Lossless Compression of Cloud-Cover Forecasts for Low-Overhead Distribution in Solar-Harvesting Sensor Networks

Christian Renner
Institute of Computer Science
Universität zu Lübeck
renner@iti.uni-luebeck.de

Phu Anh Tuan Nguyen
Institute of Computer Science
Universität zu Lübeck
nguyenp@informatik.uni-luebeck.de

Abstract

Combining local harvest patterns and global weather forecasts, e.g., cloud-cover forecasts, makes solar harvest predictions and online duty cycle adaptation more reliable. For this purpose, an energy and bandwidth efficient network-wide distribution of those forecasts is required. To meet this end, we propose compression methods for cloud-cover forecasts, so that they can be piggy-backed on regular network packets. We evaluate compression performance based on data collected from an online weather service for more than 14 months. We find that (i) cloud-cover forecasts can be compressed by up to 76%, (ii) fit into an average of 5 B for a one-day and 21 B for a seven-day forecast horizon, so that (iii) network-wide distribution leveraging, e.g., software acknowledgments used by prominent low-power data collection algorithms is achievable.

1 Introduction

Energy-harvesting is a key enabling technology for sustained, autarkic, reliable, and maintenance-free sensor networks. Especially in outdoor sensing scenarios, solar power is an abundant resource and has hence been used for several energy-harvesting sensor nodes [11, 3, 9]. However, the amount of harvested energy—the *harvest*—is limited, since 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 set up these protocols—mainly in terms of configuring their duty cycle—the harvest has to be well known. Unfortunately, the electric power produced by a solar cell cannot be predicted precisely prior to network deployment. The reasons for this can be divided into two categories:

ENSsys'14, November 6, 2014, Memphis, TN, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-3189-0/14/11…$15.00
http://dx.doi.org/10.1145/2675683.2675686

1. Each node has a specific harvest pattern that is influenced primarily by its positioning and environment. For example, shades of buildings and trees reduce the actual harvest of a node. Their influence varies throughout the year due to, e.g., solar altitude and foliation. Hardware aging and dirt deposits may also change the harvest pattern temporarily.
2. Weather conditions and seasonal effects impact the actual harvest. The usefulness of statistical information based on weather records is limited, because it only provides an uncertain picture of the future weather. Hence, pessimistic assumptions about the future harvest must be used to prevent node depletion.

In recent efforts, researchers have concentrated on capturing the harvest pattern via online algorithms [1, 12, 16]. These algorithms exploit the fact that solar irradiance generally follows a diurnal pattern, which they try to learn online and use them to predict the future harvest. However, predictions are only based on locally available node data—e.g., a node compares the beginning of a day with a set of learned harvest patterns to select the most likely future course of harvest in this day. Regardless of the actual harvest prediction algorithm, its results are used to adjust the duty cycle of sensor nodes at runtime as in, e.g., [10, 6].

These approaches share in common that they are likely to fail—i.e., produce massively wrong harvest predictions—in changing weather conditions with the worst-case result of unexpected node depletion. Prediction algorithms combining local harvest patterns and global weather forecasts are hence required. In our previous work [14] we have shown that using cloud-cover forecasts improves harvest predictions. However, distribution of the weather forecasts into the network is a mandatory yet unmet prerequisite for adopting our prediction algorithm in real-world networks.

In this paper, we close this gap by proposing and evaluating lossless compression methods for cloud-cover forecasts, so that they can be distributed efficiently—i.e., by consuming low bandwidth and by causing a small energy expenditure only. Since the distribution of cloud-cover forecasts is generally delay-tolerant in the range of minutes (forecasts usually have a prediction horizon of hours our even days), we aim at sending cloud-cover forecasts piggy-backed on regular network packets. In particular, we envision network-wide distribution by leveraging software acknowledgments used in duty-cycled, low-power routing protocols. Hence, no extra network protocol is required for forecast distribution.

2 Lossless Cloud-Cover Forecast Compression

Cloud cover is a metric to describe the relative cloudiness of the sky, in eighths. Forecasts are available through several online weather portals. They are usually updated once every hour, have a time resolution of one hour, and a forecast horizon of several days. Since cloud cover already is a rather coarse metric and the effect of precision loss for harvest forecasts has not been explored, we concentrate on lossless compression in the following.

