Automated Encounter Detection for Animal-Borne Sensor Nodes

Björn Cassens
TU Braunschweig
cassens@ibr.cs.tu-bs.de

Simon Ripperger
Museum für Naturkunde Berlin
simon.ripperger@mfn-berlin.de

Martin Hierold
FAU Erlangen-Nürnberg
martin.hierold@fau.de

Frieder Mayer
Museum für Naturkunde Berlin
frieder.mayer@mfn-berlin.de

Rüdiger Kapitza
TU Braunschweig
kapitza@ibr.cs.tu-bs.de

Abstract

Tracking animals is mandatory for a better understanding of their social interactions. Social behavior of bats is of particular interest, since it can be used for improved epidemic avoidance and forecasts. Animal-borne sensor nodes are enabling the monitoring and analyzing the social contacts. Attached to bats, these nodes can only be equipped with small batteries, dictating tight requirements regarding the energy demand of the soft- and hardware infrastructure.

In this paper, we present our software architecture of a wireless sensor network that allows a fully-automated tracking of the social behavior of free ranging animals. This software was extensively tested during a two month lasting field test in the tropical rain forest of Panama, thereby monitoring 38 free-ranging bats. We documented more than 300,000 biologically relevant events, generated by animal-borne sensor nodes running up to 15 days. Furthermore, we support runtime reconfiguration, allowing to adapt the system during campaigns without disturbing the animals. Showcasing that our system and especially the software works under realistic conditions, we give an in-depth insight of our experience during the field test.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and behavioral science; D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*

General Terms

Experimentation, Measurement, Design

Keywords

Energy aware systems, Wearable, Firmware, Animal tracking, Wireless sensor networks

1 Introduction and Motivation

The observation of the (social) behavior of animals in their natural environment is a great challenge for biologists. Often they rely on technical solutions to study wildlife in detail since direct observation may not be feasible for cryptic or highly mobile species. Wireless Sensor Network (WSN) consisting of miniaturized animal-borne sensor nodes (called Mobile Node (MN)) give several new opportunities for wildlife tracking as different prior works shows [19, 21]. A high degree of automation allows for creating extensive data sets with moderate effort. Wireless sensor nodes can collect and store accessory physiological or environmental data sensed by on-board sensors. Remote data access minimizes the impact on the observed animals ensuring the documentation of unbiased behavior. Finally, communication among MNs gives the opportunity to study group dynamics and social behavior of gregarious animals.

However, this increase in functionalities comes at the cost of higher energy demand compared to the state of the art radio telemetry. Current high-performance tracking devices carry large batteries and are in turn rather heavy (the weight ranges from 30 g [19] up to 200 g [21]). The high weight limits their applicability to medium or large-sized vertebrates since the weight of the MNs should never exceed 5-10 % of the observed animal's body weight [1]. Sophisticated software architecture is key for energy-efficient on-board data collection and processing, thus allowing for the miniaturization of MNs. We propose a system that is suitable for automated tracking of social interactions of small, highly mobile animals in the wild. Our aim was to study social behavior of free ranging bats in a Panamanian tropical rainforest to demonstrate the performance of our system under real-world conditions.

In Panama, we tracked 38 individual bats from 3 species (fringe lipped bats (*Trachops cirrhosus*), common vampire bats (*Desmodus rotundus*) and pale spear-nosed bats (*Phyllostomus discolor*)). We chose these medium-sized bat species (25-45 g) because they show very distinct forms of dietary specializations, foraging strategies, and social organizations. These contrasting behavioral traits gave us the opportunity to test the functionality of our system in diverse conditions.

Authors Version - to be appear. EWSN'17,20–22 February 2017, Uppsala Sweden.

In this paper, we present our developed software infrastructure for MNs, which is tailored to the needs of the biologists. These nodes are composed of off-the-shelf components and are capable of documenting social interactions among free living bats fully automated and at high resolution. The system is suitable to observe free ranging bats that are as light as 20 g over time periods of 10-15 days. As the software interacts closely with the hardware, we also give insights of the used hardware and its parameters like clock drift and energy demand. These parameters are important for our software, as one of our goals is to save energy to achieve a long runtime for each node. With our collected data we discuss some of the most important parameters like the occurrence of different meeting durations and memory utilization which can be used as a baseline for further applications. Furthermore, packet losses while exchanging monitoring data are decreasing the coverage of our collected data during the runtime and is, therefore, an important value. We discuss our observed packet losses during the field tests as both good case and bad case scenarios.

In Section 2 we discuss the requirements from the biologist point of view, which also outlines the state of the art. Afterwards we give an overall view of how the system works in Section 3. As the software interacts very close with the hardware we introduce the used hardware in Section 4. Based on the previous sections, we present the software architecture in Section 5. We evaluated our system intensively during a 2-month lasting field test in the tropical rain forest in Panama which are discussed in Section 6. The related work is discussed in Section 7 and Section 8 concludes our paper.

2 Biological Requirements

Our goal to track animals and especially bats is because bats form the second largest group of mammals with more than 1000 species while the majority is living in groups. Group size may vary depending on the species from few individuals to several millions and social systems may vary strongly in their complexity [12]. Furthermore, bats play central roles in ecosystems by providing crucial services such as pollination, seed dispersal, and pest control [7], but at the same time many species are endangered [9]. Even though detailed knowledge on foraging strategies is scarce, there is increasing evidence that social interactions contribute to foraging success [16, 3]. More recently, bats also received negative publicity for being reservoirs of many emerging infectious diseases and being involved in spillovers to humans and livestock [2, 6]. Detailed knowledge about (social) behavior is crucial to understand the interdependencies and dynamics within bat populations and with their environment. This knowledge in turn is key to develop successful conservation strategies in order to preserve the valuable services provided to humans and to understand dynamics of infectious diseases to prevent future outbreaks [20].

State of the art in bat tracking is still radio telemetry because the weight of animal-borne sensor nodes is the most critical factor for the study of bats. These MNs are not supposed to weigh more than 5-10 % of the body weight of the study animal in order not to restrain the bats natural behavior [1]. Thus, a weight of 2 g or less is desired for the observation of medium to large-bodied bats.

A common way of observing gregarious animals is the use of Global Positioning System (GPS)-nodes and miniaturized loggers are available at a weight of 1 g. However, loggers need to be retrieved for data access and an option for remote download multiplies the weight [11]. A miniaturized 1.3 g version of the 'EncounterNet', an automated solution for encounter logging [17], would be light enough for the application in many bat species, but the tag miniaturization limits the runtime to less than 24h [13]. It is essential to observe a significant share of the individuals of a population over longer time periods at minimal levels of disturbance in order to draw population (or even species) wide conclusions from behavioral data.

Therefore, systems for automatic data collection of bats must fulfill the particularly challenging task of tracking interactions among group living bats.

- Smart energy management that allows data collection over a period of 1-2 weeks at high sampling rates and at a MN weight of < 2 g
- Remote data download which limits disturbances of the animals by the researchers to the tagging event thus allowing for observations on unbiased, natural behavior.
- Full automation of the system and operation of at least 30 MNs at a time, enabling the observation of entire social groups at a fraction of the costs of conventional methods.
- Two-way communication of MNs allowing for direct encounter logging creating dyadic data sets that form the basis of state-of-the-art social network analysis. On board collection and processing of accessory data (maximum Received Signal Strength Indicator (RSSI) and duration of encounters) that may be used to weight social networks.

Every encounter among two individual bats must include the Identifier (ID) of the participating bat, the start time and the duration of the encounter. Along with these basic data a max. RSSI value should be stored that can be used as a proxy for the minimum distance among the involved bats after careful calibration of the system [18]. A distance estimate is crucial to evaluate the biological meaning of an encounter (e.g., close (body-)contact vs. flying past). Similarly important, the activity, i.e., flying or roosting, of the encountered bat gives insights in whether the observed animals were actively hunting in a group or just sharing a roosting site. In addition to meeting data, the cumulated active time within a predefined time window gives the opportunity to study individual differences in behavior or so called "personality traits". The opportunity for remote data access at multiple foraging and roosting sites is crucial for data recovery since bats often cover a large area, are highly mobile and frequently switch roosting and foraging sites.

