C-SDF: Practical Solar-aware Distributed Flow Control

Immanuel Schweizer, Tobias Petry, Max Mühlhäuser
Technische Universität Darmstadt, Telecooperation Lab
Email: schweizer@cs.tu-darmstadt.de

***Abstract*—Energy harvesting in wireless sensor networks leads to an interesting optimization problem: Given the spatial-temporal distribution of the energy harvested, how can nodes adapt their consumption to reach energy neutral operation, i.e., consume exactly the harvested amount of energy over time.**

For many applications, nodes need to forward data for other nodes. Hence, adapting the energy consumption of one node equates to controlling the amount of data produced elsewhere in the network. For field monitoring applications, different flow control approaches have been proposed that tackling this problem.

In this paper, we present the first hardware implementation of the solar-aware distributed flow control (C-SDF) approach. Different runs on two distinct testbed sites show that C-SDF works even under adverse network conditions. The sampling rate stabilizes quickly. Energy utilization is more than 80%, while ensuring no node is overloaded.

I. Introduction

Energy harvesting is often cited as a solution to the issue of battery-limited lifetime in wireless sensor networks [1], [2]. With batteries the challenge is to prolong the lifetime of the network. With harvested energy the challenge shifts to providing the best possible service, while no node will ever deplete. This is called the energy neutral operation.

This leads to the following optimization problem: Given the spatial-temporal distribution of the energy production, how can nodes adapt their energy consumption to reach energy neutral operation. Here, the definition of best possible service and the consumption of each node depend on the application. We want to focus on one of most prevalent applications in sensor networks; periodic reporting. In periodic reporting, all nodes periodically collect data and sent it to a common base station. The service quality is given by the time between collected data points called sampling rate. Higher sampling rates lead to better sensor coverage, hence, a better service. Now, consumption of each node is mostly driven by the sampling rate of the sensor and by sending and forwarding data packets. This is also the main challenge. Forwarding data packets implies that some energy consumption is driven by remote nodes.

This challenge was first introduced for homogenous sampling rates by Kansal et al. [1]. And it has lead to the introduction of different flow control mechanisms [3], [4], enabling nodes to control their sampling rate and the flow they need to forward for remote nodes. [4] offers a very comprehensive centralized and distributed solution with very interesting theoretical results. Unfortunately, since all flows need to be completely recalculated for even small updates in the routing tree, this approach is impractical. We have introduced SDF [3] as the first practical flow control heuristic to answer these shortcomings. With SDF, nodes need to send control broadcasts only to their children in the routing tree. This enables almost instant adaptability to changes in the routing tree with neglectable overhead.

In this paper, we want to validate the practicality of SDF on real hardware. We have implemented SDF for the Contiki[1] operating system using RPL [5] as routing protocol. We present results from four runs on two different testbeds and up to 7 days. On average, nodes utilize around 80% of the energy available, even under poor network conditions. With delivery rates in the testbed dipping below 70%, SDF was still able to stabilize fast and increase the sampling rate from 5 to 30 messages per hour on average.

The remainder of this paper is organized as follows. Section II introduces the related work. The implementation of C-SDF is described in Section III. Section IV presents the results of the evaluation. Section V concludes the paper.

II. Related Work

Energy harvesting describes the concept of powering sensor nodes using renewable energy sources. The feasibility of energy-harvesting wireless sensor networks has been studied extensively. This chapter aims at providing an overview by first introducing possible power sources and hardware for energy harvesting. Afterward, different algorithms that are solar-aware will be introduced. Last but not least, the problem solved by SDF is introduced, and related work is discussed.

Roundy et al. [2] review a great variety of potential power sources for wireless sensor networks. Besides the classical power storage in batteries, micro fuel cells, and micro heat engines, they also research the potential of different energy-harvesting techniques. These techniques include photovoltaics (i.e., solar panels), temperature gradients, human power, wind/air flow, and vibrations. Of these, solar panels offer the highest power density and are, therefore, the most likely candidate for reliably powering wireless sensor networks. Another advantage of solar energy is the predictability [1].

Predicting the availability of harvested energy is the basis of any harvesting sensor network. The prediction is used to adapt the duty cycle during times with no or insufficient energy [6] or to assign tasks to nodes with more energy [7].

[1]We are, thus, calling the implementation *C-SDF*. C-SDF is open source and can be downloaded at: https://github.com/ischweizer/C-SDF

Other research has focused on overcoming problems posed by powering wireless sensor nodes with solar panels. One of those problems is that simple charging circuits keep the solar panel's working point fixed. This implies that the battery can only be charged under ideal weather conditions. Alippi and Galperti [8] develop a power transferring circuit for optimally conveying the energy from a solar panel into a rechargeable battery, even in not optimal weather conditions, using a maximum power point tracker. This further increases the feasibility and usefulness of solar-powered wireless sensor networks, because solar power can be harvested and stored more efficiently.