The nine distinct cloud-cover values can be easily encoded as a 4-bit integer. Overall message length scales linearly with the time resolution of the forecast and the forecast horizon. When distributing a typical forecast with a horizon of only 24 h naïvely in the network, this gives an overall length of $24 \cdot 4\,\text{bit} = 96\,\text{bit} = 12\,\text{B}$. This already implies a notable but likely tolerable overhead when piggy-backing this data on normal data packets or acknowledgments. For example, software acknowledgments (beacons) used by the TinyOS-implementation of ORiNoCo [18] have a length of 17 B. Although cloud-cover forecasts will be attached to a fraction of beacons only, these few beacons may suffer from a higher packet loss probability. Moreover, providing forecasts longer than a day is only possible at an even increased packet loss rate or a reduced time resolution (to maintain forecast data size).

Therefore, we have investigated methods to compress cloud-cover forecasts so as to reduce the bandwidth and energy expenditure caused by network-wide forecast distribution. In the following, we present a brief analysis of the characteristics of a cloud-cover forecast trace recorded from an online weather forecast portal; based on the presented findings, we subsequently present light-weight compression methods. We present an evaluation of the achievable compression performance in Sect. 3.

2.1 Characteristics of Cloud-Cover Forecasts

As a first step towards compressing cloud-cover forecasts, we analyzed their characteristics and statistical properties in order to choose promising compression methods. Figure 1 shows two example cloud-cover forecasts for the same place but on different days. Despite the presence of 72 individual forecast values each, adjacent forecast values change infrequently, i.e., there are only 12 and 16 value changes in the example forecasts, respectively. Moreover, values change by only ± 1 in these examples.

A statistical analysis of more than 10 590 forecasts that we collected for more than 430 days revealed a more detailed and dependable picture (c.f. Sect. 3.1 for more details about the data trace). In particular, we found that cloud-cover values are distributed unevenly as shown in Fig. 2a. This means that a coding based on value frequency, such as Huffman coding, is a promising approach. In addition, we observed that more than 82% of all adjacent cloud-cover values are equal; and 99% of all forecast values have a difference (which we will call delta in the following) of an absolute value no more than 1. This is documented in Fig. 2b. The entropy of cloud-cover values is 3.17 as opposed to only 0.87

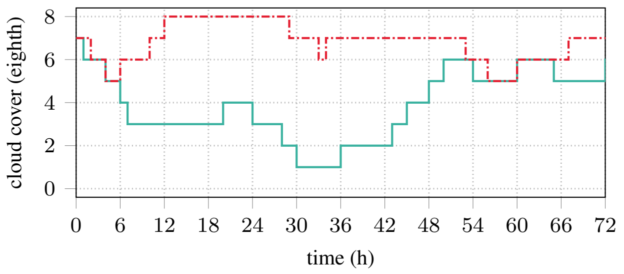


Figure 1. Two example cloud-cover forecasts with a resolution of one hour and a horizon of three days.

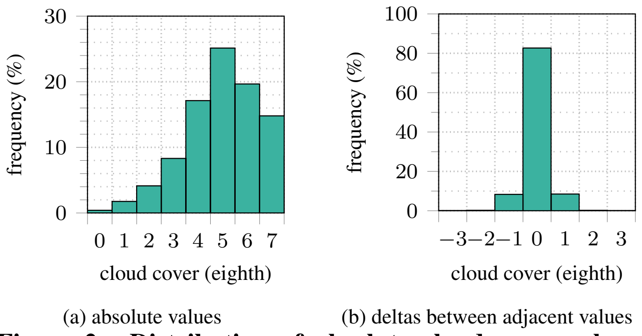


Figure 2. Distribution of absolute cloud-cover values and absolute changes between adjacent cloud-cover values within a forecast.

for the deltas of adjacent slots. Therefore, using a compression method based on value deltas is more appealing than encoding the absolute values.

While it may appear tempting to investigate the actual change of subsequent forecasts and only transmit differential forecasts, we do not follow this approach for the sake of distribution simplicity. Such an approach would introduce problems if, e.g., a sensor node runs out of energy or misses a forecast. In this case, any differential forecast would be useless for that node. Therefore, recovery mechanisms would be required to enforce the distribution of a complete, non-differential forecast. To avoid this, we only consider the distribution of complete forecasts.

