INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

CHARON: Convergent Hybrid-Replication Approach to Routing in Opportunistic Networks

Efficient Collection Routing for Low-Density Mobile WSNs

Jorge Miguel Dias de Almeida Rodrigues Soares

Resumo Alargado da Dissertação para obtenção do Grau de Mestre em Engenharia de Redes de Comunicações

Júri

Presidente:	Prof. Doutor Rui Jorge Morais Tomaz Valadas
Orientador:	Prof. Doutor Rui Manuel Rodrigues Rocha
Vogal:	Prof. Doutor Luis Filipe Lourenço Bernardo

Setembro de 2009

Abstract

Combining wireless sensor networks (WSNs) with delay-tolerant networking (DTN) has the potential to extend their use in a multitude of previously impossible applications. However, and despite numerous proposed solutions, there is still wide debate as to how to best route messages in these networks and, more importantly, how to do it in an energy-efficient way.

This paper proposes CHARON (Convergent Hybrid-replication Approach to Routing in Opportunistic Networks), an approach that focuses on maximizing efficiency in addition to delivery statistics. CHARON uses delay as a routing metric, and provides basic QoS mechanisms, with both a quasi-single-copy mode for general traffic and a multi-copy mode for urgent data. It also integrates time synchronization and radio power management mechanisms. Simulation results show that it is able to achieve good delivery statistics with lower overhead than comparable solutions.

Keywords

Wireless Sensor Networks, Opportunistic Communications, Routing Protocols, Opportunistic Routing, Delay-Tolerant Networks, Energy Efficiency

Notice

An earlier version of this paper has been submitted to the IFIP Wireless Days ’09 conference, and is currently under review.

1 Introduction

Wireless sensor networks (WSNs) have been slowly moving into the mainstream as valid solutions to real-world monitoring problems. These problems have arisen in a multitude of fields, including industry, health, agriculture and environmental sensing. WSNs generally work by periodically collecting data over long time spans and relaying it to a reception point (the sink) where it is analyzed.

Certain applications, such as wildlife monitoring, feature implicit mobility, as some or all sensory nodes on the network are attached to mobile elements. Other applications require the deployment of sensor nodes over large areas, usually with low node density, making it expensive or unpractical to maintain full connectivity by the traditional means of adding nodes or sinks. One way to achieve connectivity, although not in a time-continuous way, is by using mobile nodes to carry the data. Networks like these are called opportunistic, as communication only takes place when an opportunity comes up, i.e., when nodes are in range. They are also delay tolerant, as it can take an arbitrary amount of time for a message to reach the destination.

Routing in opportunistic networks presents additional challenges when compared to more traditional networks. Not only does it have to deal with an additional variable (time) but also, in most scenarios, with no knowledge of the future network evolution. Decisions must thus be made based on random criteria or on analysis of the contact history.

There have been numerous approaches to opportunistic routing. One of the earliest is Epidemic Routing [1], wherein each message is replicated to every node encountered, effectively spreading it over the entire network. Although for low data rates this mechanism provides good results in terms of delivery statistics, it is very inefficient. Spray and Wait [2] is another flooding-based approach, although it is a case of controlled flooding, in the sense that it limits the number of copies. Data MULEs [3] introduce a more efficient three-tiered architecture, in which special nodes (MULEs) move around collecting data from sensors and delivering it to sink nodes. History-based protocols, such as PROPHET [4] and SCAR [5], are the most popular and work by routing messages according to the history of previous encounters, though the specific methods vary.

While all existing solutions have their strong points, there is ample room for improvement. Most were not designed with WSNs in mind, but with conventional ad-hoc computer networks, in which the communication paradigm is very different. Efficiency is also frequently forgotten, and some algorithms depend on generally unavailable information or engage in much higher complexity than is feasible with the scarce resources typical of WSNs. Furthermore, real-life deployments can often yield good results using the simplest protocols. One example is ZebraNet [6], a wildlife monitoring network that routes messages using a very simple historic analysis: an integer counter is periodically increased if the node is

in range of the sink or decreased otherwise. A node forwards its messages every time it encounters another with a higher score.