Several working prototypes of wireless sensor nodes using solar power have been developed, showing that solar-powered wireless sensor networks are indeed feasible. Raghunathan et al. [9], [10] present the *Heliomote* (cf. Fig. 1), a Mica2 [11] mote enhanced with a circuit board equipped with a solar panel and NiMH batteries for solar energy harvesting.

Fig. 1: Heliomote [9], [10]

They show that, in principle, their device is capable of nearly perpetual operation. Jiang et al. [12] design and implement *Prometheus*, a wireless sensor node based on the Telos-mote [13] with a solar panel, supercapacitors and Li-ion batteries. They present and discuss results from a 10-day period. Minami et al. [14] develop and present *Solar Biscuit*, a wireless sensor node without a battery, solely relying on a solar panel and a supercapacitor. They do not present any results of an actual long-term deployment. Sikka et al. [15] present *Fleck1*, a solar-powered wireless sensor node using a solar panel and NiMH batteries. Their main contribution is the incorporation of a DC-DC converter enabling deeper battery discharge cycles between periods where solar power is available. The authors in [16] present results of a network operating 24/7 for over two years. Alippi et al. [17] develop and present a wireless sensor network framework based on their own solar-powered sensor nodes. The wireless sensor network is deployed in Moreton Bay, Brisbane, Australia, to deliver temperature and luminosity data of the marine ecosystem. They present and discuss results from a four-day period.

To make the operation of environmentally powered wireless sensor networks more efficient, several solar-aware protocols have been developed. Some focus on solar-powered wireless sensor networks, while others use a more general approach.

Lin et al. [18] develop E-WME (Energy-opportunistic Weighted Minimum Energy), an energy-aware routing algorithm for wireless sensor networks with any environmental energy sources. They show that their routing scheme is asymptotically optimal. Voigt et al. [19], [20], [21] present two variants of directed diffusion [22] modified to route in a solar-aware manner. Both protocols route packets preferably via nodes that are currently solar-powered and, thus, can relay the packet without using their battery. The authors present simulation results of both protocols and conclude that they offer significant energy savings. In another work, Voigt et al. [23] present solar-aware clustering protocols. Some more theoretical works focus on channel performance. Rajesh et al.[24] for example find the Shannon capacity when data is transmitted over an additive white gaussian noise channel. However, none of these approaches on optimizing environmentally powered wireless sensor networks takes the utilization of harvested power into account.

Some related work exists on the topic of assigning sampling rates in wireless sensor networks. Shu et al. [25] try to optimize the network sampling rates in terms of a scheduling problem without taking energy consumption into consideration. Bandyopadhyay et al. [26] analyze what trade-offs in sensor density, energy usage, throughput, and delay have to be made to achieve certain temporal and spatial sampling rates. But those sampling rates are always fixed and predefined. A more sophisticated idea is introduced by Lachenmann et al. [27]. They define different lifetime goals for a network and adjust the operation of any node to meet these goals by introducing different levels of operation. However, harvested energy is not taken into account. Thus, infinite lifetime goals and dynamic sampling rates are not considered.

The problem of maximizing sampling rates in solar harvesting sensor networks was first discussed by Kansal et al. [1]. They describe a field-monitoring application with the goal of maximizing homogeneous sampling rates while remaining energy neutral. They model the flow in the network using linear equations. Solving the linear equations yields one sampling rate and a corresponding flow. This flow can be sustained by all nodes using no more than the energy neutral consumption rate. To calculate the solution, global knowledge over all nodes, flows, and energy consumptions in the network is required. This is not feasible in a real-world sensor network.

A more viable approach was presented by Fan et al. in [4]. Their approach leads to a number of interesting theoretical results. However, this approach suffers heavily if there are any changes in the routing tree[2]. With SDF [3], we presented a new approach with much lower overhead and higher resilience against adverse routing performance.

III. Implementation

SDF was first introduced in [3]. The contribution in this paper is a first implementation for the popular Contiki platform and corresponding evaluation results from different comprehensive testbed runs, illustrating the performance of the approach. Implementing SDF on hardware leads to significant changes to the protocol.

We will first give a short introduction of SDF (for more details, please refer to [3]), before introducing and discussing different these crucial adaptations required for SDF to run in hardware.

