2 A Window into Badger Movement Patterns

One of the advantages of our automatic monitoring system has been that we were able to capture data with high temporal resolution from our fixed detector sites. This allowed us to produce records of daily badger activity for future zoological analyses.

A density plot of badger 'sightings' is shown in Fig. 11(a). The horizontal axis shows the time in 24-hour format. The vertical axis shows the day of year. The intensity of each dot represents the average amount of time that badgers were observed at the detection nodes.

Fig. 11(b) and Fig. 11(c) show badger detections at sett and latrine located detection nodes, respectively.

In the evening, badgers exit their setts (indicated by the strong dark line at dusk in Fig. 11(b)). They then visit the latrine nodes probably foraging for food in between. At the end of the night, they return to their setts, producing a high density of activity on the right side of Fig. 11(b).

Regarding seasonal trends, and as expected, it can be seen that the length of time that badgers are out of their setts decreases, reaching a minimum around day 170 (corresponding to June 18). From this point on, the average trip time starts to increase again, with decreasing day-length.

7.3 Badger Co-location

We extracted pairwise co-locations between badgers from the detection node records: our assumption was that two animals were within 0-60m of each other if they were recorded contemporaneously by the same detection node. Because we do not have any indication of the type, if any, of social interaction between the animals, we must be cautious in any assumptions we infer.

Since setts and latrines have different social functions for badgers, co-locations are divided into three datasets: (a) setts and latrines together; (b) setts only; (c) latrines only. To investigate the broad social structure we create a weighted social graph for each co-locations dataset where nodes represent badgers and the weight of each link is proportional to the amount of time for which the two animals were co-located.

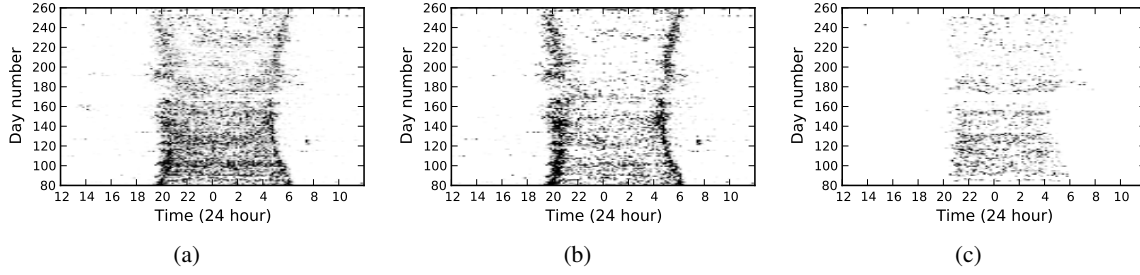Fig. 12 illustrates the resulting graphs for each dataset,

Figure 11. Badger activity captured at detection nodes. Horizontal axis is time of day and vertical axis is day of year. (a) Badgers detected at any detection node. (b) Badgers detected at nodes placed near setts. (c) Badgers detected at nodes placed near latrines.
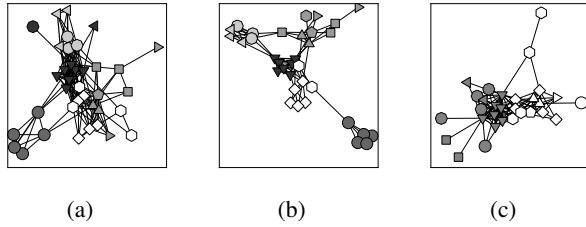

Figure 12. Badger social networks: The shade of a node (node = badger) represents the social community it belongs to while its shape denotes the sett it lives in. Different networks are created by using (a) all co-locations between animals at setts and latrines, (b) only co-locations at setts, and (c) only co-locations at latrines.

where communities have been detected using the algorithm described in [5]. The network defined by all co-locations in Fig. 12(a) depicts 5 discrete 'communities', but each interlinked with one-another. This is not as evident for the network defined from sett co-locations (Fig. 12(b)) where the 5 similar communities are more discrete with fewer links between them, giving greater separation. Conversely, for the network defined from latrine co-locations (Fig. 12(c)), only two communities were in evidence.

