**Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin**

**Copyright statement**

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

**Liability statement**

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

**Access Agreement**

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

# Using Trust for Environment Adaptive Model-Based Energy Saving Algorithm for Wireless Sensor Networks

**Mithileash Mohan**

A Dissertation submitted to the University of Dublin, Trinity College

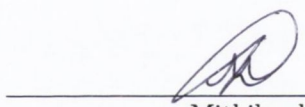
in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

June 2015

# Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.



Mithileash Mohan

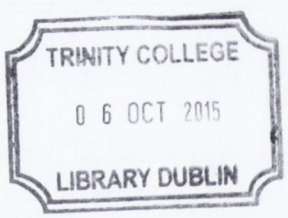
Dated: June 2, 2015

## Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this Dissertation upon request.

Mithileash Mohan

Dated: June 2, 2015



# Acknowledgements

I have faced a lot of difficulties during my time doing the PhD. It would have been impossible for me to survive through it if it was not for help from my supervisors my family and all the amazing friends I have made during these years. Thanks to my supervisors Prof. Vinny Cahill who always forced me to look at the bigger picture, whenever I was stuck down on minute details and Dr. Melanie Bouroche in always making time for me to talk about my research/registration problems and with all the help she has given me through these years.

My dad, Mr. Surendra Mohan has always been an inspirational role in how he struggles his way out of problems he faces in life through sheer strength and will power, and he has helped me do the same in writing this thesis. My mom, Mrs. Manjula Mohan has given me immense courage by always believing in me for which I have made it this far in life. A special thanks to Dr. Peer Mohammed for always telling me to face fear head on because: *The first duty of man is to conquer fear; he must get rid of it, he cannot act till then - Thomas Carlyle.*

I am also extremely grateful to all my friends who have shown me the meaning of being kind and in helping me being shaped into this strong person that I am today. In particular, my flatmates Satheesh Sivasankaran and Atul Nauityal for treating me as family and taking care of me when I had to undergo a surgery. Guoxian Yang, for discussing "weird" research in our travel trips together, encouraging me to drink alcohol by spiking my drinks and above all, always being there for me. Luca Longo, for having

to wait so long so I could party properly with him, the wait is over bro and in always encouraging me and eating steaks with me when I was down. Dan Marinescu and Jan Curn in motivating me by submitting their theses earlier than me and for all the extra motivational hugs given. Serena Fritsch for always asking me whether I needed food when working late and for reading my thesis and encouraging me by always saying it is awesome. Arnab Bhattacharya for showing me how bad I am with statistics and giving advices as my elder brother. Marcin Karpinski, Farrukh Mirza and Ricardo Carbajo for sharing their insights in WSNs (PS: Marcin stop sharing those old pics now that I have acknowledged you). As'ad Salkham for answering questions about Boltzman constant and making the first few years of my PhD fun. Servesh Muralidharan and Aravind Vasudevan for always inviting me to their home and forcing me to have a conversation during Indian society events in college. Colm Bhandal for always beating me in chess and helping me with graphing algorithms. My friends/colleagues in lab 2.15: Andrei Marinescu, Adam Taylor, Ivana Dusparic, Niall O'Hara, Dirk Hasselbach, Franck Franck and Fatemeh golpayegani for always socialising with me during lunch and taking the stress out of PhD for a bit, especially Adam Taylor for providing his amazing facts of the day during lunch. Finally thanks to all the ex DSG'ers: Tim Walsh, Neil O'Connor, Jenny Munnely, Shiu Lun Tsang, Raymond Cunningham, Dominik Dalhem and the rest (Sorry there are too many of you to write down) who welcomed me to DSG and have always been a huge support throughout my PhD days.

**Mithileash Mohan**

*University of Dublin, Trinity College*

*June 2015*

# Abstract

Wireless sensor networks (WSNs) are used in environmental monitoring applications to obtain fine-grained spatial and temporal information concerning environmental phenomena. This requires continuous sensing and communication by the sensor nodes, potentially draining their energy. As energy is provided to the sensor nodes by batteries or energy harvesting devices, which have a limited capacity, it needs to be conserved to extend the lifetime of the WSN.

Energy is mostly consumed by communication operations and possibly by sensing operations. Communication operations can be further divided into transmit and receive operations. Prediction models can be used to reduce either the transmit operations only or both the transmit and the sensing operations by using prediction estimates of the sensor readings rather than the actual readings, thus saving energy. To obtain good prediction estimates of sensor readings, the prediction model should be chosen such that the relationships of the sensor readings can be modelled appropriately. The chosen prediction model then needs to be trained to model the relationship of the sensor readings before it is possible to predict estimates of sensor readings. Training prediction models requires a set of sensed data, and therefore consumes energy as the sensor nodes are required to turn on sensing and transmission operations. Changes taking place in the environment might render the trained prediction models out of date requiring retraining of the prediction model. Furthermore, changes in the environment affecting the prediction model's performance need to be identified to reduce either the transmit/sensing

operations and maintain the sensing fidelity requirements of the application. Currently in the literature, changes affecting a prediction model's performance are identified by either having the sensing operations always on or by turning on the sensing operations at a fixed interval. Enabling the sensing operations all the time consumes more energy than turning on these operations at a fixed interval. Turning on the sensing operations at a fixed interval can also consume a lot of energy if the interval is small, but conversely if the interval is large there is the possibility of a loss in sensing fidelity.

This thesis presents the Less Battery (BLESS) algorithm, which adopts techniques to deal with an environment that sporadically changes. To reduce energy consumption and to improve sensing fidelity when using prediction models, BLESS adopts the use of a simple linear regression model trained with a small set of sensed data that is retrained opportunistically. Based on the performance of the prediction models some nodes are put to sleep while the sensor readings from the other nodes are used to predict the sensor readings of sleeping nodes. To mitigate the energy costs in checking for environmental changes affecting the prediction models' performance, BLESS adopts an adaptive approach to turning on the sensing operations in the sleeping nodes. The sensing operations are turned on based on the trust in each of the prediction models, the lower the trust, the more often the sleeping sensor nodes turn on the sensing operations. Trust in a prediction model is estimated by identifying a trend in the prediction errors of the prediction model using an exponentially weighted linear regression model. The prediction models are opportunistically retrained using the sensor readings obtained when the sensing operations are turned on. Thus retraining of a model happens more quickly when its trust is low, enabling to react to environmental changes affecting prediction model's performance quicker. The BLESS algorithm is evaluated using datasets from three real-world WSN deployments. The results show that BLESS can reduce sensing and transmission operations compared to approaches that have their sensor operations always on or turn on at a fixed interval for saving energy for environmental monitoring applications of WSNs. For example BLESS algorithm can save about 73% of sensing

and transmission operations with about 0.3% error in prediction for temperature sensor with an accuracy level of $1°C$ deployed in the Intel lab data.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

This thesis proposes an approach to reduce sensing and communication operations of wireless sensor networks (WSNs) used in environmental monitoring applications. Reducing sensing and communication operations of WSNs can lead to energy savings in the WSNs. It proposes the design of the Less Battery (BLESS) algorithm, which uses prediction models and a technique for evaluating the trustworthiness of each of these prediction models to save energy in a WSN whilst maintaining the required sensing fidelity of the WSN as specified by the user. Specifically, the sensing fidelity is maintained by turning on/off the sensor nodes appropriately and re-training the prediction model, as sporadic changes take place in the environment. In BLESS, the changes that affect the prediction model are identified based on the trust identified in the prediction model. The effectiveness of the BLESS algorithm is evaluated using three data sets obtained from real-world WSN deployments for environmental monitoring. This chapter motivates the work, provides background information on prediction models, outlines the main contributions of this work, and presents a roadmap for the remainder of the thesis.