[2]In fact, it was not possible to implement it in our testbed to compare it to C-SDF due to dynamics in the routing graph.

A. SDF Recap

The goal of solar-aware distributed flow (SDF) is to maximize the sampling rates achieved by each node, while ensuring the energy neutral consumption operation. SDF assumes a periodic reporting application. The nodes form a collection tree. Each node has one successor or parent node and might have multiple predecessor P in the routing tree. We note that, in this scenario, energy is mainly consumed by either generating or relaying flow. Each node has to send sampling messages, as well as forward the message from all predecessor P in the collection tree.

The idea of SDF is that each node will issue control messages to the direct predecessors granting them an *allowed flow*. From a node's perspective, SDF consists of three main steps, which are repeated periodically.

- Predict the *consumption rate* c.
- Calculate the own *sampling rate* and the *allowed flow* of the predecessors P.
- Send control messages to all direct predecessors in P and set the own sampling rate.

Algorithm 1: Solar-aware distributed flow

Require: Allowed flow f_a from successor
Ensure: Sampling rate x and allowed flow f_a to direct predecessors
$c \leftarrow$ CONSUMABLE-POWER
$x \leftarrow$ CALCULATE-FLOW
$D \leftarrow$ GET-DIRECT-PREDECESSOR-SET(P)
for all $p \in D$ **do**
  $P_p \leftarrow$ GET-FORWARDED(P, p)
  SEND-FLOW-UPDATE-MESSAGE$(pred, x \cdot$SIZE$(P_p))$
end for
SET-SAMPLING-RATE$(max(x_{min}, x))$

Algorithm 1 gives a sketch of one round of SDF. This will be repeated periodically on each node to yield a maximized sampling rate, utilizing as much harvested energy as possible.

The hardware implementation leads to a number of changes to SDF. Most are highlighted in the next section. The changes that are purely driven by shortcomings in the testbed, e.g., no solar cells and batteries are connected, will be highlighted in the testbed description next section.

B. C-SDF

After introducing the basic mechanism of SDF, we will now describe the implementation on Contiki (called C-SDF for Contiki-SDF). The goal is to maximize the sampling rate of each node during a sampling period S given a prediction of the amount of harvested energy (consumption rate c). Here, S is the time between two SDF control messages and, thus, equivalent to one round of SDF. The sampling rate or message flow is measured as sampling messages m per sampling period ($\frac{m}{S}$). It is important to note that a node can only send a discrete number of messages per sampling period. Here, we chose to implement a conservative approach and, thus, floor the calculated messages per sampling period to the next integer.

C-SDF is built upon ContikiRPL [28], the RPL [5] implementation for Contiki, as data routing layer. RPL builds a destination oriented directed acyclic graph (DODAG) where each node chooses one parent and knows about it's children. This topology is perfectly suited for SDF. RPL builds upon IPv6. We use uIPv6 [29] and the IPv6 stateless address auto-configuration standard [30] to assign IP addresses in a distributed fashion. All messages are sent by UDP, hence, reliability is only given on a per-hop basis by the CSMA/CA MAC of Contiki. Since control messages are only send to child nodes one-hop reliability is sufficient.

To calculate the allowed flow, SDF needs knowledge about the required power for sending (P_{tx}) and receiving (P_{rx}). P_{tx} and P_{rx} are not constant, but fluctuate over time. Each send and receive requires different amounts of energy, due to e.g., the MAC preamble, retransmits for unacknowledged messages, packet size constraints (e.g. splitting a message into multiple frames for sending), etc. We will approximate P_{rx} by calculating a running average over the past m messages received. In Contiki, the application layer has no knowledge of the exact number of messages received and sent on the routing layer. Since each node controls the number of messages it is able to forward for every child node, the maximum number of allowed messages is used as an approximation. P_{tx} can be approximated similarly by dividing the amount of energy needed to send all messages divided by the approximated number of messages forwarded and sent by the node itself.

Since synchronization cannot be assumed, the sampling periods of the parent and child node may not be synchronized, leading to errors in the calculation of the required energy. To ensure synchronized sampling periods every node starts a new sampling period whenever a control message update is received. At the end of each sampling period a small time buffer is introduced to counter small synchronization offsets. Most of the sampling period is dedicated to sampling uniformly given the calculated maximized sampling rate.