2.2 Compression Methods

The previous results motivate the compression of cloud-cover forecasts to decrease the size of the forecasts and thus reduce the overhead when sending them piggy-backed on data packets or software acknowledgments. In addition to a high compression ratio, the decoding step should be light-weight enough for contemporary sensor node hardware, since every node in the network has to perform this step after reception of a new forecast, whereas encoding may be performed on a server connected to the sink node. The implementation of the decoder should hence have properties such as a small memory footprint and a fast computation time. In the following, we present three such methods.

2.2.1 Delta Coding (DC)

Since cloud cover within a forecast changes slowly, a Huffman coding of the deltas between adjacent values within a forecast is a promising approach. For this purpose, we as-

Table 1. (Selected) Codes for Delta Coding

delta	frequency (%)	code	code length
...	...	...	...
-2	0.23...	`1110`	4
-1	8.31...	`110`	3
0	82.67...	`0`	1
1	8.49...	`10`	2
2	0.22...	`11110`	5
...	...	...	...

Table 2. Codes for Partial Delta Coding

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

signed codes to all possible deltas from -8 to 8 (eighth) based on their frequency. Table 1 gives an overview of the corresponding frequencies and codes. Since delta coding requires knowledge about the first value, we transmit the latter with a 4-bit integer encoding for the nine possible values. Note that using an implicit value (e.g., the value 5, which occurs with highest frequency as indicated above) would be possible, but we found that the savings are marginal. Not considering the first absolute forecast value, the expected average code size per forecast value is 1.27 bit for this coding.

While the implementation of the encoder is straight forward, decoding generally requires some effort. The latter may, e.g., be implemented using a tree structure for the used codes, where leaf nodes contain the corresponding delta values. In this concrete instance, codes may be chosen in a way that they are defined by a series of 1's followed by a single trailing 0. While this simplifies decoding considerably (to counting the number of consecutive 1's before a zero), codes have a length of up to 17 bit. Although the frequency of such codes is rare, they will increase data size by 4 bit w.r.t. to our current implementation. We hence did not opt for this approach.

2.2.2 *Partial Delta Coding (PDC)*

Decoding complexity of the previously introduced delta encoding is considerably higher than a simple integer encoding of the deltas. Moreover, codes consume up to 13 bit. For these reasons, we devised a partial delta coding that consists of four different codes only, that have a maximum length of 8 bit, and that can be decoded efficiently. Table 2 gives an overview of the corresponding frequencies and codes. The first three codes are used for delta changes of -1, 0, and 1; the fourth is used for any other delta and contains an absolute forecast value encoded as 4-bit integer. As with DC, the first cloud-cover value of the forecast is encoded as a 4-bit integer. Not considering this value, the expected average code size per forecast value is 1.28 bit.

The corresponding decoder can be implemented very efficiently. After reading the first absolute 4-bit value, the decoder has to count the number of consecutive 1's before a 0 is read or until a maximum of three consecutive 1's has been received. In the former case (a 0 is read), the delta can be directly determined from the number of 1's. If three consecutive 1's were received, a 4-bit-encoded absolute forecast value follows.

```
1: procedure DecodeDDC($N$, sunrise, sunset)
2:     isDay ← sunrise > sunset;
3:     for $i \leftarrow 0,\ldots,N-1$ 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 }
```

Figure 3. Decoding algorithm for a DDC-encoded cloud-cover forecast with a horizon of N values (hours).

2.2.3 *Daytime Delta Coding (DDC)*

The previous two codings disregard the fact that, over a period of a complete year, approximately half of the forecast values are obsolete, since they forecast cloud cover during nighttime when no solar energy is harvested. To cope with this particular characteristic, we devised another encoding that does not transmit these obsolete values.

We realized this by adding information about sunrise and sunset while omitting all forecast values that fall into nighttime. With this information, a node can correctly decode the received forecast. We implemented this practically by attaching the offset (in hours) of sunrise and sunset from the time of forecast creation to the (compressed) forecast. Both values fit in a 5-bit integer and hence counter the saving due to value omission by a fixed 10 bit. The remaining encoding is equal to PDC (with the exception that deltas during nighttime are omitted). Note that the first slot after the night may be encoded as a delta of -1, 0, or 1, if applicable, or as a prefixed absolute value as described for PDC.

