An Energy-Efficient and Reliable Mechanism for Data Transport in Wireless Sensor Networks

Erik-Oliver Blaß
Institute of Telematics
University of Karlsruhe, Germany
Email: blass@tm.uka.de

Lars Tiede
Institute of Telematics
University of Karlsruhe, Germany
Email: tiede@tm.uka.de

Martina Zitterbart
Institute of Telematics
University of Karlsruhe, Germany
Email: zit@tm.uka.de

***Abstract*— Transportation of data between nodes in a sensor network is expensive, as wireless radio transmission depletes finite battery capacity. In addition, wireless data transmission is prone to errors, like static, making reliable data exchange between sensor nodes even more expensive. This paper describes a novel transport scheme that allows sensors to *predict* data from other sensors. Thereby, communication can partially be omitted, which in return results in reduced radio traffic, less energy consumption, and thus improved network lifetime. In addition to that, simple techniques to ensure reliable communication become much more affordable. The proposed scheme seamlessly integrates into in-network data aggregation. The prediction mechanism is based on the evaluation of polynomials derived from simplified Kalman filters.**

I. Introduction

Extracting information from a sensor network requires communication among sensor nodes, often along a path starting from a measuring sensor node to a data *sink* or *base station*, respectively. In most scenarios, the collection of data towards a base station is done on a regular, i.e., periodic base, as continuous measurements are required from sensors. Popular examples are the measurements of temperatures in a room or the heartbeat rate of a patient's heartbeat sensor. Typically, timed communication is used to provide constant data rates towards the base station. This type of communication is, however, expensive, as sensor hardware is powered by tiny batteries with finite capacity. Frequent communication using a radio interface consumes energy and degrades a node's lifetime. Besides, packet loss is an inherent problem of wireless sensor networks. Sensors utilize cheap and primitive radio interfaces, prone to errors and collisions. Experiments show that packet loss in sensor networks can be quite high, for example, [1] encounter 50%. High packet loss makes reliable communication in a sensor network, e.g., by sending additional overhead like acknowledgments (ACKs), even more expensive.

To reduce traffic towards the sink, sophisticated techniques for the minimization of communication are required in order to maximize the overall network lifetime. One typical technique to achieve this is in-network data *aggregation*. Bunches of sensor nodes are logically grouped, one among them being an *aggregation node*. This node collects all the data from group members and produces an aggregate which is then used as the collective data from that group and further forwarded towards the sink. But even in a network performing data aggregation, there is still a considerable communication overhead, sometimes highly redundant. In a real-world environment, where measurements from nodes are reported to a sink, which collects and evaluates all data, there is often communication transporting data of little value or low entropy through the network. For example, imagine a network observing temperatures in a building on a regular basis: as temperatures at single sensor nodes in different rooms often behave fairly well-predictable or do not change at all during short periods of time, there is mostly redundant data transfer, reporting insignificant information to their aggregation node. If the nodes in the network had some application-independent intelligence, enabling them to distinguish between important and unimportant or insignificant samples, periodic transmissions of the actual temperature values would become unnecessary, thus saving a huge number of expensive radio transmissions. Instead, a measuring node would only send new data to its aggregation node, if the new data appears to be *significant*. A node decides about the significance of a new measurement by comparing the actual measurement with the data the aggregation node might expect or predict. Only if there is a significant difference between the actual measurement and the predicted value, the node sends data to the aggregation node. For example, an increase in temperature is periodically measured, like $30, 32, 34$ degrees Celsius, and this is transmitted to the aggregation node. As the aggregation node can now, with a certain error probability, predict the next measured temperature to be 36 degrees, there is no need to actually send this information from the sensor to the aggregation node.

This paper introduces a fairly general approach to utilize a simple, yet powerful prediction ability in sensor networks. The approach is suited for typical sensor network scenarios and supports in-network data aggregation. Prediction will allow the whole network to save a lot of energy and battery-consuming transmission costs, thus making communication and additional reliability overhead much more affordable. Basically, this prediction is made up by a *Kalman Filter* [2] on a measuring node and a *predictor polynomial* which is employed on both, the measuring and the aggregation node (cf. Fig. 1).