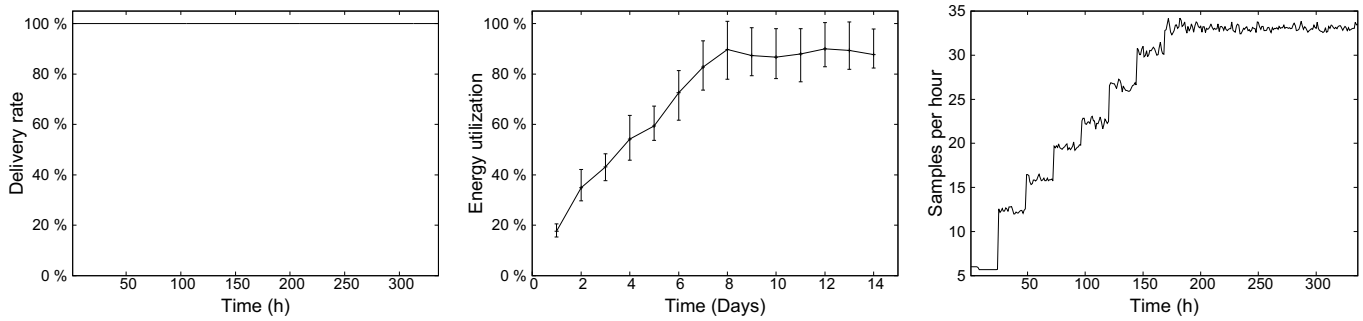
IV. EVALUATION

To illustrate the feasibility, C-SDF has been evaluated on different testbeds and with different wireless link conditions[3]. Before presenting the results, we will introduce the setup and models used to evaluate C-SDF. Afterward, the results on two different testbeds are given. Emphasis was put on generating a wide variety of network characteristics to validate the practicality of C-SDF.

A. Setup & Models

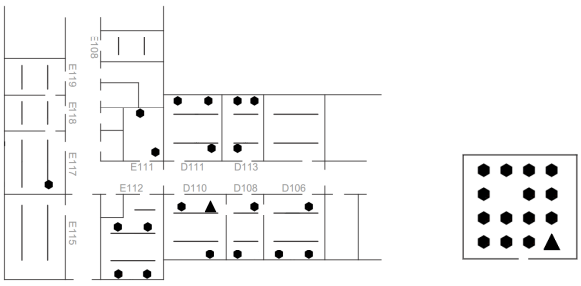
1) Testbed TUDμNet: We used two distinct sites of the TUDμNet testbed in Darmstadt to evaluate SDF under completely different conditions. The first testbed is located inside the computer science building (*Piloty*) at the Technical University Darmstadt. 20 TMote Sky nodes have been deployed over eight rooms on a single floor. The location of each node is displayed in Figure 2(a). All sampling nodes (hexagonal) are transmitting their data to the base station (triangle). This testbed site suffers from high interference.

[3]We also tried to evaluate C-SDF against the work by Fan et al. [4]. However, their work requires static routing trees, which was not feasible given the testbed conditions.



(a) Delivery rate (b) Energy utilization (c) Samples per hour

Fig. 3: Performance of C-SDF (TIZ)



(a) TUDμNet testbed placement (b) TIZ testbed placement

Fig. 2: Placement in the testbeds

The second testbed site is located at *TIZ*, Darmstadt. At that time, the TIZ site had 15 TMote Sky nodes deployed in a single room. They have been arranged in a grid as displayed in Figure 2(b). Again all sampling nodes (hexagonal) are transmitting their data to the base station (triangle). Here, almost no interferences is present.

2) Energy consumption, storage, and harvesting: The nodes in the TUDμNet testbed are connected to a router via USB. The nodes are neither connected to a solar panel nor a battery. Thus, we require means to measure energy consumption, simulate the solar cell and model the battery.

One complication for calculating such values on the mote are that floating point numbers are deactivated by Contiki. We implemented a fixed-point arithmetic to circumvent this limitation. All floating point operations are calculated with a Q15.16 numbers, which logically splits a 32bit integer into 15 integer bits, 16 fractional bits and the two's complement sign bit. The range of Q15.16 numbers is $[-32767.99998474122109375, 32767.99998474122109375]$ with a maximum fraction accuracy of 2^{-16}.

Energy consumption can be estimated using the Energest framework, which is included with Contiki [31]. The Energest framework delivers the number of ticks each hardware module was active. The important modules are cpu (active, sleep) and radio (transmit, receive). In order to be more realistic, we have also added GPS and CO and CO_2 sensors as consumers. For each sample, the sensors as well as the GPS device are activated for a given time. This time is picked uniformly at random and is between $[1, 5]$, $[0.5, 1]$, and $[10, 30]$ seconds for GPS, CO_2 and CO respectively. The time ranges, as well as the energy consumption, are extracted from the respective data sheets [32], [33], [34], [35]. The energy consumption of each module is summarized in Table I.

