Delay-Tolerant Networking for Long-Term Animal Tracking

Philipp Sommer, Branislav Kusy, Philip Valencia, Ross Dungavell, Raja Jurdak
CSIRO, Autonomous Systems, Brisbane, Australia

Abstract
Enabling Internet connectivity for mobile objects that do not have a permanent home or regular movements is a challenge due to their varying energy budget, intermittent wireless connectivity, and inaccessibility. We present a hardware and software framework that offers robust data collection, adaptive execution of sensing tasks, and flexible remote reconfiguration of devices deployed on nomadic mobile objects such as animals. The framework addresses the overall complexity through a multi-tier architecture with low tier devices operating on a tight energy harvesting budget and high tier cloud services offering seamless delay-tolerant presentation of data to end users. Based on our multi-year experience of applying this framework to animal tracking and monitoring applications, we present the main challenges that we have encountered, the design of software building blocks that address these challenges, and examples of the data we collected on flying foxes.

Introduction
Tracking wildlife at continental scale has received high research interest for decades due to its significance for ecological conservation and disease spread monitoring. Ecologists are interested in understanding the movement and condition of animals across different landscapes and environmental conditions, as this understanding can lead to better conservation and management decisions. The significance of wildlife tracking is further amplified for species that can carry virulent and potentially deadly diseases, as the movement of animals correlates highly with the likelihood that the disease spreads across the landscape. For instance, fruit bats are known carriers of viruses such as Nipah, Ebola, and Hendra. Australian federal and state governments have set up a national monitoring program to understand where these highly mobile animals travel for better understanding of Hendra virus dynamics, through a combination of individual animal tracking tags and manual surveys of roosting sites.

Effectively tracking wildlife involves temporal, spatial, and functional aspects. On the temporal dimension, ecologists need individual tracking of animals that is long-term, high frequency, and delay-tolerant. Long-term operation is required because the animals move in nomadic patterns that vary over time scales of weeks to months. High frequency position and condition sampling are important for fine-grained understanding of the animals’ movement patterns and behaviours. Delay-tolerance ensures that the data from tagged animals with prolonged disconnections from the communication infrastructure is reported once connectivity returns. Spatially, tracked animals can move large distances over short time periods. For instance, flying foxes can typically move tens of kilometers over a night, and have been reported to move up to 600 kilometers in one night. The communication infrastructure for retrieving data from these animals must therefore be spatially spread across large geographical areas and must provide seamless and efficient download capability from
any tracking animal. Finally, at an operational level, wildlife tracking is clearly a long-term activity where many of the research questions evolve with inflow of new data, which necessitates highly reconfigurable and re-taskable systems.

Recent work in wildlife tracking, however, falls short on addressing the above requirements. Most tracking projects collect either short-term frequent data or long-term sparse data, subject to very limited battery energy that is constrained by the weight of tracking nodes. Delay-tolerance has been partially addressed in some efforts by logging data continuously and offloading it when connections resume. For tracking systems over large spatial scales, however, base stations that serve as the data collection points typically act independently, without coordinating with other base stations to check whether data from the tracking node had been previously downloaded. This creates potential for redundancies in data download as well as bandwidth and energy inefficiencies in the system. Finally, currently available systems do provide a degree of remote reconfiguration of sensor sampling schedules, such as GPS, but they do not support retasking or fully reprogramming nodes remotely. This limits the versatility of tracking studies once the nodes are deployed.

This article addresses the above challenges by proposing an architecture for long-term delay-tolerant networking, consisting of three tiers: mobile nodes; gateways; and cloud services. Our mobile nodes include solar panels, GPS, and many low-power sensors, in addition to algorithms for energy-based sensor scheduling and delay tolerant data collection. Our gateway nodes that are spatially dispersed synchronise data availability through the cloud services tier to ensure that data is downloaded only once throughout a continental-scale deployment. The architecture supports, in addition to remote reconfiguration, full remote reprogramming of the nodes once they are in contact with a base station, providing maximum versatility for long-term tracking studies. We showcase the features of our architecture through the motivating application of tracking flying foxes across Australia, which has inspired its original design.

