Adaptive Model Selection for Time Series Prediction in Wireless Sensor Networks

Yann-Aël Le Borgne *,1

ULB Machine Learning Group

Department of Computer Science

Université Libre de Bruxelles (U.L.B.)

1050 Brussels - Belgium

Silvia Santini [2]

Institute for Pervasive Computing

Department of Computer Science

ETH Zurich

ETH-Zentrum, IFW D41.2

CH-8092 Zurich, Switzerland

Gianluca Bontempi

ULB Machine Learning Group

Department of Computer Science

Université Libre de Bruxelles (U.L.B.)

1050 Brussels - Belgium

Preprint submitted to Elsevier *10 June 2007*

Abstract

Many practical applications of wireless sensor networks require sensor nodes to report approximations of their readings at regular time intervals. Time series prediction techniques have been shown to effectively reduce the communication effort for such applications, while guaranteeing user-specified accuracy requirements on collected data. The achievable communication savings offered by time series prediction, however, strongly depend on the type of signal sensed, and an inadequate *a-priori* choice of a prediction model can in practice lead to poor prediction performance. This paper describes our adaptive model selection (AMS) algorithm, a lightweight online algorithm that allows sensor nodes to autonomously select the statistically most suitable model among a set of candidate models. Experimental results obtained on the basis of 14 real-world sensor time series demonstrate the efficiency and versatility of the proposed framework in improving communication savings.

Key words: Wireless sensor networks, time series prediction, prediction models, adaptive model selection.

* Corresponding author.
Email addresses: yleborgn@ulb.ac.be (Yann-Aël Le Borgne), santinis@inf.ethz.ch (Silvia Santini), gbonte@ulb.ac.be (Gianluca Bontempi).

[1] Supported by European Community COMP2SYS project (MEST-CT-2004-505079).
[2] Partially supported by the European Community Embedded WiSeNts project (FP6-004400) and by the Swiss National Science Foundation (NCCR-MICS, grant number 5005-67322).

1 Introduction

In many wireless sensor network deployments, sensor nodes are distributed at various locations over a region of interest and collect data at regular time intervals [1–3]. Each sensor on a node captures a time series representing the development of the sensed physical variable over space and time. Reporting these time series to a sink [3] through the sensor nodes' on-board radio represents a significant communication overhead. Since the radio channel has limited capacity [4] and radio communication is known to be the dominant factor of energy consumption in wireless sensor networks [5, 6], the development of adequate data gathering techniques able to reduce the amount of data sent throughout the network is recognized as a key factor for allowing long-term, unattended network operation. This topic has therefore gained increasing attention in the wireless sensor network research community [7–14].

In many practical application scenarios for sensor networks, users are interested in observing physical phenomena with a pre-specified, application-dependent accuracy. Thus, gathered sensor data are usually accepted to lie within a known error bound, say $[-\epsilon, +\epsilon], \epsilon \in \mathbb{R}^+$. A sensor node regularly collecting local measurements can fit a prediction model to the real data and communicate it to the sink, which can then use the model to compute estimates of future sensor readings [7–14]. The sensor node can then reproduce the same readings estimations and transmit a model update to the sink only if the current measurement differs from the predicted by more than $\pm\epsilon$, thus avoiding unnecessary communication. In the absence of notification from the

[3] We refer to a *sink node* as either the central server or as a sensor node responsible to relay data further to the central server.

sensor node, the sink implicitly assumes that the value obtained from the shared prediction model is within the required error bound. This strategy, which we will refer to as *dual prediction scheme* or *DPS* henceforth [4], may lead to high communication and energy savings if adequate prediction models are used [8,9,11,13,14]. Typically, the model to use (e.g., constant or linear) is fixed a-priori, while model parameters are estimated on the basis of incoming data [11, 13, 14].

Fig. 1. The DPS acting on a temperature time series with error threshold set to $e_{max} = 0.5^oC$

Figure 1 illustrates how the DPS behaves on a temperature time series obtained from a real world sensor deployment [16], when the required data accuracy ϵ is set to $0.5°C$ and a simple autoregressive model is used. We can observe that the predicted data is within $\pm 0.5°C$ of the real data up to the 1261st time step. At time $t = 1262$, the prediction error exceeds the tolerated threshold ϵ and the sample collected at time $t = 1262$ is sent to the sink. The prediction model is then updated to take into account the new acquired data and from time $t = 1263$ to $t = 1272$, the predicted measurements are again close enough to the real ones, making further communication between the

[4] Other authors dubbed this strategy *dual prediction reporting (DPR)* [15].

sensing node and the sink unnecessary. At $t = 1273$, the sensor node realizes again that the sink is predicting the sensor measurements with an error bigger than ϵ and thus transmits the current reading. The procedure is repeated again at $t = 1286$, when a new update is sent to the sink. In this example, out of 35 collected readings, only 3 were effectively transmitted by the sensor node, which amounted to about 90% of communication savings. Obviously, the achievable communication savings depend, among others, on the particular sensed phenomenon, on the data sampling rate and, last but not least, on the used prediction model.

Approaches to perform time series forecasting in wireless sensor networks range from simple heuristics to sophisticated modeling frameworks [8, 9, 11, 13, 14]. These methods typically allow to improve upon the simple monitoring approach in which measurements are continuously reported at fixed time intervals. However, they also overlook two relevant issues. First, complex prediction techniques, like, e.g., Kalman filtering, rely on parameters whose identification proves to be difficult in practical settings, particularly when no a-priori knowledge on the signals is available. These difficulties increase with the flexibility of the model, or, equivalently, with the number of parameters necessary to specify it. Therefore, the more flexible the model, the less usable in practice. Second, as the DPS requires the sensor node and the sink to run the same prediction model, all the parameters of the model must be sent each time an update is needed. There exists therefore a tradeoff between the ability of a model to properly fit the signal, so as to lower the number of data transmissions and model updates, and the number of parameters that need to be computed locally at the sensor node and that are then sent to the sink when an update is needed.