Component	mAh	mAs
cpu (active)	1.8	0.0005
cpu (sleep)	0.0545	0.00001514
transceiver (send)	20.0	0.0056
transceiver (receive)	17.4	0.0048
CO sensor	3.0	0.0008
CO_2 sensor	50.0	0.0138
GPS modul	20.5	0.0057

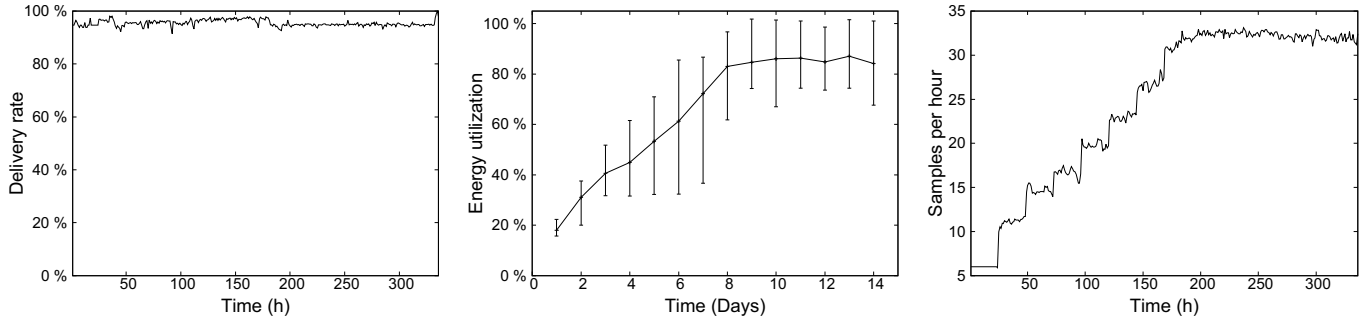
TABLE I: Energy consumption of sensor modules

To simulate the harvested energy coming from the solar panel, we first approximate the solar energy per square meter using Brock's model [36], [3]. The model can calculate the solar energy per square meter for any given day and any given location. However, given the constraint nature of the motes this was approximated for May 7 in Darmstadt only. The approximation in fixed-points number for the solar energy in $\frac{\text{Watt}}{m^2}$ for this day is given by $-9.5455099942671 * 10^{-6} * x^3 + 0.0101828 * x - 567.954$ for the x^{th} minute of the day. This is accurate for the first 12 hours and is then mirrored for the remaining 12 hours. The maximal error between Brock's formula and our approximation is only 0.03%.

This inbound solar energy is damped by two factors per node. The first factor g is taken from an interval $[0.9, 1.1]$. It models persistent difference in sun exposure, e.g., through placement. The second factor d is from the same interval, but recalculated per day. It models weather effects, e.g., clouds. To calculate the energy harvested, we also need to specify the area of the solar panel and the energy efficiency. For our following runs, the area is set to $100cm^2$ and the efficiency of the solar panel to 4%.

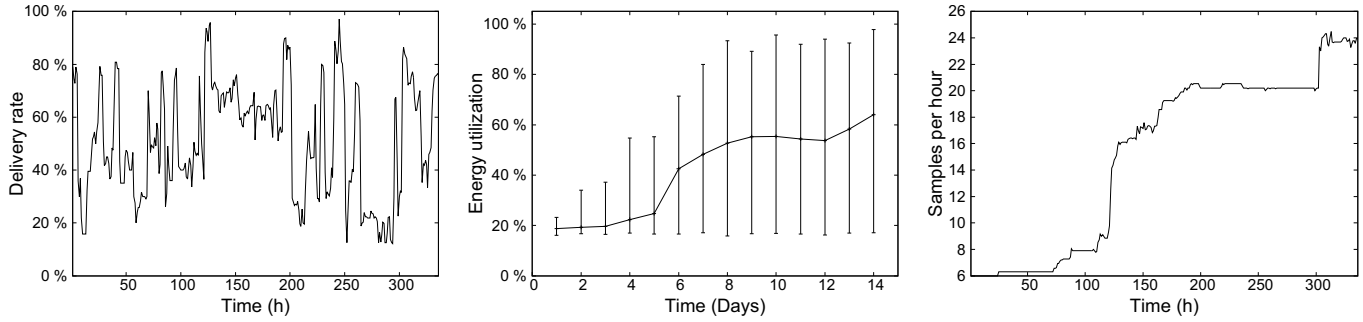
Using these models, we can calculate the consumed and harvested energy. The last step is storing the energy. We simulate the battery as perfect energy storage with a capacity of $1000mAh$ and an initial charge of 70%.

