Seal-2-Seal: A Delay-Tolerant Protocol for Contact Logging in Wildlife Monitoring Sensor Networks

Anders Lindgren, Cecilia Mascolo
Computer Laboratory
University of Cambridge
{anders.lindgren, cecilia.mascolo}@cl.cam.ac.uk

Mike Lonergan, Bernie McConnell
Sea Mammal Research Unit
University of St Andrews
{mel5, bm8}@st-andrews.ac.uk

Abstract

Sensor networks are now enabling the monitoring of various environmental phenomena with more accuracy than the previous labour intensive and less technological solutions. This paper is concerned with the application of opportunistic networking techniques to wildlife monitoring, where the sensors are attached to animals moving in their habitat. We present Seal-2-Seal, a novel protocol for logging of node (i.e., animal) contacts in mobile networks and for dissemination of that information to sinks for further analysis. The protocol utilises an efficient data summary mechanism to reduce the amount of information that needs to be transmitted, thus reducing energy consumption. To evaluate the performance of the protocol, we implemented it for the Contiki operating system on sensor devices and ran simulations based on real-life mobility traces using the Cooja emulator.

1. Introduction

Recent developments in the area of mobile and sensor networks have generated interest in the application of these technologies to various domains. In particular, protocols and applications for intermittently connected networks offer interesting solutions to problems in wildlife monitoring. In projects related to wildlife, researchers are interested in knowing how animals interact with each other to gain a better understanding of the behaviour of the species. This information can also be used to address specific problems such as how the behaviour of the animals might contribute to the spreading of diseases or to track animal survival.

Our target scenario is to model the social contact patterns of grey seal (*Halichoerus grypus*) pups in the UK within their first year. There is an increasing awareness that the social contact patterns within a population of animals have an important bearing on the spread and fate of disease outbreaks.The motivation behind this paper is to devise a system that can collect and relay ashore contact pattern data from a large *sample* of animals that would then allow us to model the contact patterns of the *population* of animals. To estimate population parameters with precision we need to tag a large number of animals. Thus there is a requirement for the tags to be cheap, as well as being small. Most previous marine mammal telemetry studies have collected data about the movements and behaviour of individual animals either through retrieval of tags (often impractical for many species) or by transmitting the data directly ashore. The later strategy may use Argos satellite modems [2] or GSM modems [8]. However the costs of these systems are high and this limits the number of animals that may be tagged.

We propose a cheap and light-weight solution, by using nodes attached to animals to exchange data with each other. This means that most of the animals can be tagged with cheap and energy-efficient devices that are only capable of local communication. Through this communication system, data can then be delivered to sinks, which are points from where the researchers can collect the data easily. A sink can either be a fixed node somewhere in the environment, or one of the animals that in addition to the local communication interface also has some other long haul communication possibility (such as a satellite link). Since only a small fraction of the animals need to be tagged with these more complex and expensive nodes, it is possible to tag a larger sample of animals.

In this paper, we present Seal-2-Seal (S2S), a protocol for efficient logging and collection of encounters between mobile nodes (i.e., the tagged animals). The protocol has been devised with very specific application constraints in mind and is specifically tuned to contact data dissemination rather than other type of data, which allow a precise optimization of its performance. Seal-2-Seal allows contact data to be disseminated to sinks over an intermittently connected network (e.g., a network of animal contacts) in an energy efficient way. The protocol utilises a novel mechanism to create efficient data summaries to reduce the amount of

978-1-4244-2575-4/08/$20.00

data that has to be transmitted by the nodes. We evaluate the protocol using mobility traces generated from real observations of animal mobility.

2. The Seal-2-Seal Protocol

In this section we describe the Seal-2-Seal (S2S) protocol. S2S is a protocol for logging contacts between mobile nodes, and in a delay-tolerant manner transmitting information about such contacts hop by hop to one of the sinks.
The protocol consists of two phases:

- In the *contact detection phase*, nodes periodically attempt to detect the presence of other nodes within transmission range and log such contacts.
- When a new contact is detected, the *data dissemination phase* is initiated, in which neighbouring nodes exchange information about the currently stored contacts and synchronise the contacts between the two nodes. The data dissemination phase has two parts:
  - First, data summaries are exchanged between the two nodes. These allow the nodes to determine which contacts are stored at the other node, and thus, which contacts need to be transmitted. The data summaries also contain information about which contacts have been delivered to a sink and acknowledged. This information can then be used to purge acknowledged contacts from the network to avoid unnecessary extra transmissions of them.
  - Finally, the actual data are exchanged between the two nodes.