In this paper, we address both these issues by introducing a generic procedure for adaptive model selection, henceforth referred to as AMS. The rationale of our approach is to use complex prediction models only if they prove to be efficient both in terms of computation and achievable communication savings, and otherwise to rely on simpler models. We consider a set of models of increasing complexity, and let the sensor nodes assess their performances in an online fashion, as sensor data are collected, on the basis of a metric that weights the number of updates by their size. It is in this way possible to select, among a set of candidates, the model that offers the highest achievable communication savings. Supported by the literature on time series prediction [17, 18], we propose an implementation of the AMS based on autoregressive models, whose parameters can be updated in an on-line fashion as new observations become available, and that are computationally thrifty to maintain. We show on the basis of 14 publicly available time series captured by real-world sensor nodes that gains achievable by complex prediction models quickly drop as the number of parameters increase, and that, therefore, very few models effectively need to be considered in practical settings. Finally, we propose to rely on a statistical procedure known as *racing* [19], to discard over time models that perform poorly so as to save sensor nodes' computational and memory resources. The obtained experimental results, presented in detail in section 5, show that the AMS provides a lightweight and efficient implementation of the DPS.

The remainder of this paper is organized as follows: Section 2 introduces the general framework of the DPS and the limits of previously proposed approaches. Section 3 describes the proposed AMS algorithm. Experimental setup and results are reported in sections 4 and 5, respectively. Finally, limits

and potential improvements are discussed in section 6.

2 The dual prediction scheme (DPS)

Continuously reporting sensor readings to a sink node at regular time intervals is the most widely used data gathering mechanism in real wireless sensor networks deployments [1, 20]. For this reason, we refer to it as the *default monitoring* data collection scheme. With respect to the default monitoring scheme, the DPS significantly reduces communication between a sensor node and the sink, while guaranteeing the data collected to be within a user-specified accuracy. The gains in communication offered by the DPS, however, depend on the ability of the used prediction model to reproduce and follow the time series captured by the sensor nodes. Providing an overview on time series forecasting techniques is beyond the scope of this paper, and an interested reader may refer to [18, 21–23]. In this section, we outline the particular problems and challenges that arise when applying the DPS in wireless sensor networks.

2.1 Time series prediction models

Let $\mathcal{X}_t = \langle X_0, X_1, X_2, \ldots \rangle$ be a time series representing the sequence of collected sensor measurements $X_t \in \mathbb{R}$ for each sampling time instant $t \in \mathbb{N}$. Let $\hat{X}_t$ indicate an estimation of the element X_t at time t and let $\mathbf{X}_{[\mathbf{0:t}]} = \langle X_0, X_1, X_2, \ldots, X_{t-1}, X_t \rangle$ be the sequence of observations up to time t. The estimate of X_{t+1} returned by the prediction model $h(\mathbf{X}_{\mathbf{h,t}}, \boldsymbol{\theta}_{\boldsymbol{h,t}})$ on the basis of $\mathbf{X}_{[\mathbf{0:t}]}$ is expressed as:

$$\hat{X}_{t+1} = h(\mathbf{X_{h,t}}, \boldsymbol{\theta_{h,t}}). \tag{1}$$

A prediction model $h(\mathbf{X_{h,t}}, \boldsymbol{\theta_{h,t}})$ for the time series $\mathcal{X}_t$ is a mapping that takes as inputs a row vector of input values $\mathbf{X_{h,t}}$ (a subset of $\mathbf{X_{[0:t]}}$), together with a row vector of parameters $\boldsymbol{\theta_{h,t}} = (\theta_1, \theta_2, \ldots, \theta_k)$, with $k \in \mathbb{N}^+$, and returns an estimate $\hat{X}_{t+1}$. We shall in the following refer to $h(\mathbf{X_{h,t}}, \boldsymbol{\theta_{h,t}})$ as h for short. Vectors $\mathbf{X_{h,t}}$ and $\boldsymbol{\theta_{h,t}}$ depend on the model h. For example, a constant model requires just one input value and no parameters, while an autoregressive model of order 2 takes 2 input values and 2 parameters. These vectors also depend on t if input values and parameters vary over time (see section 4.1). These dependencies are therefore expressed for input values $\mathbf{X_{h,t}}$ and parameters $\boldsymbol{\theta_{h,t}}$ by the means of the subscript (h, t).

A variety of different prediction models can be used to perform time series forecasting [17, 18, 23]. The choice of an adequate model depends on the nature of the time series, on the amount of available a-priori knowledge, on the required forecasting accuracy, as well as on the available computational resources.

2.2 DPS: Overview and limits

The main task of the dual prediction scheme is to run an *identical* prediction model h at both the source and the sink nodes and to use it to produce estimates of the future sensor readings, given some of the previous samples. If the predicted value differs from the actual sensor measurements by more than

a given error threshold ϵ [5], a model update [6] is transmitted to the sink.