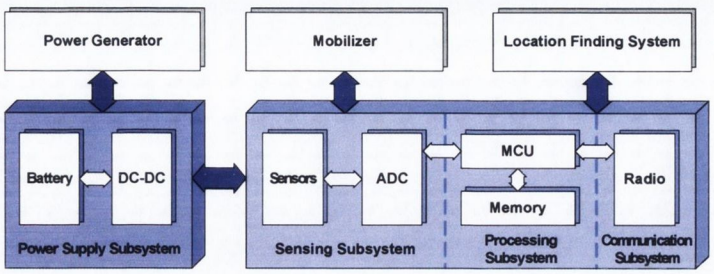



**Fig. 1.1**: Typical architecture of a sensor node (Anastasi et al., 2009)

## 1.1 Wireless Sensor Networks

A WSN is composed of a network of sensor nodes that can communicate wirelessly. A typical architecture of a sensor node in a WSN is as shown in Figure 1.1. A sensor node is usually composed of four basic components: a sensing unit, a processing unit, a transceiver unit and a power unit. Application-dependent additional components such as a location finding system and a mobilizer (for moving sensor nodes) may also be present. In general the processing and the memory units of the sensor nodes are resource constrained (Akyildiz et al., 2002). Despite such resource constraints, these sensor nodes are still capable of collecting and reporting information about the environment to one or more central entity called a sink/base station. This is made possible by sensing the phenomena in the environment through the attached sensors, processing and storing the sensor readings using the memory and processor units and finally communicating wirelessly to a specified destination through the attached transceiver. Using WSNs, data of unprecedented spatial and temporal resolution describing the phenomena under observation within the environment can be obtained (Bogena et al., 2010).

These WSNs can be utilised in many applications, for example, Yick et al. (2008) classify WSN applications as shown in Figure 1.2. Given the hardware constraints of the sensor nodes, accommodating a wide array of applications is difficult. WSNs are thus mostly tailored for specific applications. In this thesis, we focus on monitoring applica-

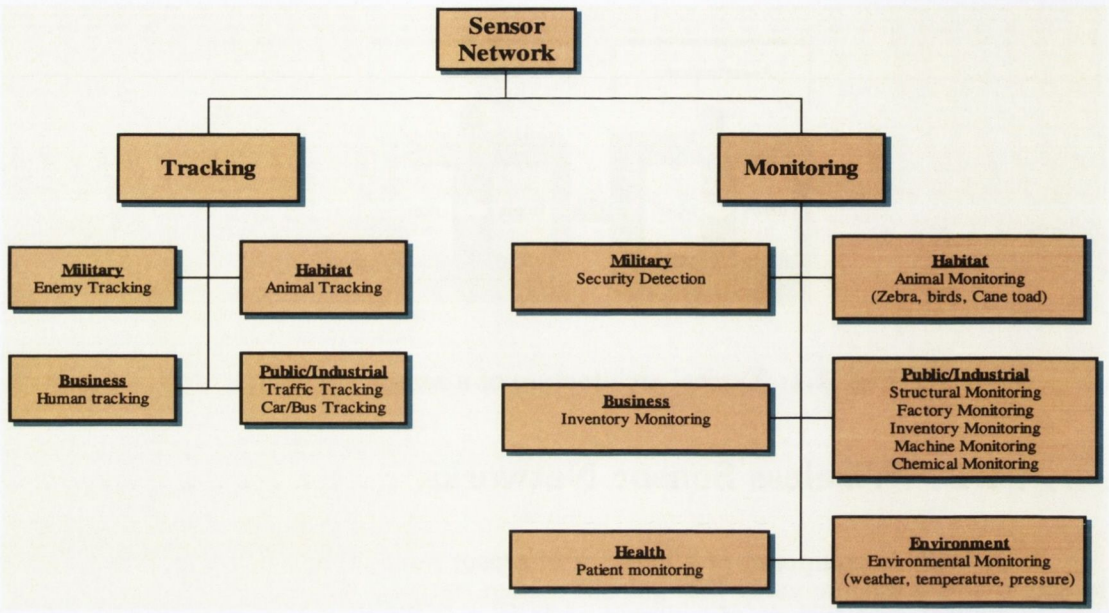



**Fig. 1.2**: Applications of WSN (Yick et al., 2008)

tions; especially monitoring of public/industrial spaces or the environment. In this thesis the public/industrial space monitoring and environmental monitoring applications are collectively referred to as environmental monitoring applications. Environmental monitoring applications are explained further in the next section.

### 1.1.1 Environmental monitoring applications

In environmental monitoring applications, sensor nodes measure a phenomenon or a set of phenomena within an environment. The monitored sensor readings can then be used to make better decisions such as controlling the light within a tunnel as implemented by Ceriotti et al. (2011) or can simply be used for observing some environment such as monitoring the habitat of plants and animals as implemented by Mainwaring et al. (2002). To take appropriate decisions or to be able to make sense of the observations, the sensor readings monitored need to be time synchronised, i.e., the sensor nodes must be sensing the phenomenon or phenomena at the same time.

Environmental monitoring applications can be further subdivided into event-based and continuous data monitoring applications. Event-based monitoring applications communicate only when a particular event occurs. Continuous data monitoring applications communicate information about the phenomenon being monitored at a specified interval called the sampling frequency. This sampling frequency is usually specified by the end user of the WSN and is based on application requirements. The higher the sampling frequency, the quicker the energy will drain from the sensor nodes. Since energy is drained quicker with high sampling frequency, this thesis focuses on reducing energy for continuous monitoring data applications requiring high sampling frequency.

For the sensor nodes to be functional in an environmental monitoring application, an energy supply is needed. The different types of energy supply possible are discussed in the next section.

### 1.1.2 Energy supply to sensor nodes

The energy required by the sensor nodes is usually supplied either through batteries or through an energy harvesting device. Batteries have limited energy capacity after which they need to be replaced. An energy-harvesting device e.g., a solar cell converts different forms of environmental energy into electricity to be supplied to a sensor node (Niyato et al., 2007). These harvesting techniques can only produce energy at a limited rate (Niyato et al., 2007) and the harvested energy typically varies with time in a nondeterministic manner (Kansal et al., 2007). Thus if the environmental monitoring application needs to survive for a long time without sensor nodes dying from lack of energy, energy needs to be conserved and managed appropriately.

The next section discusses how energy can be saved in WSNs.

## 1.2 Energy saving in WSNs

Sensor nodes need to conserve energy to extend the lifetime of the WSN. While energy expenditure depends on the specific node, in general the following statements hold:

1. Typically in a sensor node, the communication operations consume more energy than the computing operations (Raghunathan et al., 2002).

2. Depending on different factors, the energy consumed by sensing operations may be high and cannot be ignored (Raghunathan et al., 2006).