2.1. Contact Detection Phase

In the contact detection phase, a node needs to discover other nodes within transmission range (here, we choose to do it through local broadcasts of beacons, but other methods can be used if available). The beaconing interval needs to be selected carefully, as too frequent beacons will lead to many unnecessary transmissions which consume energy, which often is a scarce resource. Too infrequent beaconing, on the other hand, risks missing short contacts. The beaconing interval should be set based on the expected duration of contacts in the target environment, while also taking into consideration the available energy and required lifetime of the nodes.

Contact Logging

The basic unit of data stored in nodes is a contact entry, consisting of the tuple $(id_1, id_2, t_{start}, dur)$, where id_1 and id_2 are node identifiers. The tuple indicates that id_1 and id_2 had a contact of duration dur that started at time t_{start}. Two

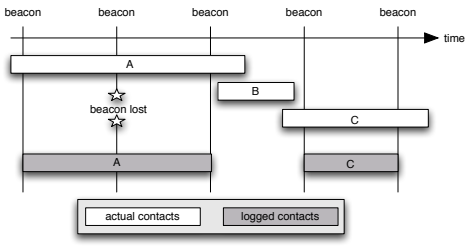


Figure 1. Example of contact logging.

contact events (receiving a message from a certain node or being notified by an underlying neighbour discovery service) are considered to be part of the same contact if they are less than $3 \cdot t_{beaconinterval}$ apart. Thus, if a contact event occurs within this time period, no new entry is added to the contact data base, but instead the duration of the last entry for that node pair is updated to reflect the new length of the contact. Figure 1 illustrates how this works through an example. A node is over time in contact with three other nodes – A, B, and C. During the contacts with nodes A and C, the presence of the node is detected more than one time through the transmission of beacons, but only one entry for each contact is stored. As the maximum time allowed between two contact events to still count them as being in the same contact is greater than a single beacon interval, the contact with A is logged as a single contact even though one of the beacons is lost in transmission. The contact with node B on the other hand illustrates the tradeoff involved in choosing the beaconing interval. In this example, the beaconing interval was longer than the actual contact, and thus the entire contact happened between two beacon transmission and could not be logged. Shorter beaconing intervals reduce the risk of missing contacts, but will also increase the amount of data that has to be sent, and thus the energy consumption of the nodes.

2.2. Data Dissemination Phase

We now describe the two steps of the data dissemination: the exchange of data summaries and the actual exchange of data leading to data synchronization between the two nodes.

Data Summaries

When a node has detected another node, it needs to be able to determine which data the other node has and what needs to be sent. In protocols such as Epidemic Routing [13], such message synchronisation is achieved by exchanging lists of identifiers of all the messages carried by the nodes. This allows a node to determine which messages are not present in the buffer of the other node and send those messages. The overhead of such a method is acceptable if the size of the message identifiers is small in comparison

to the size of the data messages themselves. In a protocol such as S2S on the other hand, each data entry is small, and a unique identifier for such an entry would not be much smaller than the data itself. Therefore it is not viable to transmit a list of all entries stored, so the data must be summarised using a more compact representation. Through our knowledge of the type of data being stored, and by imposing requirements on the way data is stored and transferred, we can create a very compact summary of all contacts stored at a node. The size of the summary does not depend on the number of contacts stored, but only on the number of node pairs for which contacts are stored. To enable this compact representation of the data, we require that the data for each node pair is stored and transmitted in chronological order. Thus, a particular data entry is never sent to a node unless that node already has all the earlier entries for that node pair (or if all earlier entries are also included in the same packet as this entry).