The simplest implementation of the DPS uses a constant prediction model, henceforth referred to as CM, which allows the sink to reconstruct a piecewise constant approximation of the real sensor signal. Using a CM, no updates are sent as long as readings collected by the sensor do not diverge by more than $\pm\epsilon$ from the last reading sent to the sink. When this difference becomes bigger than ϵ, the current reading is sent, and this process is repeated over time. This approach, proposed by [9] and [8], provides appealing communication savings with respect to the default monitoring strategy. On many time series, more complex prediction techniques may easily outperform the CM, as shown by [11,13,14]. However, all these methods depend on a number of parameters that are hard to fix on the sensor nodes without adequate a-priori knowledge. For instance, the DPS may be implemented using a Kalman filter [11], which is particularly attractive in this context as it allows to model a large variety of natural phenomena. However, its use require to define a *state transition matrix*, which describes the underlying dynamics of the signal, and two covariances matrices specifying the process noise and the observation noise. In [11] these matrices are arbitrarily specified to be diagonal with all non-zero elements being equal to 0.05. While this choice may be appropriate for a very limited and specific class of signals, it is in general hard to properly specify these matrices a-priori or to estimate them on the sensor nodes. To avoid this problem, Santini and Römer [14] propose the adoption of autoregressive

[5] To work properly, the DPS requires all sent data to actually reach the sink, thus a loss-free or acknowledged communication link between the node and the sink is required. See [14] and section 6 for a discussion of some mechanisms that allow to relax this assumption.

[6] A model update can consist in either a variation in input values and model parameters or in the choice of a brand-new model.

adaptive filters, updated by the means of the LMS procedure, which are able to learn signal statistics on the fly, and can continuously and autonomously adapt to changes. At the same time, Tulone and Madden [13] interestingly cast the DPS in a more general framework, applying autoregressive models and including the identification of outliers, whose appearance is not reported to the sink. In both cases, however, neither online procedures for setting the order of the autoregressive model nor considerations about the communication overhead caused by the transmission of the necessary model parameters are provided. As we will show in section 5, these issues are nevertheless of critical importance for an efficient and effective implementation of the DPS, as they can seriously impact the achievable communication savings.

3 Adaptive Model Selection (AMS)

In this section, we present the AMS strategy, which allows sensor nodes to autonomously select a optimal model out of a set of possible prediction models, without the need of any a-priori knowledge on the sensed signal. A sensor node running the AMS maintains a set of K *candidate* prediction models $h_i(\mathbf{X}_{h_i,t}, \boldsymbol{\theta}_{h_i,t})$, $1 \leq i \leq K$. For each model h_i, a given quality measure is recursively estimated and the model that optimizes this performance indicator is selected as the *current* model. The same indicators, presented in section 3.1, also allow to run the *racing* mechanism, which discards poorly performing models from the set of candidate models, as we describe in section 3.2. Section 3.3 finally reports the detailed AMS algorithm.

3.1 Performance estimates

The main goal of the DPS is to reduce the number of updates between a sensor node and the sink. To measure the performance of the DPS it is therefore meaningful to consider the *relative update rate*, i.e. the ratio of the number of updates effectively sent when running the DPS to the number of updates that would have been sent by the default monitoring scheme. Let $U_{h_i,t}$ be the relative update rate for the model h_i at time t, where $U_{h_i,1} = 1$, $1 \leq i \leq K$. $U_{h_i,t}$ can be recursively computed as $U_{h_i,t} = \frac{(t-1)*U_{h_i,t-1}+1}{t}$ if an update is needed at time t, or as $U_{h_i,t} = \frac{(t-1)*U_{h_i,t-1}}{t}$ otherwise. The relative update rate reflects the percentage of transmitted packets with respect to the default monitoring scheme. Note that the relative update rate for the default monitoring scheme is 1 since it requires the transmission of all the collected readings, and that any lower value indicates a gain in the number of transmitted packets.

Performance assessment in terms of update rate has been considered in several implementations of the DPS [9,11,13,14]. However, this performance indicator does not take into account the fact that while an update in the default monitoring mode only consists of the current sensor readings, updating a model h_i requires the input values $\mathbf{X_{h_i,t}}$ and the model parameters $\boldsymbol{\theta}_{h_i,t}$. Consequently, performing a single update may require sending a high number of bytes to the sink, which may become critical in settings characterized by a very limited network bandwidth. To take into account the packet size of a single model update we introduce an alternative performance indicator, *relative data rate*, which we define as follows:

$$W_{h_i,t} = U_{h_i,t} * C_{h_i}. \tag{2}$$

where C_{h_i} (henceforth referred to as *model cost*) is the ratio of the number of bytes required to send an update of model h_i to the number of bytes required to send an update in the default monitoring mode. The relative data rate $W_{h_i,t}$ measures the savings in terms of data rate for model h_i at time t with respect to the default monitoring mode.

3.2 Racing mechanism

Since it is likely that some $\{h_i\}$ will perform poorly, it would be preferable not to maintain them in order to save computational and memory resources. An effective approach to detecting prediction models that perform poorly out of a set of candidate models is offered by the *racing* mechanism [19]. The rationale of the racing mechanism is to determine, on the basis of hypothesis testing [24], what models among a set of candidate models are significantly outperformed by others. For instance, let $h_i^* = \text{argmin}_{h_i} W_{h_i,t}$ be the model with the lowest relative data rate at time instant t among the set of candidate models $\{h_i\}$, and let $\Delta_{h_i,h_i^*} = W_{h_i,t} - W_{h_i^*,t}$ be the difference between the estimated relative data rates of any model h_i and h_i^*. Relying on the Hoeffding bound [25], a distribution free statistical bound, the racing mechanism assumes with probability $1-\delta$ that h_i^* truly outperforms h_i if

$$\Delta_{h_i,h_i^*} > R\sqrt{\frac{ln(1/\delta)}{2t}}, \tag{3}$$

where R is the range taken by the random variable Δ_{h_i,h_i^*}. Thanks to the lack of parametric assumptions, the Hoeffding bound requires no other information than the range of values taken by the random variables considered, which is known in advance. As $0 \leq W_{h_i,t} \leq C_{h_i}$ and $0 \leq W_{h_i^*,t} \leq C_{h_i^*}$, it follows that