For a node to estimate the consumption rate c, it will



(a) Delivery rate (b) Energy utilization (c) Samples per hour

Fig. 4: Performance of C-SDF (Piloty, short run, almost no message loss)



(a) Delivery rate (b) Energy utilization (c) Samples per hour

Fig. 5: Performance of C-SDF (Piloty, short run, high message loss)

measure the harvested energy over the last 24 hours. The last 14 measurements are used to calculate a very conservative 10% quantile. To take the initial uncertainty into account, the consumption is further multiplied by a *sample confidence* sc (similar to [3]), but only for the first seven days. Obviously, $sc = \frac{1}{7}$ after the first day and increased by $\frac{1}{7}$ per day, until it is equal to 1. Since it is updates once per day, we expect the energy utilization to jump accordingly over the first 7 days.

The longest evaluation run lasted exactly 7 days. We felt that this was not enough to evaluate long term effects of C-SDF. Fortunately the models used for energy consumption, storage, and production can be sped up using a time factor. We ran the energy simulation at eight times speed, which meant that during the 7 day run, 56 days were covered. The sampling interval S was set to 10 minutes.

The next section will present the results from different testbed runs. Keep in mind that SDF [3] achieves around 90% average energy utilization per node. With the conservative implementation choices, e.g., flooring the number of messages per sampling interval, we assume C-SDF to be a little bit lower.

B. Results

We will present results from four different runs on the two testbed sites described above. Each result is one distinct testbed run. Average and variance are calculated over all nodes in the testbed.

First, we present one run on the TIZ site. We expect the results to be close to optimal. Nodes are placed onto a grid with no walls and almost no wireless interference. The results are given in Figure 3.

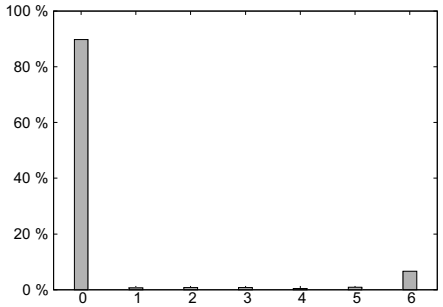
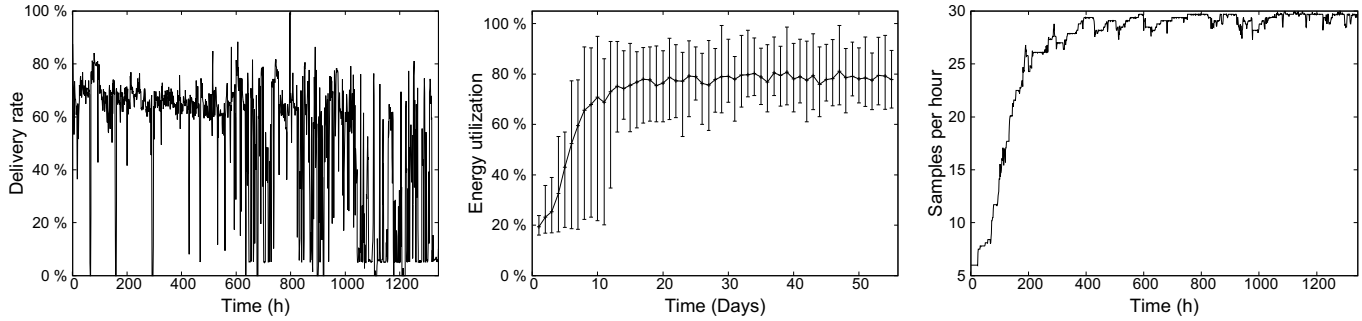


Fig. 6: Number of C-SDF messages received per hour (Piloty, short run, high message loss)

As expected the results are very good. Fig. 3(a) reports the delivery rate for all messages send in the testbed. The delivery rate is 100%, which is astonishing for a real-world testbed. C-SDF behaves exactly as expected in such an environment. The energy utilization (the percentage of harvested energy that was consumed) increases over the first 7 days (due to the sampling confidence as described above). The utilization stabilizes afterward at over 85% on average (cf. Figure 3(b)). This translates into an increase in samples per hour from 6 to around 33 (cf. Figure 3(c)). As expected the values jump over



(a) Delivery rate (b) Energy utilization (c) Samples per hour

Fig. 7: Performance of C-SDF (Piloty, long run)

the first 7 days[4]. The jump also indicates the short stabilization time of C-SDF. As soon as more energy is available the network adapts immediately.