We have developed a new approach, CHARON (Convergent Hybrid-replication Approach to Routing in Opportunistic Networks), which aims to be a simple yet efficient solution to the problem of routing messages in sparse mobile WSNs. It aims to minimize the number of message exchanges, while still providing a way for urgent messages to be delivered quickly. It also integrates time synchronization and radio power management, features seldom found but of critical importance in the achievement of energy efficiency.

We assume a scenario in which most or all nodes are resource-constrained and mobile, and as such make no distinction between them. Networks are also assumed to be sparse (node density is low, and contacts are somewhat infrequent), be highly mobile, and have stochastic evolution (the future topology can't be reliably predicted). We aim for the most common scenario in WSNs, in which the objective is to collect data from sensors. For that reason, our approach is *convergecast* (data flows from nodes to a point of collection, the sink) on an *any-sink* paradigm (more than one such collection point can be present, and delivery to any one of them is sufficient).

The remainder of this article is organized as follows: in Chapter 2 we discuss the design choices for our approach; in Chapter 3 we provide a brief overview of the prototype implementation; in Chapter 4 we show some evaluation results; finally, in Chapter 5 we draw some conclusions and finish by suggesting some future work.

2 CHARON

Our general goal is to transport data from its generating node to one of the sinks, via other sibling nodes. To do this, a node has to decide, locally, if a neighbour it meets is a better carrier than itself. This is done by using a *routing metric* or *score*. Nodes periodically broadcast beacon messages that contain the information needed to calculate their score. When a node receives a beacon broadcast by a better carrier, it starts to transfer the currently held messages.

2.1 Delay as a Routing Metric

The main routing metric used in CHARON is delay, as previously proposed in other contexts [7]. The basic idea is to forward messages from nodes with higher to nodes with lower estimated delivery delay (EDD). Sink nodes have an EDD of 0. From node A's perspective, the *perceived delay* through a node B is the sum of node B's EDD and the period of contacts between them. We call the latter factor inter-contact time (ICT), and it is determined by an exponentially weighted moving average (EWMA) of the interval between each of the previous contacts. A node v's EDD is the lowest of the estimated delays through all known nodes (the set V_v), per equation (1).

$$edd(v) = \min_{c \in V_v}\{edd(c) + ict(v,c)\} \quad (1)$$

It is easier to visualize the mechanism by using a graph, as in Figure 1. Graph nodes correspond to network nodes, with the number in the centre being the EDD. Edges correspond to "known node" relationships, and are marked with the recorded ICT. A node's EDD is equivalent to its shortest-path weight to the virtual sink, representing all real sinks.

There is a problem with this mechanism, as a node's ICT is only updated when a contact takes place. This leads to a situation in which a node's EDD does not degrade even if other nodes are not met for long periods. While there are several possible solutions, we have chosen to penalize nodes whose last contact time is older than the ICT by adding the overrun, slightly modifying (1).

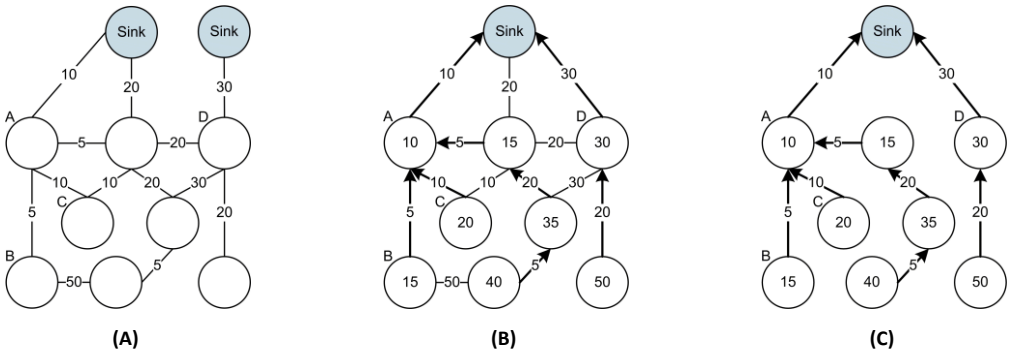


FIGURE 1: STEPS OF EDD CALCULATION FROM ICT VALUES

2.2 Utility Function

To compute a node's final routing score we are not, however, restricted to using delay alone. CHARON uses a customizable multi-variable routing metric that can be tuned to the specific scenario according to the obtainable information. Battery level and free buffer space are useful and widely-available indicators but others can be used, including application-specific ones. They can be combined in any number of ways, from simple addition to the use of individual scaling functions. To prevent loops, the system never routes a message to a node with higher EDD, regardless of its score under this scheme. This way we guarantee we're not wasting energy in forwarding a message further away from the sink.

