FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Department Informatik

Technical Reports / ISSN 2191-5008

Christopher Eibel, Sebastian Herbst, Björn Cassens, Timo Hönig, Peter Wägemann, Heiko Janker, Rüdiger Kapitza, Klaus Meyer-Wegener, Wolfgang Schröder-Preikschat

A Flexible, Adaptive System for Data-Stream Processing in Energy-Constrained Ad-hoc Networks

Technical Report CS-2015-04

November 2015

Please cite as:
Christopher Eibel, Sebastian Herbst, Björn Cassens, Timo Hönig, Peter Wägemann, Heiko Janker, Rüdiger Kapitza, Klaus Meyer-Wegener, Wolfgang Schröder-Preikschat, “A Flexible, Adaptive System for Data-Stream Processing in Energy-Constrained Ad-hoc Networks,” Friedrich-Alexander-Universität Erlangen-Nürnberg, Dept. of Computer Science, Technical Reports, CS-2015-04, November 2015.

Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Informatik

Martensstr. 3 · 91058 Erlangen · Germany

www.cs.fau.de

A Flexible, Adaptive System for Data-Stream Processing in Energy-Constrained Ad-hoc Networks

Christopher Eibel[1], Sebastian Herbst[2], Timo Hönig[1], Peter Wägemann[1], Heiko Janker[1]
Klaus Meyer-Wegener[2], Wolfgang Schröder-Preikschat[1]
[1]Department of Computer Science [2]Department of Data Management
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

Björn Cassens, Rüdiger Kapitza
Institute of Operating Systems and Computer Networks
TU Braunschweig, Germany

***Abstract*—Today's generation of sensor networks reflect several changes in system characteristics over traditional sensor networks. These changes affect three key aspects: energy as a fundamental resource, stream data processing, and the inherently dynamic structure of the overall system.**

In this paper we extract and present eight distinct challenges aligned to the key aspects which need to be addressed in the future. We use data of ongoing science and research projects to extract the most important challenges. These challenges need to be tackled in order to provide the basis for a flexible and adaptive system design which supports data-stream processing in energy-constrained ad-hoc networks.

***Index Terms*—Energy Awareness, Energy Analysis, Data Stream Management System, Query Optimization, Dynamic Code Reconfiguration, Wireless Sensor Networks**

I. Introduction

Today's generation of sensor networks reflects several distinct changes in system characteristics over traditional sensor networks. Traditional sensor networks commonly consist of heterogeneous systems with a predetermined and well-defined set of responsibilities (e.g., base station, wireless sensor node) and a static performance ability (e.g., static computing power, limited runtime). Today, however, we are challenged to build sensor networks where system characteristics such as responsibilities, performances, and runtime greatly change over the lifetime of the overall system. Naturally, the next step is making it adaptable to fit changing demands.

Three key aspects strongly impact the design of current distributed sensor networks: First, energy has not only persisted as the rarest of all system resources (e.g., for mobile and/or wireless sensor nodes), but it now also affects larger components of the system (e.g., detached base stations). As a facilitating aspect, energy harvesting (e.g., from photovoltaic, ambient radiation, and piezoelectric energy sources) has become more practical as today's energy harvesting devices have reduced losses and provide an increased level of efficiency [1]. Handling energy as a fundamental resource for the overall system eventually leads to the situation where scheduling decisions must be based not only on available time resources (time constraints) but also whilst taking into account available energy resources (energy constraints). As the system's state (e.g., harvesting, discharging) and the amount of resources (i.e., charge level) are changing over time, energy represents a highly dynamic input value for system decisions.

As a second aspect, data processing within sensor networks has changed from a „sense and deliver" scheme—commonly within a star topology—towards a hybrid structure where responsibilities of system components are changing at runtime. Within such a dynamic network topology, data often needs to be preprocessed at the source—or during the transport—and is transported as a stream. Similar to cloud-computing environments, stream processing requires new, adapted algorithms for efficient processing of the sensed data. This is an ongoing research field. For instance, in [2], robots are able to react to the battery power level. If the energy falls below a certain point, the robots can hand over their running task and start an emergency task. In our intention, we go one step further—instead of reacting to low power, we aim to prevent such a situation.