For these reasons, approaches for saving energy in sensor nodes are generally based on reducing the number of communication operations and/or the sensing operations. There are two types of communication operations, the transmit operation, which transmits data wirelessly, and the receive operation, which receives data. The energy consumed by the transmit operation varies depending on the size of the data and the distance over which the data must be transmitted. For receive operations, the energy consumed is usually a constant value. Depending on the type of radio used and the distance over which to transmit the data packets, the energy consumption might be high for either the receiving operations or the transmitting operations or both. The energy consumption of the sensing operations is dependent on the type of sensors. For example, in a typical WSN application monitoring temperature and humidity readings using mica2 motes with a Crossbow MTS400 environmental sensor board, the energy consumption for the transmit, receive and sensing operations are shown in Table 1.1. The values in Table 1.1 are derived from the assumptions of Deshpande et al. (2004) that the transceiver needs to be on for about 27ms to receive or transmit wireless packets and to sense the temperature or humidity the sensors need to be kept on for 1s.

To conserve energy, transmit, receive and sensing operations need to be reduced. Depending on the application scenario such as the application requirements and hardware used, these different operations might consume different amounts of energy. This

| Operation | Energy consumption (mJ) |
|---|---|
| **Transmit** | **0.4** |
| **Receive** | **0.4** |
| **Temperature & Humidity sensing** | **0.5** |

**Table 1.1**: Typical energy consumption of mica2 motes sensing temperature and humidity (Deshpande et al., 2004)

reduction in the operation should not affect the sensing fidelity requirements of the environmental monitoring applications. The sensing fidelity requirement can be different for different applications and this requirement can be taken advantage to conserve energy. For example, in the WSN deployed by Ceriotti et al. (2011) to control lighting in a tunnel, Raza et al. (2012) have determined that an accuracy of about 25 lux is sufficient for light sensors (including errors of light sensors). This thesis proposes the Less Battery (BLESS) algorithm, to take advantage of the sensing fidelity requirement specified by the user, to reduce the transmit and the sensing operations within the WSN to conserve energy. This is made possible by using prediction models to predict the sensor readings of certain sensor nodes and putting these sensor nodes to sleep, thus reducing the sensing and transmission operations.

In the next section some important concepts of prediction models are explained.

## 1.3 Prediction models for energy saving in WSNs

In this section, the general concepts of prediction models are described first, then the different types of prediction models are described followed by a discussion of applying these prediction models for saving energy in WSNs.

### 1.3.1 Overview

It is commonly understood that a prediction model estimates the relationships between different variables. Initially, the prediction model is trained using some values of the variables, this phase is referred to as the training phase in this thesis. In the training phase, parameters are fitted to the model so that a prediction estimate of one of the variables (response variable) can then be made given the value of one or more of the other variables (predictor variables). When parameters are fitted to a model, the accuracy of the model can be expressed using a confidence interval and for a new predicted value the accuracy can also be expressed using a prediction interval.

1. *Confidence interval of a model:* Let us assume that the data used in training the model comes from any probability distribution. Once we fit the prediction model to this data, a way to judge its accuracy is to use a confidence interval on the fitted model. The formal definition of a $(1-\alpha)\%$ confidence interval states that if the data collection process from the probability distribution is repeated several times and confidence intervals for the fitted model are computed for each of those datasets then $(1-\alpha)\%$ of those intervals will contain the true model.

2. *Prediction intervals:* A prediction interval is defined as a confidence interval for a predicted value at a new point and it is always wider than the confidence interval of the model.

The typical values used for $(1-\alpha)\%$ are 90%, 95% and 99%. For example let us consider a variable $X$, which is believed to be linearly related to a second variable $Y$, and is used to predict the value of $Y$ using a linear regression model of the form $Y = A + BX + \epsilon$, as the prediction model. $X$ is assumed not to be the perfect predictor of $Y$ because of possible chance variation, which prevents making perfect predictions and where $A$ and $B$ are the model parameters, this chance variation is denoted by white noise $\epsilon$. The linear model needs to be trained to get prediction estimates of $Y$.

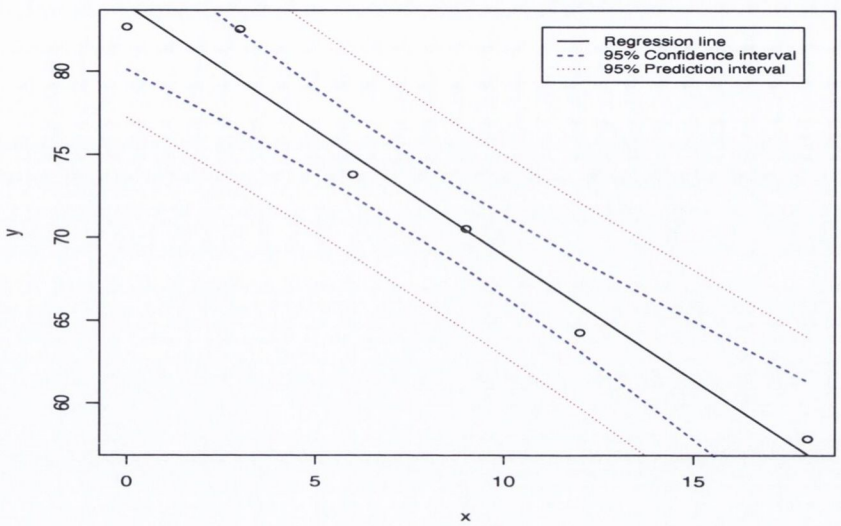



**Fig. 1.3**: Data, regression line, confidence interval & prediction interval (example is obtained from (Mullins, 2003))

By training the model $Y = A + BX + \epsilon$, estimates of the model parameters $\hat{A}$ and $\hat{B}$ are identified, which can then be used to derive the mean prediction estimates of $Y$, denoted as $\hat{Y}$. To train the model initial values of both $X$ and $Y$ are required. Now suppose the values of $X = \{0, 3, 6, 9, 12, 18\}$ and the corresponding values of $Y = \{82.7, 82.55, 73.8, 70.52, 64.28, 57.82\}$. Using these values to train the linear regression model will result in estimating the value of $\hat{A}$ to be 83.987 and $\hat{B}$ to be $-1.50529$ respectively, giving Equation 1.1. The regression line and the intervals for the model in Equation 1.1 is shown in Figure 1.3.

$$\hat{Y} = 83.987 - 1.50529X \tag{1.1}$$

Similarly, the relationship between sensor readings of the WSN can be estimated and used to predict future sensor readings. In WSNs, sensor nodes sense and transmit readings at different times and these sensor readings could be related over time and

prediction models that identify these relationships are referred to as temporal models in this thesis. Apart from sensor readings sensed and transmitted at different times, sensor readings also are sensed and transmitted from spatially distributed sensor nodes giving rise to a relationship in space and time. Prediction models that identify the relationship existing over space and time are referred to as spatio-temporal models in this thesis. Prediction models could also be classified as either parametric or non-parametric models as follows:

### 1.3.2 Parametric models

In parametric models, the model that represents the relationship is selected a priori within a finite set of model parameters. An example of such a parametric model is a regression model. In the case where the predictor variables are related linearly, a linear model as in Equation 1.2 could be used, an example of a non-linear model is Equation 1.3 and finally, if information from multiple variables is used to perform the prediction, a model such as Equation 1.4 could be used.

$$Y_i = A + BX_i + \epsilon_i \tag{1.2}$$

$$Y_i = A^{X_i} + e^{BX_i} + \epsilon_i \tag{1.3}$$

$$Y_i = A + BX_i^2 + CZ_i + \epsilon_i \tag{1.4}$$

Where $X_i$, $Z_i$ are the variables used for predicting the variable $Y_i$, and $A$, $B$ & $C$ are the parameters that describe the model. $A$, $B$ & $C$ are identified during the training phase of the prediction model and $\epsilon_i$ is the chance variability component and referred to as the model error. $\epsilon_i$ is usually assumed to be distributed normally with a standard deviation of $\sigma$ and a mean of zero. The regression line as specified by the model is a line that simply joins the means of these distributions (Mullins, 2003). The residual error $e_i$

can then be calculated by: $e_i = Y_i - \hat{Y}_i$ for each $i$. The term prediction error has been used synonymously with residual error for the rest of the thesis.