2.3 Message Replication

Most opportunistic routing approaches use a multi-copy mechanism, meaning messages may be replicated while on the network. Increasing the number of message copies tends to increase delivery ratio and decrease average latency, as it provides some tolerance against sub-optimal routing choices. It also has two negative effects: increase of communication overhead (number of transmissions per message) and significant performance degradation under high load scenarios, when buffers become full and messages have to be dropped.

With routing efficiency in mind, we opted for a primarily single-copy approach, meaning that there is at most one single copy of each message in the network. While this choice carries some consequences, as stated above, we believe it to be satisfactory for most messages in our target scenarios. Individual messages are typically non-critical and if lost don't impact the application significantly. Nevertheless, some copying is inevitable, as when a node forwards a message it still keeps a local copy. On a pure single-copy system, this message would be deleted. However, keeping it has zero energy cost, as the message is already in the buffer. There is an apparent memory cost, but that is avoided simply by deleting these messages anytime a regular one is received. We choose to keep these messages, and call them *zombies*. Even though they are in every way similar to the original message, they cannot be forwarded, and are only used for direct delivery to a sink, if one is met. Although there really are multiple copies of each message, the algorithm behaves in a single-copy manner, hence the name *hybrid replication*.

We have supposed, so far, that collected data isn't urgent or critical. Although this is true for the vast majority of messages, there may be some situations in which we want a specific message to arrive to the sink quickly, even at the cost of lower efficiency. Imagine, for instance, a precision agriculture setting, in which a network monitors environmental parameters and plant health. The first kind of data is presumably non-urgent but we may want to quickly notify operators when a plague is detected. For this reason, we have decided to support both usages at once, providing the aforementioned single-copy mode and a multi-copy mode similar to PROPHET's. In this second mode, messages are forwarded to better carriers, but the local copy is kept as an original, instead of being turned into a zombie. This allows the node to forward the message again if it comes in contact with another good carrier, clearly improving the delivery indicators. We work on the assumption that the number of messages using this multi-copy mechanism is a small fraction of the total.

2.4 QoS Classes

As mentioned above, not all data on a sensor network has the same requirements – some of it may be urgent, while most is usually very delay-tolerant. In this context, we define quality of service (QoS) in the broadest way: providing different priorities for different traffic classes. In effect, we are still providing best-effort service to all classes, but letting the user choose the goal, e.g. minimize latency or maximize energy efficiency.

We do this by introducing the concept of *message classes*. Each class has an associated replication strategy and a routing metric, as well as a specific TTL value.

Although it is possible to define an arbitrary number of classes, we propose two simple ones, which strike a good balance between flexibility and complexity:

- High-priority (or alarm) messages that use a multi-copy strategy, and are routed according to the EDD only, thereby minimizing latency.
- Low-priority (or sensing) messages that use a single-copy strategy, and are routed based on a combination of EDD, battery and buffer space, aiming to prevent saturation of the best carriers and extend network lifetime.

By using these classes, we can increase the network's global efficiency and transfer less important messages with minimal effort whilst still being able to accommodate urgent ones.

2.5 Time Synchronization and Power Management

Increasing routing efficiency isn't enough to achieve energy efficiency in opportunistic networks. Broadband radios are not only one of the largest consumers but can use as much or even more energy on idle listening than they do while transmitting. No energy savings can be achieved without taking this into account and turning off the radio when it is not being used.

There are several possible solutions including synchronous and asynchronous cycling, as well as wake-up radios. As the latter requires specific hardware, and asynchronous cycling is generally suboptimal, we have decided to use synchronous cycling. To be able to use it, we need a global time reference that can be obtained from a broadcast signal (e.g. GPS or FM), once again requiring hardware, or through a synchronization protocol. Despite there being a number of protocols for WSNs, they usually focus on high precision and are not adapted to opportunistic settings.