Challenges
Animal tracking systems push limits of the form factor, weight, and cost of electronic devices attached to objects compared to conventional sensor networks. Animal welfare considerations or deployment-of-scale requirements often limit the computation, communication, data storage, and energy capacity available on the device. Here, we distill the key challenges that we encountered in long-term animal tracking applications.

Constrained and Variable Energy Budget. Due to stringent constraints for weight and form factor, mobile nodes need to rely on batteries for energy storage. Most current work in animal tracking employs non-rechargeable batteries for high frequency sampling [2,3,13]. Energy harvesting allows to recharge the on-board batteries and thus allows for long-term operation. However, energy harvesting exposes new challenges with energy availability and sensor sampling as energy budgets are dynamic and unpredictable with an order of magnitude difference between energy harvested on sunny summer and cloudy winter days. Therefore, the software needs to gracefully adapt to a varying energy budget. To avoid missing critical data due to lack of energy, the nodes need to schedule sampling of several
on-board sensing modalities in accordance with the current battery state of charge, daily energy budget, and predicted activity of the mobile object. Nevertheless, batteries might get depleted and the device will not be able to operate before sufficient energy has been harvested again and operation can resume.

**Intermittent Network Connectivity.** Free-living animals can cover large distances each day. Communication over existing wireless infrastructure, such as mobile phone networks or satellite uplinks, is often not feasible due to lack of network coverage, limited bandwidth, strict weight constraints, limited energy, or cost constraints (subscription costs) in animal tracking scenarios. Recapturing animals to manually collect data from the device or building dense communication infrastructure to ensure continuous connectivity of each node is often impractical. While some existing commercial devices [3] offer an opportunistic wireless download of data through a portable receiver device carried by a human, labor costs of such human-based data collection methods are prohibitive for large deployments. Animal tracking projects have mainly focused on either a single base station [2,14], which limits their spatial scalability, or collection of contact logs [13] without capturing heterogeneous sensor data, or multimodal sensing without absolute position sampling [15]. Instead, our software framework should offer seamless communication, tracking and data presentation that support situations when animals leave a known area for weeks and return to a different area, possibly hundreds of kilometers away.

Flying fox tracking exemplifies this scenario. Flying foxes, also known as fruit bats or mega bats, congregate in large numbers in day roosts, where placing a gateway node provides a great opportunity to download data from the tagged animals in the roost. During nightly foraging flying foxes can fly large distances and migrate to other roosts. While many animals come in proximity with a gateway placed at a known day roosting camp every few days, it might also take several weeks before the next contact with a gateway is made. We show the histogram of the time interval between two subsequent contacts with a gateway node for 73 flying foxes in Figure 1. Therefore, mobile devices should provide on-board persistent storage to record data for multiple weeks and communication protocols need to be able to determine which data has been downloaded to date.

![Figure 1: Distribution of the time interval between successive contacts with a gateway for 73 mobile nodes attached to flying foxes (fruit bats).](image)

**Lack of Physical Access.** Due to the nature of the deployment scenarios involving animals, physical access to nodes might not be feasible after the initial deployment, e.g., it is difficult or infeasible to capture the tagged animal again after the initial deployment. Therefore, mobile nodes need to operate on a near-perpetual basis for long periods of time without physical human intervention. However, our
experience has shown that it is often necessary to verify that the nodes operate correctly and to reconfigure sensing tasks after the initial deployment. Therefore, the software framework should provide remote debugging and configuration methods that operate over the wireless channel and handle intermittent connectivity between gateways and mobile nodes.

**Network Architecture**

The main challenge that our system needs to tackle is the lack of communication infrastructure in a majority of the tagged animal’s habitat. Consequently, we based our system architecture around a sparse network of gateway nodes that communicate directly with mobile nodes to download the most recent data and use a cloud service to synchronize the metadata among gateways (see Figure 2 - top).