### 1.3.3 Non-parametric models

Unlike a parametric approach where the model is fully described by a finite set of parameters, nonparametric modelling accommodates a very flexible form of the model representation. The model representation is determined from the values of the variables. An example of a non-parmetric model is spline regression (Silverman, 1985). Though non-parametric models clearly offer the benefit of not needing to specify the model representation, some issues do exist. Determining the non-parametric model representation and identifying its parameters can be computationally expensive. Also for higher dimensions of predictor variables a large training dataset is required to identify the non-parametric model representation and identify its parameters (Györfi, 2002).

### 1.3.4 Using prediction models to save energy in WSNs

Prediction estimates of sensor readings can be obtained using prediction models which can be used to reduce sensing and transmission operations, thus saving energy. There can exist a prediction model for each sensor node or for a sensor node pair or for groups of sensor nodes depending on the application. Usually approaches employing a prediction model per node, model only the relationship of sensor readings from that node over time. Approaches employing prediction models for pairs or group of sensor nodes, usually model the relationship of inter-node readings between these nodes and are able to predict estimates of some of the sensor readings given the sensor readings of other nodes.

Regardless of the approach or the type of prediction model chosen, all models need to be trained with readings from sensor nodes before prediction estimates of sensor readings can be obtained. Obtaining readings from the sensor nodes requires them to turn on the sensing operation and possibly the transmission operations. Thus the training of prediction models consumes energy from the sensor nodes. The prediction

models used can be either parametric or non-parametric models. Parametric models require the identification of the model representation. Parametric models are usually described using few parameters, therefore they are suitable to be trained with small number of sensor readings (Györfi, 2002). However, identifying this model representation can be difficult, if the relationships of the sensor readings are not known a priori. Non-parametric regression can be used instead to solve this issue. Most sensor readings are one dimensional, in the case they have more than one dimension, a large training dataset is required for non-parametric regression approaches, making it energy expensive (Györfi, 2002). Even if the data is one dimensional, the number of parameters used to describe the model is usually higher than parametric models, thereby more sensor readings might be needed to train non-parametric models compared to parametric models. Thus requiring more energy to train non-parametric models compared to parametric models.

To alleviate these problems, BLESS proposes to use parametric models, since a good linear approximation for the sensor readings' relationships can be identified when the frequency of the sampled sensor readings is high. Thus, in BLESS a linear regression model of the form shown in Equation 1.2 is used as the prediction model. It should be noted that the modelled linear relationship might exist only for a small time window. At the end of this time window re-training needs to take place to ensure that sensing fidelity is maintained. This approach is similar to piecewise linear regression, where the prediction variables are partitioned into segments and a separate line segment is fit to each (Hastie et al., 2009). The main difference between piecewise linear regression and the approach of BLESS is that the partitions are not done a priori.

Given a trained prediction model, prediction estimates of sensor readings can be obtained. Then the sensing and transmission operations of sensor nodes whose readings are being predicted by the models can be put to sleep/idle state, reducing the sensing and transmission operations thus saving energy. In this thesis the words sleep and idle are used interchangeably. It is difficult to estimate the sensor readings accurately given the chance variation. Thus prediction models are used in WSN applications that do not

require 100% accuracy. A small loss in accuracy is a trade-off for reducing sensing and transmission operations when prediction models are used. Application users define the acceptable loss in accuracy, as a bounds on the maximum acceptable prediction error which is termed user-specified error ($u_{error}$), in this thesis. Approaches using prediction models try to monitor the accuracy of the prediction estimates within the bounds of $u_{error}$.

Changes in the relationship between the sensor readings identified by the trained prediction model can occur making the prediction model obsolete. The next section describes these issues, followed by some existing approaches used in dealing with these issues and then describes the approach used by BLESS.

## 1.4 Relationship changes in the identified prediction model

In environmental monitoring applications, changes take place continually in the environment in which the sensors are deployed, such as switching on and off air conditioning, opening windows (for temperature sensors), turning on/dim/off lights, sun light (light sensors). Apart from these trivial changes, (Koushanfar et al., 2006) mention that changes can also happen depending on weather patterns, workday, holidays etc. As these changes take place, changes might also take place in the relationship between different sensor readings represented by the prediction model. Such changes in the relationship need to be identified. Identifying these changes is crucial for the following reasons:

1. The change in relationship might not allow the prediction models to give a prediction estimate within the bounds of $u_{error}$; thus compromising sensing fidelity.

2. The change in relationship might enable predicting the estimates of certain sensor readings previously not possible within the bounds of $u_{error}$; thus further reducing the sensing and transmission operations.

The performance of prediction models thus needs to be monitored to detect significant changes in the relationship of the sensor readings modelled by the prediction models. Once these changes are identified, actions need to be taken to improve the sensing fidelity and to reduce sensing/transmission operations in the WSN. Currently in the literature, there are two methods proposed to monitor changes in the performance of prediction models and take appropriate actions: the dual-prediction and the frequent-monitoring approaches.

### 1.4.1 Dual-prediction

The prediction models used in most of the dual-prediction approaches model only the relationship of sensor reading within a sensor node across time. This trained prediction model resides in each sensor node and the sink node, and is used to predict future sensor readings from that node. In order to monitor the performance of the prediction model, the sensing operations are always kept on to check the actual reading against the predicted reading. If the predicted reading is not within $u_{error}$, the actual reading is transmitted. Usually, in these approaches, once the errors are not within $u_{error}$ for a significant period of time, re-training of the model takes place. This re-trained model's new parameters are then transmitted to the base station, so the sensor node can turn off transmit operations, thus saving energy.

An issue with such an approach is that the sensor operations are not minimised and are always on, which might lead to energy drain.

### 1.4.2 Frequent-monitoring

The frequent-monitoring approaches eliminates the problem of always-on sensing operations by frequently switching on the sensors to check the performance of the prediction model. If the interval between monitoring is small, energy might be wasted by checking for poor performance in prediction models when no relationship changes happen. Conversely if the interval is large, the prediction model's performance might have decreased

during that interval, which might lead to errors in the prediction. In these approaches, usually one of two methods are used for switching on/off the sensor nodes as changes take place:

1. Once an error above $u_{error}$ is detected, the nodes are turned on and re-training of the models take place appropriately so the nodes can be put back to sleep.

2. The prediction models are re-trained at a frequent interval. The sensor nodes are turned on/off according to the newly trained prediction models.

A fixed frequency of monitoring sleeping sensor nodes to assess the performance of the prediction model can increase sensing and transmit operations or reduce the sensing fidelity. Making the monitoring interval adaptive by monitoring the sleeping sensor nodes more often as the performance of the prediction model is suspected to decrease, could further reduce the sensing and transmission operations and the sensing fidelity. These monitored sensor readings can then be used to re-train the model quicker during low performance of the model and based on the re-trained model, the sensor nodes can be turned on/off appropriately to reduce the sensing and transmission operations or sensing fidelity. The BLESS algorithm proposes a solution for such an approach and is explained in the next section.