Since we now have some more knowledge about the data stored at each node, it is sufficient to add to the data summary a tuple $(id_1, id_2, first_{start}, last_{end})$ for each node pair. The value $first_{start}$ is the start time of the earliest contact for that node pair that is stored, and $last_{end}$ is the end time of the latest contact for that node pair that is stored. Because we require nodes to exchange information about contacts in a chronological order within each node pair, we can guarantee that a node that transmits such a report not only has the first and last contact as listed above, but also has all contacts that have been logged in between them.

Upon reception of a data summary message (which might have been spread over multiple network level packets; acknowledgements are sent to ensure reliable delivery of summaries), a node can now determine what data is stored at the other node, and what data needs to be transmitted. This node also creates a *summary diff* message, which is constructed exactly like a data summary message, but to reduce the amount of data that has to be transmitted, this message only sends the differences between the data stored in both nodes, which will result in a smaller message if large parts of the stored data is the same at both nodes.

Data Exchange

After receiving a summary or summary diff messages, the nodes now create one or more packets (depending on the amount of data that has to be transmitted and the allowable packet size on the underlying network) with the entries that are not present at the other node. How to select which data entries to include in packets can be chosen by the implementer, as long as the requirement of chronological transmissions is followed (in the implementation used for the evaluations in this paper, entries are selected sequentially). A node **must not** send any other entry other than the one with the lowest timestamp that the receiving node does not already have. As this entry has been added to the packet, the node may now add the next entry in sequence for that node pair, or an entry from another node pair (for which the same requirements of timestamp ordering applies). Data can now be sent between the nodes. To ensure that data is not lost, the protocol uses acknowledgements to be able to retransmit lost packets. If both nodes have data to send, acknowledgements can be piggybacked to data messages, but if one node sends more packets, the other will have to generate separate acknowledgement packets.

Acknowledgements and Data Purging

On small sensor devices, memory can be a limited resource. Therefore it is important to reduce the amount of data that has be kept in the network as much as possible. Storing less data also reduces the amount of data transmissions needed, so data aggregation also implies energy savings. If nodes come in contact with sink nodes and delivers the data, such data can be safely purged from the network. Because of the way the data summaries are created, this is easily achieved. As data are transferred to a sink and deleted from the transmitting node, the data entries with the smallest time stamps are the first to be deleted. Therefore, the $first_{start}$ part of that node pair's entry in the data summary will also be increased. When a data summary is received, nodes can delete all entries with time prior to the first time reported in the data summary. As the data is deleted on this node, data summaries created by it will also be affected, and thus, the acknowledgement will spread through the network, purging all delivered data from nodes.

3. Evaluation

We have implemented Seal-2-Seal in the open source, highly portable, multi-tasking operating system for memory-constrained networked embedded systems, Contiki [3]. Small-scale tests have been performed on the Tmote Sky sensor platform [9] to verify that the protocol works on real hardware, and larger scale deployments are planned for the future. To evaluate the performance of S2S in larger scenarios, we used the Cooja simulator [11] on which we run the real Contiki code but with the benefit of being able to perform larger-scale testing.

In order to emulate mobility of the sensors, we use a mobility trace generator developed in the ZebraNet project, that allows us to emulate zebra mobility [5]. We simulate a 20 node network during a simulated time of around 31.5 hours. All nodes in the network run the S2S protocol, and some nodes (a varying number) also act as data sinks (there are no stationary sinks). For the mobile sinks (e.g., the animals with a GSM link installed too), we assume that their uplink is instantaneous so that data can be acknowledged as soon as it has been sent to the mobile sinks. We study the performance of the protocol for a number of different beacon intervals and data sink ratios. The nodes that act as

sinks were selected randomly.

For comparison purposes, we have also included a simple *Direct Transmit* (DT) protocol and a *ZebraNet* protocol in the evaluations. In Direct Transmit, nodes log contacts as in S2S, but they never forward them to other nodes. Only when a sink is met will the nodes offload all their contacts to that sink. Such a protocol will have low overhead, as each data item is only sent at most once, but is on the other hand likely to suffer from worse performance with regard to other metrics. The ZebraNet protocol is an epidemic dissemination protocol as used in the ZebraNet project and creates data packets of contacts when needed and uses these as the unit of synchronisation.