8 Related Work

Wildlife and Environmental Monitoring A number of other wildlife monitoring deployments also exist like Zebranet [30], DuckIsland [27] and TurtleNet[11]. Sikka et al. [26] discuss the deployment of a hybrid network consisting of mobile sensors mounted on farm animals and fixed sensors measuring soil moisture and weight of food and water consumed by animals. Selavo et al.[25] describe the deployment of wireless sensor network for measuring complex light environment in thickets and also use delay tolerant networking, fault-tolerant distributed storage and custom hardware. A number of modified Mica2 motes were deployed by Gilman et al.[28] to monitor the microclimatic conditions and solar radiation in a redwood tree for 44 days. With respect to these we have developed a very integrated heterogeneous deployment which enabled us to gather very large volumes of data. Moreover, the system is able to customise the distribution of the data depending on the urgency of the delivery required.

Duty cycling: [19] propose a machine learning based ap-

proach for adaptive resource allocation for sensor networks. The sensors are modelled as self-interested agents that attempt to maximise their profit and a simulation based evaluation is presented. An adaptive sampling technique has been used to adjust the sampling rate depending on the predictability of the phenomena [7]. The incoming data is modelled using time series models and the data rate depends on the residual error between the predicted model and the actual measurements. Our work is different because we use adaptive sampling for exploiting temporal correlations for node discovery in mobile environments.

Data collection: The MRME algorithm [10] schedules mobile sinks to visit static nodes before data delays expire. When data is close to expiration, multi-hop routing is used to guarantee timely data delivery. Unlike our approach, the MRME algorithm assumes control over sink mobility and assumes homogeneous data latency requirements for all data. The SensorScope project [3] describes the deployment of a low duty-cycle sensor network in which a central base station gathers data. SensorScope uses a non-standardised networking stack that is designed for remote areas that cannot be frequently accessed. Hui et al. [12] demonstrated the usability of the IPv6 standard for sensor networks as a flexible networking layer whilst maintaining a very low duty cycle. Our choice of network stack was strongly influenced by their findings.

In Lance [29], each data unit has an associated value, as well as a cost for multi-hop data delivery to a single basestation. Values and costs are taken into account to determine download scores, i.e. the priorities of data units for data delivery. Unlike Lance, we not only use priorities to rank data units, but also to rank storage nodes. In addition, we send data of different priorities to different storage nodes immediately, instead of delaying their delivery to a single node (the basestation). Jiang et al. [13], propose EMA, an energy management architecture that enables prioritized enforcement of policy directives. If, for example, there are sufficient energy resources in the network, a sample-and-send directive is used, whereas the system gracefully degrades to sample-and-store when energy resources become scarce. Their framework could be combined with ours to offer a greater variety of policies. For example, a directive could suggest that when energy resources are scarce, a class of data must be demoted to a lower priority. As a result, this data will be delivered to a closer but less frequently visited storage node, and will incur

a lower energy cost. Unlike existing systems, in which prioritization results in a binary decision (store vs. download), our system uses data priorities to select among a wide variety of data delivery options.

Evolution: With any design, it is very difficult 'to get it right' for the first time, despite a lot of planning and effort. We have shown how our systems developed over time, and how we have managed to reduce the maintenance cost to a tenth of the initial costs, while still collecting substantial amounts of data.

The authors of the ZebraNet project describe in [30] the different stages of hardware upgrade they went through in their deployment. They deployed 3 different sensors, each improving on the capability of their previous ones. The improvements included solar panels, changing the radio to a more energy efficient one and increasing the on-board memory. The Glacsweb Project [21] aimed at monitoring glacial dynamics through the use of WSN. They have had yearly deployments from 2001 to 2008 in different regions and countries (including Norway and Iceland). Their deployments relied on a number of 'probes' embedded in the ice, and a base station, relaying data back from the sensors to the scientists. The base staion turned out to be their single point of failure, they redesigned it from deployment to deployment to improve on reliability and robustness.

Although we detail similar evolutions to the aforementioned projects, our overall aims are different. Here we not only focus on the long-term maintainability of our system and generally improving its reliability, but we also reconcile the inherent relationships between the necessity for specific software and hardware evolutions and highlight the resulting cost savings and benefits from such actions.

9   Conclusions and Lessons Learned

Although there is currently a lot of work on building real sensor systems, very few attempts have been made to deploy them in the field and then maintain and develop them. In this paper, we provide details of the first distributed active RFID-WSN hybrid system for wildlife tracking. We undertook an iterative process of software and hardware designs and developments, while still maintaining backwards compatibility.

Maintenance Costs We gained invaluable experience from our deployment. System maintenance is a key to a long-term deployment, and the costs associated with it should be factored in from the initial design stages. Though our first stage was very successful in collecting large quantities of high quality data, maintaining it turned out to be more expensive than expected. For a wildlife monitoring application, continuous operation is essential therefore maintenance is unavoidable.