The next three runs we present have been executed at the Piloty site, since we expect the behavior of the network to be more interesting here. First, we present the results of two shorter runs, before diving into the final long run. The results of the short runs are presented in Figure 4 and 5 respectively.

The first run we want to present is almost identical to the run on the TIZ site. The delivery rate is not 100%, but still very good at around 95% (cf. Fig. 4(a)). We can still observe the jumps in sample per hour for the first 7 days (cf. Fig. 4(c)). There is a slight increase in variance for the utilization (cf. Fig. 4(b)), but overall the results are comparable.

This is not true for the second run as illustrated in Figure 5. The delivery rate is around 50% on average, but dipping below 20% on occasion. Compared to an average utilization of over 80% in the last run, here we see only 64% (cf. Fig. 5(b). The interesting point is that the utilization is still constantly increasing. Given these delivery rate it is sage to say that the network is operating under the worst possible conditions. To illustrate how bad the conditions really were, we take a closer look at the number C-SDF control messages received. Figure 6 presents the number of C-SDF control messages received per hour over all nodes. Nodes receive either all (six) control messages or none. Unfortunately, 90% of the time nodes receive no control messages. Even though the message loss was really hampering the overall performance, we can note that the average number of samples per hour is also increasing steadily and jumps to around 24 at the end (cf. Fig. 5(c)). Due to the low overhead of C-SDF, we could afford shorter update cycles. C-SDF works even under adverse conditions.

The last run we present (cf. Fig. 7) was 7 days long and executed on the Piloty testbed site. Due to the scaling factor described last section, this translates to 56 days of C-SDF operation. Figure 7(a) presents the delivery rate over time. It starts off at around 70% with some dips, but drops off rapidly later on. This short period of acceptable delivery rates is enough to stabilize after 7 days at around 80% average energy utilization (cf. Fig. 7(b)). This leaves us with a much improved performance of around 30 samples per hour (cf. Fig. 7(c)). Again, this run illustrates that C-SDF works despite harsh network conditions. We fully achieve our goal of using the harvested energy to increase the service quality. 80% average energy utilization is already an excellent result and, according to other runs, we lose approximately 10.6% in utilization due to flooring the calculated samples per sampling period to the next integer. Applying a more sophisticated method would, thus, boost our performance to over 90%. In line with the simulation results presented in [3].

In summary, C-SDF works even under adverse network conditions. It has neglectable overhead even as we send control messages every 10 minutes. The sampling rate stabilizes quickly. Energy utilization is more than 80%, while ensuring no node is overloaded.

[4]Please note that the utilization does also jump, but is only measured once per day. Hence, the line drawn for readability makes it appear linear.

V. Conclusion

In conclusion, C-SDF is the first implementation of a practical flow control approach for field monitoring applications. It stabilizes even under adverse conditions, where more than 90% of all control messages are lost. The overhead is neglectable even though control messages are send every 10 minutes. The sampling rate improves significantly. Energy utilization is more than 80%, while ensuring no node is overloaded. The utilization could even be increased to over 90% with a more aggressive approach when calculating the message per sampling period. As a next step, we want to pro-actively adapt the RPL routing topology to enable mulit-path routing.

Acknowledgment

This work has been co-funded by the DFG as part of the CRC 1053 MAKI.

References

[1] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 6, no. 4, Sep. 2007.

[2] S. Roundy, D. Steingart, L. Frechette, P. Wright, and J. Rabaey, "Power Sources for Wireless Sensor Networks," in *Wireless Sensor Networks*, ser. Lecture Notes in Computer Science, H. Karl, A. Wolisz, and A. Willig, Eds. Springer Berlin / Heidelberg, 2004, vol. 2920, pp. 1–17.

[3] I. Schweizer, N. Fleischhacker, M. Mühlhäuser, and T. Strufe, "SDF - Solar-Aware Distributed Flow in Wireless Sensor Networks," in *2011 IEEE 36th Conference on Local Computer Networks*, 2011.

[4] K.-W. Fan, Z. Zheng, and P. Sinha, "Steady and fair rate allocation for rechargeable sensors in perpetual sensor networks," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008, pp. 239–253.

[5] T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks (RFC 6550)," 2012.

[6] C. M. Vigorito, D. Ganesan, and A. G. Barto, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," in *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2007, pp. 21–30.

[7] A. Kansal and M. Srivastava, "An environmental energy harvesting framework for sensor networks," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, 2003, pp. 481–486.

[8] C. Alippi and C. Galperti, "An Adaptive System for Optimal Solar Energy Harvesting in Wireless Sensor Network Nodes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 6, pp. 1742–1750, Jul. 2008.