To determine the performance of the protocol, we study the following metrics. We study the *total contact time logged* as that shows how accurately the protocol is able to detect contacts as the beacon interval varies. The *percentage of contacts reported to a sink* and *delay to sink* are the main performance metrics, indicating how efficiently the protocol can disseminate the data to the sink. As energy is a scarce resource in the target environment for S2S, the power consumption should be kept low. Thus, it is of interest to keep the amount of data transmitted as low as possible. We measure the *communication overhead* as the average number of bytes transmitted by the nodes during the experiment. Finally, we look at the *maximum buffer usage in the nodes*, which allows us to evaluate the memory requirements of the protocol, and also study how that is affected by varying number of sinks available to acknowledge the data.

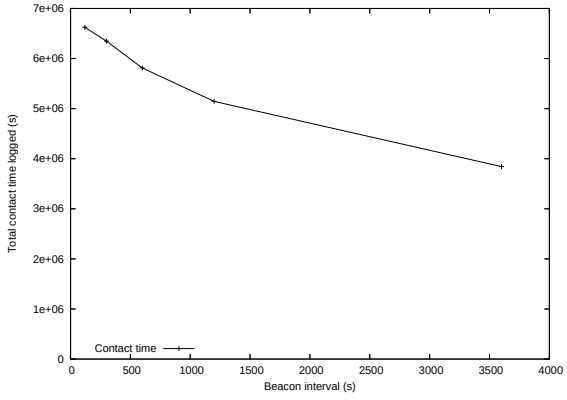


Figure 2. Total contact time logged by any node for different beacon intervals.

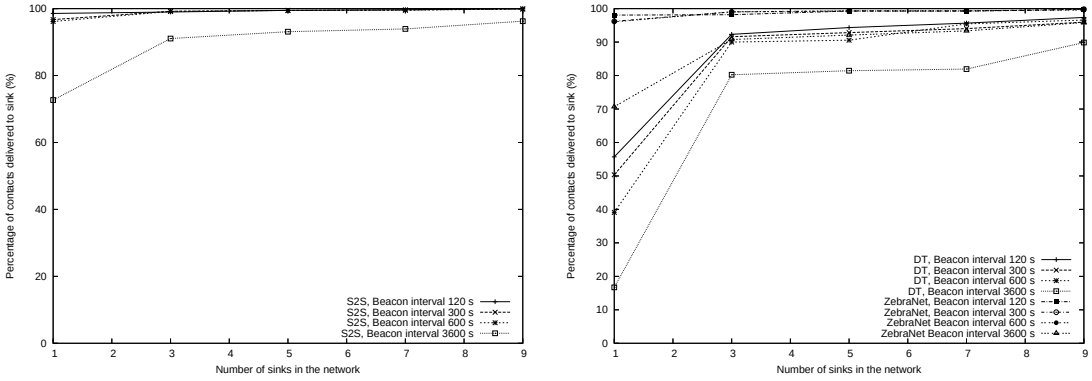
3.1. Results

We start the evaluation of the protocol by looking at how much of the contacts that are present in the system can be successfully logged by the protocol, and how that is affected by the different parameters.

This can be seen in Fig. 2, where the total contact time logged by *any* node for different beacon intervals is shown. As expected, a shorter beacon interval results in a larger portion of the contacts being logged. For a beacon interval of 120 seconds, almost all the contact time is detected, but as the beacon interval increases, the logged contact time drops.