Third, the highly dynamic structure of the overall system is an aspect that requires new mechanisms to reconfigure individual system components. To achieve a

common goal of the overall system, the responsibilities of individual components need to be adapted at runtime. This requires reconfiguration operations—or even operations to update the entire affected system component. Updating system components, a widely addressed research topic [3], is necessary when a reconfiguration of previously deployed program code is no longer sufficient to transfer the component to its new system state.

Recently, science and research projects from various different areas have implemented complex systems which adapt to changes of their environmental conditions. Furthermore, such systems have to carry out highly diverse experiments over the lifespan of the project. For example, the computing systems of research projects such as the comet lander *Philae* (see Section V) and the Mars rover *Curiosity* were built to adapt their mode of operation dynamically. Similar to this, more *down-to-earth* research projects such as the BATS project (see Section II) provide important data to study the challenges which are accompanied with the change and combination of the three aspects presented above.

The paper is organized as follows. We present the usage scenario of our system in Section II. In Section III we extract challenges, grouped by categories (holistic energy awareness, energy-aware code updates, and energy-centric query optimizations) and propose solutions for the challenges in Section IV. Section V presents related work, Section VI concludes.

II. Operating Environment

To showcase the aforementioned challenges of a heterogeneous sensor network that is subject to continuous reconfiguration and stringent energy requirements while delivering live data using advanced data-stream processing, this section gives a brief overview of the BATS[1] project. The main goal of this interdisciplinary project composed of biologists, electrical engineers and computer scientists is the automated and scalable tracking of bats in their natural habitat to get in-depth knowledge about their social behavior. For behavioral biology this is a major step forward as up to now a biologist typically has to catch and chip single animals and manually track them using radio telemetry. Analyzing complex social and behavioral patterns is almost impossible this way. In the context of BATS a biologist gets live data from the habitat and a set of individually identifiable animals. She can crosscheck data during experiments, adapt the experiment setting online (e.g., different monitoring interval)

[1]Dynamically Adaptive Applications for Bat Localization Using Embedded Communicating Sensor Systems, http://www.for-bats.org/

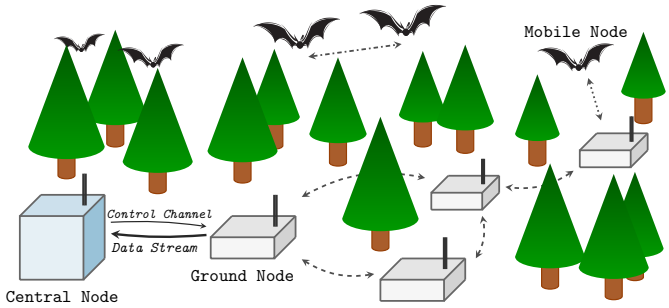

Fig. 1. In our usage scenario the overall system is spread across central, ground, and mobile nodes.

and refocus research questions with growing knowledge about the social behavior or, for example, due to live events (e.g. special climate conditions or meteorological phenomena such as thunderstorms).

Ground and Mobile Nodes. To achieve this goal we use a combination of stationary *ground nodes* that span a network over an area of several hectares (i.e., a hunting ground) and ultra-lightweight mobile nodes that are attached to the animals. Since bats can only carry up to 10 % of their own body weight, a mounted sensor node (i.e., transceiver, MCU, battery, PCB, etc.) must not weigh more than 2 g—assuming a body weight of only 20 g for the species under investigation. Designing a mobile node that fulfills such strict requirements is a challenge by itself and out of scope of this paper.