**Mobile Nodes (Tier 1).** The mobile nodes are attached to the monitored animal using specially designed collars or halters, as shown in Figure 2 (bottom left and center). When designing the mobile node units, the form factor and weight restrictions heavily depend on the physical size of the tagged objects (e.g., animal ethics regulations require collars to weigh less than 5% of the animal’s body weight). We use the TI CC430 low-power microcontroller architecture to prolong node lifetime when running from small batteries integrated into the collars. Consequently, the computational power and storage capabilities of such devices are highly constrained. Mobile nodes feature sensor chips for multiple sensing modalities, persistent data storage capabilities and a short-range wireless transceiver.

![Figure 2: Three-tier device architecture for the delay-tolerant animal tracking and monitoring](image-url)
Gateway Nodes (Tier 2). The second tier consists of gateway nodes, which are equipped with short-range wireless radios to communicate with the mobile nodes (e.g., IEEE 802.15.4 radios). As the communication range of the mobile nodes is restricted to a few hundred meters, gateway nodes are placed at strategic locations where animals tend to congregate (e.g., flying foxes roosting camps, cattle drinking troughs) to maximize the chance of wireless connectivity with mobile nodes. We implemented a framework for remote method invocations with low overhead using bi-directional radio packets between a client (gateway node) and a server (mobile node). A remote procedure call (RPC) is initiated by a radio packet transmitted by the client containing the command identifier and optional arguments, and is acknowledged by a response packet [10]. We used RPC commands to implement several network services such as data download, remote reconfiguration and reprogramming (see Figure 3). Thereby, mobile nodes support a small set of RPC commands to execute simple tasks, which provides flexibility to implement high-level communication protocols by changing the sequence of RPC commands at the gateway nodes [4].

In contrast to mobile nodes, relaxed form factor and weight restrictions for gateway nodes allows to include larger electronics, batteries, antennas, and solar panels (see Figure 2 - bottom right) and to run standard operating systems (Linux) and communication protocols (TCP/IP) on gateway nodes. Gateway nodes can be connected to the Internet using cellular networks, point-to-point WiFi links, or Ethernet, depending on which access technology is available in the area of interest.

Cloud Services (Tier 3). The third tier in our system consists of a several web services located in the cloud. As bandwidth is less restricted between gateway nodes and the cloud services, we employ standard Internet protocols such as HTTP and encode data traffic into JavaScript Object Notation (JSON) objects. At the core of Tier 3 is a collection of RESTful [5] web services which provide access to sensor data, global view of the node download state, and node configuration information. The data ingest service accepts HTTP POST requests containing the sensor data in a JSON object wrapper. We further implemented a RESTful API to provide hierarchical access to resources and sensor data, which is also used to provide a system dashboard for engineers and domain scientists. We use the HDF5 hierarchical storage system [6] for archival data storage as it provides large storage capacity and is an excellent archival data storage system, as all data and metadata is contained in a single file.

Figure 3: Software architecture of our framework: Sensing tasks can be remotely configured using a set of simple rules that are evaluated at runtime by the energy- and context-aware task
scheduler. Sensor readings from different sources are encoded into a TDF stream on the mobile
node and decoded on the gateways before the data is forwarded to the cloud service. Gateway
nodes can update the task configuration and program image remotely using RPC commands.

Energy Constraints
The main challenge of the software architecture is to integrate devices operating in different system
tiers and with different energy budgets. With the exception of the cloud services in Tier 3, individual
devices might be duty-cycled and operate according to a schedule that depends on the available
energy budget and mobile-node related activity.

Mobile Nodes. The mobile nodes need to operate in a highly efficient manner to meet their strict
energy constraints. They must use ultra-low power in sleep mode, optimize sensor sampling for
maximum information gain, such as tracking animal location only when the animal is moving, and
minimize idle listening in their communication protocols. We designed the Contiki OS [11]
application deployed on the mobile nodes to provide unsupervised long-term operation with energy
awareness. The battery state of charge is subject to fluctuation, which largely depends on the energy
consumed to execute sensing and communication tasks, and on the amount of harvested energy [7]. At
the core of the mobile node’s software is a task scheduler, which manages different sensing tasks
based on the available energy and context, as illustrated in Figure 3.