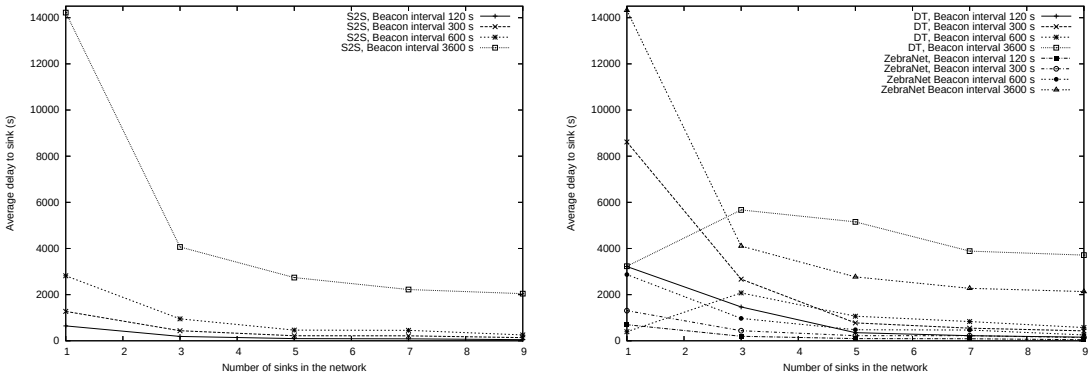
In Fig. 3, we show how many of the logged contacts make it to a sink. This is an encouraging result. Even if only a very small part of the node population are sinks, Seal-2-Seal is still able to deliver most of the logged contacts to a sink where they can be used. Direct Transmission requires more sinks to achieve a good delivery ratio, and never quite reaches the same level as S2S. Not only is more of the data disseminated to the sink by S2S as compared to DT, but is also delivered in a more timely manner as shown in Fig. 4. S2S yield significant reductions in the average time it takes a contact to be delivered to a sink as compared to only making direct transmissions to the sink (the reason that the delay for DT is very low with beacon intervals of 600 and 3600 seconds with just one sink is that only a small number of contacts are delivered to the sink, and many of these are contacts between the mobile nodes and the sink, which thus has zero delay, reducing the average delay). For both of these metrics, the gains are extra evident when there are only a small number of sinks in the system. This showcases the benefit S2S has in reducing the required number of sinks in the system to achieve the desired performance. Thus, the cost of deployments is reduced and larger populations can be studied. In terms of delivery and delay, ZebraNet exhibits similar performance to S2S as it is an epidemic protocol, so good performance is to be expected as buffer and bandwidth is not limiting here (due to the higher overhead, the delay is also slightly higher as more data must be transmitted).

In the simulations, nodes had infinite buffers for storing contacts, but in real deployments, a proper buffer size must be selected. Figure 5 shows the maximum number of contacts that was stored at a single node at any point of the simulation. This indicate the smallest possible buffer size that can be used without having to discard data entries. This shows one of the aspects of the usefulness of the acknowledgements from the sinks, as we can see that with more sinks continuously acknowledging the data received, and allowing it to be removed from the other nodes, the buffer size requirements decrease. However, even with only a few sinks, the buffer requirements are quite small, and in this scenario, a buffer of 10 kB would be fully sufficient. It is also good to see that the buffer requirements do not vary that much depending on the beacon interval. For very long beacon intervals, the buffer is naturally kept small as



(a) Seal-2-Seal (b) Direct Transmission and ZebraNet

Figure 3. Percentage of the logged contacts that were reported back to a sink.



(a) Seal-2-Seal (b) Direct Transmission and ZebraNet

Figure 4. Delay to sink. Average delay for a contact to be reported back to a sink.

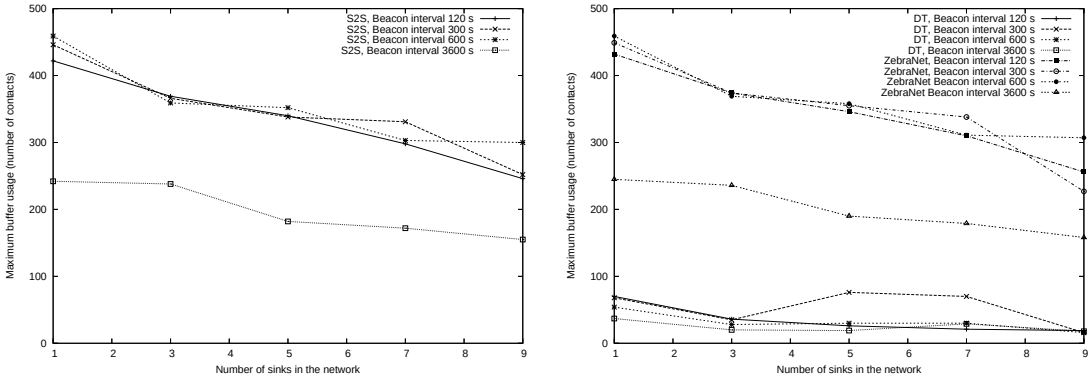
only a small number of contacts is ever logged. For short to intermediate beacon intervals, the buffer usage has less variance. While very short beacon intervals is likely to generate more detected contacts, it also provides more communication opportunities for contacts to be sent to the sinks (and thus acknowledged and removed from the buffer).