Updating Using a Data-Stream Management System. Figure 1 shows a deployment scenario and depicts interactions of all involved components. The mobile nodes continuously log occurrences and durations of meetings with other mobile nodes. If a mobile node is in communication range of the ground network, meeting data is transmitted to the ground stations. This way the social interactions of bats can be recorded even if meetings take place miles away from the ground network. Once the meeting data reaches the ground network it becomes accessible for a central node that acts as a gateway to the end user, typically a biologist. This central node also hosts the data-stream management system (DSMS), which requires this platform to be more powerful than the other ground nodes. The DSMS gives the biologist the aforementioned flexibility, she simply controls the system's configuration via queries. These queries are composed of operators which are deployed over the network to reduce network traffic and energy demand [4].

Besides handling meeting data provided by the mobile nodes, ground nodes are responsible for a) forwarding as well as preprocessing of localization data, and b) collecting environmental data (e.g., temperature, humidity) as

well as network conditions. If a mobile node is within reach of the ground network, phase-based range estimation and field-strength measurements are used to compile trajectories (showing the bats' flight paths), which are provided to the end user. Albeit not being as restricted as mobiles nodes, the ground nodes are deployed in remote locations, such as rain forests, and are battery-powered, thus energy is scarce. Nevertheless, they are required to have considerably more computational power since range estimation algorithms inherit some complexity. Moreover, higher transfer rates within the ground-node subnetwork are necessary to handle multiple connections in parallel and still be able to deliver collected data to the central node in a timely manner.

Figure 2 depicts the control-circuit–based class of a system that conforms to our use-case scenario. The DSMS is in constant dialogue with the energy manager to decide how the ground network and its nodes need to be re-adjusted (i.e., reconfigured or updated) by the control unit. The network system emits a continuous data stream that contains all collected data but also acts as the feedback channel in the control circuit. In the other direction, the channel is used for controlling the network's configuration by means of updating or reconfiguration of both mobile and ground nodes. However, for mobile nodes, this is quite more challenging because of even less resources being available. More precisely, reconfiguration and update management is necessary on all types of nodes; though, the functionality differs among them. For instance, the ground-node–specific control unit not only accepts update and reconfiguration requests (as is the case with mobile nodes) but also can initiate and forward them. At the other end, the DSMS continuously processes a stream of data (e.g., identifiers, timestamps, movement coordinates) it receives from the ground-node network. Continuous queries (CQs) look similar to queries in database management systems (DBMSes). Queries are realized as operator graphs which transform input streams into output streams. The operators are distributed over the network. However, CQs run for an extended amount of time. Thus, query optimization is even more beneficial than in a DBMS. As compiling queries improves the performance in DBMSes [5], using code updates to deploy compiled operators will save energy on the nodes.

We distinguish between three types of trigger categories for reconfiguration and update requests:

Automatic. This is the trigger type we are particularly interested in. In this case, certain events are specified a priori (however, might also be reconfigurable) that trigger

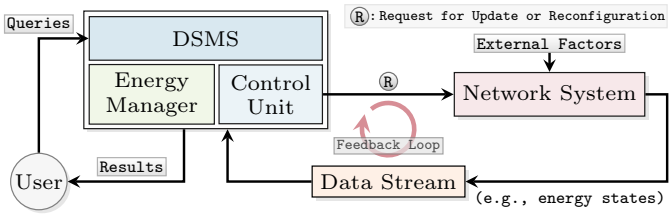

Fig. 2. At a high level of abstraction our system is an extended control circuit of continuous operation.

a reconfiguration or update automatically. Particularly, from continuously ongoing data-stream analyses, we aim to extract specific patterns that lead to re-adjustments of specific ground and mobile nodes. The type of node (mobile, ground, central) and its specific configuration and placement have an influence on the initiation of a re-adjustment as well. For instance, a ground node is aware of all mobile nodes in range. With further information, such as the individual energy state and current conditions in the compound (e.g., severe weather), the ground node may initiate a configuration switch within those mobile nodes. In addition, we envision that central and ground nodes maintain code-update databases, whose content is automatically generated based on the data stream.