The rest of this paper is structured as follows: After giving an overview of related work in Section II, Section III describes the basic idea of the new approach. As it relies on Kalman

filtering, the math behind such filters is roughly sketched in Section IV. An example application with real-world data has been evaluated and results are presented in Section V. This section also evaluates the feasibility and performance of the prediction algorithm on a typical constrained sensor hardware. Although a general prediction mechanism is suitable for most sensor network scenarios, there are certain types of situations that will not benefit from this approach at all. This is briefly discussed in Section VI. Finally, Section VII concludes this paper.

II. Related Work

Saving energy by prediction of to-be-transmitted values is a relatively new area of research in sensor networks. In WSDP [2], the authors apply the technique of Kalman filtering in wireless sensor networks: WSDP's main purpose is to lessen the impact of packet loss at the information sink by estimating missing measurements lost due to faulty communications. Similarly, [3] theoretically analyzes the behavior of Kalman filter's estimation errors with intermittent transmissions. The authors of [4] or [5] propose to lessen the impact of lost transmissions on estimation by using distributed filtering.

All of the before mentioned work focuses on the estimation of lost measurement values using Kalman filters at an information sink. In contrast, this paper concentrates on the energy savings possible due to predicting and therefore omitting data transmissions on a hop-by-hop basis in sensor networks.

The theory of Kalman filters was published first in [6]. Today they are found in virtually any application, reaching from communication processors to PLL radio tuners, autonomous or assisted navigation and tracking applications.

Some research has been done to save energy by minimizing the *size* or encoding of transmitted data values, for example by sending only differences of preceding values to an aggregation node [7]. Yet, the total number of packets to be transmitted will not be reduced, and, as packets have a certain minimum length, which is larger than the size of typically measured values, cf. TinyOS [8], the expected energy gains are negligible in most sensor network scenarios.

ADPCM [9] allows an efficient encoding of new pulse code modulated speech data based on old received data, however, application to arbitrary sensor network data is unlikely.

Also, effort has been made to investigate general reliability problems in sensor networks, cf. [10] for an overview. This work, however, aims at saving energy by the prediction of to-be-transmitted values, therefore making the impact of necessary reliability mechanisms much more affordable.

III. Basic Idea of our Approach

Using a Kalman filter, sensor nodes measuring data are able to produce a special mathematical function: a prediction polynomial or simply a *predictor*. A Kalman filter uses actually sensed measurements as input, processes them, and outputs the predictor polynomial. This polynomial allows to predict future to-be-measured values based on all measurements taken so far. Sensor nodes send their computed predictors to their associated aggregation nodes, giving them the ability to predict measurement data from sensor nodes without further communication. This basic setup is shown in Fig. 1.

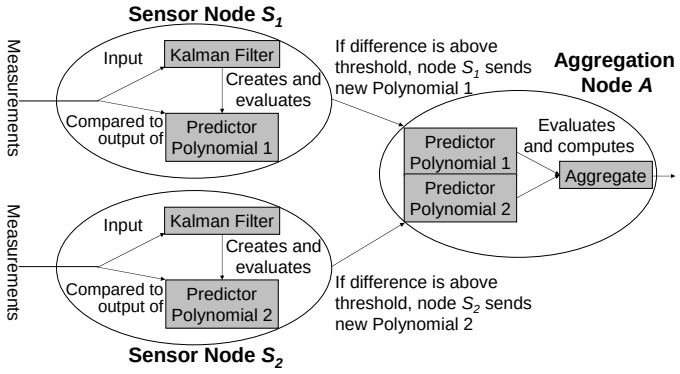

Fig. 1. Setup of two measuring nodes and one aggregation node

A sensor node will not discard the predictor, which has been sent to its aggregation node, but use it to compare actual measurements with the evaluations of the predictor. After a certain time, prediction using the predictor loses accuracy: the difference between actual measurements on a node and predicted values on the aggregation node exceeds a user or application defined threshold. As soon as a measuring node detects that its predictor used by the aggregation node becomes too imprecise, it has to provide an updated polynomial to its aggregation node. A sensor has to know when the predictor in the aggregating node needs to be updated and therefore the measuring node needs to track and evaluate the polynomial used by the aggregation node. Thus, a node *knows* which data the aggregation node currently estimates for its measurement.