The benefit of the sinks' ability to acknowledge data that subsequently can be purged from the network, can be even more clearly seen in Fig. 6, which shows the average number of bytes that each node sends during the entire experiment. As more sinks are introduced into the system, the amount of data to be sent between nodes drops quite rapidly. This drop is most remarkable for short beacon intervals, where the interactions are frequent and thus larger amounts of data are sent, but it is also relevant when longer intervals are used. The amount of data transmissions is the strong point of DT, as each data entry is only sent at most once (when meeting a sink), and the curves here show the low levels of transmissions. As data is only sent when meeting a sink, the data sent by DT increases as the number of sinks increase. When looking at the data transferred, we can here see that S2S is more efficient than ZebraNet as it sends less data while achieving the same delivery rate at comparable delays.

4. Related Work

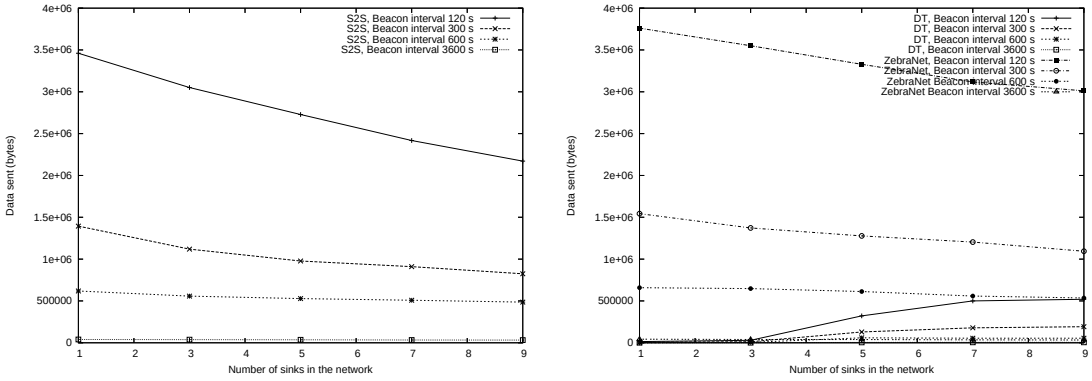
Contact Logging

There has been a number of measurement experiments performed in order to log mobility and contact patterns between mobile nodes. While it is possible to use infrastructure-based data to infer contacts between nodes, such measurements will not be able to log contacts that occur out of range of the infrastructure [1]. Thus, contact gathering will be biased towards certain areas and not a biologically good sample in many cases. This is particularly problematic in situations where the network is expected to be sparse and dispersed over a large geographical area or in



(a) Seal-2-Seal (b) Direct Transmission and ZebraNet

Figure 5. Maximum buffer usage by nodes.



(a) Seal-2-Seal (b) Direct Transmission and ZebraNet

Figure 6. Communication overhead. Average amount of data transmitted per node.

harsh environments, such as in wildlife monitoring applications. Therefore, a number of measurement studies have been conducted in which direct contacts between nodes have been logged. The Haggle project has performed several such data collection deployments in which iMotes were given to university students or conference attendees [1, 6]. In the RealityMining project, smartphones logging information about Bluetooth contacts were deployed to students and staff at MIT during a nine month period [4].

The target of the protocol presented in this paper is similar to that of the contact logging experiments described above, but has some important differences that makes it more flexible and efficient. A big problem with previous deployments of mobile contact loggers has been the need to collect all the nodes at the end of the experiment in order to retrieve the data. In many cases, this is not possible. If nodes are given to humans, nodes might be lost or fail during the experiment, or the logging software might have been installed on nodes that belong to the users (such as mobile phones), which the user is not willing to give away for data collection. This problem is even more prevalent in wildlife monitoring applications. In such applications, it can often be difficult or impossible to recapture the same animals that have initially been tagged, and data loggers might frequently be expected to fall off the animals after a certain period of time, making it impossible to retrieve those devices. Thus, there is a need for a mechanism for detecting and logging node contacts that also transfer that data between nodes to spread it through the network until it reaches a sink where it can be retrieved by the user.