Decoding is similar to that of PDC, but the decoder must keep track of the beginning (sunrise) and end (sunset) of daytime. In particular, the decoder has to check whether it is currently day or night (it can do this by checking if sunrise or sunset is smaller, i.e., closer to the currently evaluated time in the future). Figure 3 shows a pseudocode algorithm for decoding DDC-encoded forecasts.

3 Evaluation

In the following, we evaluate the compression performance of the methods proposed in Sect. 2.2. First, we introduce the evaluation setup and metrics, before we present the results. We conclude the section with a discussion of the results and their limitations.

3.1 Setup and Metrics

To evaluate compression performance, we recorded cloud-cover forecasts for the location of our university from an online weather forecast service[1] from April 2013 to July 2014 with a gap of 4 weeks in November 2013 due to hardware failure. Updated forecasts were available once every

[1] `http://www2.wetterspiegel.de`

hour. They have an hourly resolution and a forecast horizon of up to 10 days. Although a few forecasts were not available or incomplete, we were able to reconstruct approximations from previous forecasts. In total, the evaluation data set has a size of 10 590 individual cloud-cover forecasts.

From these data, we created hourly cloud-cover forecasts with forecast horizons ranging from one to seven days. We compressed these with the methods from Sect. 2.2 and recorded the size of the encoded data in bits and bytes (rounded to the next larger number of bytes). We assume that forecasts that are distributed in the network have a constant horizon, so that no extra length information field (for the forecast itself) has to be sent alongside the forecast. For DDC, these sizes also contain the static offset required to encode sunrise and sunset. Unfortunately, we did not record the required times of sunrise and sunset together with the cloud-cover forecasts. However, these times were calculated using the so-called sunrise equation[2].

3.2 Results

3.2.1 *Compression Performance and Comparison*

First, we studied the absolute size and distribution of compressed cloud-cover forecasts to assess the general feasibility of transmitting them piggy-backed on regular data packets or software acknowledgments. Figure 4 shows the results for the three different methods and various forecast horizons. The results exhibit an expected linear increase of encoded data size for all methods.

The comparison between DC in Fig. 4a and PDC in Fig. 4b indicates that both methods achieve an almost equal compression performance. Only in a few instances, i.e., when cloud-cover forecasts change by more than one eighth between adjacent values, PDC requires up to two extra bytes. On average, both methods achieve savings over uncompressed data of 58%, 63%, and 66% for forecast horizons of 1, 3, and 7 d, respectively. Compared to an optimal entropy encoding of absolute cloud-cover values, these figures remain at 44%, 48%, and 51%.

For a forecast horizon of only a single day, DDC in Fig. 4c performs comparable to DC and PDC. With an increasing forecast horizon, however, DDC achieves a higher compression. For example, the median for a three-day horizon using DDC is 10 B vs. 13 B for (P)DC. The advantage increases to 7 B (which equals a 25% saving) for a seven-day horizon. On average, DDC achieves savings over uncompressed data of 60%, 72%, and 76% for forecast horizons of 1, 3, and 7 d, respectively. Compared to an optimal entropy encoding of absolute forecast values, these figures are 47%, 60%, and 66%, respectively.

3.2.2 *Optimality Study*

Second, we compared the average compression results of our methods with an optimal entropy encoder based on delta values. For this purpose, we calculated the entropy of the delta values, which gives a lower bound for the average number of bits needed to encode a single delta value. As for all presented compression methods, we assumed that encoded delta values are preceded by a single absolute value encoded as four-bit integer. Hence, the overall average code length decreases with an increased forecast horizon, since the impact of this static size offset is reduced.