We have devised a simple scheme that uses two fields on the periodic beacons broadcast by each node and allows synchronization to the sink's clock. When two nodes meet, each processes the other's time reference as shown in Figure 2.

```
algorithm update_time (c) is
    if localTimeAge ≥ timeAge(c) + stepPenalty then
        localTime ← time(c)
        localTimeAge ← timeAge(c) + stepPenalty
    end
end
```

FIGURE 2: TIME SYNCHRONIZATION ALGORITHM

Sinks are considered to always have fresh references, that is, they have zero reference age. The *stepPenalty* parameter is only indented to reduce the number of average synchronization steps, as there is an error introduced with each one. Because there is no drift correction, the error will also tend to quickly increase with reference age. Current digital clocks can, however, maintain a useful reference for days, which is long enough for most scenarios.

The acquired global time is then used to generate synchronous rounds on all nodes. During these times, the radio is turned off and all system activity is suspended. The node must wake up frequently enough not to miss too many connection opportunities, requiring some care during the definition of sleeping periods. Once again, a balance must be stuck between energy-efficiency and network performance, considering the specific conditions of the deployment scenario. There are other uses for a global time reference, for instance it can be used to timestamp messages in a way that allows them to be sorted and correlated at the sink. This timestamp is also used to sort messages in the buffers, and check for TTL expiration.

3 Reference Implementation

A reference implementation was developed primarily as a proof of concept, aiming to validate and evaluate, in a real setting, our proposed solution. It also allowed us to better assess the difficulty of implementing our solution in real WSN hardware, as well as system requirements. We used Sun Microsystems' SPOT sensor nodes[1]. These are relatively powerful nodes, featuring an ARM9 processor, 512KB of RAM, 4MB of Flash memory and an 802.15.4-compliant CC2420 radio. They get their power from a rechargeable battery, and are shipped with a sensor board containing a 3-axis accelerometer, temperature and light sensors, as well as LEDs, switches and several input and output pins. No operating system is used, with nodes running a bare-metal Java VM (Squawk) instead.

Our architecture can be seen on Fig 2. CHARON is implemented as a bundle layer, sitting atop the included network stack and using the bundled datagram-based protocol (`Radiogram`) for all single-hop exchanges.

CHARON uses four main threads:

- `BeaconThread` periodically broadcasts beacons containing the node's EDD, battery level, free memory and time.
- `ListenThread` listens for beacons, uses the received information to update the encounter history and, if necessary, updates the routing table and triggers message sending.
- `ClientThread` forwards message to the next carrier, if available.
- `ServerThread` listens for incoming messages, and stores them in the buffer.

Applications use the system by instantiating a new `CharonConnection` object with a chosen stream ID, used by the sink to tell apart messages from different applications. As regular nodes cannot

[1] http://www.sunspotworld.com/

receive messages, this class only provides methods for sending messages. Message class is controlled by calling the `sendMessage` or `sendAlarm` methods.

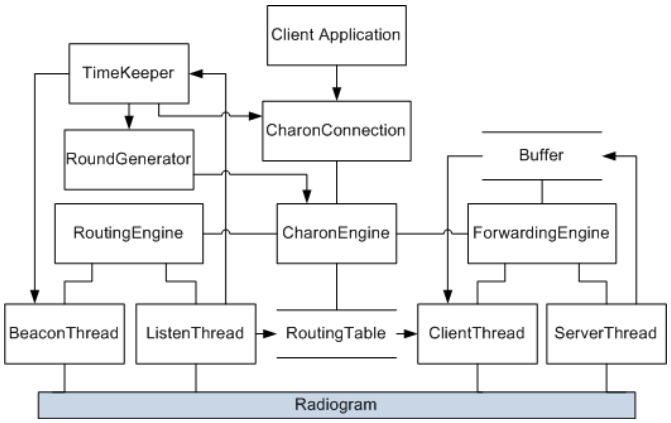


FIGURE 3: CHARON IMPLEMENTATION'S ARCHITECTURE

Each node keeps a list of known nodes, and their last received EDD, battery and memory data, as well as their computed ICTs. When another node comes in range (i.e. when a beacon is received), its score for each class is checked. If it is better than the node's own score and that of all other current neighbours, it is added to the routing table. At most, one entry per class is present in the table at any time.