$R = C_{h_i} + C_{h_i^*}$, and the bound for discarding model h_i is therefore given by:

$$\Delta_{h_i,h_i^*} > (C_{h_i} + C_{h_i^*})\sqrt{\frac{ln(1/\delta)}{2t}}. \quad (4)$$

The racing mechanism discards poor performing models from the set of candidates among which the AMS chooses the current model. Since the bound gets tighter as t increases, only one model is eventually maintained on the sensor node.

3.3 AMS algorithm

Table 1 shows the pseudocode of the AMS algorithm. It takes as inputs the error tolerance ϵ, the number of candidate models K, the set of models $\{h_i\}$, and their corresponding costs $\{C_{h_i}\}$ [7]. The first model sent to the sink is that with the lowest model cost. When the sensor collects a new reading X_t, the AMS runs the function *simulateModel*, which estimates the relative update rates $U_{h_i,t}$ for all candidate models h_i. This function first determines whether an update is necessary or not by checking if the current reading estimation $\hat{X}_t = h_i(\mathbf{X_{h_i,t-1}}, \theta^*_{\mathbf{h_i,t-1}})$, computed by model h_i at time t, is more than $\pm\epsilon$ off the actual sensor value X_t. The relative update rate $U_{h_i,t}$ is then computed as described is section 3.1. Moreover, since the parameters of a candidate model may be updated recursively as new sensor readings become available, the function *simulateModel* maintains two sets of parameters for each model h_i: $\boldsymbol{\theta_{h_i,t}}$ and $\boldsymbol{\theta^*_{h_i,t}}$. Parameters $\boldsymbol{\theta_{h_i,t}}$ are continuously updated with incoming data so that the model is constantly refined (e.g., using the recursive least

[7] The model costs must all be set to 1 if the relative update rate is used as performance indicator

square procedure for autoregressive models, as detailed in section 4.1). On the contrary, as long as no update is necessary for model h_i, parameters $\boldsymbol{\theta}^*_{\boldsymbol{h_i,t}}$ remains unchanged since they represent the parameters that would be shared by the sensor node with the sink if h_i were the current model.

After running to completion, the function *simulateModel* returns control to AMS, which then behaves as it were a "classical" DPS scheme. It therefore checks wether the absolute value of the difference between the reading estimation $\hat{X}_{t+1} = h^*(\mathbf{X}_{\mathbf{h^*,t}}, \boldsymbol{\theta}_{\boldsymbol{h^*,t}})$, computed at the sink using the current model h^*, and the actual sensor value $\hat{X}_t$ does not exceed the tolerated error threshold ϵ. If this threshold is exceeded, the current model h^* is assigned the model in $\{h_i\}$ that minimizes the chosen performance indicator, and an update composed of the input values $\mathbf{X}_{\mathbf{h^*,t}}$ and the parameters $\boldsymbol{\theta}_{\boldsymbol{h^*,t}}$ is sent to the sink.

Table 1
Adaptive model selection algorithm

Adaptive model selection algorithm	Algorithm for virtual model updates

Algorithm AMS($K, \{h_i\}, \{C_{h_i}\}, \epsilon$)
$U_{h_i,1} \leftarrow 1$ for $1 \leq i \leq K$
$h^* \leftarrow \text{argmin}_{h_i} C_{h_i}$
While True
$X_t \leftarrow$ getNewReading()
For (i in $1:K$)
$h_i \leftarrow$ simulateModel(h_i, X_t)
endFor
$\{h_i\} \leftarrow$ racing($\{h_i\}$) see Equation 3
$\hat{X}_t \leftarrow$ predictValue(h^*)
if ($\lvert\hat{X}_t - X_t\rvert > \epsilon$)
$h^* \leftarrow \text{argmin}_{h_i} U_{h_i,t} * C_{h_i}$
sendNewModel(h^*)
endIf
endWhile
	Algorithm simulateModel(h_i, X_t)
	$\hat{X}_{t+1} \leftarrow$ predictValue(h_i^*)
	update(h_i, X_{t+1})
	if ($\lvert\hat{X}_t - X_t\rvert > \epsilon$)
	$U_{h_i,t} \leftarrow \frac{(t-1)*U_{h_i,t-1}+1}{t}$
	$\boldsymbol{\theta}^*_{\boldsymbol{h_i,t}} \leftarrow \boldsymbol{\theta}_{\boldsymbol{h_i,t}}$
	else
	$U_{h_i,t} \leftarrow \frac{(t-1)*U_{h_i,t-1}}{t}$
	endIf
	Return h_i

4 Experimental Setup

In this section, we describe the setup we used to assess the performance of the AMS algorithm. The corresponding experimental results are reported in section 5.

4.1 Prediction models

We based the experimental evaluation of our AMS algorithm on autoregressive (AR) models. We tested how AR models, whose parameters can be recursively updated, can improve upon a CM when running the DPS. AR models have been chosen for two reasons. First, they have been shown to be both theoretically and experimentally good candidates for time series predictions [17, 18]. Second, model parameters can be estimated by the means of the recursive least square (RLS) algorithm [26], which allows to adapt the parameters to the underlying time series in an online fashion, without the need of storing large sets of past data.