Opportunistic Data Dissemination

There have been many protocols proposed for data dissemination in intermittently connected network. Such protocols try to deliver application messages to their destinations either by spreading them through the entire network as in Epidemic Routing [13], or by using different methods to reduce the overhead in the network while still delivering the messages [7, 10].These protocols make no assumptions

about the data sent through the network. This means that they can be used in most scenarios, but also that they are not making use of knowledge that could make operation more efficient.

Wildlife Monitoring

Related sensor networking techniques have been used for wildlife monitoring in other projects. The projects that is most closely related to ours are the ZebraNet [5] and wildCENSE [12] projects. In these projects, zebras and the Indian Nilgari antelope are tagged with GPS receivers to log the position of the animals. The tags also contain communication hardware that is then used to send all collected data using standard epidemic forwarding protocols over the wireless network to the researchers that study the animals. While there are similarities between those projects, and the anticipated use of Seal-2-Seal, there are also important differences. Since S2S focuses on the logging of animal contacts, and not of other data, we were able to perform optimizations in the way the data was aggregated and stored, which have not been considered in the other projects. Instead, these projects, focusing on larger animals which may carry bigger batteries, rely on more traditional data dissemination protocols such as the ones outlined above, which instead are optimized in terms of bulk data transfer. S2S considers each node contact to be the smallest data unit, and thus can do more efficient and flexible synchronizations between peering nodes. Moreover, by not incorporating GPS receivers, nodes can be smaller and consume less energy, properties that are especially beneficial when doing deployments on smaller animals. This also reduces the cost of nodes, which makes larger deployments possible.

5. Conclusions and Future Work

We have presented Seal-2-Seal, a dissemination protocol for contact data gathering through sparse intermittently connected networks. Our wildlife scenario of animal tracking and the peculiarity of the contact data to be collected have influenced a number of protocol decisions and have indicated a number of possible optimizations.

As the protocol is light-weight, has low requirements on the system it is running on, and only needs long-haul communication capabilities in a limited number of instances, it enables deployments to a larger population than before at the same cost.

We have described an implementation of the protocol over sensor nodes and presented a large scale evaluation over a simulator using real mobility traces.

In terms of future work, we want to combine the protocol with some application aware duty cycling techniques and to deploy the protocol on animals: this will allow the zoologists to gather the needed data and us to evaluate further the capabilities of the protocol in real life.

Acknowledgements: We would like to acknowledge the support of EPSRC through the project WILDSENSING.

References

[1] A. Chaintreau, P. Hui, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. Technical Report UCAM-CL-TR-617c, University of Cambridge, Computer Lab, February 2005.

[2] CLS Argos, Toulouse, France. *Guide to the Argos System*, 1989.

[3] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, Nov. 2004.

[4] N. Eagle and A. Pentland. Reality mining: Sensing complex social systems. *Journal of Personal and Ubiquitous Computing*, September 2005.

[5] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proceedings of Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), San Jose, CA*, October 2002.

[6] J. Leguay, A. Lindgren, T. Friedman, J. W. Scott, and J. Crowcroft. Opportunistic content distribution in an urban setting. In *Proceedings of the ACM SIGCOMM workshop on Challenged Networks (CHANTS 2006)*, September 2006.

[7] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. In *Proceedings of The First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR 2004)*, August 2004.

[8] B. McConnell, E. Bryant, C. Hunter, P. Lovell, and A. Hall. Phoning home - a new GSM mobile phone telemetry system to collect mark-recapture data. *Marine Mammal Science*, (20):274–283, 2004.

[9] Moteiv Corporation. *Tmote Sky Datasheet*, November 2006.

[10] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks. In *Proceedings of the IEEE 6th International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM 2005). Taormina, Italy.* IEEE press, June 2005.

[11] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, Nov. 2006.

[12] P. Ranjan, P. K. Saraswat, A. Kumar, S. Polana, and A. Singh. wildcense – sensor network for wildlife monitoring. Technical report, Dhirubhai Ambani Institute of Information and Communication Technology, May 2006.

[13] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, April 2000.