Each time the routing table is updated or a new message arrives, `ClientThread` is notified and begins transfer of all suitable messages. The system uses a cross layer approach, having no bundle-layer ACKs and relying instead on the MAC-level ones. When a message transfer fails, the message is put back into the buffer. Due to an optimization on the SPOT network stack, a listening connection, even after being closed, still acknowledges incoming packets. This presents a problem, as there is no easy way to quickly notify a sending node if out of buffer space. The solution we adopted is to place a limit on the number of sent messages between each beacon. That limit has to be a fraction of the number of messages that fit in the free buffer space, in order to prevent buffer exhaustion by several sending nodes. Its ratio has to be tuned to the scenario, and presents a trade-off between radio capacity and packet loss but, if chosen reasonably, does not have a noteworthy effect on the system's performance.

Time synchronization is handled by the `TimeKeeper` module, which receives contacts' time references from `ListenThread`. This information is then used by `RoundGenerator` to shut down the radio and block all threads in synchronization primitives, thereby allowing the SPOT power management system to enter deep sleep mode.

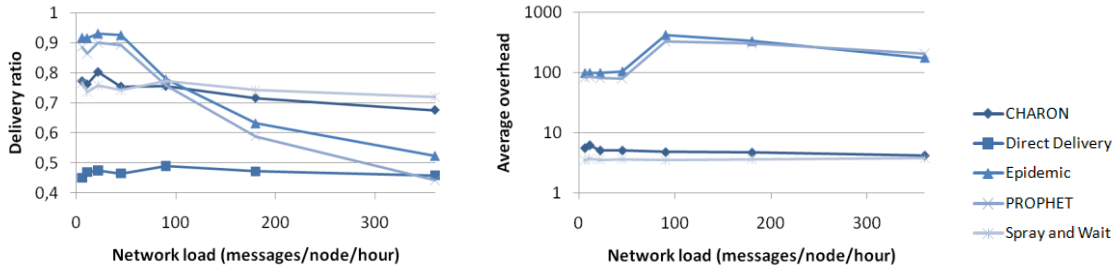
4 Evaluation

4.1 Simulation

For the evaluation of CHARON, we have decided to use the Opportunistic Network Environment (ONE) simulator [8]. ONE is a delay-tolerant network (DTN) simulator, designed specifically for routing protocol evaluation and written in Java, making it easier to port the source from our prototype implementation. It is distributed along with implementations of several routing algorithms, of which we chose Spray and Wait [2], Epidemic Routing [1], PROPHET [4] and Direct Delivery (best described as the absence of routing in the sense that messages are only delivered if the source and destination meet). We believe this set covers the most representative classes of routing algorithms and allows for a fair comparison with our solution.

Settings for the simulation were extracted from our high-level target scenario specification. The movement area is an 80 km^2 square, in which 60 nodes are distributed at random, and a sink is placed at the centre. There are six node groups with different movement patterns, representing different cohabiting populations. Each group has a pool of waypoints, from which nodes select their next destination. Their movement speed is randomly selected for each trip, ranging from 1.8 km/h to 18 km/h. Each node has 200 kB of buffer space, and the radio range (40 m) and speed (250 kb/s) were chosen to reflect a typical 802.15.4 radio. Messages are generated every 60 s. Each simulation ran for a simulation day. Similar simulations for different protocols were run with the same pseudo-random number generator seed, guaranteeing result comparability. All simulation sets were repeated with different seeds and the results averaged, in order to reduce the chance of artefacts.

The results of the simulation can be seen in Figure 4 which presents, for each protocol, the delivery ratio (percentage of messages delivered) and average overhead (number of excess transmissions per message).