Manual. Expects a user to trigger a reconfiguration or update request. For instance, a biologist knows that—possibly only during a single experiment that lasts a few days—the timestamp granularity does not need to be as accurate as possible. Therefore, the biologist initiates a reconfiguration request for using more coarse-grained timestamps, which saves energy as fewer bits per transmission have to be sent. For manual code updates, a user does not necessarily have to be a biologist. Occasionally, it is also necessary that computer scientists or electrical engineers initiate an update; for example, to add or exchange code because of new functionality or for error-correction reasons. As long as the changes are minor enough, it is more desirable to make an online instead of an offline update, because the latter would require to catch the bat with the respective mobile node first.

Semi-Automatic. There will be some configurations for which the system may not be able to decide on its own whether they are acceptable to the user or not. In that case, the user will be prompted and asked for approval.

The BATS scenario implies a few characteristic constraints, which are either fully or partially applicable to a number of other projects discussed in Section V. As a consequence of limited resources available on a mobile node, its transceiver can only be designed with very small possible down- and upstream. Since the mobile

nodes are expected to send much more data than they receive, the upstream (i.e., upload from ground to mobile nodes) is even more restricted for energy-saving reasons. Consequently, the moments of an update or reconfiguration need to be efficiently and precisely determined. Furthermore, bats movement, signal collisions, and the specific nature of the terrain lead to a certain degradation of network quality. Thus, packet losses are likely to occur, which makes adequate fault-tolerance mechanisms (e.g., erasure coding) necessary. Ad-hoc connections are an immediate consequence of moving bats (i.e., moving mobile nodes). This type of connection in combination with limited transfer rates forces us to take a decision on which data is essential to be transmitted and which data may be dropped safely if it cannot be sent later on.

III. Challenges

We present eight challenges (C1–C8) that are divided into the categories *energy awareness*, *code updates* (incl. reconfiguration), and *data-stream processing*. The first category is closely linked with the other categories as energy is one of our key aspects (cf. Section I). In the next section, we show how we plan to solve them.

A. Holistic Energy Awareness

Being aware of the whole system's energy consumption is a key challenge. This incorporates analyses of each node as well as a comprehensive analysis of the whole network including interactions between nodes and resulting communication costs.

C1: Local Energy-Resource Analysis. Local energy resources refer to the available energy budgets on mobile and ground nodes. The local analysis considers the energy consumption of the processor, the memory, and the peripherals during code execution. To ensure a long-running node, it is periodically put into a deep sleep mode leading to energy savings. Consequently, the node has a common duty cycle consisting of a wakeup signal from a sleep mode of the processor, the processing phase (e.g., sending data to the ground network), and again entering the sleep phase. The energy consumption of all possibly configurable duty cycles must be determined precisely in order to best exploit the available resources and to guarantee long-term operation. A self-measurement system on each mobile node enables requests of the current state of charge.

C2: Global Energy-Resource Analysis. The global resource analysis is based on information gathered through the local analysis. It considers the interaction between nodes, for instance, simple packet routing and code-update requests. Consequently, this analysis step determines the costs considering the communication between all nodes and the code executed on each node. To get sound results, the radios, their link qualities, the drivers, and the operating systems must be simulated.

B. Energy-Aware Code Updates

On account of strict resource limitations, applying a code update (or reconfiguration) inherently needs to be as energy efficient as possible—especially on mobile nodes. This is aggravated by the fact that the nodes are heterogeneous at software and hardware level. Under these prerequisites the subsequent challenges arise.

C3: Distributed-System Properties. Since we are dealing with a distributed system, we have to cope with typical distributed-system properties, such as consistency, reliability, and heterogeneity. Although a multitude of different solutions exist, in general, it is hard or even impossible to maintain all properties at once. Saving energy in accordance with these properties is even more challenging.