In conclusion, a measuring sensor node S_i conducts the following:

1) Each node S_i utilizes a Kalman filter, which enables S_i to generate a predictor polynomial P_i at any time. Generally, P_i is based on all measurements S_i has taken so far. It is further assumed that S_i has sent P_i to S_i's aggregation node A in the past, so A also possesses P_i.
2) Now, at time t, S_i takes a new measurement m_i. This measurement is compared to the actual prediction value v_i, that is, the evaluation of predictor P_i: $v_i = P_i(t)$. (Details on polynomial evaluation are described in Section IV-C.) Node S_i compares the difference between the predicted value and the actual measurement to $\epsilon > 0$, a user defined or application specific error threshold: $|m_i - v_i| \leq \epsilon$.
3) If $|m_i - v_i| \leq \epsilon$, then S_i does not transmit this value. The prediction is precise enough for the user. If, however, $|m_i - v_i| > \epsilon$, then S_i generates a new predictor polynomial P_i' based on all previous measurements and m_i, utilizing its Kalman filter. The new predictor P_i' replaces P_i and is transmitted reliably to A.
4) At time $t' > t$, whenever a new measurement has to be taken, the whole procedure is repeated at step 2.

Meanwhile, all aggregation node A has to do is to evaluate

all received predictor polynomials P_i and to compute their aggregate, cf. Figure 1. Node A will continue to do so until it receives an update, that is, a new predictor P_i', which replaces P_i. The estimation error, which can occur at an aggregation node A, is equal to the maximum error threshold ϵ of the associated predictor.

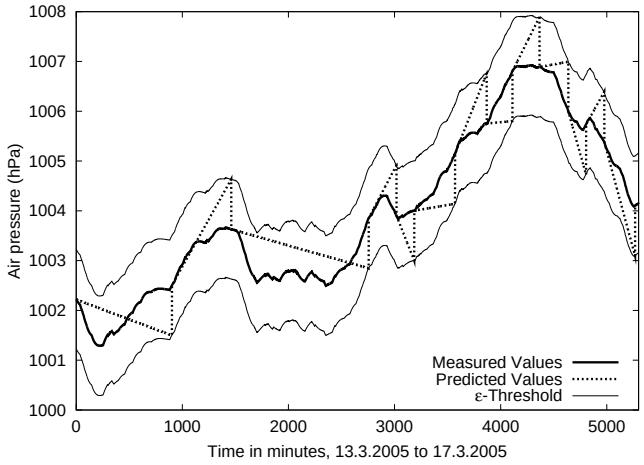

Fig. 2. Application of simple predictor to air pressure measurements

For a better understanding, Fig. 2 illustrates a zoomed view on real-world measurements, cf. [11] for details. The solid line represents actual measured air pressure values, enclosed by an ϵ-tube, dotted in grey, with $\epsilon = 1$hPa. The darkly dotted line represents the output of a simple first-order predictor polynomial, that is, a straight line. Instead of sending measured values, e.g., once a minute, you can clearly see the points in time where predictor updates need to be transmitted: every time the line appears to be discontinuous, for example just before $t = 1000\text{min}$. As this appears to be very rare, energy savings because of fewer transmissions are to be expected.

Having the latest predictor, the aggregation node can now serve as an information *proxy* for all the measuring nodes in its group. Particularly, it can perform an aggregation function on its group's data at any time, using predicted values. If an aggregate is queried explicitly from a group, the only node that has to work and communicate is the aggregation node.

