A System for Efficient Dissemination of Weather Forecasts for Sustainable Solar-Powered Sensors

Christian Renner and Phu Anh Tuan Nguyen
Institute of Computer Engineering
Universität zu Lübeck
Lübeck, Germany
renner@iti.uni-luebeck.de

***Abstract*—Depletion-safe and efficient operation of sensor networks powered by miniature solar harvesters demands precise prediction of the future energy intake. Only recently, methods that join local knowledge and global weather forecasts have been shown to improve the prediction performance. However, making use of global weather forecasts requires network-wide dissemination with low energy and resource consumption. We present and evaluate a system architecture that automatically obtains global weather forecasts from freely available online resources and disseminates them into the sensor network. We implemented our system for sensor network hardware based on TinyOS and Java. We conducted a system integration test to verify the functionality of all components and evaluated the performance of the dissemination algorithm with testbed experiments. While we find that energy consumption is negligible and resource usage is low, dissemination was reliable with a low delay compared to the hourly interval of forecast updates.**

I. Introduction

Wireless sensor networks enable seamless and highly integrated observation and control in several scenarios. Due to their miniature size and wireless communication, they have been used for, e.g., wildlife [1], environmental [2], agricultural [3], and structural [4] applications. They have also started drawing the attention of industry, where sensor networks are employed as cyber physical systems (CPS) [5] to monitor and control production processes.

While the individual sensor nodes are constrained devices with low power consumption, their energy budget is also limited, since sensor nodes are usually battery-powered. To overcome this limitation, a plethora of so-called energy-harvesting sensor nodes [6, 7] have been developed. The term energy harvesting implies in this context, that nodes are powered from renewable energy sources, solar power being most widely used. To bridge periods of low or no energy harvest, rechargeable batteries or supercapacitors are used as energy buffers, where the latter are particularly environmentally friendly and allow a concise estimation of their residual energy by means of their terminal voltage [8].

However, the amount of harvestable energy is restricted, as solar panels must usually meet the tiny dimensions of sensor nodes to maintain non-intrusiveness and low cost. Therefore, sensor nodes still rely on, e.g., energy-efficient medium access and routing protocols. To configure these protocols—usually by defining their duty cycle—the harvest has to be well known. Yet, the energy produced by a solar cell cannot be predicted precisely before network deployment and operation. This is due to local and global influences. The former encompass positioning and environment effects that lead to a specific harvest pattern of each individual node, caused by, e.g., shadows from buildings and plants. The latter is due to weather conditions (e.g., snow and rain) and seasonal effects (e.g., height of the sun) that usually affect all nodes in a network.

As a result, algorithms to adapt a node's consumption to match the available energy harvest have been devised [9, 10]. These rely on another category of algorithms to predict the future harvest. In the past years, most effort has been spent on making use of locally available data for harvest prediction [11, 12]. In changing weather conditions, these approaches lead to a poor performance and may even fail, leading to (temporary) node depletion. Only recently, we have proposed a method to incorporate weather forecasts to improve harvest prediction quality [13]. However, dissemination of weather forecasts into the network is a mandatory yet unmet prerequisite for adopting our prediction algorithm in real-world networks. To the best of our knowledge, there are no solutions to close this gap.

In this paper, we present a system solution to disseminate weather forecasts based on cloud cover into an energy-harvesting sensor network, so that the nodes in the network are enabled to produce more precise harvest forecasts, shrink the chance of depletion, and increase the utility of harvestable energy. Our approach exploits existing network traffic and compresses the weather forecasts to reduce the amount of additional bandwidth utilization and energy expenditure. Moreover, a base station application automatically obtains weather forecast from a freely available online weather forecast service and compresses the data. This data is subsequently disseminated in the sensor network with low delay (compared to the update intervals of weather forecasts) and decompressed on the sensor nodes with low resource consumption.

II. Background and Related Work

This section presents related research in context of this paper. First, we briefly discuss the basics of sensor networks and, second, motivate the need for online consumption adaptation of sensor nodes. Third, we sketch techniques for harvest forecasting, and fourth, we discuss existing methods for data dissemination in sensor networks.

978-3-901882-70-8

A. Energy-Harvesting Sensor Networks

Wireless sensor networks consist of tiny sensor nodes that are, in addition to the sensors, equipped with low-power and resource-constrained computing and wireless communication devices. Their typical field of application is that of data collection, where all nodes in the network create sensor samples at fixed intervals. All data is collected by a central node, the sink, which forwards the data to a more powerful computing device, frequently called base station.

Traditionally, sensor nodes have been powered with batteries, limiting their lifetime to several months. In the last decade, researchers started replacing the batteries with micro-sized renewable energy sources, so-called energy harvesters [6, 7], to achieve an unattended, uninterrupted, and virtually unlimited operation. Here, solar cells have been frequently adopted, as solar energy is available in most outdoor sensor network deployments. To maintain the tiny dimensions of the sensor nodes, the solar cells yet have to be relatively small, hence leading to an average energy intake smaller than the consumption of a fully operational node.