[9] 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*, 2005, pp. 457–462.

[10] K. Lin, J. Yu, J. Hsu, S. Zahedi, D. Lee, J. Friedman, A. Kansal, V. Raghunathan, and M. Srivastava, "Heliomote: enabling long-lived sensor networks through solar energy harvesting," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005.

[11] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM SIGPLAN Notices*, vol. 35, no. 11, pp. 93–104, Nov. 2000.

[12] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," in *Fourth International Symposium on Information Processing in Sensor Networks*, 2005, pp. 463–468.

[13] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Fourth International Symposium on Information Processing in Sensor Networks*, 2005, pp. 364–369.

[14] M. Masateru, T. Morito, H. Morikawa, and T. Aoyama, "Solar Biscuit: A Battery-less Wireless Sensor Network System," in *The 2nd International Workshop on Networked Sensing Systems*, 2005.

[15] P. Sikka, P. Corke, and L. Overs, "Wireless sensor devices for animal tracking and control," in *29th Annual IEEE International Conference on Local Computer Networks*, 2004, pp. 446–454.

[16] P. Corke, P. Valencia, P. Sikka, T. Wark, and L. Overs, "Long-duration solar-powered wireless sensor networks," in *Proceedings of the 4th workshop on Embedded networked sensors*. ACM Press, 2007, pp. 33–37.

[17] C. Alippi, R. Camplani, C. Galperti, and M. Roveri, "A Robust, Adaptive, Solar-Powered WSN Framework for Aquatic Environmental Monitoring," *IEEE Sensors Journal*, vol. 11, no. 1, pp. 45–55, Jan. 2011.

[18] L. Lin, N. B. Shroff, and R. Srikant, "Asymptotically Optimal Energy-Aware Routing for Multihop Wireless Networks With Renewable Energy Sources," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1021–1034, Oct. 2007.

[19] T. Voigt, H. Ritter, and J. Schiller, "Utilizing solar power in wireless sensor networks," in *28th Annual IEEE International Conference on Local Computer Networks*. IEEE, 2003, pp. 416–422.

[20] T. Voigt and H. Ritter, "Solar-aware routing in wireless sensor networks," *Personal Wireless Communications*, vol. 2775, pp. 847–852, 2003.

[21] J. Schiller, A. Liers, H. Ritter, R. Winter, and T. Voigt, "ScatterWeb - Low Power Sensor Nodes and Energy Aware Routing," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 2005.

[22] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[23] T. Voigt, A. Dunkels, J. Alonso, H. Ritter, and J. Schiller, "Solar-aware clustering in wireless sensor networks," in *Proceedings of the Ninth International Symposium on Computers And Communications*, 2004, pp. 238–243.

[24] R. Rajesh, V. Sharma, and P. Viswanath, "Capacity of fading gaussian channel with an energy harvesting sensor node," in *IEEE GLOBECOM*, 2011.

[25] W. Shu, X. Liu, Z. Gu, and S. Gopalakrishnan, "Optimal Sampling Rate Assignment with Dynamic Route Selection for Real-Time Wireless Sensor Networks," in *Real-Time Systems Symposium*, Nov. 2008, pp. 431–441.

[26] S. Bandyopadhyay and E. Coyle, "Spatio-temporal sampling rates and energy efficiency in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp. 1339–1352, Dec. 2005.

[27] A. Lachenmann, P. J. Marrón, D. Minder, and K. Rothermel, "Meeting lifetime goals with energy levels," in *Proceedings of the 5th international conference on Embedded networked sensor systems*, 2007, pp. 131–144.

[28] N. Tsiftes, J. Eriksson, and A. Dunkels, "Low-power wireless IPv6 routing with ContikiRPL," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010.

[29] M. Durvy, N. Finne, A. Dunkels, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, and N. Tsiftes, "Making sensor networks IPv6 ready," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008, pp. 421–422.

[30] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration (RFC 4862)," 2007.

[31] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th workshop on Embedded networked sensors*, 2007, pp. 28–32.

[32] u-blox AG, "ANN-MS active GPS antenna," 2011.

[33] ——, "MAX-6 u-blox 6 GPS Modules," 2011.

[34] Figaro USA Inc., "TGS 2442 - for the detection of Carbon Monoxide," 2005.

[35] ——, "TGS 4161 - for the detection of Carbon Dioxide," 2005.

[36] T. D. Brock, "Calculating solar radiation for ecological studies," *Ecological Modelling*, vol. 14, no. 1-2, pp. 1–19, Nov. 1981.