Finally, the prediction mechanism integrates seamlessly into "multi-leveled" or cascading aggregation scenarios as well: an aggregation node on a *higher* level of aggregation can use predictors for its aggregation nodes on *lower* levels, thereby again omitting transmissions and saving energy.

Remarks

It is quite obvious that the proposed scheme will imply additional costs in terms of CPU time and therefore energy, as the predictor polynomial has to be created and evaluated. However, as shown in Section V-A, these energy costs are far smaller compared to the gains by reducing the more expensive radio traffic.

The prediction mechanism requires completely reliable communication, e.g., by sending ACKs, for the polynomial updates. A polynomial update must not be lost, because otherwise the prediction-error may grow beyond ϵ. However, reliable communication implies additional energy costs, cf. Section V-A, so applying the proposed scheme is especially suitable for scenarios which demand reliable communication anyway. Reliability is mandatory for polynomial updates – in return you get reliable predictions within an ϵ-tube.

As a sensor's *duty cycle* is typically very low, i.e., a few percent: sensors are sleeping most of the time and wake up, e.g., periodically. Awakened, they do their duty, like measuring, sending, or receiving data and fall to sleep afterwards. This duty cycle is supported by the proposed scheme: on the one hand, the measuring node wakes up, takes its measurement and decides whether to send an update, sends it if necessary, and falls asleep. On the other hand, the aggregation node wakes up and listens for a predictor update, if there is no update, it simply evaluates its current polynomial and sleeps again.

Although periodic transmissions are supported, the proposed prediction scheme is not limited to it. If it is expected that a measuring sensor can measure and send new data spontaneously, the receiving aggregation node has to be awake all the time – just as it would have been without using prediction. Still, transmissions can be omitted saving a lot of energy.

IV. Mathematics behind Prediction

The prediction in this scheme is mathematically formed by two different parts, one being the steady-state Kalman Filter and the other one being the prediction mechanism itself. On a measuring node, a Kalman Filter processes sensor data and thus continuously generates new prediction polynomials, i.e., predictors, that can be transferred to aggregation nodes if necessary.

The next two sections will only give a very rough, informal, and incomplete overview of the filter and predictor math as this is necessary to understand its use. However, their complex theory is not the scope of this paper, and a more comprehensive description can be found in [12].

A. Steady-State Kalman Filter

A Kalman filter takes an input signal and transforms it into an output. It is executed for every incoming input signal, i.e., in our case, a measurement, so it can be understood as working iteratively, one iteration per measurement. In general, its primary use is to filter out the stochastic component of a time-discrete periodical signal, usually referred to as *noise*, leaving the pure signal as filter output. Thereby, it improves the quality of the signal for the application incorporating the Kalman filter. The signal filtering is done using a dynamic weighing mechanism which weights the filter input against a prediction function, the latter being a part of the filter itself. Noise variables, one for each of these two components, determine the impact of the associated component on the filter behavior: the greater the noise for a component, the less the components impact on the filter. For example, a high noise associated to the systems input signal normally results in greater *smoothing* the filter applies to the output curve.

If the noise variables are taken as constants, the wheighing mechanism of the Kalman filter will eventually converge, resulting in a constant *smoothing intensity* being applied to the output curve. This special case is referred to as the *Steady-State* Kalman Filter and offers some opportunity to simplify the underlying math to a noticeable extent. While the most expensive computation in a full-featured Kalman filter is a matrix-matrix multiplication, it is only a matrix-vector multiplication for the simplified steady-state filter. It is fair to assume constant noise for the input signal, i.e., the actual measurement, because filtering noise is out of scope for this prediction mechanism. An application, however, could decide to incorporate a full-featured filter, if it can take advantage of the signal filtering flexibility compared to the downside implied by the additional computing overhead. Since our primary concern here is energy-efficiency and not signal filtering, the simplified and cheaper (in terms of computing overhead) steady-state Kalman filter is used.