3 Scenario

In our system, we use an active MN which is attached to a bat and is able to store and forward the collected data to a Base Station (BS). This way all encounters among tagged individuals can be detected without deploying a large BS net-

work. Once a meeting is detected, its data can be stored on the microcontroller and be forwarded to a BS if the BS network comes into receiving range. The BS itself stores the raw data and may remotely be accessed for download by the biologist at any time thus keeping disturbances of the observed animal low.

Therefore, an almost unbiased behavior can be expected if the weight limits are met. However, this approach has the drawback that the MN becomes more complex compared to the state of the art of radio telemetry. Thus, our system is also prone to errors which is important, while errors in a deployed WSN are complicated to cure. This is even more problematic when monitoring animals while the system has been deployed. Catching a bat repeatedly may cause avoidance of a particular habitat and in consequence effect an altered behavior. Furthermore, due to the behavior of the animal, some parameters must be reconfigurable while the node is deployed since the behavior alters for each individual and cannot be known a priori.

3.1 Encounter Detection

Figure 1 shows the designated system with all components and is explained in the following. However, a more detailed explanation on our presented scenario can be found in the work by Dressler et. al. [5]. For simplicity, we assume that only two MNs are attached to two different bats with the IDs 1 and 2 in our scenario. To collect all necessary data, the exchange of their IDs is mandatory to know who met whom as the number of bats is not limited to two. For this reason, each individual MN is sending out their ID periodically. If two bats come into receiving range (approximately 10 meters), which is limited due to a small transmission power to save energy and the low sensitivity of the Wake-up Receiver (WuRx), we can assume that both bats are meeting each other, refer to Figure 1 ①. Once a beacon is received, the MN begins to update the data of a running meeting or to create a new meeting data structure.

In our scenario, the bat with the ID 2 flies to its tree roost and start to rest (Figure 1, ③). In this case, the temporal granularity of the data can be decreased in order to save energy. Doing this way, we assume that if a bat is resting, the information does not alter frequently thus a decreased granularity results in a minimal effect on the data.

3.2 Data Download

If a MN comes into the communication range, the stored data is transferred to the BS (Figure 1, ②). For this reason, each BS sends out a Base Station Beacon (BSB) to indicate that a BS is present. When such a BSB is received by a MN, all collected data is sent to the BS. In order to prevent collisions, the MN checks if another ongoing transmission is available (this information is encoded in the BSB since the BS can communicate in full-duplex). If an ongoing data transmission is detected, the MN does not send its data to the BS.

For further analysis it is mandatory to keep track on how long a bat spent its time in this area. Therefore, a Pseudo Localization Packet (PLP) is sent, independently to the acquired data, to the BS. This is important, as other data downloads of other MNs can prevent a data transmission of this particular

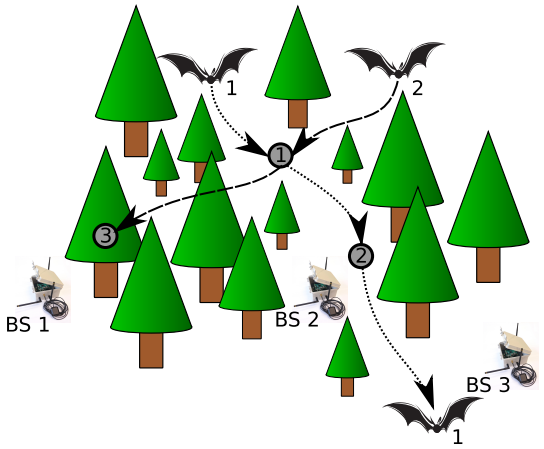

Figure 1. System overview, containing the base stations (BS 1 - BS 3) and two bats with attached sensor-nodes. Circle numbers representing points of interest which are important for the software design.

MN or if no data is available on the MN. Thus, it is assured that a bat is localized if it comes into a receiving range of the BS. During daytime, the biologist can remotely connect to the BSs in order to download the received data. The data is then processed offline with a decoder, which decodes the data and stores all received events in a MySQL data base. From this point on, the biologist is able to analyze the events acquired by the system.

4 Hardware Architecture

The biologist's requirements dictate a strict size, weight limit with a maximized runtime of the nodes. This also impacts the available energy of the circuitry as the battery constitutes a significant amount of the overall size and weight budget. Therefore, energy efficiency is of great importance as the battery capacity (55.5 mWh at a weight of 0.46 g) cannot be extended due to weight limits. For the hardware we chose components with a high energy efficiency and low quiescent currents that are as light weight as possible. The whole MN consists of an energy distribution system, different transceivers, microcontroller and an accelerometer which is depicted in Figure 2 and explained in the following.

In order to keep the nodes energy demand as low as possible, the energy distribution system is used to supply the most energy demanding devices with its minimal supply voltage. As the software follows a strict duty cycle, in that most of the time the transceiver can be switched off, the energy distribution system is partly turned off in order to save energy. The energy distribution system consists of a Direct Current (DC)-DC-converter TPS62261 from Texas Instruments and a Low-Dropout Regulator (LDO) ADP160 from Analog Devices. The LDO supplies all components with 2.1 V, with the exception of the main transceiver which runs at 1.8 V provided by the DC-DC-converter. However, in order to save energy the DC-DC converter is turned off during idle times. During this times, we power the main transceiver via the microcontroller to ensure a permanent power supply. Doing so, offers power savings determined by the ratio between the output voltage of the DC-DC-converter and the battery volt-

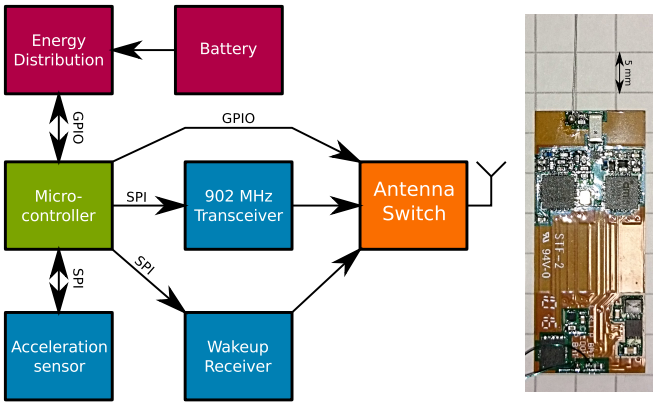

Figure 2. Graphical overview of the hardware and its connections between each module (left). Manufactured and assembled PCB of the Node (right).

age. The efficiency of the converter is more than 90 % at tens of mA which is the transceiver's active supply current. This results in a high efficient power supply for the transceiver as quiescent currents are minimized and a high efficiency is achieved during active times.

A low power accelerometer ADXL362 from Analog Devices is used to detect whether a bat is hanging upside down or not. This is important, as this devices delivers data on the activity state of the bat which is used by the biologists and it can be used to save energy. If a bat is hanging upside down, it can be assumed to be resting or sleeping and thus the systems functions may be executed at a reduced rate.

The communication between the MNs and the BSs is handled by a sub gigahertz front-end from Silicon Labs, the Si4460. Furthermore, a WuRx is used for the communication among the MNs. The WuRx allows a pure asynchronous communication and makes it obsolete to turn on the main transceiver in order to check if another node is in communication range. Therefore, the energy demand is decreased as the main transceiver can stay in idle for a longer period. The WuRx that is used is an AS3933 from *AMS* in conjunction with an envelope detector which makes the device suitable for the detection of signals in the 915 MHz band. However, for the communication to the BS, the WuRx is not sufficient as the sensitivity is too low as a bat can fly with a high velocity. Therefore, if a BSB is recognized late, a reliable data transmission cannot be ensured. In order to receive a BSB we exploit the higher sensitivity of the main transceiver which is turned on to receive data for a small amount of time. In order to use one single antenna for the WuRx and the main transceiver, we used the Radio Frequency (RF) switch SKY13350 from Skyworks Solutions.