A task configuration describes the behavior of a specific Contiki process defined by a set of entry and
exit conditions such as time of day, voltage and context (e.g. motion detected). The task scheduler
periodically checks all task configurations and starts the corresponding task when its entry conditions
are fulfilled. Similarly, currently running tasks are checked for the specified exit conditions (timeout,
number of samples acquired, depletion of energy resources) and stopped if necessary.

The execution of sensing tasks is based on the current state of charge and the amount of sensor data
collected is gracefully reduced when the battery charge level is decreasing. We show an example of
this scheduling mechanism for a mobile node deployed on a flying fox in Figure 4. The scheduler
adapts the sampling rate of the on-board GPS receiver based on the measurement of the battery
voltage. In case of a low battery voltage reading (3.7V in this example), the GPS is sampled less
often, e.g., only periodic samples are taken instead of motion-triggered continuous tracking. This
allows the battery to be recharged through the solar panels. Once the battery voltage recovers to an
acceptable level (3.9V), the GPS can be scheduled more often again. Multi-modal sensing capabilities
(e.g. inertial, audio, temperature, pressure) can classify the node context further for optimal
scheduling of energy intensive operations, which can result in significant energy savings [8].
Figure 4: Battery voltage measurement and GPS sampling activity as reported by a mobile node attached to a free-living flying fox. The task scheduler decreases the GPS activity as soon as the battery voltage falls below 3.7V on Day 6. This allows the battery to recharge and the voltage increases again. GPS sampling returns back to normal sampling as soon as the battery voltage reaches a value of 3.9V on Day 16.

Gateway Nodes. Although the form factor of gateways is less restricted than for mobile nodes, cost considerations and mounting restrictions will result in size constraints for batteries and solar panels, thus limiting the energy supply for gateway nodes. Therefore, it might become necessary to operate the embedded PC and cellular network transceiver at a duty cycle, which will result in intermittent connectivity to the cloud services. We adapt the duty cycle of the gateway node based on its battery state of charge. To optimize power usage at gateways, we prolong gateway operation when a lot of packets are waiting to be downloaded from mobile nodes, while gateway nodes go back to sleep immediately when no mobile nodes are present nearby.

Intermittent Connectivity
As mobile nodes spend only short time periods in areas that have wireless connectivity, we need to provision for logging sensor data to a temporary storage on mobile nodes and provide delay-tolerant mechanisms to upload the data to the cloud services. Furthermore, the framework should support intermittent connectivity not only with the mobile nodes, but also with base stations that may be energy constrained themselves.

Data Storage. Different technologies that vary in their energy efficiency and data capacity can be used on mobile nodes for local storage. For example, we use an 8 MBit flash chip in the flying foxes monitoring application due to its high energy efficiency, while we use a 4 GB microSD card in the cattle application due to its large capacity.

We implemented a logging abstraction that allows applications to access the local storage in a unified way without needing to know the structure of the underlying hardware architecture. The logging abstraction is based on pages, whose size corresponds to the page size of the physical storage medium (e.g., 256 Bytes for common flash chips). We use increasing page numbers to retrieve data from the logger and identify data that has been downloaded from the logger.
The data format used to store sensor readings is an important consideration as it influences the energy efficiency of communicating the data over the radio. For transmission efficiency reasons, we store and transmit data in a binary format, which makes it difficult for humans to interpret directly, but introduces significant energy savings. We use the Tagged Data Format (TDF) [10] to encode multiple sensor readings into a byte stream. Each sensor sample is stored with a unique sensor identifier, which defines how the sample will be interpreted. All sensor samples are timestamped at a granularity that fits application accuracy needs and storage constraints [4]. In line with recent approaches to store and transmit information from IoT devices [9], we advocate for separation of the actual data from metadata that defines how the data can be interpreted. In our case, the metadata information is stored at the cloud service to which gateways periodically synchronize (see Figure 3). Mobile nodes, on the other hand, keep a static version of the metadata and backwards compatibility is ensured through creating new data types in the metadata rather than modifying the old ones.