## 1.5 Approach

Unlike the dual-prediction and the frequent-monitoring approaches, BLESS adaptively increases or decreases the monitoring of the sensing operations to check the performance of the prediction model as needed. The increase or decrease of the monitoring frequency is based on the current estimated trust in the performance of the prediction model. The trust in the performance of the prediction model is identified based on the trends of the residual errors ($e_i$). In time varying data such as WSN sensor readings, relationships exist in the $e_i$ values and a trend in this relationship is identified in BLESS by an

exponentially weighted linear prediction model. Using this modelled trend, BLESS can predict estimates of the future residual errors based on the previous and current values of residual errors appropriately. Based on the predicted estimates of the residual errors in the future, the trust in the prediction model is increased or decreased. The value of the trust can lie between 0 and 1; 0 indicating no trust and 1 indicating complete trust in the model.

In order to identify the trend of the residual errors, the sensing operations must be monitored, leading to a paradoxical situation. To solve this issue, BLESS proposes a default monitoring interval $K$. When the trust of the model is 1, the performance of the prediction model is monitored at every $K$. As the trust in the performance of the model declines, the value of k decreases, thus increasing the frequency of monitoring. Each time the performance of the prediction model is monitored, the trust in the prediction model is updated and the monitoring frequency adapted accordingly.

In BLESS, the prediction model used is a simple linear regression (SLR) model as discussed in Section 1.3.4. Based on the trust identified in the SLR models, the BLESS algorithm selects certain sensor nodes to be active and puts the rest of the nodes into sleep/idle state, reducing sensing and transmission operations. In this thesis the words sleep and idle are used interchangeably. Using the SLR models and the active nodes' sensor readings, the sensor readings of the sleeping nodes are estimated. Depending on the trust in the prediction model, the monitoring of the sleeping nodes is done at the identified monitoring frequency. The monitored sensor readings are used to check the performance of the prediction model and are also used to re-train the prediction model after sufficient monitored sensor readings are collected. In BLESS, the re-training of the models representing the relationships between all the sensor nodes takes place whenever sufficient sensor readings to re-train the models are collected. The trust is then re-estimated in all the re-trained models. The lower the trust in the model, the higher the frequency of monitoring of the sleeping nodes thus enabling re-training of the prediction model quicker. Based on the values of trust identified across the re-trained prediction

models, BLESS increases or decreases the number of active nodes required to improve the sensing fidelity and reduce sensing and transmission operations.

In the next section, we discuss the scope and assumptions of the BLESS algorithm.

## 1.6 Scope and assumptions

The scope of applicability of the BLESS algorithm and its main assumptions are described below.

### 1.6.1 Time synchronisation

Environmental monitoring applications, which take actions or observe the environment based on sensor readings monitored by sensor nodes, require sensor nodes to be time synchronised. If the sensor readings sensed by the sensor nodes are not time synchronised, inference about the phenomenon being monitored might prove difficult or at times a wrong inference could be made. Similarly in BLESS, time synchronisation is required to infer the relationship between the sensor readings during the training phase of a prediction model. For this reason, we assume that there is a time synchronisation algorithm running in the WSN, such as (Maróti et al., 2004) . The energy consumption of the time synchronisation algorithm used is not considered in this thesis. It is assumed negligible as the purpose is to evaluate the performance of BLESS and not the time synchronisation algorithm.

### 1.6.2 Energy management

It is assumed that sensor nodes can turn 'on' and put the different components to 'sleep' in order to manage energy better. It is also assumed that the energy consumed during the 'sleep' state is considerably low than the 'on' state. This is typically the case as seen in the Telosb datasheet (Crossbow, 2013). The energy consumed in transitioning from the 'on' to the 'sleep' states or the 'sleep' to the 'on' state is considered negligible.

### 1.6.3 Sensor types

Mobile or audio/video sensors are currently not considered in this thesis. Energy efficiency in mobile nodes might not be of major concern as it can possibly be recharged. Unlike traditional sensor nodes, the energy of audio/video sensor nodes might be dominated by processing operations (Akyildiz et al., 2007). Different techniques are thus required to improve energy efficiency for those sensors. For this reason, we do not consider the use of audio/video sensors in the envisaged applications.

### 1.6.4 Transmission of sensed reading

We assume that whenever sensing operations take place a subsequent transmit operation is done which transmits the sensed reading. This enables the applications to have the latest information about the environment and also the applications can quickly react to changes in the environment if required.

### 1.6.5 Linear relationship of sensor readings

The prediction model used to predict sensor readings is a linear model. Thus, the sensor readings of idle nodes can only be predicted accurately if they exhibit a linear relationship with the active nodes' sensor readings. As shown in Appendix A, linear models will have less approximation errors as the sampling of the sensors are increased. BLESS will thus perform the best for those WSN applications that require sensors to sample frequently. It should also be noted that the energy usage is likely to go up as the sampling frequency of sensors increase, as more sensing and transmitting operations are required. Thus, there is a higher need for energy saving algorithms for WSN applications that require sensors to sample frequently.

### 1.6.6 Continuous data monitoring applications

Sampling of sensors in event-based monitoring applications might take place sporadically unlike, continuous data monitoring applications. Thus, identifying a linear relationship between sensor readings to predict sensor readings accurately might not be possible. As a result, BLESS is only applicable for continuous data monitoring applications.

### 1.6.7 Sporadic changes

BLESS assumes that changes in the environment affecting the relationships of sensor readings happen sporadically. So, BLESS can take advantage of the trained prediction model to predict sensor readings. If changes were taking place continually, the trained prediction models by BLESS will not be able to predict sensor readings accurately. This is because the relationship represented by the prediction model would not be valid as a result of the change that took place in the environment.

### 1.6.8 Routing algorithm is known at the sink node

Usually a routing algorithm identifies which intermediary nodes to choose and specifies a route from a sensor node to the sink node. An assumption is made, that the routing algorithm used in a WSN, provides the sink node with all routes that each node needs to take to deliver messages to the sink node. Based on these routes, the intermediary nodes required for the representative node sets are identified at the sink node. The sink node can then add the appropriate required intermediary nodes to the representative node set.

### 1.6.9 Energy in sink nodes and cluster heads

We assume that the energy available in sink nodes and cluster heads are significantly larger than the sensor nodes. As a result BLESS does not need to conserve energy on sink nodes or cluster heads.

### 1.6.10 Computation and memory in sink nodes and cluster heads

We assume that the computation power and memory available in sink nodes and cluster heads are adequate to perform all the different operations required by BLESS.

## 1.7 Thesis contributions

- The first contribution of this thesis is a demonstration that simple linear regression models can be used to predict sensor readings of sensor nodes in a WSN used in environmental monitoring applications involving high sampling rates. In particular, this thesis shows how a small training dataset can be used to train SLR models and re-train the prediction models whenever enough training data is available to give good prediction estimates of the sensor readings.
- The second contribution of this thesis is to demonstrate how the trust in the SLR models can be estimated and how it can be used to adapt the monitoring frequency, hence improving energy efficiency while maintaining sensing fidelity.
- The third contribution of this thesis is utilising the trust calculated in the models to adaptively increase or decrease the number of active nodes as changes in the relationship of the sensor readings happen.