Therefore, interrupted and unlimited operation requires low-power protocols to reduce the consumption of a node. Since the radio contributes highly to a node's consumption, low-power MAC and data collection protocols have been devised in the past [14, 15, 16, 17]. Their common principle of operation is to duty-cycle a node's radio. To reduce protocol complexity, duty cycles are asynchronous. While nodes wake up periodically to check for incoming packets, this implies that a node willing to transmit a packet has to wait until a receiver wakes up. The corresponding rendezvous is achieved by two different concepts. Low-power-listening (LPL) protocols let the sender of a data packet either send a long preamble or repeat the data packet until a receiving node wakes up, receives the packet, and sends an acknowledgment. Low-power-probing (LPP) protocols let the sender of a data packet wait until it receives a beacon from a node that wakes up, where each node is required to send such a beacon when it wakes up periodically. Based on LPP, Unterschütz et al. have presented a routing protocol in [17] that does not require link quality estimation and routing tables.

B. Online Consumption Adaptation

Energy harvesting generally enables perpetual node and network operation. Solar energy is the predominant energy source, since most sensor networks are deployed outdoors. Unfortunately, solar energy has an unsteady nature, exhibits a diurnal pattern, and is highly location-dependent. Therefore, the amount and times of harvested energy may heavily vary from one node in a network to another. It is hence impossible to plan the amount of harvested energy for each node in the network prior to deployment. As a consequence, the tolerable consumption of a node is not known and the consumption-deciding protocol parameters cannot be chosen.

To solve this problem, algorithms for online consumption adaptation such as [10, 18, 19] have been proposed. These algorithms aim at two contracting goals. On the one hand, they try to reduce the risk of (temporary) node depletion in order to avoid network disconnects and data gaps. On the other hand, they try to maximize the utility of a sensor node in terms of using as much harvestable energy as possible—e.g., faster MAC reaction times lead to a higher consumption while reducing the latency. While it is possible to implement and run such algorithms based on the residual node energy only, they profit highly from predictions of the future harvest.

C. Harvest Prediction

Several researchers have proposed and investigated methods for prediction algorithms that are light-weight enough to be run on sensor nodes. While a few works exist that propose model-based prediction algorithms [20, 21], the predominant approach is to divide a day into time slots and collect historical data for each time slot [18, 11, 12]. By evaluating this historical data, predictions for the energy harvest in the next time slot or in time slots of the next day are generated. Existing algorithms are restricted to local information only, i.e., each node tracks its own history of energy harvest and creates predictions solely based on its own local knowledge.

While these algorithms are suitable to identify the local harvest pattern of the nodes in a network in stable weather conditions, the fail upon changing conditions, e.g., when cloudy days follow a series of sunny days. In this particular instance, nodes tend to continuously overestimate the energy intake and may hence experience (temporary) depletion. In our previous work in [13], we have shown that combining algorithms to identify local harvest patterns can be effectively merged with global cloud-cover forecasts to improve predictions and hence enhance the performance of online consumption adaptation algorithms. To make global weather forecasts available on each node in a sensor network, these forecasts have yet to be disseminated.

D. Data Dissemination

For the general purpose of data dissemination in sensor networks, protocols such as Trickle [22] and CodeDrip [23] have been developed. They aim at rapid dissemination (usually within a few seconds) and thereby counteract the benefits of low-power MAC protocols by an increased energy consumption. However, cloud-cover forecasts have long forecast horizons of several days and are updated infrequently, hourly to be the lowest known update interval. A latency in the order of a few minutes is hence acceptable.

An approach such as RoCoCo in [24], which enables low-power command dissemination in sensor networks at the cost of an increased delay, is hence appealing. However, RoCoCo does not support the dissemination of generic data and adds overhead for addressing, which is not required for disseminating harvest forecasts relevant for all nodes in the network. Moreover, RoCoCo does not offer means to track the latency, or delay, of a harvest forecast between its creation and reception by a node. In the absence of time synchronization, which is the case in many sensor network deployments, this

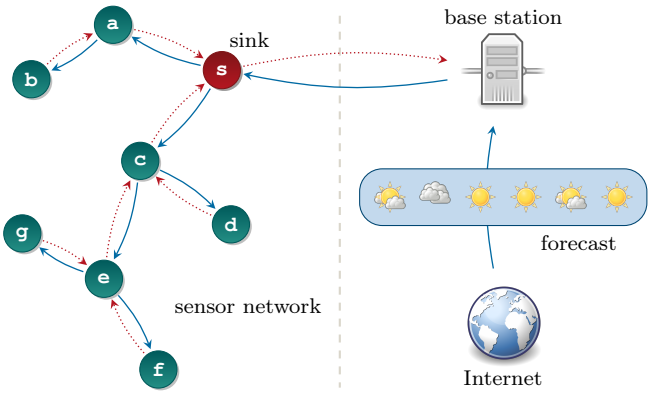

Fig. 1. System architecture: Data collection sensor network consisting of energy-harvesting sensor nodes. The base station obtains cloud-cover forecasts from online weather portals, compresses and forwards them to the sensor network sink s. The data collection network paths (red dotted arrows) are used to disseminate weather forecasts (solid blue arrows) in the network.

information is yet required to align the forecast with the locally generated, time-slot pattern.

III. System Architecture

In this section, we present our system architecture to provide all nodes of the energy-harvesting sensor network with up-to-date weather forecasts in order to improve their harvest predictions and enable a more reliable operation that uses the available solar energy to an improved capacity.

The presented system architecture is illustrated in Fig. 1. The sensor network on the left-hand side of the figure consists of solar-powered nodes that adjust their consumption (e.g., their radio duty cycle or sensing rate) based on their current energy level and the expected future energy harvest. The main purpose of the network is to measure and collect sensor data without measurement gaps due to (temporary) depletion.