Table 3. Average Number of Bits per Forecast Value

horizon (d)	avg. compression code length (bit/value)			
	optimal	DC	PDC	DDC
1	1.00	1.50	1.51	1.43
2	0.94	1.41	1.42	1.14
3	0.92	1.37	1.38	1.04
4	0.91	1.35	1.36	0.99
5	0.90	1.34	1.35	0.95
6	0.90	1.32	1.33	0.93
7	0.89	1.31	1.32	0.91

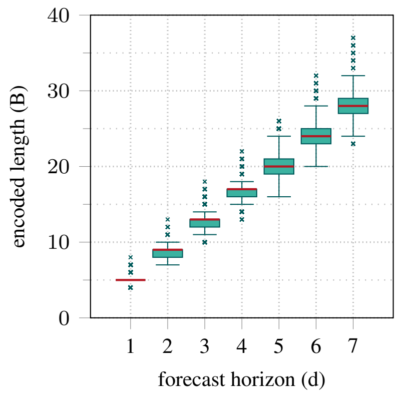
Table 3 displays the average code length per forecast value in bits for forecast horizons from 1 to 7 d. It shows that optimal average code lengths are at most 1 bit per forecast value, which cannot be achieved with a real entropy encoder, since the minimum bit length of each forecast value is 1 bit already. In fact, DC causes an overhead of 47–50% w.r.t. an optimal entropy encoding. This overhead equals the average bit length of DC codes when considering the size offset caused by the first (non-delta) value (c.f. Sect. 2.2). Since PDC is a slight modification of DC only, the numbers are similar. In contrast, DDC approaches the compression performance of an optimal entropy encoder. This is achieved through omitting unnecessary values at nighttime combined with entropy coding of delta values.

3.2.3 *Seasonal Daytime Influence*

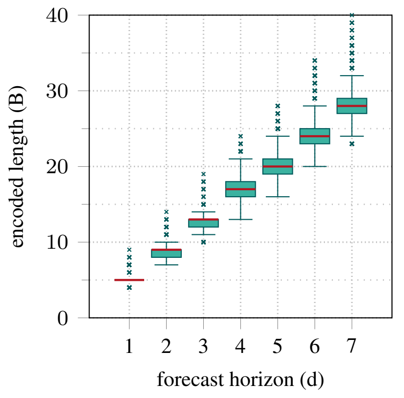
Finally, we compared compression performance of PDC and DDC in more detail by studying the development of compressed data sizes over time, i.e., depending on the seasons. The expected result is that DDC performs best in the winter, when days are short (and nights long), and worst in summer, when days are long (and nights short). However, it is uncertain if it is desirable to implement a hybrid-compression method that may choose for each forecast among PDC and DDC—for simplicity we assume that there is always room for another single bit to mark whether PDC or DDC was used for encoding. We hence want to answer the question if and when such a solution is favorable over a single method.

Therefore, we compared all individual compression results of PDC and DDC and analyzed the savings of DDC over PDC. Selected results are displayed in Figs. 5a and 5b. The general trend of the results confirms that during periods with more daylight (late spring, summer, early fall) DDC performs better than during darker periods of the year. However, the forecast horizon decides whether a hybrid solution is worthwhile. For a horizon of one day, PDC outperforms DDC from May to early August (negative savings of DDC over PDC in Fig. 5a). In the remainder of the year, DDC is the better choice. In contrast, when the forecast horizon is at least three days, as shown in Fig. 5b, DDC never produces compression results inferior to PDC. We can hence conclude that for a forecast horizon of up to two days, it would be favorable to apply both methods and only distribute the smaller result of the two in the network. However, this requires to add an extra bit to enable the receiver to pick the right de-

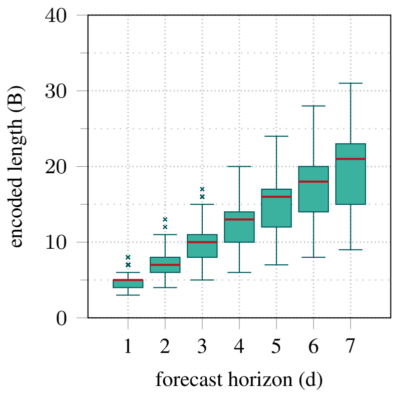
[2]`http://lexikon.astronomie.info/zeitgleichung/`