## 1.8 Thesis organisation

The remaining chapters of the thesis are organised as follows. In Chapter 2 we show the problems with some of the current approaches. Chapter 3 provides the design and implementation of the BLESS algorithm that identifies solutions to the problems identified in chapter 2. Chapter 4 then evaluates the effectiveness of the BLESS algorithm for different scenarios and is compared against results from a related approach. Finally, the thesis is concluded by summarising all the above chapters and by discussing the future work in Chapter 5.

# Chapter 2

# State of the Art

This chapter discusses some of the approaches described in the literature for saving energy in a static WSN for environmental monitoring applications requiring continuous sensing and transmission of sensor readings. The different approaches are classified based on the principles employed in saving energy. Finally, the advantages and disadvantages of the different approaches are discussed motivating the BLESS protocol.

## 2.1 Overview

Energy in sensor nodes is saved by reducing the number of transmit, receive and/or the sensing operations. In this thesis, we classify the energy saving approaches based on the number of operations reduced as shown in Figure 2.1. In the following sections, the different approaches used for saving energy in transmitting, sensing and receiving operations are detailed.

## 2.2 Energy saving for the transmit operations

Sensor nodes in a WSN transmit data packets in the following scenarios:

1. Transmitting a sensed reading of the phenomenon under observation.

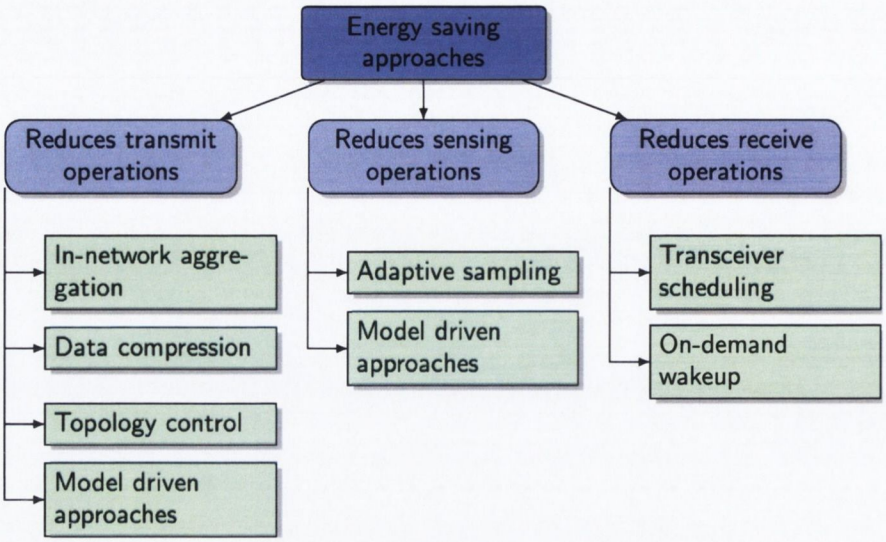



**Fig. 2.1**: Energy saving classification

2. Transmitting a control message. A control message is generally used by the WSN to control the network and its applications.

3. Acting as a bridge node transmitting readings or control messages of other sensor nodes to their intended destination.

The energy required to transmit data varies depending on the size of the data to be transmitted, the expected distance to the receiver and the type of transceiver used. Thus, in general, a reduction in energy usage by transmission operations are observed by reducing the number and the size of sensor readings to be transmitted or by optimising the transmission distance through a multi-hop network. Energy saving in the transmission operations is further classified by us as shown in Figure 2.2. It should be noted that the overhead of transmission operations due to control messages are entirely dependent on the protocols implemented in the WSN and is thus not included in the classification of reducing transmit operations.

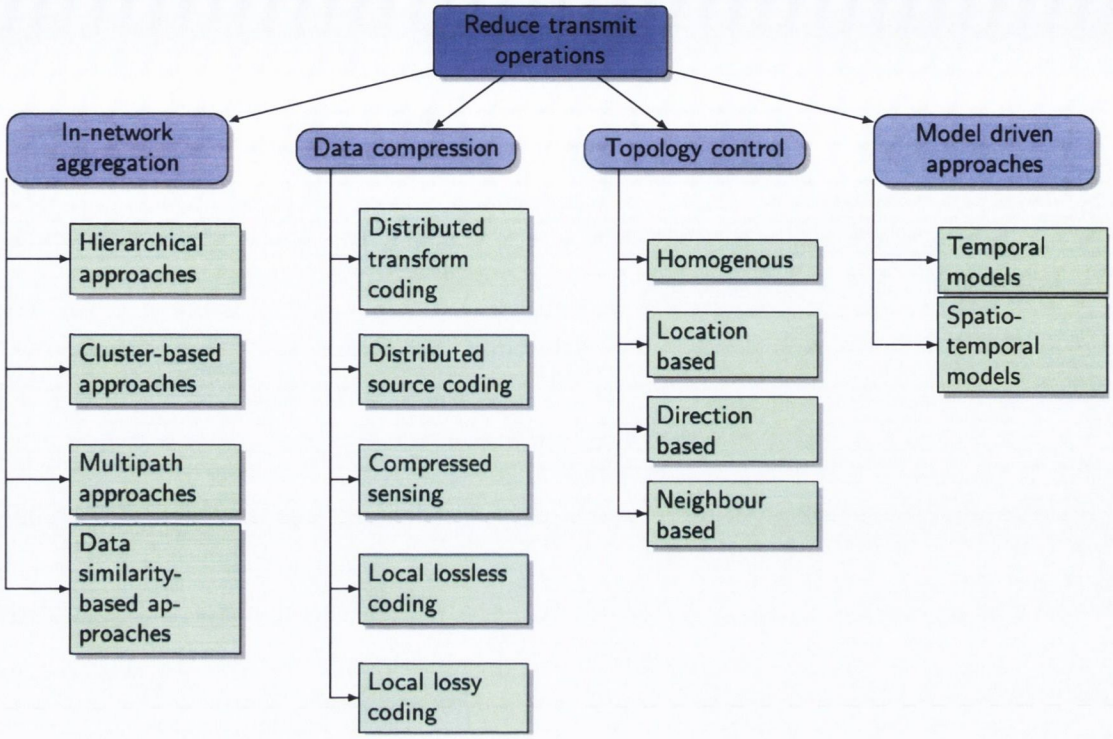



**Fig. 2.2**: Classification of transmit operations

### 2.2.1 In-network aggregation

In-network aggregation is defined as: *"The global process of gathering and routing information through a multi-hop network, processing data at intermediate nodes with the objective of reducing resource consumption (in particular energy), thereby increasing network lifetime."* (Fasolo et al., 2007).

Data aggregation techniques generally tradeoff transmit operations for computational operations. Data aggregation achieves energy savings as computation operations in WSNs generally consume negligible energy (Raghunathan et al., 2002). In-network aggregation is comprised of at least a routing protocol and a data aggregation function. Routing protocols for in-network aggregation require routing of data to appropriate nodes for aggregation, which might not necessarily constitute the shortest path in the

WSN. Introducing delays within the routing of data can provide more opportunities to aggregate data as the nodes propagate the data within the WSN. This is because by introducing delays within the routing of data, one or more sensor nodes are forced to wait until it receives readings from different sensor nodes as the readings are propagated in the WSN (Roedig et al., 2004). The sensor nodes that has received multiple readings from other nodes can then aggregate the sensor readings as the readings propagate within the WSN. The aggregation function merges the data from different nodes or from the same node into smaller packets and can either be lossy or lossless (Fasolo et al., 2007). The aggregation function is lossy if at the receiving end it is not possible to reproduce all the original sensor reading with the aggregated value. For example, some commonly used lossy aggregation functions are: average, sum, min and max values. The aggregation function is lossless if at the receiving end it is possible to completely reproduce all the original sensor reading using the aggregated value. For example, extracting sensor readings from multiple sensor readings (from different nodes or the same node) and putting into a single wireless packet. The aggregation function is usually specific to the application of the WSN. Some of the main approaches using in-network aggregation are discussed below.