Time series forecasting using AR models is performed by regressing the value X_t of the time series $\mathcal{X}_t$ at time instant t against the elements of the time series at the previous p time instants $(X_{t-1}, X_{t-2}, ..., X_{t-p})$. The prediction at time $t+1$ is thus obtained as:

$$\hat{X}_{t+1} = \theta_1 X_t + \theta_2 X_{t-1} + ... + \theta_p X_{t-p+1} \tag{5}$$

where $(\theta_1, \theta_2, ..., \theta_p)$ are the autoregressive coefficients and p is the *order* of the AR model, thus denoted as AR(p). Following the notations introduced in

section 3, let $\boldsymbol{\theta_{AR(p),t}} = (\theta_1(t), \theta_2(t), ..., \theta_p(t))$ be the row vector of parameters and $\mathbf{X_{AR(p),t}} = (X_t, X_{t-1}, X_{t-p+1})$ be the row vector of inputs for a model AR(p) at time instant t. Then the scalar product [8]:

$$\hat{X}_{t+1} = \boldsymbol{\theta_{AR(p),t}} \cdot \mathbf{X^T_{AR(p),t}} \tag{6}$$

returns the prediction at time instant $t+1$. The parameters $\boldsymbol{\theta_{AR(p),t}}$ can be computed by means of the RLS algorithm, which consists in a computationally thrifty set of equations that allows to recursively update the parameters $\boldsymbol{\theta_{AR(p),t}}$ as new observations X_t become available. The computational cost for an update of the vector $\boldsymbol{\theta_{AR(p),t}}$ is $3p^3 + 5p^2 + 4p$.

4.2 Datasets

The experimental evaluation is based on a set of 14 publicly available datasets, collected in real sensor network deployments. The datasets vary in terms of the nature of the observed phenomenon, signal dynamic, sampling frequency and length, and are briefly listed in Table 2.

4.3 Generic error threshold for performance comparison

To be able to compare results obtained from different datasets regardless of the specific physical quantities being examined, the influence of the threshold parameter ϵ is analyzed by considering it as proportional, through a given factor k, to the range r of the signal. The range r was computed by taking the

[8] The superscript $'T'$ stands for the transposition operator.

Table 2
Data sets

Data set	sensed quantity	sampling period	period	number of samples	source
S Heater	temperature	3 seconds	-	3000	[27]
I Light	light	5 minutes	8 days	1584	[20]
M Hum	humidity	10 minutes	30 days	4320	[16]
M Temp	temperature	10 minutes	30 days	4320	[16]
NDBC WD	wind direction	1 hour	1 year	7564	[28]
NDBC WSPD	wind speed	1 hour	1 year	7564	[28]
NDBC DPD	dominant wave period	1 hour	1 year	7562	[28]
NDBC AVP	average wave period	1 hour	1 year	8639	[28]
NDBC BAR	air pressure	1 hour	1 year	8639	[28]
NDBC ATMP	air temperature	1 hour	1 year	8639	[28]
NDBC WTMP	water temperature	1 hour	1 year	8734	[28]
NDBC DEWP	dewpoint temperature	1 hour	1 year	8734	[28]
NDBC GST	gust speed	1 hour	1 year	8710	[28]
NDBC WVHT	wave height	1 hour	1 year	8723	[28]

difference between the maximal and minimal values in the time series. The case $k = 0.01$ accounts for scenarios in which high precision is required, while $k = 0.2$ corresponds to a very rough bound on the tolerated error.

5 Experimental results

In this section, we report extensive experimental results to assess the performance of the AMS algorithm. First, we report the communication gains achievable running the "classical" DPS with the constant model (CM) and with autoregressive models of orders 1 to 5 (AR1, ... AR5), both in terms of relative update rate (in subsection 5.1) and in terms of relative data rate (in subsection 5.2). Along with these results we highlight the benefits of using the AMS, which for all time series was able to select the best model. Results for the convergence rate of the racing mechanism are reported in section 5.3, and average gains in data rate obtained on all 14 time series as a function of the tolerated error threshold are presented in section 5.4.

5.1 Gains in update rate

Table 3 reports the percentage of packets sent when running the DPS with the CM and AR models with orders from 1 to 5 (AR1, ... AR5). The error tolerance was fixed at $0.01 * r$, and results are reported for each of the 14 time series presented in section 4.2. Bold faced figures indicate models that are not significantly outperformed by the model with the lowest update rate [9].

We remark that in most cases, AR models outperformed the CM, and that performances of AR models are statistically equivalent regardless of the model order. However, the CM performed significantly better than AR models for three time series, namely I Light, NDBC DPD and NDBC WSPD, and yielded similar performances for NDBC AWP and NDBC GST. These apparent deficiencies of AR models are due to the nature of those time series, qualitatively characterized by sudden and sharp changes. These abrupt changes cause the variance in the estimation of AR coefficients to increase, making the models unstable and thus allowing a simple CM to provide better performances in terms of update rates (with gains of about 15% with respect to AR models for NDBC DPD and gains up to 8% for NDBC WPSD over a one year period). The last column of Table 3 contains the model that yielded the lowest update rate, and that was consequently selected by the AMS procedure.