(a) DC

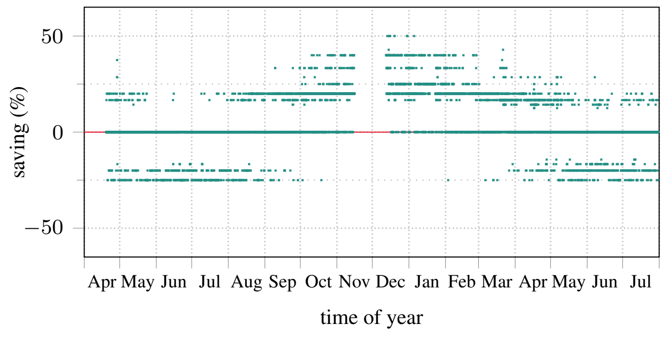


(b) PDC

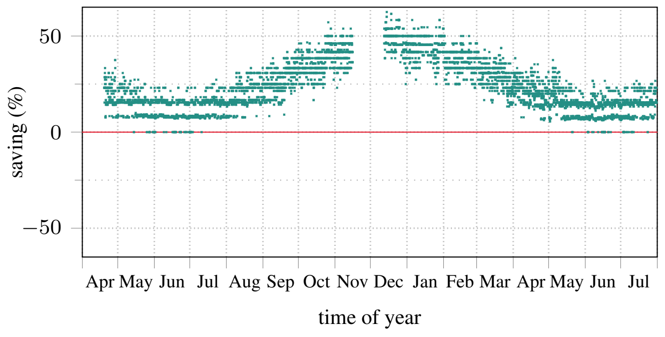


(c) DDC

Figure 4. Comparison of compressed data sizes for the three compression methods and different forecast horizons with an hourly forecast resolution. The boxplots indicate the median (red line), upper and lower quartiles (box), and outliers (marks). Note that uncompressed forecasts require 12 B for every forecasted day.



(a) 1-day cloud-cover forecast horizon



(b) 3-day cloud-cover forecast horizon

Figure 5. Compression improvement of DDC over PDC for different forecast horizons. Negative values indicate a larger size of DDC-compressed forecasts.

coder and will increase firmware size slightly, since two decoding methods are required.

3.3 Discussion and Limitations

Our evaluation has shown that cloud-cover forecasts can be compressed with savings of up to 76%. On average, the daytime delta coding (DDC) achieves best results while it can be implemented very efficiently. Forecasts of up to 3 d consume at most 17 byte, which allows piggy-backing them on regular data packets or software acknowledgments. A non-compressed forecast with equal horizon already requires an intolerable 36 byte. Increasing forecast horizons is likely to improve adaptive load adaptation algorithms w.r.t. making a more efficient use of harvested energy while avoiding accidental node depletion. However, we consider even longer-term forecasts as being too large for piggy-backing in most cases—e.g. a 7 d forecast requires up to 31 B, which equals one fourth of the maximum payload of many packet-based low-power radios.

Although our evaluation data is limited geographically to central Europe, we are confident that our results are valid for a wide range of possible deployments with a similar distribution of day lengths over the year. In regions where day lengths tend to vary less, PDC will be the better option for short-term forecasts, whereas an even wider variation of day lengths will profit from an hybrid compression as discussed in Sect. 3.2.

In this paper, we did not consider compression by means of reducing the time resolution of cloud-cover forecasts, because the impact of a resolution reduction on harvest forecast precision has not been fully explored. In particular, it is not known if reducing time resolution requires an increased resolution of cloud-cover values—e.g., reducing time resolution by a factor of 2 generally requires increasing cloud-cover resolution by a factor of 2; hence requiring longer codes.

4 Related Work

Power management for energy-harvesting sensor networks enables sensor nodes to prevent depletion [15] and to maximize the utility of harvested energy [10, 6]. Performance evaluation of energy-harvesting hardware platforms and algorithms has been enabled through the GreenCastalia simulator [2]. Due to the unsteady energy production of (solar) harvesters, several researchers have investigated and proposed methods for future harvest prediction based on local knowledge. Such methods are usually based on time slots as in [1, 12] or pattern creation and recognition as proposed by Spenza et al. [16]. Only recently, we have shown the advantage of exploiting global weather information, such as cloud-cover forecasts, in [14].

Such global weather forecasts must be distributed in the network. For this purpose, data dissemination protocols, such as Trickle [8] and CodeDrip [4], have been developed. However, they rarely make use of low-power, duty-cycled MAC protocols and hence increase energy consumption. Another option is to perform distribution by means of piggy-backing forecasts on, e.g., software acknowledgments used by low-power data collection protocols, such as [7, 18].