All data is forwarded, possibly over multiple hops, to the sink. The latter is connected to a more powerful computing device, the base station. This base station periodically obtains weather forecasts from an online weather portal. All necessary information is extracted and compressed to reduce bandwidth usage and energy overhead during dissemination in the network. Next, the base station forwards the compressed forecasts to the sink, from where the forecast is disseminated in the multi-hop sensor network. For this purpose, the existing routing infrastructure—i.e., the paths used for data collection—is exploited.

In the following, we discuss the overall idea in more detail but at a high level perspective. We provide technical and implementation details and an evaluation in subsequent sections.

A. Weather Forecasts and Compression

The Internet offers a plethora of weather portals with freely available weather forecasts. The latter contain information such as cloud cover and hourly sunshine duration. Forecast horizons are multiple days and the typical granularity is an hour. As indicated in our previous research in [13], there is a high correlation between cloud-cover information and harvested solar energy. We hence developed a server application that periodically reads the latest cloud-cover forecast from an online weather portal[1]. Here, we rely on parsing the generated HTML forecast output that is freely available. Unfortunately, the APIs for direct data access are liable to pay costs.

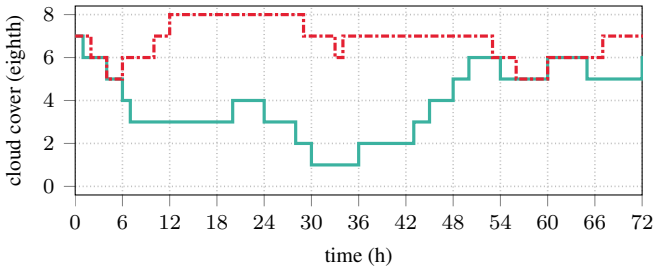

Fig. 2. Two example cloud-cover forecasts with a resolution of one hour and a horizon of three days.

Cloud cover is a metric that indicates the cloudiness of the sky in eighth, leading to nine discrete values. A naive encoding of these forecasts with an hourly resolution yields a size of 12 B per forecasted day. Considering that state-of-the-art sensor network radio chips [25] offer a packet size of up to 127 B, it is hence possible to transmit multi-day cloud-cover forecasts within a single packet into the network. However, cloud-cover forecasts are only relevant for daytime and slowly change over time, cf. Fig. 2, hence leaving room for compression. Using a simple block-to-block code with nighttime omission, compression savings of up to 60%, equal to an average compressed forecast length of 10 B for three-day forecasts, are achievable [26]. The major benefit of compression is the possibility to attach, or piggy-back, these forecasts to already existing network packets. Doing this avoids additional (consumption) overhead inherent to low-power MAC protocols, cf. Sec. II-A. Besides, bandwidth usage and energy consumption during dissemination are decreased when the amount of transmitted data declines. The risk of packet loss is decreased at the same time.

B. Network-Wide Forecast Dissemination

Following the compression of the forecasts, the latter must be disseminated in the multi-hop network. Naturally, this distribution should lead to as low communication overhead as possible in order to limit the extra energy expenditure and bandwidth utilization. To meet these ends, we propose to leverage the already existing network traffic resulting from data collection. While data packets travel from the nodes towards the sink, acknowledgment packets travel in the opposite direction and therefore match the required flow of forecast dissemination. We exploit this fact and aim at attaching, or piggy-backing, the compressed forecasts to these acknowledgments similar to RoCoCo, cf. Sec. II-D.

[1] for our prototype implementation, we used the forecasts available from http://www2.wetterspiegel.de/

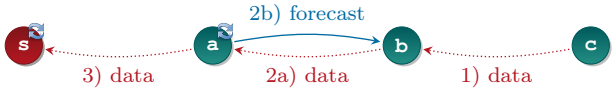

Fig. 3. Combined data collection and forecast dissemination. While the data packet from node c travels all the way to the sink (dotted red arrows), the forecast only propagates one hop further into the network (solid blue arrow). Numbers on the edges indicate the sequence of events, circled arrows mark nodes already holding the latest forecast update before communication.

TABLE I
CODES FOR PARTIAL DELTA CODING

delta	frequency (%)	code	code length
0	82.67...	0	1
1	8.49...	10	2
-1	8.31...	110	3
else	0.53...	111xxxx	7

The underlying idea is to propagate forecasts in the network similar to Trickle [22] but without the need for extra packets and timers. Technical details are provided in Sec. IV-B. These advantages come at the cost of an increased delay, since for each packet traveling along a path, the forecast can only propagate one hop into the network. This fact is depicted in Fig. 3, where node a has already received the updated forecast from the sink (indicated by the circled arrows in the upper right node corner), whereas nodes b and c hold an outdated forecast. In the example, node c has generated a new data packet that travels to the sink with two intermediate hops. Since node c sends its packet to node b, before node b receives the updated forecast from node a, node c can only receive the forecast upon the following packet transmission. We present theoretical delay limits in Sec. IV-C and evaluate real-world performance in Sec. V.

IV. TECHNICAL REALIZATION

As discussed in Sec. III-B, we intend to send compressed harvest forecasts piggy-backed on the acknowledgments of collection data traffic. In this section, we explain the used compression method and its implementation, before we provide details about the dissemination strategy. Finally, we derive theoretical limits for the expected dissemination delay that we will analyze in Sec. V.