Several kinds of faults must be considered. Most of these faults are complicated by the presence of constraints as presented in Section II (e.g., unreliable channel). Errors during update or reconfiguration can be a result of hardware or network-communication failures. Generally, occurrences of all types of faults (i.e., persistent, intermittent, transient) are expected, which requires mechanisms for prevention or detection and recovery. Consistency is very important as experimental results depend on it. It must be guaranteed for single nodes after applying an update. Beyond that, the system must remain functional while nodes are updated independently. In general, compromises have to be reached; for instance, between strict and weak consistency.

C4: Deployment Strategies. It is not a trivial task to deploy updates just in time, especially when fault tolerance is necessary. Therefore, a scheduling and update plan is essential to ensure reliable dispatching of updates. To prevent delays, the plan creation itself must be as performant as possible. Furthermore, different strategies are desirable, such as a slow-deployment strategy saving energy or a fast-deployment strategy consuming more resources. During plan execution it is not always possible to update some of the nodes (e.g., they are not in transmission distance). As plan execution takes time, another update request may arrive which must be integrated into the current plan.

C5: Message Format. The message format defines which data can be found at which position in a sent and received message. Most applications use a predefined format with fixed positions and lengths for each data field. The deserialization of such packets is a straightforward task; however, it cannot be applied to our scenario because the data to be sent changes frequently. Besides different data widths for a certain data field, also the number and order of the data fields may change—therefore, the message format must be adaptive and its overhead as low as possible.

While updating it is possible to receive messages with outdated formats due to network latencies or partial failures. Thus, it is necessary to facilitate different versions of a deserializer until all messages with outdated formats have been received. This problem is amplified due to the fact that different configurations of different nodes are possible and the common case. For this reason, it must be tackled during the development of such a system.

C. Energy-Centric Query Optimization

Aside from algebraic, rule-based optimization, energy saving and data quality are competing goals. When energy is scarce, it makes sense to consider reducing the results' quality to save energy. However, the quality degradation must be estimated, as reducing the quality too much may render the whole query useless. The goal is to find cheaper queries, which produce similar results with a negligible quality loss.

C6: Semantic Optimization. Typically, stream-processing optimizations increase the performance without changing the semantics [4]. The notable exception is load shedding, which is an emergency measure to counter load spikes by dropping packages. We want to develop methods to create alternative queries which trade accuracy for reduced energy consumption. Unlike load shedding, the impact on the result should be minimal. The better the DSMS understands a query's purpose, the better it can determine which information is essential. For example, instead of requiring both participants to send down their meeting data, only the participant with the higher ID could send it. This would save half of the transmission cost but increase the risk of lost meetings. Still, losing some meetings is better than losing all data after the battery has been depleted. Consequently, methods to create alternative queries must be developed that provide similar results while saving energy. In particular, defining an adequate similarity measure is challenging and may vary greatly depending on the operational scenario.

C7: Situational Awareness. The impact of an optimization strongly depends on the situation. Assessing the situation of each node is essential for making good optimization decisions. For instance, the chance of successfully transmitting data can also be estimated by the system. A node attached to a bat currently resting in the middle of the observed area will most probably succeed in sending data. The above optimization would negligibly reduce accuracy if both participants are in this area. Yet, if a bat spends a lot of time outside the observed area, its attached node will probably have an overfull transmission queue. Then, there is no sense in relying on this node to successfully send down new meeting data. This challenge comprises the identification of relevant situations from queries and query results, and rating them according to their optimization potential.

C8: Forward Planning. Situations change over time. This is crucial to two aspects. First, applying an update is neither free nor instant. As it takes some time to pay off, it is imperative to estimate how long a situation will be in favor of an optimized configuration. The mere fact that a slightly worse configuration is likely to be be applicable for a longer time can make it the better choice over a configuration which is superior in the short term. Second, if the situation changes, updating costs may rise or updating may be impossible. In the BATS scenario mobile nodes can spontaneously leave the observed area. This makes updates over ground-to-bat communication impossible. Usually, this also means that the situation has changed in a way that the current configuration has become inadequate. In this example, as part of its configuration, the node should be directed to reconfigure itself to safer (less efficient) operation if it receives a beacon from a ground node at the border of the observed area. In summary, the system must predict future situations for all nodes and prepare them for predicted changes.