B. Prediction Mechanism

It is expected that, at least on short term, multiple subsequent measurements of typical sensor data, e.g., temperatures or pressures, can be approximated by a polynomial. Therefore, the filter-internal prediction function is made up by a Taylor series [13]. The Taylor series is merged into the filter in a way that the filter's internal state, represented by a vector, holds the filter's output and its first $N-1$ derivatives with N being the size of the vector. Such a setup makes it possible to extract an $(N-1)$th-order polynomial out of the current filter state. This can easily be done at any time and results in a polynomial function which can be used to predict future measurements. The filter's internal state does always represent information from the present as well as the past: At time t, a newly arrived sensor measurement does not solely impact the filter's internal state at time t, but is offset by a prediction mechanism working with the filter's state evaluated during iterations before time t. Thus, information from the past is preserved in the filter's internal state which makes it well suited to serve as the building block for a predictor. Descriptive: the filter's prediction mechanism can be extracted from the filter at any point in time and then be used independently. Our scheme extracts polynomials out of the filter's state and uses them as predictors.

C. Computational Cost

The evaluation of an Nth-order polynomial can efficiently be computed with Horner's scheme [13] using N additions and N multiplications. Furthermore, the extracted polynomial $P(t)$ is adjusted to the time t_0 of the last measurement, i.e., $P(0)$ gives a prediction for the measurement at time t_0. So, to predict a measurement at time $t' > t_0$, you have to evaluate $P(t'-t_0)$. Therefore, N additions, N multiplications, and 1 subtraction are necessary for one Nth-order predictor evaluation. During an Nth-order predictor update, its $N+1$ new coefficients have to be transmitted to the aggregation node as well as t_0; together, a total of $N+2$ values has to be transmitted.

While applying the proposed prediction scheme to real-world data, cf. Section V, the authors selected simple first-order polynomials for prediction: Although polynomials of higher order were tested, the first-oder polynomials surprisingly achieved the best results in terms of prediction precision and thus omitted transmissions. Therefore, in the rest of this paper, only first-order predictors are used.

An efficient implementation of a steady-state Kalman Filter, which can produce first-order predictors, requires 4 additions and 3 multiplications. To evaluate a first-order predictor, 1 multiplication, 1 addition, and 1 subtraction are required. The extraction of a new predictor from the first-order setup filter state requires 2 additions and 1 multiplication. This, however, is only necessary, in the case of $|m_i - v_i| > \epsilon$, cf. Section III.

V. Real-World Example

For a realistic or tangible evaluation of our scheme's efficiency and energy savings, the proposed prediction mechanism was applied to real world data. The meteorological office of Norwegian University of Tromsø measures weather data on a per minute basis [11]. This data, like air temperature, humidity or pressure, is archived and can be downloaded for free.

To verify the practicability and performance of the prediction scheme in general, air temperature and air pressure measurements of March 2005 were used as an example together with a first-order predictor. Imagine a situation where sensor nodes measure this weather data and reliably send them to an aggregation node for further processing once in a minute. The question is, how much transmissions, transmitted data, and finally energy can be saved on the nodes if applying a first-order predictor. This will be discussed, firstly, in an "ideal world" without packet errors or the necessity of ACKs: only the number of saved transmissions and total transferred volume will be compared. Subsequently, in Section V-A, packet errors, e.g., because of static, as well as overhead arising from ACKs will be considered. Finally, the gain in energy has to be offset against the computational overhead, CPU time, and thus energy consumption which comes with the proposed scheme.