A. Forecast Compression and Decompression

We have shown the theoretical advantage of Daytime Delta Coding (DDC) in our preliminary work [26]. Before providing implementation details, we briefly revisit the function of DDC.

DDC makes use of two properties of cloud-cover forecasts. First, it considers differential changes between each two adjacent, hourly cloud-cover forecast values rather than absolute numbers. The predominant case ($> 82\%$) is that of no change between two values. Changes of ± 1 are the next most frequent cases with less than 9% occurrence each. All other cases—i.e., absolute changes of more than 1—have an accumulated relative frequency of less than 1%. The average number of

```
1:  procedure DecodeForecast(N, sunrise, sunset)
2:      isDay ← sunrise > sunset;
3:      for i ← 0, ..., N − 1 do
4:          if i = sunrise then
5:              isDay ← true;
6:              sunrise ← sunrise + 24;   { next sunrise in 24 hours }
7:          else if i = sunset then
8:              isDay ← false;
9:              sunrise ← sunset + 24;   { next sunset in 24 hours }
10:         if isDay then
11:             forecast[i] = decodeNextValue();   { daytime }
12:         else
13:             forecast[i] = DEFAULT_NIGHT_VALUE;   { nighttime }
```

Fig. 4. Decoding algorithm for cloud-cover forecasts with a horizon of N values (hours) and the first relative occurrence of sunrise and sunset in hours [26].

bits for encoding a single forecast value can be efficiently reduced using the Huffman code as shown in Table I. Note that in case of absolute changes higher than one, the absolute forecast value is encoded using 4 bit and a prefix. Due to the choice of the codes, decoding of the individual forecast values boils down to counting the number of one-bits before the next zero-bit in the compressed forecast bit stream.

Second, DDC makes use of the fact that, over a period of a complete year, approximately half of the forecast values are obsolete, since they fall into nighttime when no solar energy is available. Therefore, DDC omits all forecast values that fall into nighttime but adds information about sunrise and sunset. Here, sunrise and sunset times are the offset (in hours, rounded to the previous or following full hour, respectively) of sunrise and sunset from the time of forecast creation. Our practical implementation requires 5-bit integers for both values, hence leading to a static overhead of 10 bit. Figure 4 shows the fundamental algorithm behind the decoding performed on the sensor nodes.

At this point, we want to point out that compression is only performed on the more powerful base station, whereas only decompression is executed on the energy- and resource-constrained sensor nodes. Depending on the frequency of consumption adaptation, it may be useful to perform the decompression only once upon receiving an updated forecast. In this case, however, each node needs to store both the compressed and decompressed forecast.

Our reference implementation of the decompression algorithm for TinyOS occupies less than 600 B of ROM and up to 20 B of RAM, of which 10 B are only locally used in functions. Runtime is linear in the size of the (cloud-cover) forecast in bits. Storage space for the forecast is linear in the number of forecast days and cloud-cover values, respectively.

B. Network-Wide Forecast Dissemination

Harvest forecasts are updated periodically and disseminated into the network. To decide whether a forecast is newer than another, we assign version numbers to all forecasts. These version numbers are incremented by the base station when it generates a new forecast. For practical implementation, we use a ring counter to keep memory consumption and packet

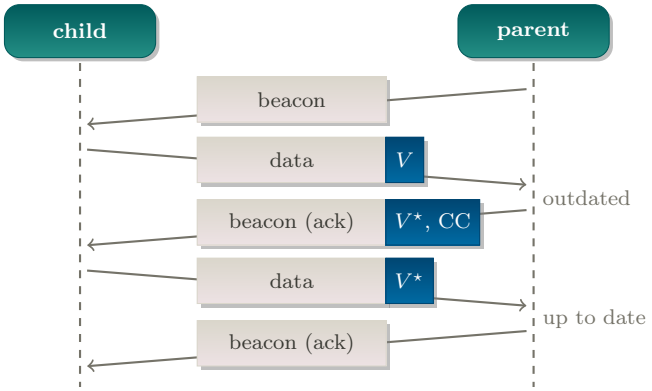

Fig. 5. Cloud-cover (CC) forecast dissemination during data collection for a receiver-initiated routing/MAC protocol. Upon forecast version mismatch (the parent holds the newer forecast with version $V^\star > V$), the parent attaches the new forecast to the collection data acknowledgment.

overhead low. In our current implementation, we use an 8-bit counter. Since old forecasts become obsolete once an update is received, each node is only required to store the latest forecast. Local storage is hence limited to a few ten bytes.

The general concept of dissemination is as follows. Whenever a child node transmits a data packet to a parent node—i.e., when forwarding data towards the sink—it attaches the version number of its most recent cloud-cover forecast. Upon reception of a data packet, the parent compares the attached forecast version number with its local forecast. Only if the received version number indicates that the child holds an outdated forecast, the parent piggy-backs its forecast on the software acknowledgment sent out to the child.

This concept is applicable to all routing protocols based on either low-power-listening (LPL) or low-power-probing (LPP) MAC protocols. Figure 5 shows the concrete packet flow for an LPP-based collection protocol similar to ORiNoCo/RoCoCo. For an LPL-based collection protocol, the first beacon is replaced by several (identical) data packets from the child until the parent answers with its first acknowledgment.