IV. Systems Approach

Our systems approach is based on the setup introduced in Section II: The system consists of heterogeneous ground and mobile nodes, which are the target for energy optimizations, and a central node coordinating them. Subsequently, we describe ideas for solving the challenges presented in Section III.

A. Energy Analysis (C1–C2)

For the local analysis, we propose static-analysis techniques to determine the worst-case energy consumption (WCEC) [6] of tasks. These bounds consider the

consumed energy on the mobile node including all peripherals. Precise energy models are yielded through accurate energy measurements of all components [7]. These models are exploited by the static analysis of program code. Nodes are aware of their local state of charge. The current state and the knowledge about WCECs are used for local scheduling.

The global energy-resource analysis uses the information about the WCEC, the state of charge on each mobile node, communication costs, and the global system activities (i.e., ongoing experiments) in order to best predict the behavior of the network. To accomplish this task, we consider extensive use of wireless–sensor-network simulators [8]. These computation-intensive simulations run on the central node of the network where energy is not a limiting factor. For example, if the user requests a code update, the analysis uses the current state of the actual network as input configuration for the simulation [9]. The simulation yields information about the suitability and costs of the requested update.

B. Middleware and Operating System (C3–C5)

For energy-aware reconfiguration and updating on the very limited mobile nodes in particular, as a basis, we build upon the real-time operating system SLOTH [10], which is highly optimized for both efficiency and effectiveness. More precisely, by exploiting underlying hardware features such as the hardware interrupt subsystem, SLOTH can efficiently schedule and dispatch tasks—and still outperforms comparable software-based solutions.

To attack the challenges C3 to C5, we implement a middleware layer that is coalesced with the control unit that triggers and initiates updates or reconfigurations. This middleware exists on all nodes but requires to be specialized depending on the node type (e.g., on mobile nodes, only reduced functionality is possible). The problem of general consistency across all nodes (C3) can be addressed based on existing techniques, such as proposed by Levi et al. [11], whose algorithm relies on meta-data broadcasting declaring the current software version of a node. For challenge C4, the middleware needs to be steadily aware of all nodes in the network and their current state. Based on global energy-resource analyses (C2), the DSMS can find the most suitable update-and-reconfiguration plan. This plan includes which nodes are updated or reconfigured as well as the order (i.e., at which priority) and points of time the updates or reconfigurations are scheduled and applied. The desired functionality thereby has to be guaranteed among all nodes when an update is applied

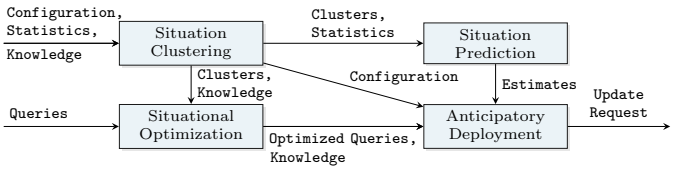

Fig. 3. Subsystems and their data flow needed to realize energy-centric query optimizations.

or a configuration is changed. Hence, for this type of consistency (C3) these plans need to be enhanced with, for instance, directed acyclic graphs (DAGs) [12]. Edges model the dependencies and nodes represent updates or reconfigurations. By means of DAGs, the control unit will be able to efficiently execute the enhanced update-and-reconfiguration plans. In order to choose the right deserializer and its version (C5), an identifier for the message layout can be used. To tackle network latencies or partial failures, this identifier must be sent for each message, which is then evaluated in the middleware.

C. Query Optimizer (C6–C8)

As argued in the challenges C6–C8, situations play a key role in energy-centric query optimizations. To identify them, the system collects statistics[2] from the query results. The statistics then have to be interpreted with respect to the active configurations and domain-specific knowledge. This knowledge comprises use-case–specific quality measures, their weighting, heuristics and causal relations. By employing techniques known from data mining, the system will identify relevant correlations. Through clustering, we hope to identify stereotype situations of the nodes. This is depicted as *situation clustering* in Figure 3.