All devices are controlled by a Cortex M0+ microcontroller KL05 from NXP which supports different interrupt priorities, small sleep currents and an internal Low-Power Oscillator (LPO). Interrupt priorities allow an event driven task execution. Between these events the microcontroller can be set into an efficient sleep mode that consumes only 2.5 μA. Additionally the energy efficient internal LPO, makes an external oscillator obsolete. This saves space and weight on the printed circuit board (PCB). However, using the LPO results in a high clock drift ($\pm 10\,\%$), [14]. As the desired runtime of a node is approximately 15 days, this clock drift results in a difference of 1.5 days at maximum per MN. Furthermore, the clock drift between two MNs can accumulate to a drift of $\pm 20\%$ which makes a drift compensation by the software necessary. Without countermeasures the clock drift will propagate into the timer modules of the microcontroller, leading to a drift in the estimated time in all MNs up to $\pm 10\,\%$ of its runtime.

The whole circuitry is assembled on a flexible substrate. This provides enough area for all components and allows for a compact setup of the node by folding the circuitry around the battery. The assembled PCB weighs only 0.55 g. The node is encapsulated by a 3D-printed housing adding another 0.58 g. So in total the node's weight adds up to 1.65 g including wiring the battery to the PCB.

5 Software Architecture

The software needs to fulfill several requirements of the biological and the electrical engineering part. This includes especially the low energy demand, a minimal set of data and also a robust data acquisition. To meet these requirements which may stay in conflict to each other, the software needs to implement a good trade-off between them. Our application uses the SLOTH operating system [8], which allows an energy efficient task switch. However, as the used Cortex-M0 provides only four different interrupt priorities, which are not enough for our application, we used a state-machine to dispatch more than four tasks. This gives us additionally the opportunity to port our software to other operating systems easily.

The general software layout is depicted in Figure 3 and shows the most important modules for our application. The main data path (marked with yellow arrows) begins with the ***WuRx*** and ends at the ***Base-Station Handler*** which directly interacts with the ***Main Transceiver***. Almost every module inside the data path is decoupled by First In First Outs (FIFOs)-buffers to allow an asynchronous operation of each module.

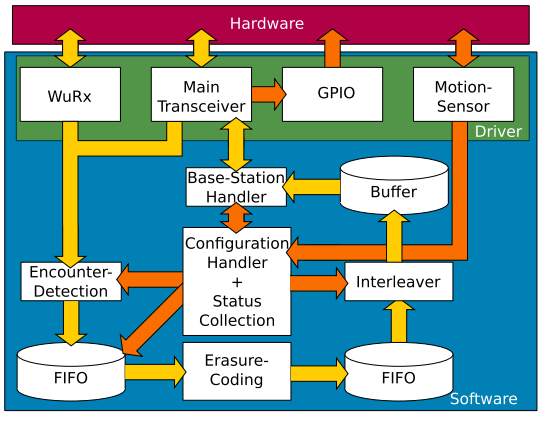

Figure 3. Graphical overview of the software and the data flow between all components.

Our software basically reacts on data packets, received by the transceiver, and wakeup-sequences received by the WuRx. An overview of all message types are depicted in

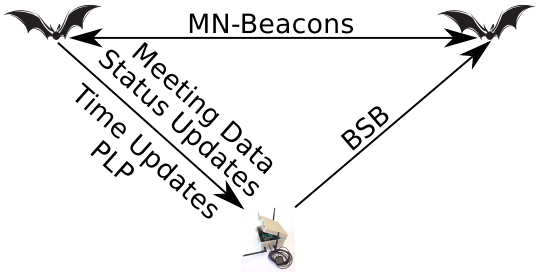

Figure 4. Graphical overview of exchanged data between mobile nodes and base stations.

Figure 4. We firstly describe how a meeting is generated and downloaded to a BS. Afterwards, we describe the other communication packets. In order to generate a meeting, our software requires a received MN-beacon. Each beacon consists of two parts, a wakeup sequence in order to wake up the microcontroller and a data packet which is sent with a small delay after the wake up sequence.

If two nodes come into a receiving range, an interrupt is generated by the WuRx in order to wake the microcontroller. As the WuRx is only able to trigger interrupts without receiving any data, the main transceiver is enabled to receive the ID and active state of the encountered bat. Afterwards, the ***Encounter Detection*** is invoked, which derives meeting information from this data.

Firstly, the ***Encounter Detection*** looks up if a meeting with the same identifier exists. If no meeting with the same identifier is available, a new dataset is created for this particular meeting. Otherwise, the meeting is updated. In order to prevent a high number of generated meetings, we accumulate subsequent beacons.

However, packet losses are common which prevents a successfully received beacon due to a multitude of influences like the antenna orientation or fading effects. Therefore, we implemented a timeout allowing several packets to be lost until a meeting is closed. As all MNs send their beacons in a predefined period, we can define a timeout which must be expired until a meeting is closed.

If the time out is exceeded in which no beacon has been received, we do a post processing of the data in order to keep the memory effort low. This in particular includes a compression of the meeting duration as we do not limit how long a meeting can last. In order to keep the error low, we successively adapt the granularity of the meeting durations to fit into a 16 Bit value. For example, if a meeting lasts longer than 1h, the resolution can be decreased to a granularity of 15 seconds. This leads to an error of less than eight seconds (0.2 %) of the meeting duration. After the post processing, the data is stored into a FIFO and the data acquisition is finished.

For downloading the stored data to the BS the data is encoded with an erasure coding [4] to increase the number of received meetings on the BS. For this purpose, the module ***Erasure-Coding*** is used which encodes two original packets to two redundant packets. Additionally to the erasure coding, we concatenate multiple packets to a so-called burst in the ***Interleaver*** module. As all packets in a burst is send in one transmission, only one message header and one transceiver start is needed. Together, this decreases the energy demand of the data transmission per packet from 1.8 mJ to 0.94 mJ.

The ***Base-Station Handler*** is the last module inside the data path. Its purpose is to detect whether a BS is available or not. After receiving a BSB (Figure 4), the ***Base-Station Handler*** decides upon the received RSSI whether a burst transmission should be performed or not. In order to allow a drift compensation, the ***Base-Station Handler*** sends a time update after a predefined interval. A time update packet (refer to Figure 4) contains the current local time of the MN which is transmitted directly (without any buffering inside the software). This way, the assumption: $T_{MN} = T_{BS}$ is met. The time T_{MN} represents the local timestamp of the MN and T_{BS} is the timestamp of the BS which is synchronized to the GPS time. Thus, the drift can be compensated while the data is decoded. We use a similar technique as presented in the work by Levin et. al. [13]. Besides, time updates, PLPs (Figure 4) are generated by the ***Base-Station Handler*** in order to indicate the presence of a bat independently to any running data transmissions. These PLPs are only sent when a BS is detected in order to keep the energy demand as low as possible.

Our software provides various configuration parameters like RSSI thresholds, timeouts and sample rates. These parameters highly influence the functionality of the system and must be set carefully. However, ideal parameters highly depend on the environment and the behavior of the bats that is not known a priori. Furthermore, exchanging parts of the software requires reprogramming of the microcontroller and as a consequence physical access to the node. Once a MN is deployed, catching the bats can potentially change their behavior and must be avoided at any circumstance. Therefore, a ***Configuration Handler*** is implemented, that allows to change parameters of the system without physical access to the nodes (refer to Figure 3, orange arrows).