With the proposed dissemination strategy, harvest forecasts travel slowly from the sink to the nodes. The actual delay between forecast creation and reception by a node depends on its distance to the sink, the sensing rate, and the network density (cf. Sec. IV-C). It is expected to range from a few seconds to several minutes. However, nodes need to know the time of forecast creation for a proper alignment with their local harvesting pattern (cf. Sec. II-C). To enable nodes to assess dissemination delay without the need for time synchronization, we attached a 16-bit field to the forecasts that indicates its age w.r.t. the time of its creation. The age is measured in seconds and can track a dissemination delay of several hours.

In short, the forecast age is tracked by summing up the individual delays introduced by each intermediate hop from the sink to a node. Technically, this works as follows. The base station initializes the age with 0 and forwards the forecast immediately to the sink, which records its local time upon reception of the forecast. Whenever the sink piggy-backs the

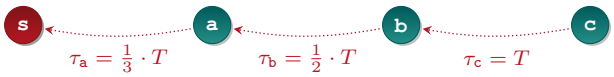

Fig. 6. Average send interval of nodes arranged in a line topology, where all nodes create a new data packet with common interval T.

forecast on an acknowledgment, it reads its local time and calculates the forecast age from the time difference since it received the forecast. The latter is written to the age field of the transmitted forecast. All nodes in the network proceed analogously. In contrast to the sink, however, they also store the age of a received forecast upon reception and add this value to the locally recorded delay when they forward a forecast.

C. Theoretical Dissemination Delay

To put the numbers of our performance evaluation in Sec. V into context and to enable estimating the dissemination delay prior to deployment, we carried out a theoretical but simplified analysis of the dissemination delay inferred by the proposed dissemination strategy. In particular, we assume that the waiting times caused by the underlying asynchronous, duty-cycled MAC protocol are small compared to the average send interval of the nodes and can hence be neglected. We also assume that all nodes take samples in common but non-synchronized intervals T and create a new data packet at the same interval. Moreover, we ignore the influence of packet loss and link fluctuation. We present and summarize our results in the following.

An upper bound of the delay $d_\mathtt{n}$ experienced by node $\mathtt{n}$ can be derived from the one-hop delay of each node in the network. Children of the sink hence receive an updated forecast with a delay of at most T. For each hop in the network, dissemination delay increases iteratively by T. If node $\mathtt{n}$ has a distance of $h_\mathtt{n}$ hops to the sink, its delay is bounded by

$$\sup d_\mathtt{n} = h_\mathtt{n} \cdot T\,. \tag{1}$$

With typical sampling intervals of a few minutes and low hop counts, achieving worst-case dissemination delays of only a few minutes is possible. In consideration of the fact that forecasts are updated at most once an hour, the proposed dissemination strategy hence appears feasible.

Furthermore, the average dissemination delay is smaller than indicated by Eq. (2), since intermediate nodes may receive updates considerably faster. This results from their task of forwarding packets from nodes in their subtree towards the sink. Since the calculation of the average dissemination delay heavily depends on the actual (subtree) topology, we concentrate on the worst-case average that is achieved for the leaf node of a line topology—here, the number of hops is maximized for a constant number of nodes.

Given N nodes in a line topology, the average send interval $\tau_\mathtt{n}$ of node $\mathtt{n}$ with distance $h_\mathtt{n}$ (in hops) to the sink is

$$\tau_\mathtt{n} = \frac{T}{h_\mathtt{n}}\,. \tag{2}$$

Figure 6 shows an example with 4 nodes. The average delay is hence calculated through the summation of the individual delays. For a leaf node $\mathtt{n}$ in a N-node line topology with distance $h_{\mathtt{n}} = N$ to the sink, the average delay is given by

$$d_{\mathtt{n}}^{\varnothing} = \sum_{i=1}^{h_{\mathtt{n}}-1} \frac{1}{i} \cdot T \tag{3}$$

For any node $\mathtt{n}$ with $h_{\mathtt{n}}$ hops to the sink, Eq. (3) can be used as an (upper bound) approximation for the expected average dissemination delay.

V. Evaluation

In the following, we present our evaluation of the presented system. First, we analyze the performance of forecast dissemination in the sensor network. Second, we explain how we validated the functionality of the entire system.

A. Forecast Dissemination

1) Evaluation Setup: We evaluated the performance of our dissemination approach using WiseBed (Lübeck site) [27]. This testbed is currently comprised of 48 TelosB nodes, of which we used up to 15 nodes for our experiments. With a larger number of nodes, we experienced consistency problems with the produced log files. However, the used number of nodes allowed us to produce general observations.

At this point, we concentrated on the dissemination aspect, so that we deployed a modified version of the sink that generates forecast updates of a constant, given size every 5 min. Note that this is a fraction of the anticipated forecast dissemination interval of 1 h in real deployments. We chose this value in order to speed up the evaluation process and gain more results. Since the nodes were able to receive all forecast updates (in time), the results are also applicable to longer update intervals.

Each experiment was run for at least 3 h, leading to 30 generated and disseminated forecasts. Unless otherwise noted, we used the following parameter set for the ORiNoCo data collection layer. Each node maintained a buffer queue of 30 (data collection) packets for the case of temporary connectivity loss. Unless congestion occurred, collection data were sent at the next transmission opportunity. On the data collecting nodes, the sleep interval between beacons was set to 750 ms with a random variation of 10% to avoid node synchronization. To achieve higher delivery, the sink has been configured to send beacons every 125 ms. After sending a beacon, nodes waited 8 ms for an incoming data packet before returning to sleep. We used the standard path cost metric—i.e., the hop count—for ORiNoCo and configured the duplicate detection to maintain a packet history of 10 packets.