For each stereotype, alternative queries are determined in the *situational optimization*. We plan to use heuristics or metaheuristics on algebraically optimized queries, as operator distribution, which is part of this problem, is NP-hard [13]. The cost function will be deduced from the domain-specific weighting of quality measures and energy consumption. As some situations might not last long enough to exploit optimizations, alternative queries for combinations of clusters will also be determined. The performance of alternative queries in unsuitable situations will be determined to provide numbers for bad-case estimations.

For each node the expected temporal flow of situations must be determined in the *situation prediction*. For this,

[2]This may be expensive, but energy is not an issue in the central node and biologists need most of this data anyway.

we hope to successfully employ Bayesian networks. As they need a lot of training data, we plan to use several networks in different levels of detail and select the most reliable one, for instance, if there is not enough data on a single bat, we use data on all bats for prediction.

With the predicted situations and the optimized queries, the expected savings for different updates can be determined and compared with the current configuration and its costs. If an update is expected to pay off, the *anticipatory deployment* will send an update request. It also determines situations which are harmful to the intended configuration. To prevent damages from these situations, triggers for local updates will be integrated into a request.

V. Related Work

Reducing energy consumption in wireless sensor networks is an extensively addressed research area. For instance, Cappiello et al. [14] propose a method which adaptively aggregates data while finding a tradeoff between quality and consumed energy. Their method is one approach to generate alternative queries.

Regarding state-of-the-art code updating, Ferri et al. [15] give insights into the Rosetta comet mission from a technical point of view. For the comet lander Philae, the scientists use solar panels for energy harvesting. When the battery level is very low, the current content in RAM is transmitted to a non-volatile memory and retransmitted back when enough energy has been obtained. In our targeted class of system, energy-harvesting methodologies are envisioned, too, to prolong the nodes' battery lives.

The BATS scenario we presented is one of several wildlife-tracking research projects. The biodiversity of tracked animal species ranges from big zebras [16] over badgers [17] to smaller animals, such as rats [18] or birds [19]. All solutions are strongly influenced by the species, as habits, environment, and carrying capacities vary. However, whenever the species live in rural environments, energy scarcity is a fundamental issue. With smaller animals, tracking becomes even more challenging, as hardware components—including batteries in particular—are limited in size and weight.

Cyber-physical systems (CPSes), as proposed in [20], have a broad field of usage scenarios. Similar to our scenario, sensor data is gathered from mobile and from stationary nodes. Afterwards, the acquired data is used to extract knowledge. Their approach for energy saving is to turn off unused sensors which is also a fundamental part of our approach. Thus, CPSes have similar challenges, as presented in our paper, and can therefore benefit from the results of our targeted approach.

VI. Conclusion

In this paper, we illustrate opportunities to save energy in wireless sensor networks by adapting the system to the current needs. With advanced query-optimization techniques, holistic energy analyses, and powerful system software for updating, a highly efficient and flexible distributed system can be built. However, to build such a system, research challenges must be overcome. We spell out these challenges and clarify them with a scenario taken from wildlife monitoring. The approaches we list give starting points for further research. Solving these problems allows to build smarter sensor-network systems. As we highlight, the techniques will not only assist biological research projects, as a wide range of different scenarios benefit from increased operation time and reliable update mechanisms.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG) under grant no. FOR 1508 and grant no. SCHR 603/13-1.

References

[1] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*, Apr. 2005, pp. 457–462.

[2] F. Dressler and G. Fuchs, "Energy-aware Operation and Task Allocation of Autonomous Robots," in *Proceedings of the 5th IEEE International Workshop on Robot Motion and Control (RoMoCo '05)*, Jun. 2005, pp. 163–168.

[3] E. Miedes and F. D. Munoz-Escoı, "A survey about dynamic software updating," Instituto Universitario Mixto Tecnológico de Informática, Universitat Politecnica de Valencia, Tech. Rep. ITI-SIDI-2012/003, 2012.