**Protocols for Delay-tolerant Network Operation.** Organisation of our system into three distinct tiers (mobile nodes, gateways, cloud services) enables us to tailor communication between different tiers to meet the resource constraints imposed by the application scenario. At the mobile node layer, we employ a unicast communication protocol based on the RIME stack of the Contiki operating system [11] for node discovery and data transfers. We designed our radio communication protocol to take into account the highly asymmetric energy resources available at mobile and gateway nodes. Consequently, mobile nodes employ duty-cycling of the radio transceiver to reduce the power consumption, while gateway nodes can afford to keep their radio enabled for longer periods. Thereby, the mobile node’s radio remains powered off most of the time and is only active for a short amount of time to send a radio packet containing node status information. Unless the gateway requests the mobile node to keep its radio enabled using a RPC command, it will go back to sleep and wait until the next beacon is due.

In contrast to many data gathering protocols for static wireless sensor networks, data transfers are not initiated by the mobile node, but by the gateway node. This design decision has been made to simplify the application logic in the mobile node so that it is not required to keep track which parts of the buffered sensor data have already been downloaded. We implemented a mechanism to download sensor readings from the flash logging abstraction using our RPC framework. Data stored in flash pages is sequentially transferred by requesting small chunks that each fits into a single radio packet. In real-world scenarios we achieve a throughput between 4 and 6 pages downloaded per second.

**Delay-Tolerant Operation with Multiple Gateways.** Gateway nodes will forward data downloaded from mobile nodes to a web service, where sensor readings are then stored into the database. Mobile nodes periodically broadcast their maximum available page number, which is overheard by the gateway nodes when in proximity. We use a central web service, which provides the lowest page number that has not been downloaded yet and the range of pages waiting for download. The gateway node will report the successful download of each page to the web service. This ensures that other gateways will not attempt to download the same memory region again in the future. Shifting the burden of keeping track of which pages have been downloaded to the web service allows to keep the application logic on the nodes and gateways stateless.

In addition to data downloaded from mobile nodes, gateway nodes will also periodically report gateway health information to the server such as uptime, battery state of charge, and storage capacity, which are used to monitor the reliable operation of our network of duty-cycled gateways.
Gateway nodes always keep a local copy of the global node state in persistent storage, which will also allow to operate gateway nodes when no cellular connection to the Internet is available. In this case, sensor data downloaded from mobile nodes is temporarily buffered on the gateway until an Internet connection becomes available and data can be synchronized to the cloud services.

**Remote Debugging and Configuration Management**

In wildlife applications it is often not feasible to re-capture individual tagged animals as it is difficult to precisely locate the animal or it would cause additional stress for the animal. Therefore, long-term operation requires remote inspection and control of the software running on the devices. For energy efficiency, the system should support reconfigurability at different levels of the software stack: changing simple parameters such as thresholds, reconfiguration of application-level logic, and replacement of the whole program binary running on the node. Therefore, remote instrumentation and debugging techniques have proven to be a crucial tool during the initial prototyping and the actual deployment phases. We leverage our RPC command framework for interactive memory inspection and to query the state of program execution, such as acquiring battery voltage readings, logger status information, reading/writing of task configurations, and information about running tasks.

**Remote Configuration.** We use a multi-level approach for remote configuration and modification of applications running on inaccessible mobile nodes. Small changes to an application, such as modifying sensor sampling rates, can be performed without the need for updating the whole program image stored in ROM. Task configurations can be added, modified or removed using RPC commands. We keep a dedicated flash memory area for persistent storage of configuration parameters, which the application loads at startup. This proved to be very useful when testing and debugging novel software components in the field, as they can be enabled and disabled remotely without the need to update the whole program image.