TABLE I
Effectiveness of prediction on March 2005 weather data

Type	ϵ	# of Transm.	% of Transm.	# of Values	% of Volume
Temp.	0.10	11252	25.24	33756	75.73
	0.25	3425	7.68	10275	23.05
	0.50	1086	2.44	3258	7.31
Press.	0.10	8163	18.31	24489	54.94
	0.25	1230	2.76	3690	8.28
	1.00	130	0.29	390	0.87

In March, a total of 44575 temperature and air pressure measurements were archived. The first-order Kalman filter was run with three error ranges $\epsilon = 0.10°\text{C}$, $0.25°\text{C}$, $0.50°\text{C}$

for temperature data and $\epsilon = 0.10$hPa, 0.25hPa, 1.00hPa for pressure. The results of applying such a first-order predictor in an "ideal world" are shown in Table I. Ideal means that faulty transmission, retransmission or mandatory ACKs are not taken into account yet.

As you can see, the total number of transmissions using the prediction mechanism is reduced significantly even with small ϵ. For example, using $\epsilon = 1.00$hPa only 130 predictor updates are necessary, which represents 0.29% of the original 44575 transmissions. As 3 values have to be transferred per first-order polynomial, a total of 390 values, e.g., each 32Bit wide, have to be transmitted. Compared to the original periodic transmissions once in a minute, this represents only 0.87% of data volume. Because the radio interface has to send only this small fraction of data, a lot of energy is expected to be saved.

A. Application to MICA2 and TinyOS

To further investigate energy savings, the above-mentioned scenario is analyzed using typical real-world sensor network platform parameters. A simulation written in Python evaluates the impact of first-order predictors on weather data, using the quite popular *MICA2* [14] hardware parameters and running *Tiny Operation System* (TinyOS) [8].

This platform is based on the ATMEL ATmega128L microcontroller CPU and utilizes a ChipCon Single IC Transceiver at the 433MHz band, which gives a data rate of 38.4KBit/s. Such a radio interface consumes 16mA for transmit operations while the CPU drains additional 5mA [15]. Every packet TinyOS sends has a constant size of 56Bytes, including 29Bytes of payload [16]. Sending one packet of data takes about 11.67ms, resulting in 245μAs of energy consumption. Simply sending one month of, e.g., pressure data would cost 10.92As capacity. Not taken into account here are the minimal energy costs for turning on and shutting down the radio interface before and after the transmission – such costs are usually neglected, cf. [16], [17].

Now, assume the demand for a reliable communication. For simplicity, each packet is acknowledged by a separate *ACK* from the aggregation node. ACKs are represented as ordinary data packets in TinyOS, thus, a simple, naive ACK based reliable communication scheme, e.g., *Stop and Wait*, effectively doubles the total number of packets to 89150 packets. Without sources of error, March 2005 transmission of packets therefore costs $89150 * 245\mu\text{As} = 21.84\text{As}$.

First experiments of deploying a sensor network in the wild using similar radio hardware show that packet error rates are as high as 50% [1]. A transmission of one measurement m between sending sensor node S_i and its receiving aggregation node A is successful, if the sender S_i receives an ACK from the receiver A. To successfully receive one ACK, on average, sender S_i has to transmit m 4 times. As 50% percent of these transmissions fail, only 2 transmission are received by A. So, A sends 2 ACKs back to S_i, who, on average, receives only one of them. Together, a total average of 6 packets has to be exchanged between sensor and aggregation node for the successfully acknowledged, reliable delivery of one data value.[1]

One month of temperature or pressure measurement requires an average of $6 * 44575 = 267450$ packets, costing $267450 * 245\mu\text{As} = 65.53\text{As}$. Table II compares this radio energy consumptions to the savings possible using the proposed prediction scheme – again based on a first-order polynomial. Note: Although less transmissions mean less packet loss, e.g. due to less collisions, it has been assumed that the transmissions taking place during the use of the prediction scheme are also prone to 50% packet loss. Table II, column *Radio Energy Savings*, shows the savings using the proposed scheme. As each packet's payload has a constant size of 29Bytes in TinyOS, the number of packets and transmissions does not increase if sending three 32Bit values, the whole predictor, instead of sending just one.