#### 2.2.1.1 Tree-based approaches

In tree-based approaches the WSN is organised into a tree structure and at each level of the tree hierarchy aggregation is performed as the data is routed up from the child node to the destination. One such approach is TAG (Tiny AGgregation) proposed by Madden et al. (2002) that allows users to pose declarative SQL-style queries over a WSN for applications that require continuous data. TAG is divided into two phases: a distribution phase and a collection phase. During the distribution phase the SQL-style query is propagated using a tree-based routing scheme from the sink node. In the collection phase the sensor readings of the sensor nodes are aggregated at each level in the hierarchy of the tree as the readings are routed up in the direction of the sink. For

aggregation to take place during the collection phase, parent nodes need to wait until the information is received from their child nodes before data is propagated towards the sink. This is accomplished by having the parents subdivide the reporting time duration such that the children are required to deliver their state records during a parent-specified time interval.

#### 2.2.1.2 Cluster-based approaches

Similar to TAG, Low-Energy Adaptive Clustering Hierarchy (LEACH), proposed by Heinzelman et al. (2002), aggregates data at each level in a hierarchy. Unlike TAG, which uses a tree structure, LEACH uses a cluster structure. Cluster-heads are first identified at random based on a parameter $P$ that specifies the percentage of cluster-heads required within the WSN. The identified cluster-heads then advertise themselves to neighbouring nodes using a carrier sense multiple access (CSMA) MAC protocol[1]. Sensor nodes join a particular cluster-head depending on the signal strength of these advertised messages. When all nodes are either part of a cluster or are cluster-heads, the sensor nodes start sending data to the cluster-head using a time division multiple access (TDMA) MAC scheme. The readings are then aggregated by the cluster-head and transmitted directly without the help of other nodes to the sink. A cluster-head will appropriately increase the transmission power depending on its distance to the sink. LEACH also improves energy balance within the WSN by rotating the cluster-heads. Similarly, Cougar, proposed by (Yao and Gehrke, 2002) and improved by (Y.Yao and Gehrke, 2003), uses a cluster structure for aggregating data. In the case of Cougar, the cluster-heads are chosen by taking into account the cost of data delivery from source sensors to the cluster-head and the delivery cost from the cluster-head to the sink node. In Cougar, the cluster-heads do not report sensor readings to a sink until they have received all readings from source nodes within $T_{send}$. $T_{send}$ specifies the time for cluster-heads to wait before they can

[1]MAC protocols tries to ensure that no two nodes are interfering with each other's transmissions and deals with the situation when they do (Van Dam and Langendoen, 2003)

transmit the received sensor readings. A prediction mechanism is used to determine the value of the $T_{send}$. Timeouts and backoffs are implemented to deal with wrong predictions of $T_{send}$.

#### 2.2.1.3 Multipath approaches

Device/link failures in the above-mentioned approaches are likely to incur high costs as a result of the need to re-organise a cluster or tree. Multipath approaches are used to deal with dynamic networks and device/link failures by making use of the multiple paths available within the WSN. Synopsis diffusion proposed by Nath et al. (2004) is one such approach where the sensor readings are propagated through all possible paths by utilising the broadcast characteristics of the wireless medium. There is the possibility of receiving duplicate readings in this approach causing errors in the aggregated values depending on the aggregation function used. For example, if average was used as the aggregation function there will be no issues regards to duplicate values but if sum was used instead as the aggregation function duplicate reading can give wrong aggregation results. Nath et al. (2004) also propose a solution to deal with duplicate sensor readings.

#### 2.2.1.4 Data similarity-based approaches

One of the drawbacks of all the above approaches is that all the nodes within the WSN need to transmit sensor readings. This consumes more energy compared to only some nodes transmitting the sensor readings. Data similarity between sensor readings is taken advantage of in Clustered aggregation (CAG), so that only sensor nodes whose data are not similar transmit the sensor readings (Yoon and Shahabi, 2007). The threshold for data similarity is specified by the user. In CAG, a user query specifying the data similarity threshold is disseminated in the network. If the disseminated query contains a cluster-head sensor reading, then the sensor node receiving the query compares it with its reading. If its reading is within the specified threshold, the node joins the cluster otherwise it becomes a cluster-head. If the disseminated query does not contain a

cluster-head sensor reading the node receiving the query will become a cluster-head. Only cluster-head from each cluster transmits the sensed readings. CAG's aggregation function is thus lossy in nature. Sensor nodes update/repair the clusters at every cluster adjustment interval, as the sensor readings changes over the time and become inconsistent for the current cluster configuration. Cluster-heads transmit the data using a tree structure where the readings are aggregated according to the aggregation function specified in the query by the user at each level in the hierarchy (similar to TAG mentioned in Section 2.2.1.1).

#### 2.2.1.5 Discussion

Most of the in-network aggregation techniques require all the sensor nodes to transmit their sensor readings. These sensor readings are aggregated as they are routed through the WSN. The number of the transmitted sensor readings are minimised through aggregation functions as they are routed through the WSN. Some of these aggregation functions can be lossy and hence the original data can never be recovered, which could be problematic if the application user ever wanted the original sensor reading. CAG utilises data similarity-based clustering which is a lossy aggregation function to eliminate the need for all the sensor nodes to transmit their sensor readings. CAG rechecks the data similarity within its clusters at fixed intervals, if nodes within a cluster are not sensing similar readings between the fixed intervals there will be a loss in sensing fidelity. If the interval size is small higher energy is consumed in checking the validity of the clusters. These intervals introduce a tradeoff between accuracy and energy saving.

Another tradeoff that is general to all in-network aggregation techniques is between latency and minimisation of the number of the sensor reading packets. Latency occurs in these approaches because sensor nodes might have to wait to receive the sensor readings from other nodes before they can be aggregated. The major short-comings of in-network aggregation approaches are the need for latency and the possible loss of accuracy in aggregation functions. These short-comings can be eliminated by using a prediction

model to estimate the sensor reading of the sensor nodes. Approaches using prediction models are further discussed in Sections 2.2.4 and 2.3.2

### 2.2.2 Data compression

Similar to in-network aggregation techniques, data compression techniques also generally tradeoff communication for computational complexity. The computation operations can be further sub-divided into processing and memory access; compression algorithms that require a lot of memory access might actually increase the energy consumption instead of saving energy (Kimura and Latifi, 2005). It is thus crucial to select a data compression algorithm that requires only little memory access during compression. In this thesis data compression approaches in WSNs are classified as shown in Figure 2.3 and described in the following paragraphs.