**Wireless Reprogramming.** Techniques for remote updating of application images are known as wireless reprogramming and have been previously adopted for wireless sensor networks [12]. In order to update the program image running on the node, we split the binary application image into several smaller chunks that are transmitted over the radio. At the node, the image is reassembled from small chunks and written to the program ROM. While this approach allows to replace the complete application, a considerable number of radio packets is required to transfer the new image, so it is only used in more significant cases, such as critical bug fixes to low-level components.

**Node Configuration Updates.** Users might want to change the configuration of one or several nodes to adapt their sensing tasks to a changing context. However, the specified nodes might not be in proximity of a gateway for several weeks. Therefore, we use a delay-tolerant approach for updating the task configuration or to update the program image running on a node, which does not require manual intervention. Upon receiving a beacon packet from a mobile node, the gateway node requests a description of each node’s up-to-date task configuration from the web service and uses RPC calls to compare it to the actual configuration present on the mobile node. If the two configurations differ, the new task configuration is written to the node, and the gateway notifies the cloud service that the node’s configuration has been updated. Similarly, we also check for the version number of the program image currently running on a node and update the program binary if necessary.
Discussion and Lessons Learned

We have learned important lessons from our multi-year experience in animal monitoring applications, at the technical, operational, and collaborative levels. From a technical perspective, it has become clear that a multi-tiered architecture is necessary to shift the computational and communication complexity towards the cloud and away from inaccessible and resource-constrained nodes. We have also established the value of multi-level reconfiguration features for nomadic devices, ranging from simple parameter updates to full node reprogramming, both in real-time or with delay-tolerance, to maximize versatility as application requirements evolve over time.

This versatility is critical for scientific investigation, where future experimental parameters and required data may differ from that of the original experiment configuration. Domain experts typically have some expectations about the animal behavior based on previous studies and knowledge and make certain assumptions for how reliably the technology performs in the field. As the experiments progress and first data comes back from the experiments, sampling parameters need to be adjusted to better fit animal activity as well as technical and logistical constraints. The delay-tolerant node reconfiguration proved essential for scalable updates to all active nodes without accessing them physically. Equally important is seamless support for multiple active versions of the embedded software. Our deployments use a phased approach where nodes are progressively added to the system, with potentially new software versions and features for every new batch. Version tracking within our software architecture supports the co-existence of multiple software versions and gracefully adapts responses to each node’s software capabilities.

We have successfully implemented the proposed framework in the context of animal tracking and monitoring applications. Several services and abstractions turned out to be essential during prototyping, testing, and operation of these applications. First, the typed data storage abstraction offered the necessary flexibility with adding different sensing modalities to applications at various sampling rates. Next, the methods that we proposed for autonomous and delay-tolerant operation of mobile nodes proved to be of critical importance for tracking flying foxes, as data had to be buffered locally possibly for weeks at a time until the animal returned to a roosting camp with gateway nodes, as shown in the GPS traces presented in Figure 5a. It turns out that observed gaps in the trajectory can occur due to spurious mismatches between the sensitivity of the accelerometer-triggered GPS sampling and the accelerometer reading when an animal actually moves, which highlights the importance of remote reconfiguration to optimize these sensitivities over time.

We also learned that while nodes can maintain a healthy battery voltage for a given sensor sampling configuration thanks to solar harvesting, long absences from the base station result in a backlog of data that requires significant energy for download once the node returns (see Figure 5b after day 23). Sensor sampling frequencies can then be reduced temporarily after this bulk download to allow the node to recover its energy supplies. Adaptiveness of the software provided our domain scientists with an interactive tool to adjust the way the sensors are sampled based on retrospective data analysis or new scientific discoveries, which provides great benefits over deploying several generations of devices during the scientific discovery process.

While animal monitoring has helped motivate and mature our network architecture, we expect it to be useful more broadly, for instance to the traceability of foods products in transit, and more generally to objects that move beyond urban regions.
Figure 5: Movement tracking for an individual collared flying fox based on GPS samples at 1 Hertz (a). Timeline of data acquisition and data download at the gateway node, and variations in the mobile node’s battery voltage during the deployment period (b).