However, this requires data compression to limit the size overhead of these acknowledgments. Reinhardt et al. presented an adaptive Huffman coding approach for sensor networks in [13], where they concentrate on limiting tree size and the required bandwidth to distribute updated Huffman trees due to code adaptations. Tsiftes et al. proposed a compression method for bulk data, particularly firmware updates, in [17]. A method for compressive sensing using a basis transformation to reduce the entropy of sampled data is presented in [5]. We anticipate a low compression gain only, if any, when applying these methods to cloud-cover forecasts, for the latter can be highly compressed by exploiting day/night times and the low fluctuation of consecutive values (c.f. Sect. 3). Furthermore, the previously mentioned methods come at a higher implementation and networking cost.

5 Conclusion

Due to infrequent changes of adjacent cloud-cover values within a single forecast and unneeded nighttime values, cloud-cover forecasts offer a high compression potential. By exploiting these properties consequently, we were able to compress those forecasts by up to 76%. Even multi-day forecasts only consume a handful of bytes—e.g., a three-day forecast requires 8 to 11 B in 50% of all cases and a maximum of 17 B. As a result, it is possible to piggy-back cloud-cover forecasts on existing data packets or software acknowledgments of low-power, duty-cycled data collection protocols. Therefore, no extra, complex data distribution protocol is required, hence saving computation power, radio usage, bandwidth, and, most importantly, energy expenditure. Moreover, we have shown that decoding can be implemented efficiently in a few lines of code.

Motivated by these results, we are currently integrating a mechanism for network-wide distribution of compressed cloud-cover forecasts in the ORiNoCo data collection protocol. We will evaluate distribution reliability, latency, and overhead in a follow-up work.

6 References

[1] M. Ali, B. Al-Hashimi, J. Recas Piorno, and D. Atienza. Evaluation and Design Exploration of Solar Harvested-Energy Prediction Algorithm. In *Proc. Conf. on Design, Automation and Test in Europe*, DATE, Mar. 2010.

[2] D. Benedetti, C. Petrioli, and D. Spenza. GreenCastalia: an Energy-Harvesting-Enabled Framework for the Castalia Simulator. In *Proc. 1st Intl. Wksp. on Energy Neutral Sensing Systems*, ENSSys '13, Rome, Italy, Nov. 2013.

[3] D. Brunelli, C. Moser, L. Thiele, and L. Benini. Design of a Solar-Harvesting Circuit for Batteryless Embedded Systems. *IEEE Transactions on Circuits and Systems I (TCAS-I): Regular Papers*, 56(11), Nov. 2009.

[4] 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*, EWSN, Oxford, UK, Feb. 2013.

[5] J. Höglund, T. Voigt, B. Wei, W. Hu, and R. Karoumi. Compressive Sensing for Bridge Damage Detection. In *Proc. 5th Nordic Wksp. on System and Network Optimization for Wireless*, Åre, Sweden, Apr. 2014.

[6] 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*, ISLPED, Oct. 2006.

[7] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low Power, Low Delay: Opportunistic Routing meets Duty Cycling. In *Proc. ACM/IEEE Conf. on Information Processing in Sensor Networks*, IPSN, Apr. 2012.

[8] 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*, NSDI, Mar. 2004.

[9] 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*, 1(3), Sept. 2011.

[10] 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*, DATE, Apr. 2007.

[11] 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*, IPSN, Apr. 2005.

[12] 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*, VITAE, May 2009.

[13] A. Reinhardt, D. Christin, M. Hollick, J. Schmitt, P. S. Mogre, and R. Steinmetz. Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks. In *Proc. European Conf. on Wireless Sensor Networks*, EWSN, Feb. 2010.

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

[15] P. Sommer, B. Kusy, and R. Jurdak. Power Management for Long-Term Sensing Applications with Energy Harvesting. In *Proc. 1st Intl. Wksp. on Energy Neutral Sensing Systems*, ENSSys '13, Rome, Italy, Nov. 2013.

[16] 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*, MASS, Oct. 2012.

[17] N. Tsiftes, A. Dunkels, and T. Voigt. Efficient Sensor Network Reprogramming through Compression of Executable Modules. In *Proc. IEEE Conf. on Sensor, Mesh and Ad Hoc Communications and Networks*, SECON, June 2008.

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