All configuration updates are send to the MN as part of the BSB. Therefore, the configuration data is received by the ***Base-Station Handler*** and forwarded to the ***Configuration Handler***. In order to keep the length of a BSB low, the ***Configuration Handler*** uses a set of predefined configurations which are determined before deployment. However, upon a reconfiguration, the ***Configuration Handler*** checks whether an altered configuration will harm already collected data or any running data acquisition. Only when it is save, the configuration is updated. Furthermore, as the ***Configuration Handler*** keeps track of all modules, it is used to collect statistical data about the system like memory usage and activity times. This data is sent to the BS in form of status-updates and is used to check if the system is working correctly. If the system is misbehaving, this data may indicate which module is misbehaving, thus allowing us to correct the issue. For example, the RSSI thresholds for sending data to a BS are determined before deployment. However, the channel properties can alter dramatically, depending on where the BS is placed since not all channel characteristics can be acquired beforehand. Therefore, an adaption of the thresholds may become necessary after deployment of the nodes in order to keep the number of lost packets low.

6 Evaluation

We evaluated our system in the rain forest in Panama to check if our system is working correctly under real conditions. This evaluation lasted for more than two months in which biological relevant data was collected. In our evaluation, we ran six campaigns with a different amount of MNs and/or species which are depicted in Table 1. All campaigns delivered data, albeit only a subset of the campaigns (especially campaign 4 and 6, which covered a wide range of good and bad case scenarios) are discussed in our evaluation. As in our campaigns it is not possible to acquire reference data, our evaluation discusses the data without comparing the values to a baseline with the exception of Subsection 6.2.1.

The evaluation is structured into three subsections. In Subsection 6.1 the energy demand of our system is discussed. Subsection 6.2 discusses the acquired events and their properties. The following Subsection 6.3 discusses the error rates of our data transmission to estimate the quality of the acquired data. The last Subsection 6.4 presents the sources of errors we discovered during our field test and gives advices how to prevent such errors (if possible).

6.1 Energy Demand

The energy demand of our system is of particular importance as it dictates the achievable runtime of our nodes. However, an estimation of the energy demand in a fine grained manner would require the Analog Digital Converter (ADC) to sample the current and voltage over the time. The current of the ADC exceeds our measured idle-current by a factor of 170 ($I_{ADC} = 1.7\,mA$ Vs. $I_{sleep} = 10\,\mu A$). This makes an energy estimation while the node is deployed counterproductive. In order to provide comparable results, we present the worst case energy demand of our experiments. However, the estimated values are much higher compared to a real-world scenario.

The software runs multiple tasks in a predefined order until they repeat (cycle). The cycle time t_{cyc} can be adapted during the deployment and ranges from 2 s up to 255 s. Adapting the t_{cyc} changes only sleep times, as the number of executed tasks remains unchanged. In each cycle, a MN-beacon is sent and consumes $E_{TX_{ID}} = 208\,\mu J$. Furthermore, the node checks whether a BS is in communication range by turning on the main receiver for 5 ms ($E_{RX_{BSB}} = 171\,\mu J$). These tasks are executed in every cycle and can be summarized to $E_{Tasks} = E_{TX_{ID}} + E_{RX_{BSB}} = 379\,\mu J$.

At most once in a cycle, data is uploaded to the BS, which consumes $E_{TX_{DL}} = 115\,\mu J$. Furthermore, for each received MN-beacon, the receiver is turned on in order to receive data from the encountered bat and adds $E_{RX_{ID}} = 44\,\mu J$ to the energy demand. With a power consumption during sleep times $P_{sleep} = 37\,\mu W$ we can express the energy demand to:
$E_{Node}(n,b,t_{cyc}) = E_{Tasks} + t_{cyc} \cdot P_{sleep} + b \cdot E_{TX_{DL}} + n \cdot E_{RX_{ID}}$.
In this equation, the $n = [0,1,2,...,29]$ represents the number of simultaneous encounters during a cycle and $b = [0,1]$ indicates whether a BS is available or not.

This simplified formula, gives a rough estimate on the energy demand for one cycle. Software overheads for processing beacons or encoding data is negligible as the computations lasts in sum less than 0.5 ms. This in turn corresponds to an energy demand of $E_{comp} < 5\mu J$ at worst. Compared to the base energy demand of at least $E_{Node}(0,0,2) = 453\mu J$ the error would be less than 1%. During our campaigns we usually used the settings $t_{cyc,act} = 2\,s$ during active times and $t_{cyc,inact} = 10\,s$ while the bat is considered to be inactive. As the energy demand strongly depends on the behavior of the bats, we estimated the greatest possible energy demand, which can be reached for only short periods. We assume 29 nodes to be in communication range (the maximum supported by our system) and permanent presence of a BS. This translates into a maximum energy consumption per cycle of $E_{inactive} = E_{Node}(29,1,10) = 2.14mJ$ and $E_{active} = E_{Node}(29,1,2) = 1.844mJ$ for the aforementioned cycle times. The average power consumption during a cycle is given by $P_{inactive} = \frac{E_{inactive}}{t_{cyc,inact}} = 214\mu W$ and $P_{active} = \frac{E_{active}}{t_{cyc,act}} = 922\mu W$. Furthermore, we assume that a bat is resting for at least 16 hours a day and is active for 8h at the night. This results in an average current consumption of 450 μW per day ensuring a runtime of at least 5 days using the aforementioned battery. As this pessimistic calculation does not regard any generation rates of data within cycles, the power consumption is expected to be much lower in a real-world scenario. Therefore, we observed runtimes of 9 days or more.

6.2 Data Acquisition

In this subsection, we discuss the acquired data from our MNs. The first part checks the accuracy of our activity detection by comparing the accumulated activity to a baseline. We furthermore, analyze the occurrence of different meeting durations, which can be used for further applications. At the end of this subsection, we analyze the memory utilization of our MNs which depends on the individual bat.

6.2.1 Activity Detection

As described in Section 3, an activity detection is used to decrease the granularity of collected data while the bat is resting. Thus, activity detection is a critical element in our application, as an unreliable activity detection would decrease the entropy of the collected data. Therefore, we evaluated the activity detection via the collected data and a camera which has been installed in front of a flight cage (campaign 3). In this cage, we put an already tagged common vampire bat which has been filmed for three hours. Unlike other bat species we have monitored this species, as these bats are able to walk and to hang horizontal in a flight cage. Therefore, the flight-cage can be much smaller which also enables the use of a camera to check their activity.

The collected activity times for three hours are depicted in Table 2. We, furthermore, collected the event count, to get a rough estimation on how often the bat changed its state from active to inactive and vice versa. In this experiment, we got an accumulated error of 27 seconds (0.19 %), compared to our baseline (video analysis), which is acceptable for our application. The wrong activity detection is mainly induced by the fact, that the acceleration sensor assumes the bat as inactive when it is hanging upside down (within an angle of $\pm 37°$).

Within the last hour of our experiment, the bat spend approximately seven seconds in the active state. This was not

Table 1. Overview of all campaigns including important information like number of tagged individuals and anomalies, which took place in Panama.

Campaign	# Individuals	Species	Anomalies	Runtime
1.	2	*Trachops cirrhosus*	Bats left roosts	15 days
2.	1+4	1 *Desmodus rotundus* 4 *Trachops cirrhosus*	Two species	Roost left after 9 days
3.	1	*Desmodus rotundus*	Flight cage observation	Captured in Campaign 2
4.	15	*Desmodus rotundus*	-	13 days
5.	9	*Trachops cirrhosus*	-	12 days
6.	6+2	*Phyllostomus discolor*	Tagging on two days	Aborted after 4 days

detected by our node as it showed an activity time of zero seconds. A closer inspection of the bat's behavior showed repeated changes from horizontal to vertical hanging from the ceiling. In this particular case, a bad timing can result in a false activity detection. This is, because the software samples multiple acceleration values to switch to the active/inactive state. If six samples show the same state, a switch of the software state is performed. In this case, the bat was active for approximately seven seconds and hung upside down for small periods. Thus, it is feasible that at least one event was sampled as inactive and as a consequence no switch to the active state was performed and eventually not recognized by our software. To prevent repeated switches to active mode when the bats show moderate movements, e.g., grooming or shaking, during resting the timeouts are reconfigurable. Therefore, an adaption to the behavior of the bats can be performed to increase / decrease the granularity of these events.