Acknowledgments
We would like to thank Ben Mackey, Christopher Crossman and Luke Hovington for their contributions towards the implementation of the software architecture this work is built on. We further thank Adam McKeown, David Westcott, Jiajun Liu and Kun Zhao for their valuable inputs regarding the system architecture for the flying foxes monitoring system. This work has been supported by the CSIRO Sensor and Sensor Networks TCP.

References
[1] D. Anthony, W. P. Bennett, M. C. Vuran, M. B. Dwyer, A. Lacy, M. Engels, W. Wehtje, “Sensing through the continent: towards monitoring migratory birds using cellular sensor networks”, Proceedings of IPSN, pg. 329-340, 2012.

[2] P. Zhang, C. M. Sadler, S. A. Lyon, M. Martonosi, “Hardware Design Experiences in ZebraNet”, Proceedings of SenSys, pg. 227-238, 2004.

[3] e-obs GmbH, Franzmitter, http://www.e-obs.de/products.html

[4] P. Sommer, B. Kusy, A. McKeown, R. Jurdak, “The Big Night Out: Experiences from Tracking Flying Foxes with Delay-Tolerant Wireless Networking”, Proceedings of REALWSN, pg. 15-27, 2013.

[5] R. T. Fielding, R. N. Taylor, “Principled Design of the Modern Web Architecture”, ACM Transactions on Internet Technology, vol. 2, no. 2, pg. 115-150, 2002.

[6] The HDF Group, http://www.hdfgroup.org/
[7] P. Sommer, B. Kusy, and R. Jurdak, "Power Management for Long-Term Sensing Applications with Energy Harvesting", *Proceedings of ENSSys*, 2013.

[8] R. Jurdak, P. Sommer, B. Kusy, N. Kottege, C. Crossman, A. McKeown, D. Westcott, “Camazotz: Multimodal Activity-based GPS Sampling”, *Proceedings of IPSN*, pg. 67-78, 2013.

[9] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, D. Culler, “sMAP: A Simple Measurement and Actuation Profile for Physical Information”, *Proceedings of SenSys*, pg. 197-210, 2010.

[10] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, D. Moore, “Environmental Wireless Sensor Networks”, *Proceedings of the IEEE*, vol. 98, no. 11, pg. 1903 - 1917, 2010.

[11] Contiki: The Open Source OS for the Internet of Things, [http://www.contiki-os.org](http://www.contiki-os.org)

[12] Q. Wang, Y. Zhu, L. Cheng, “Reprogramming wireless sensor networks: challenges and approaches”, *IEEE Network*, vol. 20, no. 3, pg. 48 - 55, 2006.

[13] Lindgren, Anders, et al. "Seal-2-Seal: A delay-tolerant protocol for contact logging in wildlife monitoring sensor networks.", *Proceedings of MASS*, 2008.

[14] Z. Butler et al. Virtual fences for controlling cows. *In Proc. ICRA*, pages 4429–4436, 2004.

[15] V. Dyo et al. Evolution and sustainability of a wildlife monitoring sensor network. *In Proc. Sensys*, pages 127–140, 2010.

**Biographies**

**Philipp Sommer**, Postdoctoral Fellow, Autonomous Systems, CSIRO, Brisbane, Australia. E-mail: philipp.sommer@csiro.au.

**Brano Kusy**, Principal Research Scientist, Autonomous Systems, CSIRO, Brisbane, Australia. E-mail: brano.kusy@csiro.au.

**Philip Valencia**, Software Engineer, Autonomous Systems, CSIRO, Brisbane, Australia. E-mail: philip.valencia@csiro.au.

**Ross Dungavell**, Senior Software Engineer, Autonomous Systems, CSIRO, Brisbane, Australia. E-mail: ross.dungavell@csiro.au.

**Raja Jurdak**, Principal Research Scientist, Autonomous Systems, CSIRO, Brisbane, Australia. E-mail: raja.jurdak@csiro.au.