TABLE II
MICA2/TINYOS ENERGY SAVINGS OFFSET AGAINST CPU OVERHEAD

Type	ϵ	Radio Energy Savings/As	Total Savings/As	% of Savings
Temp.	0.10	48.85	48.70	74.31
	0.25	60.32	60.17	91.82
	0.50	63.56	63.41	96.76
Press.	0.10	53.38	53.23	81.22
	0.25	63.54	63.39	96.73
	1.00	65.34	65.19	99.48

However, you have to pay for using the prediction scheme in terms of energy: every polynomial evaluation and Kalman filter update results in extra CPU overhead which drains the battery. As mentioned in Section IV-C, a first-order polynomial evaluation requires one floating-point addition, one subtraction, and one multiplication on the sensor and the aggregation node respectively. This evaluation has to be made 44575 times, every time the sensor takes a new measurement – and the aggregation node processes it to output a new aggregate. One filter state update, necessary only on the measuring sensor, needs 4 additions and 3 multiplications, as well as the one subtraction for the threshold comparison. Finally, in case a new polynomial is required, further 2 additions and 1 multiplication have to be computed on the measuring node for extraction. MICA2's microcontroller ATmega128 is typically clocked at 4MHz and has the following *average* 32Bit floating-point operation properties:

1) Addition: takes 117 CPU cycles and therefore 27.89μs of time, resulting in an average energy consumption of 139.45nAs.
2) Subtraction: in 96 cycles $\rightarrow 22.89\mu\text{s} \rightarrow 114.45\text{nAs}$
3) Multiplication: in 362 cycles $\rightarrow 83.31\mu\text{s} \rightarrow 431.54\text{nAs}$

So, one polynomial evaluation costs 685.44nAs for the sensor and aggregation node respectively, one filter update costs

[1]The total number of 6 transmissions in the average can also be verified by modeling this system as a Markov Chain and by computing its expectation value.

1.97μAs, and polynomial extraction costs 710.44nAs. Per measurement step, a total of 3.34μAs has to be paid for CPU overhead, plus 710.44nAs for polynomial extraction if required. One month of operation, 44575 measurements, requires a total overhead of 0.15As of battery power, plus polynomial extractions. The number of necessary polynomial extractions are read off Table I: they are equal to the total number of transmissions necessary. For example, in the temperature scenario with $\epsilon = 0.5°$C, 1086 extractions costing 0.77mAs are necessary. The savings due to reduced transmissions must be offset against these additional costs. Table II shows the remaining energy savings in column *Total Savings*. Finally, the last column in Table II contains the percentage of energy that can be saved using the first order prediction setup. For example, using a threshold of $\epsilon = 1.00$hPa, 99.50% of all energy necessary for a simple, reliable transport mechanism can be saved.

VI. Discussion on Applicability

This section discusses the general applicability of the proposed prediction scheme. Typical sensor networks scenarios or applications are perfectly suited to apply the prediction mechanism, and significant energy savings due to omitted communications can be made.

In general, this scheme is appropriate for typical sensor network scenarios, where sensors continuously monitor their environment, periodically reporting measurements towards a sink. As soon as the measurements taken follow some kind of "pattern", e.g., if they can be approximated by a polynomial, significant gains can be made. For the class of phenomena like environmental temperatures, pressure, humidity and so on, applying a first-order predictor gives the best results. However, different data may require predictors with higher order for better results. In addition, data from a completely different kind, like periodic functions, may need a different prediction function, cf. Section IV-B, not based on a Taylor series, but, e.g., on a Fourier series. If the user knows the type of data in advance, which usually is a realistic assumption, it is relatively simple to choose an appropriate prediction function.

Yet, in some special cases, prediction of values does not make sense. Therefore, applying prediction would not give any benefit in terms of energy consumption, but might worsen it even more. For example, the proposed prediction scheme may not be suitable for pure event- or alarm-based sensor network scenarios: imagine the situation, where a sensor node is connected to a light switch, reporting only one bit every time the switch is pressed. The sensor sends, e.g., *1* to its aggregation node, as soon as the switch changes from *off* to *on* and *0* vice versa. Other examples are temperature sensors sending data only in case of a fire or motion trackers reporting alarm only in case of a movement. In these scenarios, not only the entropy of the transported data is high, but also the fact that a sensor does send something as soon as an event occurs is of importance. In such a situation, a sensor network would not benefit from prediction. An alarm or the event of a user pressing a switch at a certain time can not be predicted.