6.2.2 Meeting Durations

For further applications, it becomes important to analyze the commonness of the durations for each meeting. These values can be used to decrease the amount of data to be sent as field lengths for meeting durations can be adapted.

Figure 5 shows in sum two histograms for campaign 6 and 4. Both campaigns show a significant amount of meetings that last zero seconds. In our implementation, this means that only one beacon was received from another bat. In such cases, the average RSSI value was -44.97 dBm, which is close to the minimal RSSI value of the WuRx. Therefore, small changes of the antenna orientation may cause a huge difference of the received signal strength which causes a too low signal strength at the WuRx. Since individual bats constantly move and jiggle inside the roost, meetings may be frequently interrupted.

In our datasets we observed that the bats were active for 51.15 % in the 4th campaign and 59.22 % in the 6th campaign. This, in sum, indicates a high rate of generated meetings due to variations in the antenna alignment among two animals. As the active and inactive state does not dominate the other we also analyzed the distribution depending on the state. For both cases, the distribution of the meetings stayed the same, therefore, any further optimization can be done for active and inactive states.

Table 2. Comparison of the accumulated activity time of a mobile node to a video captured baseline.

hour [h]	duration (video) [MM:SS]	events	duration (tag) [MM:SS]	dif. [s]
1	2:28	4	2:34	6
2	1:26	3	1:12	14
3	0:07	1	0:00	7
acc	4:01	8	3:46	27

In the work, presented by Levin et. al. [13], it has been mentioned that a meeting between two MNs does not necessarily be the same on both MNs. This issue, occurred also in our field tests. Table 3 shows an excerpt of our acquired events where both recorded meetings are interrupted. If all duration values are accumulated from Table 3, we also discovered that both duration values are equivalent and differ only in small values. In this example, the accumulated meeting duration of node ID 4 is 26,820 seconds whereas the accumulated duration of node ID 2 is 27,915 seconds which results in a difference of 4.08 %. This result confirms the observation, presented by Levin et. al. [13]. Therefore, a post processing stage of this data is required to ensure a correct interpretation of the data by joining both recorded meetings.

However, in further applications, it becomes handy to limit the maximum meeting duration. If a long running meeting lasts multiple hours the importance of this event may increase. However, if a meeting lasts multiple hours, this event is only observable after the meeting has ended and also the likelihood increases to lose this event due to a bat leaving the BS coverage area, damaged nodes, etc. Therefore, meetings in further applications should be split until a meeting reaches the maximum meeting duration. However, as our data shows, a limited meeting duration bears the danger to increase memory utilization which exceeds the available memory. From our data, we propose a time limit of 40 minutes as more than 90 % of all meetings fit into this range and does not increase the memory utilization further. Albeit, this limit, a mechanism must be implemented to merge already split meetings because the impact of the remaining 10 % is very high.

Table 3. Excerpt of the decoded data, showing interrupted meetings which are not necessarily equal albeit both nodes collected the data under the same conditions.

Bat	Enc. Bat	Start [HH:MM:SS]	Duration [s]
4	2	11:15:48	17100
4	2	16:10:24	1620
4	2	16:39:49	8100
2	4	11:04:38	945
2	4	11:21:55	1770
2	4	11:52:48	15300
2	4	16:13:17	9900

For example, we observed meetings which lasts more than 56700 seconds. Storing such meetings in chunks of 40 minutes would result in a 24 times higher memory effort. Judging on the data we have collected the introduction of a meeting time limit of 40 minutes would result in a 120 % higher memory demand in average for our system. This shows the importance of a merging algorithm which must be considered in further applications.

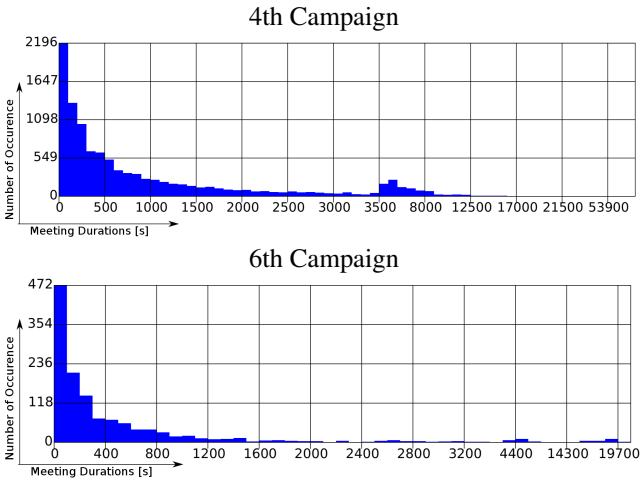

Figure 5. Histograms of the meeting durations of all meetings for each campaign

6.2.3 Memory Utilization

As described in the previous sections, we also collected system statistics, which allows a better understanding of the internal processing of each node. One of the most important values is the maximum memory utilization, as it leads to data losses if the amount of acquired data exceed the available memory. Therefore, this behavior should be avoided since it decreases the coverage of our collected data. In order to keep the communication effort and in turn the energy demand low, we estimated the maximum memory utilization within an hour. This data is stored locally and sent to the BSs together with the collected meeting data.

Figure 6 shows an excerpt of one node of the 6th campaign. The graph shows the maximum memory utilization (red line). When a bat comes into receiving range of a BS, a heat map is shown in the background. Otherwise a white background is used. In order to get an estimation of the communication effort, the transmitted data is displayed as a yellow bar. As there might be packet losses causing that no memory utilization can be displayed for this particular period, a coverage bar is displayed above each diagram. It shows the time slot when a memory utilization can be displayed (green) or not (red).

As depicted in our graph, the memory utilization increases while the bat is absent from any BS. In this time the MN accumulates meetings with other bats. When a bat comes into the transmission range of a BS, the acquired data is downloaded to the BS and the memory can be freed afterwards. This leads to a decreased memory utilization. However, at the beginning of this graph no data or at least only small amount of data is transmitted. In this case, the estimated RSSI at the receiver is very low, thus, the minimal threshold for sending data is not exceeded on the MN and no data is transmitted. The shown RSSI values are estimated at the BS of the received PLP, thus only the received RSSI of the transceiver is displayed. Therefore, the displayed RSSI can be used as a rough estimation of the RSSI estimated on the MN. Besides this, Figure 6 shows a high communication effort while the bat is within the communication range of the BS. Since new meetings can be generated while a BS is present, the downloaded data can exceed the stored data. This results in a behavior, where all generated data is downloaded to the BS immediately if at least two events were acquired.

Besides staying at the roost nearby a BS, we also frequently observed time periods where bats were absent of a BS for approximately 20h or longer if individuals switch between roosts during the observation. During this times the memory utilization increases dramatically with different slopes of the curve. These different slopes are caused by a different amount of meeting data, generated within an hour. As Figure 6 shows, the maximum memory utilization did not exceed 519 Bytes, which is compared to the main memory of 1848 low. However, with increasing periods away from a BS, the memory usage will further increase and highly depends on the behavior of the bat.

Figure 7 shows three different individuals of the same campaign (6th campaign). In this graph, only the RSSI, memory loads and the coverage of the system statistics, as described at the beginning of this subsection, are displayed. Similar to the first bat (Figure 6), different periods of the presence and absence of the BS can be observed. Summarizing Figure 7 and Figure 6 all four bats show a different behavior in terms of time spent in the roost. For example, the node ID 8 left the roost rarely and for a short time. Therefore, a small memory utilization can be observed which is lower than 200 Bytes over almost the whole runtime. In contrast, node ID 2 is only rarely in transmission range of any BS thus a high memory utilization can be observed.

As all individuals were the same species under the same weather conditions, we can assume that each bat behaves differently. The behavior of an animal is affected by the gender, individual personality and age to name some of the possible parameters which might affect the memory utilization. Thus, the memory utilization highly depends on the behavior of the tracked bat.