5.2 Gains in data rate

In this section we assess the performances of the DPS in terms of the weighted update rate $W_{h_i,t} = U_{h_i,t} * C_{h_i}$, or data rate, introduced in section 3.1. Model

[9] One tailed t-test with respect to best model, $p < .05$)

costs C_{h_i} were computed assuming that each data sample and parameter can be stored in one byte. Accordingly, the constant model requires 1 byte to be sent to the sink, while the update of an AR(p) model requires $2p$ bytes (p bytes for the initial input values and p bytes for the parameters). The packet overhead (header and footer) depends on the specific communication protocol. For our experiments, we considered a packet overhead of $P_{overhead} = 24$ bytes, which corresponds to the average overhead of the IEEE 802.15.4 protocol, a potential standard for wireless sensor networks [6]. The size of a packet carrying an update for an AR(p) model is therefore:

$$C_{AR(p)} = \frac{24 + 2p}{24 + 1}. \tag{7}$$

Table 4 reports the performances of the CM and AR(p) models in terms of percentage of bytes sent to the sink with respect to the default monitoring mode. Note that as the cost of the CM is 1, figures of the first column of Table 3 and 4 are identical. In contrast, there is a general deterioration of performances of AR models, as the cost associated with sending their parameters lead them to lose their advantage in terms of prediction accuracy over more simple models. Out of all tested time series, AR models only outperformed the CM five times (on *S Heater*, *NDBC BAR*, *NDBC WTMP*, *NDBC DEWP*), and models eventually selected by AMS were AR(2) (three times) and AR(1) (twice). The AMS column contains the model that yielded the lowest data rate for each time series.

Table 3
Percentage of transmitted packets for DPS run with different time series forecasting methods. Bold faced numbers indicate models that yielded the best performances (one tailed t-test with respect to best model, $p < .05$).

	CM	AR1	AR2	AR3	AR4	AR5	AMS
S Heater	74	75	61	**59**	**59**	**59**	AR3
I Light	**38**	40	39	40	40	39	CM
M Hum	53	53	**49**	**50**	**49**	**49**	AR4
M Temp	48	48	**45**	**45**	**44**	**44**	AR4
NDBC DPD	**65**	85	80	80	80	80	CM
NDBC AWP	**72**	**73**	**73**	**73**	**73**	**73**	CM
NDBC BAR	51	50	39	39	39	**37**	AR5
NDBC ATMP	39	39	**36**	**36**	**36**	**36**	AR3
NDBC WTMP	27	27	**21**	**21**	**21**	**20**	AR5
NDBC DEWP	57	**52**	**52**	**52**	**52**	**52**	AR3
NDBC WSPD	**74**	84	82	83	83	83	CM
NDBC WD	85	**81**	**81**	**81**	**81**	**81**	AR1
NDBC GST	**80**	81	**80**	**80**	**80**	81	CM
NDBC WVHT	58	**56**	**56**	**56**	**56**	**56**	AR3

Table 4
Percentage of transmitted bytes for DPS run with different time series forecasting methods. Bold faced numbers indicate models that yielded the best performances (one tailed t-test with respect to best model, $p < .05$).

	CM	AR1	AR2	AR3	AR4	AR5	AMS
S Heater	74	78	**68**	**70**	76	81	AR2
I Light	**38**	42	44	48	51	53	CM
M Hum	**53**	55	**55**	60	62	66	CM
M Temp	**48**	50	50	54	56	60	CM
NDBC DPD	**65**	89	89	95	102	109	CM
NDBC AWP	**72**	75	81	88	93	99	CM
NDBC BAR	51	52	**44**	47	49	50	AR2
NDBC ATMP	**39**	41	**40**	43	46	49	CM
NDBC WTMP	27	28	**23**	25	27	28	AR2
NDBC DEWP	57	**54**	58	62	67	71	AR1
NDBC WSPD	**74**	87	92	99	106	113	CM
NDBC WD	**85**	**84**	91	98	104	111	AR1
NDBC GST	**80**	84	90	96	103	110	CM
NDBC WVHT	**58**	**58**	63	67	71	76	CM

5.3 Racing mechanism

We report in this section the convergence speed obtained when relying on the racing mechanism. Figure 2 shows the average number of models, averaged

over the 14 time series and the first 1000 time instants. The weighted update rate $W_{h_i,t}$ was used to evaluate performance of competing models (as in section 5.2), with a confidence $1 - \delta = 0.95\%$. Efficiency of the racing in terms of rapidity in discarding poorly performing models depends on the nature of the time series. The convergence to the best model in less that 1000 time instants was obtained in four cases. For other cases, subsets of two or three remaining models were still in competition after 1000 time instants. The performances of remaining models were in those cases ranging from less than 1% up to 5%, and the a posteriori best model was always observed to be part of the remaining set. AR(4) and AR(5) were discarded in all cases due to the overhead incurred by sending their parameters to the sink. For five time series, AR(3) and AR(4) were in the remaining candidates models, while for the other nine time series, either CM, AR(1), or both were still competing after the 1000th time step.

Fig. 2. Number of remaining models over time.

Fig. 3. Percentage of packet transmitted as tolerance on error prediction is relaxed (proportional to the range r).

5.4 Tolerated prediction error

This section presents the gains in data rate obtained as the accuracy threshold ϵ is relaxed. Figure 3 reports the percentage of transmitted bytes for each of the 14 time series as the error tolerance is relaxed. AMS was run using the relative data rate of competing models as performance indicator. Interestingly, for a $0.05 * r$ accuracy threshold, which corresponds to good approximation of the sensed phenomenon, less than 20% of data were sent to the sink in comparison to the default monitoring scheme. This reduction decreased down to only 5% of bytes transmitted for an error tolerance of $0.2 * r$. We should also notice that as the error tolerance increases, the predictive capacity of any method tends to converge to that of the constant model. Thus, for error tolerance bigger than $0.1 * r$, the AMS does not perform, in general, significantly better than the CM (results not reported for space constraints).