[4] M. Hirzel, R. Soulé, S. Schneider, B. Gedik, and R. Grimm, "A catalog of stream processing optimizations," *ACM Computing Surveys*, vol. 46, no. 4, pp. 46:1–46:34, Mar. 2014.

[5] T. Neumann, "Efficiently compiling efficient query plans for modern hardware," in *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, vol. 4, no. 9, 2011, pp. 539–550.

[6] R. Jayaseelan, T. Mitra, and X. Li, "Estimating the worst-case energy consumption of embedded software," in *Proceedings of the 12th Real-Time and Embedded Technology and Applications Symposium (RTAS '06)*, 2006, pp. 81–90.

[7] T. Hönig, H. Janker, C. Eibel, O. Mihelic, R. Kapitza, and W. Schröder-Preikschat, "Proactive energy-aware programming with PEEK," in *Proceedings of the Conference on Timely Results in Operating Systems (TRIOS '14)*, 2014, pp. 1–14.

[8] M. Korkalainen, M. Sallinen, N. Kärkkäinen, and P. Tukeva, "Survey of wireless sensor networks simulation tools for demanding applications," in *Proceedings of the 2009 5th International Conference on Networking and Services (ICNS '09)*, 2009, pp. 102–106.

[9] M. Strübe, F. Lukas, B. Li, and R. Kapitza, "Drysim: Simulation-aided deployment-specific tailoring of mote-class WSN software," in *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '14)*, 2014, pp. 3–11.

[10] W. Hofer, D. Lohmann, F. Scheler, and W. Schröder-Preikschat, "Sloth: Threads as interrupts," in *Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS '09)*, December 2009, pp. 204–213.

[11] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, 2004, pp. 2–2.

[12] R. Mahajan and R. Wattenhofer, "On consistent updates in software defined networks," in *Proceedings of the 12th ACM Workshop on Hot Topics in Networks (HotNets '13)*, November 2013, pp. 20:1–20:7.

[13] B. J. Bonfils and P. Bonnet, "Adaptive and decentralized operator placement for in-network query processing," in *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks (IPSN '03)*, 2003, pp. 47–62.

[14] C. Cappiello and F. A. Schreiber, "Quality- and energy-aware data compression by aggregation in WSN data streams," in *Proceedings of the 7th IEEE International Conference on Pervasive Computing and Communications (PerCom '09)*, 2009, pp. 1–6.

[15] P. Ferri, A. Accomazzo, S. Lodiot, A. Hubault, R. Porta, and J.-L. Pellon-Bailon, "Preparing the rosetta deep-space operations," *Acta Astronautica*, vol. 67, no. 9, pp. 1272–1279, 2010.

[16] 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," *ACM SIGOPS Operating Systems Review*, vol. 36, no. 5, pp. 96–107, December 2002.

[17] V. Dyo, S. A. Ellwood, D. W. Macdonald, A. Markham, N. Trigoni, R. Wohlers, C. Mascolo, B. Pásztor, S. Scellato, and K. Yousef, "WILDSENSING: Design and deployment of a sustainable sensor network for wildlife monitoring," *ACM Transactions on Sensor Networks*, vol. 8, no. 4, p. 29, 2012.

[18] J. A. B. Link, G. Fabritius, M. H. Alizai, and K. Wehrle, "BurrowView – Seeing the world through the eyes of rats," in *Proceedings of the 2nd IEEE International Workshop on Information Quality and Quality of Service for Pervasive Computing (IQ2S '10)*, March 2010, pp. 56–61.

[19] C. Rutz, Z. T. Burns, R. James, S. M. Ismar, J. Burt, B. Otis, J. Bowen, and J. J. S. Clair, "Automated mapping of social networks in wild birds," *Current Biology*, vol. 22, no. 17, pp. R669–R671, 2012.

[20] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, "Review: From wireless sensor networks towards cyber physical systems," *Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 397–413, Aug. 2011.