In the 4th campaign, we also discovered data losses due to buffer overflows. Figure 8 shows two individuals with a high memory utilization. The graphs shows an excerpt of the memory utilization of one node, where the utilization exceeds the available memory. Especially, ID 27 suffers from a too low data rate to the BS and the memory utilization increases even if data is downloaded. In the middle of the graph, it can be seen that the memory utilization is near the available memory. Thus, data losses occurred due to the lack of free memory. This in turn leads to a bad coverage of status updates since data cannot be stored for later transmission. Also ID 6 suffers from this issue. Even though if this bat was in the receiving range of a BS, the memory utilization remained high.

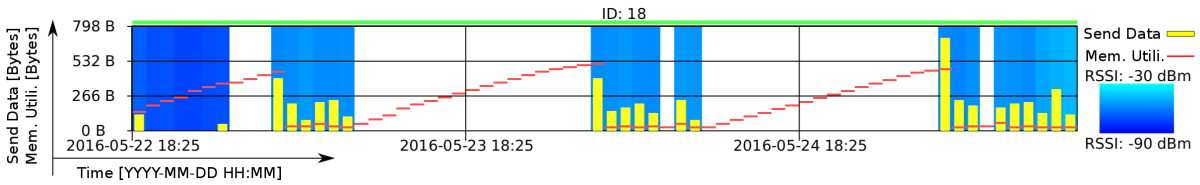

Figure 6. Graphical overview of the memory utilization, send data and the average received signal strength estimated at the base station of one node.

To conclude this section, there is no general advice for dimensioning the memory size. It depends on the expected behavior of this particular individual, which is not known a-priori and may need a reconfiguration of the system to keep the memory demand of the collected information as low as possible. During our field test we observed four times a data loss due to a too small memory which lasts in sum (all campaigns) approximately 65 hours. The majority of the time is caused by one node (Figure 8, ID 27) with approximately 53h. Compared to the (pessimistic) estimated overall runtime of all nodes (5,766 hours) a coverage of about 98.90 % has been achieved during our field tests. Therefore, the memory was enough in a retrospective view on our field test. However, to prevent such cases a reconfiguration would be more effective instead of increasing the memory as the reconfiguration can be used to decrease the receiving range of the node and in turn a decreased rate of generated meetings. Furthermore, an increased memory also shows a higher energy demand compared to a smaller one and as a consequence may reduce the runtime of the node. In addition, an increased data rate for sending data to the BS should be favored to decrease the memory utilization especially if no BSs can be placed inside a roost.

6.3 Packet-Error Rate

In our field test, we deployed the BSs in several spots where a bat might go foraging (1st campaign). We also placed some BS inside the roost to increase the likelihood of collecting all data from each node. From our received packets (in sum 44,870), we derived a minimal number of sent packets which sums up to 56,088 packets.

This corresponds to a packet loss of 20 % which is mainly caused by multipath propagation and interference which was an expected packet loss rate.

During our field test, we also discovered a weather dependent multipath propagation inside the rain forest, which results in a highly reduced communication range. On one day, we were able to communicate up to 10 meters between two MNs. After a rainy morning, on the next day, the communication were only inside an area of three meters reliable and small changes of one node in the range of two centimeters could cause a complete loss of all data. As the change is within the range of $< 10 \cdot \lambda$, we can assume that multipath propagation is the main root of error. However, we expected a weather dependent channel albeit without a big impact.

Besides the multipath propagation, we also discovered that the antenna orientation of the BS has a high influence on the packet error rate. As explained in Section 5, the MN estimates the RSSI and decides on this value if a packet should be send. In order to allow a full duplex communication of the BS, two different antennas are used. One for receiving (RX) and the other one to transmit (TX) the BSB to the MN. As the TX-Side of the BS crosstalks into the RX-Side, we arranged both antennas to be perpendicular to each other. Using this configuration has the benefit that the crosstalk is highly reduced. However, there is always an antenna configuration where the TX-antenna has the best orientation to the antenna of the MN. This results in a higher estimated RSSI at the MN when a bat receives a beacon of the BS. The perpendicular orientation of the MN antenna to the RX-antenna of the BS results in a loss of signal strength of around 30 dB on the receiver side of the BS.

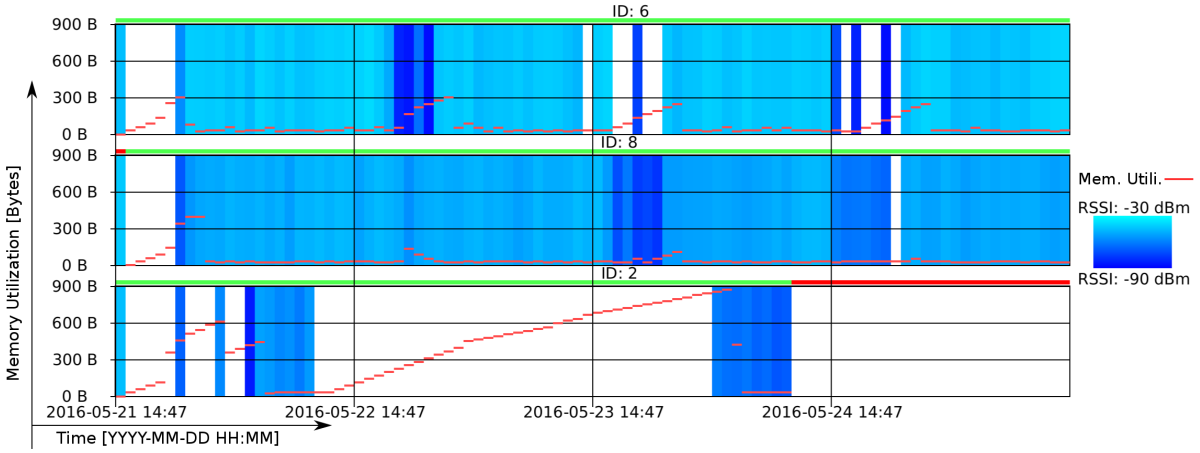

Figure 7. Graphical overview of the memory utilization of three nodes and the average received signal strength estimated at the base station.

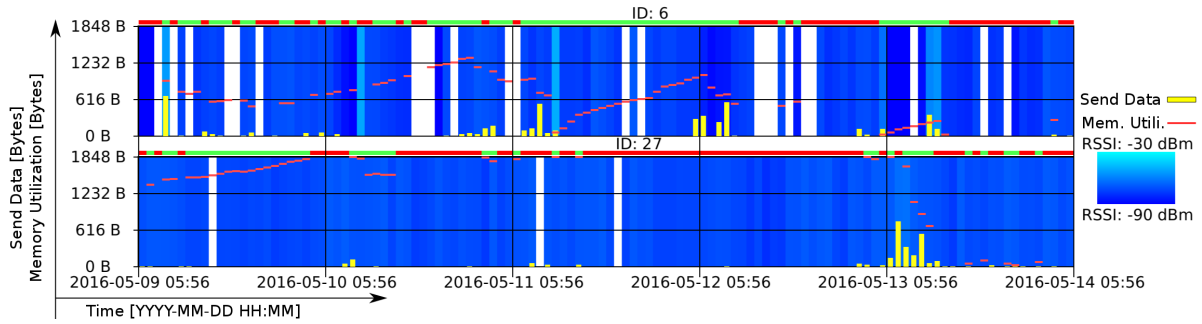

Figure 8. Graphical overview of the memory utilization of two different nodes which is very high, therefore, these nodes are not able to acquire more data.