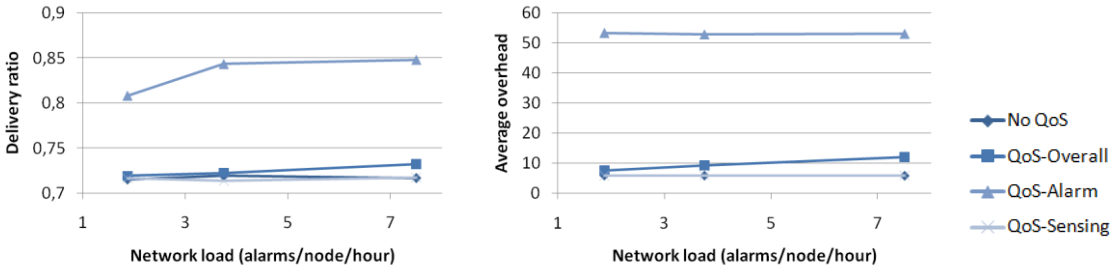
(A) (B)

FIGURE 4: PERFORMANCE COMPARISON FOR VARIOUS NETWORK LOADS

The unbounded multi-copy protocols (Epidemic and PROPHET) achieve good performance for low network loads, but their results degrade quickly as the load increases, due to resource exhaustion. Their overheads are also extremely high, generally above 100, wasting precious network resources. CHARON,

as well as Spray and Wait – a bounded multi-copy protocol – have lower but more stable delivery ratios, tolerating the load increase. More importantly, overhead is between 10 and 70 times lower than Epidemic and PROPHET's. Direct Delivery, being contingent on the co-location of the source node with a sink, has the worst overall delivery ratio, but has zero overhead.

The results show that CHARON is able to balance reliability and efficiency, as was our goal. Nevertheless, in some cases the delivery ratio might not be acceptable. The included QoS mechanism was designed to serve those cases, and its performance is shown in Figure 5.



(A) **(B)**

FIGURE 5: PERFORMANCE IMPACT OF QOS MECHANISMS

The first aspect to note is that the lines for non-QoS traffic and sensing traffic mostly overlap, showing that – in this load range – high-priority traffic does not negatively impact the regular traffic's delivery ratio. For alarm traffic, on the other hand, CHARON is able to achieve a performance level similar to the multi-copy approaches, demonstrating its flexibility.

4.2 Real-world evaluation

We have also performed several tests of CHARON using its reference implementation, some of which we'll briefly present. The synchronization scheme was evaluated by having a node broadcast messages and comparing the reception timestamps of two others. Results show that, immediately after synchronization, the average offset is of only 0.2 ms, with a standard deviation of 2.9 ms. The power management mechanism can, based on this reference, extend node lifetime by 440% for a conservative 10% duty cycle. For this duty cycle and the measured clock drift, a node could stay up to 2 days without being resynchronized and still work correctly.

5 Conclusions and Future Work

We have described CHARON, a simple but efficient and effective protocol for routing in opportunistic WSNs. In addition to the core forwarding functionality, it features integrated time synchronization and radio power management, an important, but often overlooked, component of a deployment-ready opportunistic communication solution. We also presented our reference

implementation, as well as simulation results that show our approach is able to provide a good delivery ratio at a much lower overhead than existing multi-copy protocols.

Although our solution fulfils the goals initially set, it could benefit from additional work. Exciting topics include the use of message integrity codes to provide trusted forwarding and approximate location extraction from the contact patterns, assuming static nodes are present and can be georeferenced on deployment. The synchronization mechanism could be improved by using multiple references, making it more resilient to extreme cases and, finally, more advanced uses of delay metrics should be investigated. Improvements do, however, have a tendency to increase complexity, and should be handled with caution.

References

[1] A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks," Duke University, Durham, North Carolina, USA, Technical Report CS-2000-06, 2000.

[2] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking*, Philadelphia, Pennsylvania, USA, 2005, pp. 252-259.

[3] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a Three-Tier Architecture for Sparse Sensor Networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, Alaska, USA, 2003, pp. 30-41.

[4] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic Routing in Intermittently Connected Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 19-20, Jul. 2003.

[5] B. Pásztor, M. Musolesi, and C. Mascolo, "Opportunistic Mobile Sensor Data Collection with SCAR," in *IEEE Internatonal Conference on Mobile Ad-hoc and Sensor Systems, 2007*, Pisa, Italy, 2007, pp. 1-12.

[6] P. Juang, et al., "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with Zebranet," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, California, USA, 2002.

[7] R. Baumann, S. Heimlicher, M. Strasser, and A. Weibel, "A Survey on Routing Metrics," ETH-Zentrum, Zurich, Switzerland, TIK Report 262, 2007.

[8] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques. SIMUTools '09*, Rome, Italy, 2009.