VII. Conclusion

Data transport is expensive in wireless sensor networks as radio transmissions and retransmissions drain valuable energy. This work presents a mechanism to efficiently reduce the total amount of data transported in wireless sensor networks. It uses the fact that, in a real-world scenario, most of the time sensors transmit data of little entropy towards a data sink. Thus, measurements can be predicted within a range of uncertainty. A prediction mechanism based on simplified Kalman filters is described that reduces the total amount of radio transfers to enhance the lifetime of battery-powered sensor nodes. Sensors do not send measured data to their aggregation node, but only polynomials for evaluation. As long as the prediction using these polynomials is precise enough, there is no need for further transmissions.

A first application of this mechanism to real-world environmental data shows enormous energy savings possible. It is difficult to estimate energy savings in general, as the chance to predict measurements and to omit transmissions heavily depends on the nature of the monitored phenomenon. However, in scenarios with values which can be approximated by a polynomial, like the environmental data shown above, remarkable energy savings are made.

References

[1] V. Turau, C. Renner, M. Venzke, S. Waschik, C. Weyer, and M. Witt, "The heathland experiment: Results and experiences," in *REALWSN: Real World Wireless Sensor Networks*, 2005.

[2] A. Honarbacht and A. Kummert, "WSDP: Efficient, yet reliable transmission of real-time sensor data over wireless networks," European Workshop on Wireless Sensor Networks, 2004.

[3] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," IEEE Transactions on Automatic Control, Vol. 49, No. 9, 2004.

[4] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Approximate distributed kalman filtering in sensor networks with quantifiable performance," International Conference on Information Processing in Sensor Networks, 2005.

[5] M. Coates, "Distributed particle filters for sensor networks," International Symposium on Information Processing in Sensor Networks, 2004.

[6] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, 1960.

[7] H. Cam, S. Ozdemir, H. O. Sanli, and P. Nair, "Secure differential data aggregation for wireless sensor networks," 2004, to appear in Sensor Network Operations, IEEE Press, http://www.eas.asu.edu/~hasancam/.

[8] University of California, Berkeley, "TinyOS – An open-source OS for the networked sensor regime," 2005, http://tinyos.net/.

[9] International Telecommunication Union, "40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM)," Recomm. G.726, 1990.

[10] A. Willig and H. Karl, "Data transport reliability in wireless sensor networks – a survey of issues and solutions," Praxis der Informationsverarbeitung und Kommunikation, 2005, vol. 28.

[11] Department of Computer Science, University of Tromsø, "Weather observations," March 2005, http://wserv0.cs.uit.no/.

[12] G. Welch and G. Bishop, "An Introduction to the Kalman Filter, TR 95-041," University of North Carolina, Tech. Rep., 2004, http://www.cs.unc.edu/~welch/.

[13] T. Cormen, C. Leiserson, and R. Rivest, "Introduction to algorithms," The MIT Press, 2001.

[14] X-Bow, "Mica – Wireless Measurement System," http://xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf, 2005.

[15] Atmel Corporation, "Atmel atmega128 datasheet," 2002, http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf.

[16] X-Bow, "Radio, RF concepts, and TOS radio stack," 2005, http://xbow.com/Support/Support_pdf_files/Motetraining/Wireless.pdf.

[17] K. Schwieger, H. Nuszkowski, and G. Fettweis, "Analysis of node energy consumption in sensor networks," European Workshop on Wireless Sensor Networks, 2004.

Acknowledgements: The authors wish to thank Bernhard Hurler, Roland Bless, and Curt Cramer for their support.