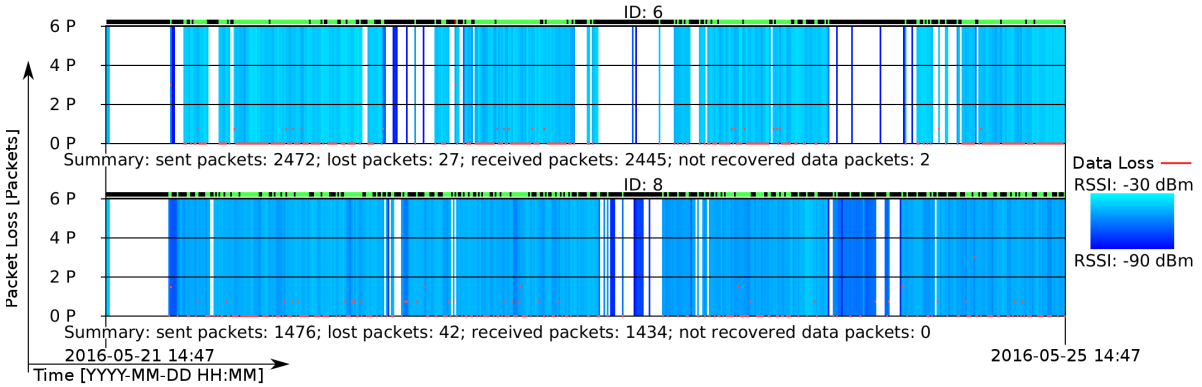

Figure 9. Graphical overview of the packet loss of two different nodes and the average received signal strength estimated at the base station.

As the threshold on the MN to send its data is set to -60 dBm this can cause the received signal at the BS to be weaker than -90 dBm which is roughly the sensitivity level of the receiver on the BS. Furthermore, in our measurements a RSSI also translates to an average distance of 10 meters between MN and BS for our chosen threshold. Decreasing this threshold further would decrease the average distance dramatically. For example, decreasing the threshold to a value of -50 dBm, would result in an average distance between MN and BS to two meters. As the bats are flying we also want to cover a bigger area because of the movement. Therefore, the -60 dBm is in our case the best trade-off between coverage and packet loss.

Similar to the memory utilization the packet loss is also an indicator of the coverage of our data during the runtime. Therefore, we estimated the packet losses over the time for our campaigns. The packet loss in Figure 9 is displayed by the red line. We compute the packet loss from the packet counter difference between two subsequent packets which is accumulated within 10 minutes. We also determine the average RSSI of the PLPs within this time frame which is displayed as a heat map in the background. If no beacon is received, a white background is displayed. Furthermore, a bar above each diagram shows whether a data transmission took place within this time frame (red/green) or not (black). If a data transmission was successfully, e.g. all events are successfully decoded a green bar otherwise a red bar is shown.

In our chosen 6th campaign, two individuals were picked from the datasets for our good-case analysis and is depicted in Figure 9. In sum 96 packets were lost in a uniform distribution which resulted in two not recoverable/restorable events. The overall packet loss rate is 1,74 % which is quite low in comparison to the average of 20 % indicating a good coverage of our collected data for those nodes. In general a relative high RSSI were observed during the data transmission for all nodes with this small packet error rate. In contrast to this good-case we also observed a relatively high packet loss in the 4th campaign. Figure 10 shows one MN

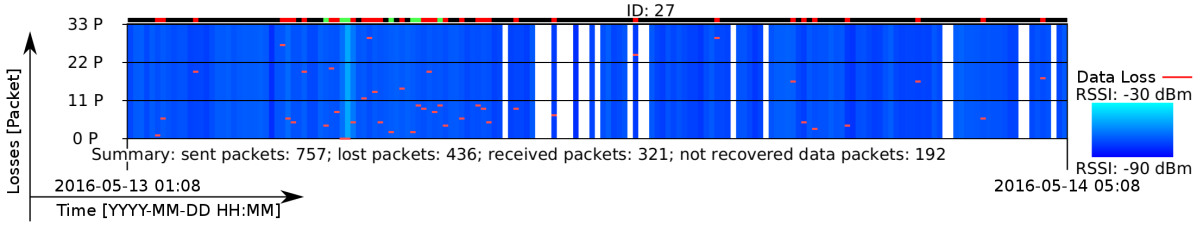

Figure 10. Graphical overview of the packet losses of one node, which lost approximately all data during two days albeit it worked fine before and after these two days.

where more than a half of the packets were lost within 1.8 days. This particular MN worked properly until the 5th day of our campaign. After two days with high packet error rate the number of lost packets returned to normal values again. This behavior has been observed on three more individuals of this campaign and is the reason, why the overall packet loss is as high as 20 %. However, the effective data loss of events is less than 20 % due to the erasure-code and also due to the fact that two nodes are recording a meeting between each other.

For this scenario we can exclude weather conditions as the reason since all nodes showed a different behavior among 10 days. Furthermore, the periods with high packet losses ranged between several hours up to two days and all cases have in common that the RSSI is relatively low. This indicates either a bad antenna orientation between MN and BS or a bad matched high frequency network on the MN. The latter can be caused by excrements on the antenna or by water condensation inside the housing. A bad antenna orientation is also possible, as in this particular campaign we tracked the *Desmodus rotundus*. Unlike *Trachops cirrhosus* which usually hangs vertically from the ceiling, *Desmodus rotundus* may adopt posture that strongly derive from a vertical position by leaning onto the inner walls of a hollow tree. This will result in a less parallel antenna orientation among MNs compared to the vertically roosting *Trachops cirrhosus*. Up to this point we assume, that both aforementioned cases are the cause for the increased packet loss.

6.4 Lessons Learned

During our field test we faced several problems, which should be shared at this point.

Antenna Orientation: We discovered, that the antenna orientation on the BS was not optimal for our tests. As the transmitter antenna and the receiving antenna of the BS is perpendicular to each other, there exists always an antenna setup of the MN, where the RSSI of the receiving antenna to the BS is up to 30 dB higher as it should be. The other way round, this decreases the rate of sent data, as the antenna orientation of the transmitter antenna to the MN causes a reduced RSSI value on the MN. This can lead to not meet the RSSI thresholds for sending data albeit the receiving antenna would have a way better RSSI value. To tackle this problem, a circular polarized antenna, where the loss would be 3 dB, can be used. Alternatively, an increased distance, for instance by using cables, between TX and RX-antenna is also a feasible solution. All in all, a proper antenna design is important, especially for the BSs.

Clock Drift Compensation: The clock drift of the MNs also made it necessary to implement a drift compensation for each node. This is implemented in the decoder which is invoked after a data download from the BS. During our field test, we found out that some MNs performed a restart. This made it hard to estimate the correct time for each node as each node uses a time stamp relative to its startup. Furthermore, a packet with the actual time of the MNs were sent like a normal PLP. As this type of message do not use any RSSI thresholds only a small subset of sent time packets was received. Therefore, the drift compensation was not able to work as desired and some drifts could not be compensated. In future designs, this issue is fixed as we send this time stamp with a RSSI threshold similar to the thresholds to send a data packet. Doing this way, the number of time updates will increase which in turn improves our drift compensation.

Data Decoding: The use of synchronized BSs by GPS turned out to be not as reliable as expected, resulting in overlapping timestamps for receiving multiple packets from one node. This makes it impossible to decode the received data, since we use a non unique packet counter were used to minimize the amount of data for each transmission. In extensive unit-tests, the recovery from a non unique packet counter worked perfectly, however, in the real environments it turned out to be prone to errors. Therefore, a unique packet counter, will be implemented in future systems.

Isolation and Fixing the Battery: Additionally, we also encountered a problem in isolating and fixing the battery into the housing during the 2nd campaign and in previous measurements. The latter led into broken wires inside the housing due to the movement of the bat. Battery isolation is also a critical task, as the small currents of the node can be exceeded by a cross current through the isolation material. The humidity inside the rain forest can also increase the cross flow current if the housing is not sufficient sealed, resulting in a reduced runtime of the node.

Data Losses: Another root of errors lies in the behavior of the animals itself, as they are highly unpredictable. This results in lost MNs due to a bat which left the roost, and in turn the collected data was also lost. For example, a *Desmodus rotundus* did not leave its roost after been caught in the 4th campaign. Individuals of *Trachops cirrhosus* were more likely to switch their roosting sites after the disturbance caused by the initial catching and tagging event. This resulted in a pair of bats which left the roost, during the first campaign, after we tagged them. In order to find these bats, we placed multiple BSs on places which were a promising foraging area inside the forest. We were able to find their foraging habitats on the first try, but the second roost remains undiscovered up to now. Therefore, a download of the data was only possible while the bat was foraging, thus spending only small amount of time within the receiving range of a BS. This scenario also shows the importance on a higher data rate for downloading data from the MN to the BS.