2) Influence of Network Size: We analyzed the influence of network size and topology on dissemination. For this purpose, we ran experiments with networks consisting of 5, 10, and 15 nodes (including the sink). Each node created a data packet every 60 s, where the first packet was created with a random offset of at most 60 s after node reboot in order to simulate the behavior of an asynchronously started and operated network.

TABLE II
Dissemination delay in a 5-node network

node	avg. hop count	avg. send interval (s)	avg. delay (s)
1	1.0	50.9	32.5
52	1.2	32.9	20.6
56	1.2	33.0	30.4
62	2.2	50.4	72.0

TABLE III
Dissemination performance for different network sizes

nodes	avg. hop count	avg. delay (s)	avg. delay error (s)
5	1.1	31.1	0.03
10	2.6	28.6	0.02
15	2.9	25.5	0.04

First, we studied dissemination success and overhead. In all experiments, all 30 forecast updates were successfully received by each node. The effort for this is measured by the ratio of sent and received forecast updates. It ranges from 1.025 (an overhead of 2.5%) in the 5-node network to 1.12 (an overhead of 12%) in the 15-node network. This is an acceptable performance in terms of packet loss and leads to almost unnoticeable energy overhead, since the frequency of piggy-backing forecasts on beacons is extremely low (less than 1% in all experiments).

Next, we analyzed dissemination delay and its influencing factors. In the small 5-node network (cf. Table II), three nodes delivered their data packets in most cases directly to the sink—i.e., they are one-hop neighbors of the sink most of the time. However, their average dissemination delay differs, confirming that it is difficult to estimate dissemination delay based on a single characteristic such as hop count. At the same time, the figures support the upper bounds of 60 s, 90 s, and 105 s derived from Eq. (3) in Sec. IV-C.

A comparison of the average delay in the three different networks, displayed in Table III, reveals that despite the tendency of larger networks to grow deeper, the average delay is only affected slightly. A deeper analysis shows that the larger networks have few leaf nodes and that the inner nodes serve as routing parents more frequently. As a result, they send data packets more frequently.

Finally, we analyzed the accuracy of the delay tracking mechanism explained in Sec. IV-B. Throughout all experiments, the average error is roughly −19 ms. In all but two cases, the errors range between −1.21 s and 1.04 s. In the two exceptional cases, errors where as large as −64 s. Unfortunately, we could not identify the source of this large deviation but suspect that data corruption during the logging process has occurred. We (still) noted this behavior in rare events.

Figure 7 displays the cumulative distribution functions (CDFs) of the delay errors for the three network sizes. All curves are similar, indicating that delay errors are almost equally distributed. However, the CDF becomes slightly more shallow for networks with more nodes, which implies a higher

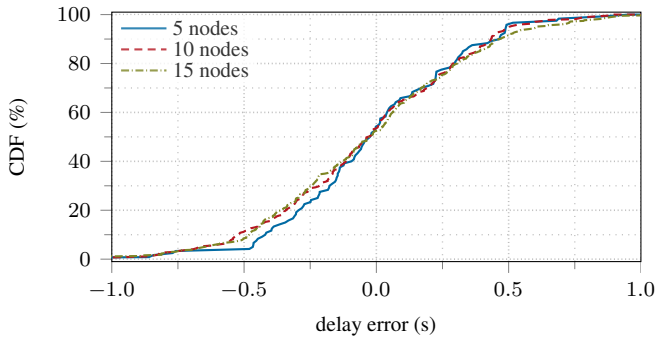

Fig. 7. Cumulative distribution function (CDF) of the delay errors for three network sizes and 30 forecast updates disseminated in the network.

frequency of larger errors. This is caused by the higher amount of long paths, on which errors are more likely to sum up. Despite this observation, delay errors are at a low level and hence acceptable for the intended purpose.

3) Data Collection Interval: We also analyzed the impact of the network traffic on dissemination by varying the data creation interval of each node from 30 s to 2 min. The per-node results are depicted in Fig. 8. The figure shows that in most cases, there is a near-linear relationship between the data creation interval of the nodes and the dissemination delay. As outlined in Sec. IV-C, this is an expected behavior, since dissemination delay is primarily impacted by the average transmit intervals of the individual nodes. The latter scale almost linearly with the per-node packet creation interval, hence the linearity of dissemination delay.

Furthermore, the results indicate that the standard deviation also increased with the packet creation interval. This is due to the fact that in networks with lower traffic, packet loss has a higher impact on dissemination delay. However, there are exceptions to this rule. As an example, node `49` reports the same (average) delay for data creation intervals of 60 s and 90 s. This is caused by a slightly changed network connectivity in between the experiment runs, leading to a different packet flow in the network. In the second case (90 s), node `49` forwarded less packets than in the other experiments.

4) Harvest Forecast Sizes: As a last step, we evaluated the influence of harvest forecast sizes on dissemination. For this purpose, we ran experiments with 15 nodes with harvest forecast sizes of 4 B, 10 B, and 20 B based on the results in [26]. Each node created a new data packet for collection every 60 s.

The reason of this investigation was to clarify whether longer forecasts promote beacon loss, leading to a larger delay, and interfere with the tight timings of ORiNoCo. Our results indicate none of these effects for the considered forecast sizes. We noted successful reception of all forecast updates and an average delay between 27 s and 28.8 s in the three experiments, where the smallest figure was surprisingly observed for 20 B forecasts updates. Average per-node delays were 44.2 s, 50.1 s, and 38.0 s for forecast sizes in increasing order.