5.5 Summary

We showed that even when relying only on the constant model, the DPS yields significant communication savings with respect to the default monitoring mode. Achievable savings range from about 50%, for a very tight error threshold (one hundredth of the sensor signal range), to 96% for rough approximations (one fifth of the sensor signal range). The introduction of AR models allows for further improvements in terms of communications savings, although we observed that models with order p bigger than three are seldom chosen by the AMS due to the additional communication overhead incurred by transmitting their parameters. In any case, the AMS procedure always se-

lects the best performing model, out of the set of initial candidates, without the need of any a-priori knowledge on the underlying time series and using a completely automated, online selection procedure.

As a general guideline for scenarios in which no a-priori knowledge on the sensed signal is available, we therefore recommend to run the AMS with a set of about four models, composed of the constant model and AR models up to order three. Considerations related to the computational cost of the AMS procedure, and to alternatives to AR models are addressed in the following section.

6 Discussion

Before coming to our conclusions, we briefly discuss a few further relevant issues concerning the DPS.

(1) **Initial set of candidate models.** In this paper, we focused on monitoring scenarios in which no a-priori knowledge on the sensed signal is available, and motivated the use of AR models on the basis of their wide applicability and their low computational and memory costs. The AMS procedure, however, can be readily applied to monitoring scenarios in which some a-priori knowledge is available, by relying on more specific modeling techniques offered in the time series prediction literature. For example, PARIMA models could be used if the observed signal is known to be periodic, or Kalman filters could provide an adequate framework if the signal underlying dynamics are known.

(2) **Computational overhead.** Although the main goal of the AMS is to

reduce communication among sensor nodes, its computational cost and memory footprint should be kept low to avoid excessive energy consumption. Running the AMS following the guidelines proposed in section 5.5 allows indeed to greatly limit computations on sensor nodes. Running a CM and AR models up to order 3 requires indeed to perform about 200 operations [10] at each time step (see section 4.1). Given that the ratio of the energy spent in sending one bit of information to the energy spent in executing one instruction has been estimated to be around 2000 for a variety of real sensor network platforms [29], sending a packet of 26 bytes (208 bits) equals the energy required to perform 416000 CPU instructions, which in turn corresponds to about 2000 iterations of the AMS algorithm. This rough estimation shows that the energy required to run the AMS algorithm is highly compensated by the energy it allows to save by reducing data communication.

(3) **Network unreliability.** The main concern about the practical applicability of the DPS resides in the fact that in absence of notification from a sensor node, the sink deems the prediction given by the shared model to fall within the ϵ error tolerance. However, an absence of notification can also derive from packet losses over the wireless communication channel or a node crash. Additional procedures must therefore be considered to deal with such possible failures. For instance, a "watchdog" regularly checking sensor activity and packet sequence numbers can be set up, as already discussed in [14]. If the sink node realizes a jump in the sequence

[10] The actual number of CPU instructions corresponding to an addition or multiplication may vary depending on the particular microprocessor. There is no loss in generality in considering here the number of additions and multiplications to be equal to the number of CPU instructions. The number of computations was obtained considering the computational cost of the RLS algorithm for AR models with orders one to three.

number of received packets, it can notify the sensor node and require retransmission of the missing message. Node failures may be detected by the absence of acknowledgment from the sensor node to the watchdog request. Choice for the watchdog period depends on the application, as critical applications like fire detection would require low latency in being warned about failure of a sensor node, whereas other applications, i.e. field watering or air conditioning system, may tolerate higher latencies.

(4) **Outlier detection.** Outliers, i.e. erroneous readings, can appear in the flow of collected readings. Detecting outliers on the sensor node is an important issue if these are frequent, as they both entail useless update packets, and jeopardize the convergence of model parameters. Detection of outliers is however a non trivial task [30]. We mention briefly as solutions to this issue the possibility of adding statistical tests on measured readings as discussed in [13], or a-priori bounds on measured readings (e.g., all temperature readings outside the range $[-20°C; +50°C]$ should be ignored). Note that in case outliers are detected the use of prediction models allows to replace the erroneous (outlier) value with a more likely value.

7 Conclusions

In this paper, we introduced and evaluated the AMS, a generic algorithm for online model selection of time series prediction models, which allows to reduce data communication in wireless sensor networks. Our adaptive selection scheme makes sensor nodes smart enough to be able to autonomously and adaptively determine the optimal model to use for performing prediction-

based data collection. In order to develop a general applicable framework, we proposed a possible implementation relying on constant and autoregressive models that works in a fully automated manner. Extensive evaluation on a set of 14 time series demonstrated the ability of our algorithm to significantly reduce the number of data transmissions while complying with the poor available memory and computational resources of common sensor network platforms.

References

[1] R. Szewczyk, A. Mainwaring, J. Polastre, D. Culler, An analysis of a Large Scale Habitat Monitoring Application, in: Proceedings of the 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys'04), Baltimore, MD, USA, 2004.

[2] M. A. Batalin, M. H. Rahimi, Y. Yu, D. Liu, A. Kansal, G. S. Sukhatme, W. J. Kaiser, M. Hansen, G. J. Pottie, M. B. Srivastava, D. Estrin, Call and Response: Experiments in Sampling the Environment, in: Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04), Baltimore, MD, USA, 2004, pp. 25–38.