7 Related Work

Extensive work has been undertaken to develop and improve animal tracking technologies. Ossi et al. [15] gives an overview of proximity sensor and GPS-based telemetry systems. The weight of the tags is, besides the runtime of the nodes, a major parameter for our field tests. Therefore, we discuss systems of comparable weight to the proposed system in more detail in the following.

The Camazotz [10] platform contains several sensors to collect accessory data on-board. However, more hardware comes along with an increase in energy demand and weight. The Camazotz nodes weigh 30 g, which is a multiple of 18 of the weight of our proposed system.

In order to ensure prolonged runtime, a solar panel is used to recharge the battery. Using a solar panel is not an option in our system, as the tracked bats are night-active and are hiding

in a roost during daytime. One of the used sensors is a GPS-module to track the movement of individuals. In order to decrease the energy-demand of the module, an accelerometer is used to decrease the energy demand by adapting the duty-cycle of the GPS-module. However, with this technique, Camazotz still has an average power consumption of 5.7 mW which is 12.6 times higher than our worst case estimation.

The closest work to our system is EncounterNet [13]. This work also uses beacons to identify who met whom and does not rely on solar panels or GPS. EncounterNet uses smaller sensor-nodes (1.3g) at the cost of a short runtime of less than one day. Our presented system weights slightly more (1.65g) but it comes with a much larger runtime of up to 15 days. The observation of longer time periods gives deeper insights into the plasticity of individual behavior and creates more representative data sets.

Besides these works, Dressler et. al. [4] analyzed the same scenario of moving bats and the data transmission to BSs. This work presented erasure codes to add redundancy to the original data resulting in an increased likelihood of successful received data. In their work, different methods were analyzed with regard to the energy demand and packet loss. This method is to our knowledge the most energy efficient way to send data at a high receiving probability ($> 90\,\%$ of sent data have been recovered) and is, therefore, used in our application.

8 Conclusion

Observing animal behavior, and especially bats, is important for improved epidemic forecasts and preventions. However, mapping social interactions among multiple individuals over longer time periods is not feasible without applying sophisticated tracking technology. To increase the coverage of observable social interactions by simultaneously keeping the disturbance of the animals to a minimum, we developed our presented system.

The key features of this system are the light-weight animal-borne mobile node and the energy aware software. The mobile node collects data about encounters between two or more individuals. This data is afterwards automatically downloaded to base stations, thus reducing the disturbances. As parameters like memory utilization or minimal received signal strength indicator thresholds cannot be known a priori we provide a subsystem to reconfigure the nodes after deployment.

We introduced the hardware and the general software layout. In our evaluation, which took place in Panama and lasted more than 2 months, our system was extensively tested and evaluated under real conditions. The results show, that our system works well for the target application.

9 Acknowledgments

We thank R. Page, G. Carter, M. Nowak, C. Toth, I. Geipel, L. Fernandez, J. Nowatzki and M. Smeekes for their great support during field work and N. Neumann for reviewing the *Desmodus rotundus* videos.

This work was partly supported by the German Research Foundation (DFG) under grant no. FOR 1508 and grant no. KA 3171/3-2.

10 References

[1] S. Amelon, D. Dalton, J. Millspaugh, and S. Wolf. Radiotelemetry: Techniques and analysis. In T. Kunz and S. Parson, editors, *Ecological and behavioral methods for the study of bats*, chapter 3. The Johns Hopkins University Press, Baltimore, 2009.

[2] C. H. Calisher, J. E. Childs, H. E. Field, K. V. Holmes, and T. Schountz. Bats: important reservoir hosts of emerging viruses. *Clinical microbiology reviews*, 19, 2006.

[3] N. Cvikel, K. E. Berg, E. Levin, E. Hurme, I. Borissov, A. Boonman, E. Amichai, and Y. Yovel. Bats aggregate to improve prey search but might be impaired when their density becomes too high. *Current Biology*, 25, 2015.

[4] F. Dressler, M. Mutschlechner, B. Li, R. Kapitza, S. Ripperger, C. Eibel, B. Herzog, T. Hönig, and W. Schröder-Preikschat. Monitoring Bats in the Wild: On Using Erasure Codes for Energy-Efficient Wireless Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 12, 2016.

[5] F. Dressler, S. Ripperger, M. Hierold, T. Nowak, C. Eibel, B. Cassens, F. Mayer, K. Meyer-Wegener, and A. Koelpin. From Radio Telemetry to Ultra-Low Power Sensor Networks - Tracking Bats in the Wild. *IEEE Communications Magazine*, 2015.

[6] J. F. Drexler, V. M. Corman, M. A. Müller, G. D. Maganga, P. Vallo, T. Binger, F. Gloza-Rausch, V. M. Cottontail, A. Rasche, S. Yordanov, et al. Bats host major mammalian paramyxoviruses. *Nature communications*, 3, 2012.

[7] S. J. Ghanem and C. C. Voigt. Increasing awareness of ecosystem services provided by bats. *Advances in the Study of Behavior*, 44, 2012.

[8] W. Hofer, D. Lohmann, and W. Schröder-Preikschat. Sleepy Sloth: Threads as Interrupts as Threads. In L. Almeida and S. Brandt, editors, *Proceedings of the 32nd IEEE Real-Time Systems Symposium (RTSS 2011)*, pages 67–77, Los Alamitos, CA, USA, 2011.

[9] K. E. Jones, A. Purvis, and J. L. Gittleman. Biological correlates of extinction risk in bats. *The American Naturalist*, 161, 2003.

[10] R. Jurdak, P. Sommer, B. Kusy, N. Kottege, C. Crossman, A. Mckeown, and D. Westcott. Camazotz: Multimodal activity-based gps sampling. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13. ACM, 2013.

[11] R. Kays, M. C. Crofoot, W. Jetz, and M. Wikelski. Terrestrial animal tracking as an eye on life and planet. *Science*, 348, 2015.

[12] G. Kerth. Causes and consequences of sociality in bats. *Bioscience*, 58, 2008.

[13] I. I. Levin, D. M. Zonana, J. M. Burt, and R. J. Safran. Performance of encounternet tags: field tests of miniaturized proximity loggers for use on small birds. *PloS one*, 10, 2015.

[14] NXP. *Datasheet: Kinetis KL05 32 KB Flash*, 2014. KL05P48M48SF1, Rev 4 03/2014.

[15] F. Ossi, S. Focardi, G. P. Picco, A. Murphy, D. Molteni, B. Tolhurst, N. Giannini, J.-M. Gaillard, and F. Cagnacci. Understanding and georeferencing animal contacts: proximity sensor networks integrated with gps-based telemetry. *Animal Biotelemetry*, 4(1):21, 2016.

[16] M. T. O'Mara, D. K. Dechmann, and R. A. Page. Frugivorous bats evaluate the quality of social information when choosing novel foods. *Behavioral Ecology*, 2014.

[17] C. Rutz, Z. T. Burns, R. James, S. M. Ismar, J. Burt, B. Otis, J. Bowen, and J. J. St Clair. Automated mapping of social networks in wild birds. *Current Biology*, 22, 2012.

[18] C. Rutz, M. B. Morrissey, Z. T. Burns, J. Burt, B. Otis, J. J. St Clair, and R. James. Calibrating animal-borne proximity loggers. *Methods in Ecology and Evolution*, 6, 2015.

[19] P. Sommer, J. Liu, K. Zhao, B. Kusy, R. Jurdak, A. McKeown, and D. Westcott. Information bang for the energy buck: Towards energy- and mobility-aware tracking. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, EWSN '16, pages 193–204, 2016.

[20] H. Whitehead. *Analyzing animal societies: quantitative methods for vertebrate social analysis*. University of Chicago Press, 2008.

[21] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware design experiences in zebranet. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 227–238. ACM, 2004.