Software and Hardware Interaction With software enhancements, we were able to increase the lifetime of the system, and thus decrease the necessary maintenance, however this resulted in the hardware becoming our limiting factor, hence our second lesson: to achieve maximum power efficiency, application-specific hardware is often necessary. With our second stage, we were able to decrease the maintenance costs to a fraction of what they were before, while collecting the

same amount of data. Optimizations that work on some hardware, however, might not perform as well on a different system, i.e. software optimizations need to take into account the capabilities and the characteristics of the hardware. The introduction of the new hardware resulted in fewer visits to the woods by the zoologists, which affected the in-network-storage. Moreover, once the new hardware was in place the detection node duty cycling could be improved with finer grain parameters which would not have been possible on the earlier version of the hardware.

Rapid Initial Prototyping and Deployment One of the most pertinent results from our deployment was the realization that no initial deployment will be perfect. This suggests that the best approach for long term monitoring systems is to design a prototype that can be rapidly deployed using commercial off-the-shelf technology. This is especially important in applications like ours, where no prior data had been collected on a similar scale in the same environment. Although our initial prototype suffered from a lot of practical issues, it was easy to get the system working in the field and this allowed us to collect suitable data to understand how things could be improved. These observations then guided the evolution of the system, allowing us to dramatically reduce the cost of system maintenance by increasing the runtime of devices. In addition, no amount of simulation or laboratory testing is equivalent to issues faced in the real deployment. Failures are common, and some failures, such as animals interfering with equipment are unquantifiable until the system is actually deployed. Thus we suggest that researchers deploy an initial version (even if it is a datalogger) as soon as possible, so that knowledge can be gained about practical problems.

Gradual versus step-change improvements In the evolution of a system, a choice has to be made whether to improve it gradually or to switch over to a new system entirely. The choices made here were influenced by the needs of the application. In our case, we had to slowly incorporate new improvements, testing them over a period of months in the field, so as to gather a continuous record of data. This was because any gaps in the data could significantly reduce their biological significance. Other applications can tolerate interruptions that allow for all effort to be concentrated on designing and deploying a new and improved version. This leads naturally to step-changes in capability and functionality, with all components of the system being upgraded simultaneously. This is an important lesson, as it dictates the type of evolutionary strategy that can be adopted.

Continuous interaction with domain scientists Our system was built as an *experimental tool*, as opposed to a proof of concept. The design of smart protocols and algorithms to reduce message overhead or energy consumption is only useful if it complies with the requirements of the eventual users of the system. Such interactions are not only useful to make sure the system works as expected, but also to provide interesting ideas for optimizations. One such key observation made when discussing system requirements with the domain scientists was that not all data had real-time requirements. In response, we formulated a priority based routing approach that reduced traffic load, particularly around network hotspots, by forming multiple routing trees. Data

were opportunistically picked up by zoologists working in the woods, another application specific factor we took advantage of. To reduce data volumes, a simple lossless compression algorithm was devised. Analysis of collected data resulted in interesting findings about badger social behavior and correlations with weather conditions. By understanding the needs of the users, we were able to tailor our system design, extending the lifetime of devices in the network, whilst still satisfying application requirements.

In summary, we learned a number of interesting lessons. First, network maintenance should not be an afterthought, but a key consideration in the original design of the system. Otherwise, maintaining a sensor network ends up being far more expensive than building it. Second, before delving into algorithmic improvements and strenuous testing of new software, it is important to carefully consider hardware limitations. Sometimes it is more cost-efficient to replace the hardware platform than to design and test new software for an existing platform. Third, the benefits of software optimization (e.g. improving sampling, storage and data collection algorithms) largely depend on the hardware. An algorithmic improvement that yields significant benefits on one platform may be less efficient or even inapplicable to another. Fourth, engineering sustainable sensor networks is an iterative process that alternates between hardware and software changes. Last, these changes must be performed in a controlled manner so that they do not disrupt the data collection process.

Finally, we believe the results and findings in this paper will provide an important insight into the workings of a long-lived outdoor sensor deployment.

10 Acknowledgments

References

[1] Wavetrend TG 100 Domino Tag. http://www.wavetrend.net.

[2] J. Aschoff. *Circadian Clocks*. North Holland Press, 1965.

[3] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. SensorScope: Out-of-the-Box Environmental Monitoring. In *Proceedings of IPSN '08*, pages 332–343, Washington, DC, USA, 2008. IEEE Computer Society.