[3] P. Buonadonna, D. Gay, J. M. Hellerstein, W. Hong, S. Madden, TASK: Sensor Network in a Box, in: Proceedings of the 2nd IEEE European Workshop on Wireless Sensor Networks and Applications (EWSN'05), Istanbul, Turkey, 2005.

[4] D. Marco, E. J. Duarte-Melo, M. Liu, D. L. Neuhoff, On the Many-to-One Transport Capacity of a Dense Wireless Sensor Network and the Compressibility of Its Data, in: 2nd Intl. Workshop on Information Processing in Sensor Networks (IPSN'03), Springer-Verlag, Palo Alto, CA, USA, 2003.

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless Sensor Networks: A Survey, Computer Networks 38 (4) (2002) 393–422.

[6] J. Polastre, R. Szewczyk, D. Culler, Telos: Enabling Ultra-Low Power Wireless Research, in: Proceedings of the 4th Intl. Conference on Information Processing in Sensor Networks: Special Track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN'05/SPOTS), 2005.

[7] S. Goel, T. Imielinski, Prediction-Based Monitoring in Sensor Networks: Taking Lessons from MPEG, SIGCOMM Computer Communication Review 31 (5) (2001) 82–98.

[8] C. Olston, J. Jiang, J. Widom, Adaptive Filters for Continuous Queries over Distributed Data Streams, in: Proceedings of the 2003 ACM SIGMOD Intl. Conference on Management of data (SIGMOD'03), ACM Press, New York, NY, USA, 2003, pp. 563–574.

[9] I. Lazaridis, S.Mehrotra, Capturing Sensor-Generated Time Series with Quality Guarantee, in: Proceedings of the 19th Intl. Conference on Data Engineering (ICDE'03), Bangalore, India, 2003.

[10] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, W. Hong, Model-Driven Data Acquisition in Sensor Networks, in: Proceedings of the 30th Very Large Data Base Conference (VLDB'04), Toronto, Canada, 2004.

[11] A. Jain, E. Y. Chang, Y.-F. Wang, Adaptive Stream Resource Management Using Kalman Filters, in: Proceedings of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD'04), ACM Press, New York, NY, USA, 2004.

[12] Y. Le Borgne and G. Bontempi, Round Robin Cycle for Predictions in Wireless Sensor Networks, in: 2nd Intl. Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP'05), Melbourne, Australia, 2005.

[13] D. Tulone, S. Madden, PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks, in: 3rd European Workshop on Wireless Sensor Networks (EWSN'06), Zurich, Switzerland, 2006.

[14] S. Santini, K. Römer, An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks, in: Proceedings of the 3rd Intl. Conference on Networked Sensing Systems (INSS'06), Chicago, IL, USA, 2006.

[15] Y. Xu, J. Winter, W.-C. Lee, Dual Prediction-Based Reporting for Object Tracking Sensor Networks, in: Proceedings of the 1st Intl. Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), IEEE Computer Society, Los Alamitos, CA, USA, 2004, pp. 154–163.

[16] Sensor Network in a Vineyard, GoodFood EU Integrated Project: Food Safety and Quality Monitoring with Microsystems, Project Website: `www3.unifi.it/midra/goodfood/`.

[17] G. Box, G. Jenkins, Time Series Analysis: Forecasting and Control, Holden-Day Series in Time Series Analysis, Holden-Day, San Francisco, CA, USA, 1976.

[18] S. Makridakis, S. Wheelwright, R. Hyndman, Forecasting: Methods and Applications, 3rd Edition, John Wiley & Sons, 1998.

[19] O. Maron, A. W. Moore, The Racing Algorithm: Model Selection for Lazy Learners, Artificial Intelligence Review 11 (1-5) (1997) 193–225.

[20] Intel Research Laboratories Berkeley: Intel Lab Data, Project Website: `berkeley.intel-research.net/labdata/`.

[21] E. S. Gardner, Exponential Smoothing: The State of the Art, Journal of Forecasting 4 (1985) 1–38.

[22] J. D. D. Gooijer, R. Hyndman, 25 Years of Time Series Forecasting, International Journal of Forecasting 22 (3) (2006) 442–473.

[23] D. Montgomery, L. Johnson, J. Gardiner, Forecasting and Time Series Analysis, 2nd Edition, McGraw-Hill, 1990.

[24] J. Hamilton, J. Hamilton, Time Series Analysis, Princeton University Press, 1994.

[25] W. Hoeffding, Probability Inequalities for Sums of Bounded Random Variables, Journal of the American Statistical Association 58 (301) (1963) 13–30.

[26] S. Alexander, Adaptive Signal Processing: Theory and Applications, Springer-Verlag New York, Inc. New York, NY, USA, 1986.

[27] A. Stenman, F. Gustafsson, L. Ljung, Just in Time Models for Dynamical Systems, in: Proceedings of the 35th IEEE Conference on Decision and Control, Vol. 1, Kobe, Japan, 1996, pp. 1115–1120.

[28] National Oceanic Atmospheric Administration's National Data Buoy Center, Historical Data Repository: `www.ndbc.noaa.gov/historical_data.shtml`.

[29] V. Raghunathan, C. Srivastava, Energy-Aware Wireless Microsensor Networks, Signal Processing Magazine, IEEE 19 (2) (2002) 40–50.

[30] M. Markou, S. Singh, Novelty Detection: A Review, Signal Processing 83 (12) (2003) 2481–2521.