5) Limitations: The current dissemination concept relies on steady data collection. One major implication is that in case of, e.g., event-detection scenarios, dissemination delay will deteriorate and become unpredictable. This can be overcome by dummy collection data. In the easiest case, such data would be created periodically. A better approach would be to exploit the periodicity of forecast updates, so that nodes start sending out dummy collection data when their current forecast turns older than the update interval. We will investigate this issue in our future work.

B. System Integration Test

Finally, we validated the functionality of the entire system in a two-tired process.

First, we tested the correctness of the compression and decompression implementations. For this purpose, we fed the compression algorithm, running on the base station, with the cloud-cover data set from [26]. The compressed forecasts were sent to a Memsic Iris sensor node, attached via the MIB520 interface board over the serial line. Upon reception of the compressed forecasts, the software running on the sensor node performed the decompression and sent the result back to the base station via the serial line. As a final step, the original input forecast was compared with the result produced by the sensor node. With this setup, we tested typical configurations. In particular, we varied the number of forecasted days from 1 to 4 and used time resolutions of 1 to 4 h.

Second, we manually compared cloud-cover online forecasts with the results produced by parsing those online forecasts and converting them to the format needed by the station for compression.

VI. Conclusion

Providing solar-harvesting sensor nodes with global weather forecasts reduces their risk of temporary depletion and increases their utility in terms of, e.g., a lower data collection delay. We presented a system architecture and a fully functional implementation that periodically obtains weather forecasts from freely available online weather portals and disseminates these in the sensor network. To achieve a low energy and resource overhead, we apply data compression and piggy-back the forecasts onto existing collection data traffic. The results of several testbed experiments show that all forecasts are eventually received by all nodes. Delays are in the region of several seconds and below 2 min in all cases. These are acceptable numbers when considering that forecasts are updated at most once an hour. We also derived theoretical worst-case estimates for the dissemination delay that we validated experimentally.

As a next step, we will deploy a solar-harvesting, multi-hop test network that makes use of the presented system. We are currently implementing the required changes for the online consumption adaptation algorithm to integrate weather forecasts disseminated into the network. Furthermore, we intend to improve our dissemination strategy to cope with low sensing rates and event-driven applications.

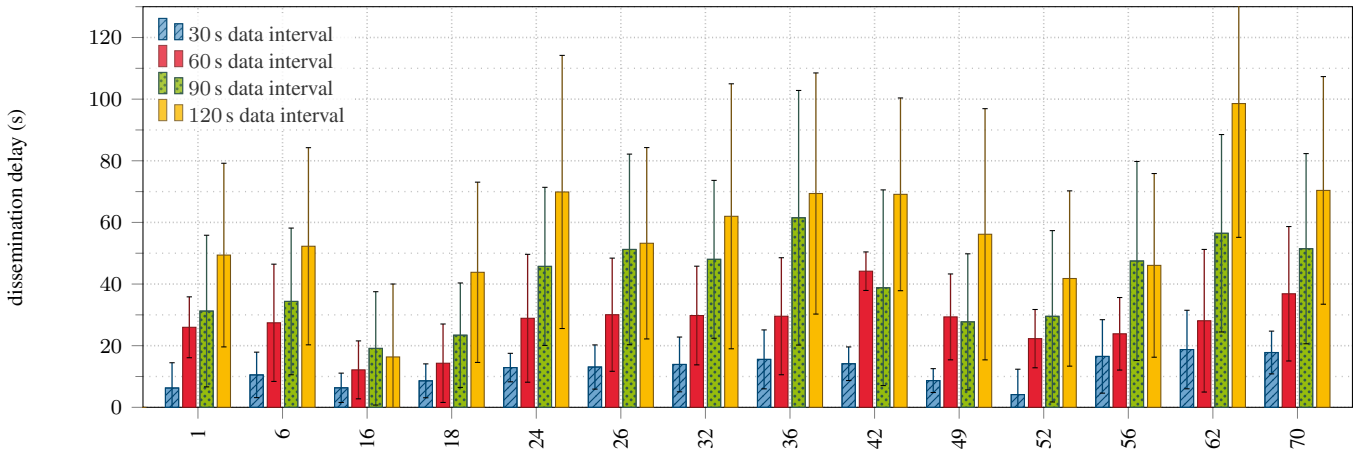

Fig. 8. Dissemination delay of individual nodes for experiments with different data creation intervals. Bars indicate the average delay, error bars show the single standard deviation. Results are based on 30 delay values (forecast updates) for each node and experiment.

REFERENCES

[1] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi, "Hardware Design Experiences in ZebraNet," in *Proc. ACM Intl. Conf. on Embedded Networked Sensor Systems*, ser. SenSys, Nov. 2004.

[2] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Wöhrle, and M. Yuecel, "Operating a Sensor Network at 3500 m Above Sea Level," in *Proc. ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks*, ser. IPSN, Apr. 2009.

[3] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard Computing: Sensor Networks in Agricultural Production," *Pervasive Computing*, vol. 3, no. 1, Jan. 2004.

[4] L. Mottola, G. P. Picco, M. Ceriotti, S. Guna, and A. Murphy, "Not All Wireless Sensor Networks Are Created Equal: A Comparative Study On Tunnels," *Transactions on Sensor Networks (TOSN)*, vol. 7, no. 2, 2010.