[4] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel. PermaDAQ: A Scientific Instrument for Precision Sensing and Data Recovery in Environmental Extremes. In *Proceedings of IPSN '09*, pages 265–276, Washington, DC, USA, 2009. IEEE Computer Society.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics*, 2008(10):P10008, 2008.

[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 *Proceedings of SenSys '06*, pages 307–320, New York, NY, USA, 2006. ACM.

[7] S. Chatterjea and P. Havinga. An Adaptive and Autonomous Sensor Sampling Frequency Control Scheme for Energy-Efficient Data Acquisition in Wireless Sensor Networks. In *Proceedings of DCOSS'08*, pages 60–78, Santorini, Greece, June 2008.

[8] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of SenSys'08*, pages 421–422, New York, NY, USA, 2008. ACM.

[9] V. Dyo and C. Mascolo. Efficient Node Discovery in Mobile Wireless Sensor Networks. In *Proceedings of DCOSS '08*, pages 60–78, Santorini, Greece, June 2008.

[10] E. Ekici, Y. Gu, and D. Bozdag. Mobility-based Communication in Wireless Sensor Networks. *IEEE Communications Magazine*, 44(7):56–62, July 2006.

[11] A. Gorlick. Turtles to Test Wireless Network, July 2007.

[12] J. W. Hui and D. E. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proceedings of SenSys '08*, pages 15–28, New York, NY, USA, 2008. ACM.

[13] X. Jiang, J. Taneja, J. Ortiz, A. Tavakoli, P. Dutta, J. Jeong, D. Culler, P. Levis, and S. Shenker. An Architecture for Energy Management in Wireless Sensor Networks. *SIGBED Review*, 4(3):31–36, 2007.

[14] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[15] R. E. Kenward. *A Manual for Wildlife Radio Tagging (Biological Techniques)*. Academic Press, 2 edition, 2001.

[16] D. W. Macdonald and C. Newman. Population Dynamics of Badgers (*Meles meles*) in Oxfordshire, U.K.: Numbers, Density and Cohort Life Histories, and a Possible Role of Climate Change in Population Growth. *Journal of Zoology*, 256(01):121–138, 2002.

[17] D. W. Macdonald, C. Newman, C. D. Buesching, and P. J. Johnson. Male-biased Movement in a High-density Population of the Eurasion Badger (*Meles meles*). *Journal of Mammalogy*, pages 1077–1086, 2008.

[18] D. W. Macdonald, P. Riordan, and F. Mathews. "Biological Hurdles to the Control of TB in Cattle: A Test of Two Hypotheses Concerning Wildlife to Explain the Failure of Control". *Biological Conservation*, 131(2):268 – 286, 2006. Infectious Disease and Mammalian Conservation.

[19] G. Mainland, D. C. Parkes, and M. Welsh. Decentralized, Adaptive Resource Allocation for Sensor Networks. In *Proceedings of NSDI '05*, pages 315–328, Berkeley, CA, USA, 2005. USENIX Association.

[20] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proceedings of WSNA '02*, pages 88–97, New York, NY, USA, 2002. ACM.

[21] K. Martinez, J. K. Hart, and R. Ong. Deploying a Wireless Sensor Network in Iceland. In *Proceedings of GSN '09*, pages 131–137, Berlin, Heidelberg, 2009. Springer-Verlag.

[22] E. Neal and C. Cheeseman. *Badgers*. Poyser Books, 1996.

[23] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM Comput. Commun. Rev.*, 24:234–244, October 1994.

[24] C. Sadler and M. Martonosi. Data Compression Algorithms for Energy-constrained Devices in Delay Tolerant Networks. In *ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2006.

[25] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter. LUSTER: Wireless Sensor Network for Environmental Research. In *SenSys'07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 103–116, New York, NY, USA, 2007. ACM.

[26] P. Sikka, P. Corke, P. Valencia, C. Crossman, D. Swain, and G. Bishop-Hurley. Wireless Adhoc Sensor and Actuator Networks on the Farm. In *Proceedings of IPSN '06*, pages 492–499, 2006.

[27] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons From A Sensor Network Expedition. In *Proceedings of EWSN '04*, pages 307–322, 2004.

[28] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 51–63, New York, NY, USA, 2005. ACM.

[29] G. Werner-Allen, S. Dawson-Haggerty, and M. Welsh. Lance: Optimizing High-resolution Signal Collection in Wireless Sensor Networks. In *Proceedings of SenSys '08*, pages 169–182, 2008.

[30] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware Design Experiences in ZebraNet. In *Proceedings of SenSys '04*, pages 227–238, New York, NY, USA, 2004. ACM.