[5] E. Lee, "Cyber Physical Systems: Design Challenges," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-8, Jan. 2008. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html

[6] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design Considerations for Solar Energy Harvesting Wireless Embedded Systems," in *Proc. ACM/IEEE Intl. Symp. on Information Processing in Sensor Networks*, ser. IPSN, Apr. 2005.

[7] C. Lu, V. Raghunathan, and K. Roy, "Efficient Design of Micro-Scale Energy Harvesting Systems," *Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, Sep. 2011.

[8] A. Weddell, G. Merrett, T. Kazmierski, and B. Al-Hashimi, "Accurate Supercapacitor Modeling for Energy-Harvesting Wireless Sensor Nodes," *Transactions on Circuits and Systems II (TCAS-II): Express Briefs*, vol. 58, Dec. 2011.

[9] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Adaptive Power Management in Energy Harvesting Systems," in *Proc. Conf. on Design, Automation and Test in Europe*, ser. DATE, Apr. 2007.

[10] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan, "Adaptive Duty Cycling for Energy Harvesting Systems," in *Proc. Intl. Symp. on Low Power Electronics and Design*, ser. ISLPED, Oct. 2006.

[11] J. Recas Piorno, C. Bergonzini, D. Atienza, and T. Simunic Rosing, "Prediction and Management in Energy Harvested Wireless Sensor Nodes," in *Proc. Intl. Conf. on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology*, ser. VITAE, May 2009.

[12] D. Spenza, C. Petrioli, and A. Cammarano, "Pro-Energy: A Novel Energy Prediction Model for Solar and Wind Energy-Harvesting Wireless Sensor Networks," in *Proc. Intl. Conf. on Mobile Ad-Hoc and Sensor Systems*, ser. MASS, Oct. 2012.

[13] C. Renner, "Solar Harvest Prediction Supported by Cloud Cover Forecasts," in *Proc. 1st Intl. Wksp. on Energy Neutral Sensing Systems*, ser. ENSSys, Rome, Italy, Nov. 2013.

[14] Y. Sun, O. Gurewitz, and D. Johnson, "RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks," in *Proc. ACM Intl. Conf. on Networked Sensing Systems*, ser. SenSys, Nov. 2008.

[15] D. Moss and P. Levis, "BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking," Stanford Information Networks Group, Tech. Rep. SING-08-00, 2008.

[16] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proc. ACM Conf. on Embedded Networked Sensor Systems*, ser. SenSys, Nov. 2009.

[17] S. Unterschütz, C. Renner, and V. Turau, "Opportunistic, Receiver-Initiated Data-Collection Protocol," in *Proc. European Conf. on Wireless Sensor Networks*, ser. EWSN, Feb. 2012.

[18] A. Kansal, J. Hsu, S. Zahedi, and M. Srivastava, "Power Management in Energy Harvesting Sensor Networks," *Transactions on Embedded Computing Systems (TECS)*, Sep. 2007.

[19] C. Renner, S. Unterschütz, V. Turau, and K. Römer, "Perpetual Data Collection with Energy-Harvesting Sensor Networks," *Transactions on Sensor Networks (TOSN)*, vol. 11, no. 1, pp. 12:1–12:45, Nov. 2014.

[20] N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy, "Cloudy Computing: Leveraging Weather Forecasts in Energy Harvesting Sensor Systems," in *Proc. IEEE Conf. on Sensor, Mesh and Ad Hoc Communications and Networks*, ser. SECON, Jun. 2010.

[21] J. Lu and K. Whitehouse, "SunCast: Fine-grained Prediction of Natural Sunlight Levels for Improved Daylight Harvesting," in *Proc. ACM/IEEE Conf. on Information Processing in Sensor Networks*, ser. IPSN, Apr. 2012.

[22] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *Proc. Symp. on Networked Systems Design and Implementation*, ser. NSDI, Mar. 2004.

[23] N. dos Santos Ribeiro Junior, M. A. M. Vieira, L. F. M. Vieira, and O. Gnawali, "CodeDrip: Data Dissemination Protocol with Network Coding for Wireless Sensor Networks," in *Proc. European Conf. on Wireless Sensor Networks*, ser. EWSN, Oxford, UK, Feb. 2013.

[24] A. Reinhardt and C. Renner, "RoCoCo: Receiver-initiated Opportunistic Data Collection and Command Multicasting for WSNs," in *Proc. European Conf. on Wireless Sensor Networks*, ser. EWSN, Porto, Portugal, Feb. 2015.

[25] Memsic, Inc., "Datasheet: IRIS Wireless Measurement System," http://www.memsic.com/userfiles/files/Datasheets/WSN/IRIS_Datasheet.pdf, 6020-0124-02 Rev. A; Last accessed: 2013/03/18.

[26] C. Renner and P. A. T. Nguyen, "Lossless Compression of Cloud-Cover Forecasts for Low-Overhead Distribution in Solar-Harvesting Sensor Networks," in *Proc. 2nd Intl. Wksp. on Energy Neutral Sensing Systems*, ser. ENSsys, Memphis, TN, USA, Nov. 2014.

[27] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, T. Braun, P. Hurni, M. Anwander, G. Wagenknecht, S. P. Fekete, A. Kröller, and T. Baumgartner, "Flexible Experimentation in Wireless Sensor Networks," *ACM Communications*, vol. 55, Jan. 2012.