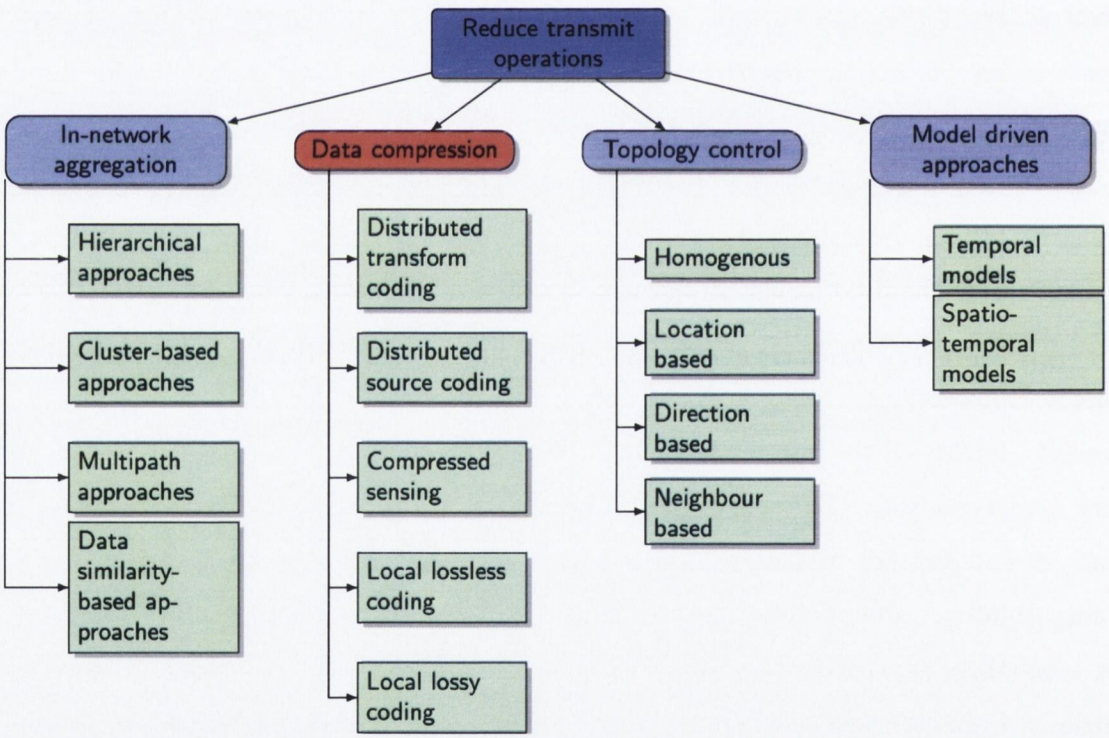



**Fig. 2.3**: Classification of data compression approaches

#### 2.2.2.1 Distributed transform coding

Well-known transform coding algorithms such as the Karhunen-Loeve transform (KLT) and the wavelet transform are applied for compressing the sensor readings in WSNs. These algorithms, consume high energy, when directly applied to WSNs. Thus several approaches in the literature either approximate or modify these classic transform-based algorithms to make these applicable in WSNs. One such approach is proposed by Amar et al. (2010b) where a local greedy version of the KLT algorithm presented by Amar et al. (2010a) is used to encode the sensor readings. These encodings are sent to the base station so reconstruction of the entire compressed sensor reading data can be done with minimal mean square error.

The wavelet transform algorithm is used by Ciancio and Ortega (2004) and is computed by exploiting the natural flow of sensor readings in a single hop network. In this algorithm first the sensor nodes are numbered. The odd-numbered sensor nodes receive samples from the neighbouring even-numbered sensor nodes to compute a prediction coefficient. These coefficients are then sent to the even number sensor nodes where it is smoothed and then sent to the central node so the original data can be reconstructed. This approach is extended to be suitable for multi-hop networks by Ciancio and Ortega (2005).

#### 2.2.2.2 Distributed source coding

Distributed source coding (DSC) approaches generally follow the Slepian-Wolf theorem (Slepian and Wolf, 1973). This theorem states that separate encoding of correlated sources is as efficient as joint encoding (a location where both the correlated information is available), for lossless compression. WSNs can thus use this theorem to effectively compress sensor readings that are correlated between sensor nodes without the need for explicit and energy-expensive inter-sensor communication. Chou et al. (2003) propose an approach where some sensor nodes will transmit their data directly to a base station while

compressing their readings with respect to their own previous readings. The remaining sensor nodes compress their data by encoding their data with respect to the other sensor nodes data that is transmitted to the base station. Sensor nodes that encode the data need to know the number of bits required to encode the data. The number of bits required to encode the data is generated in a code book based on the correlation structure of which the base station keeps track. This approach works well for simple network topologies and in networks where the decoders have no power constraints. The bigger the network, the more the memory is required to store the codebook. This leads to problems in sensor nodes with memory constraints. To solve this problem, an approach to using a smaller code book is proposed by Ramaswamy et al. (2010) by designing a bit-subset selector module in combination with distributed source coding at the base station. The module's role is to extract a suitable subset of the received bits for decoding per individual source. Using this approach requires a code book 16 times smaller than the conventional code book. The compression rate is directly related to the correlation of the sensor nodes and is not constant over time. Encoding the sensor readings with different bit rates based on the correlation structure will improve the compression; but if the correlation structure is not identified appropriately errors might occur during decoding. A low density parity check is used in combination with DSC approaches by Rezayi and Abolhassani (2009) to reduce bit error rate value in sensor readings with low correlation.

#### 2.2.2.3 Compressed sensing

When using approaches like distributed source coding based on the Slepian-Wolf theorem, knowledge of the precise correlation between sensor readings is necessary. To overcome this, other techniques make use of compressed sensing (Donoho, 2006). Compressed sensing does not rely on any prior knowledge or assumptions of the sensor readings. Quer et al. (2009) and Luo et al. (2009) use compressed sensing to gather data in a multi-hop WSN to improve energy efficiency.

#### 2.2.2.4 Local lossless coding

In this scheme, compression algorithms are run on a local node to reduce the size of the data sent by each of the sensor nodes. Unlike DSC approaches discussed in section 2.2.2.2 that use spatial correlation between the sensor nodes, temporal correlations of the sensor readings are utilised in lossless coding approaches. Dictionary-based compression is one of the lossless coding approaches used in WSNs. The dictionary-based compression algorithms involve high processing and high memory requirements, not suitable for WSNs. Sadler and Martonosi (2006) and Marcelloni and Vecchio (2009) approximated some of these dictionary-based algorithms and use then in WSNs.

#### 2.2.2.5 Local lossy coding

Lossy compression enables better compression than lossless compression. Data loss might be acceptable in certain WSN applications. For these applications, Marcelloni and Vecchio (2010) applied an adapted version of differential pulse code modulation for WSNs that exploits strong correlation between the sensor readings. Additionally de-noise techniques are used which allowed the loss of information in their algorithm to be only noise.

#### 2.2.2.6 Summary

Data compression approaches in general require more computation operations to compress the sensor readings. A balance between the computation operations versus the compression of the sensor readings achieved, is required to save energy. Some of the approaches such as the distributed source coding approaches require precise correlation information to prevent errors. This correlation information can change and an energy-efficient approach is required to track these changes. The distributed approaches to compression generally require specific assumptions or models of WSNs. In practice, these assumptions might not hold over time and thus these distributed approaches can degrade in performance (Srisooksai et al., 2012). In general sensor nodes that employ

audio/video sensing have larger data to transmit compared to sensors such as temperature and pollution. Data compression approaches might thus be more useful in saving energy of those sensor nodes that employ audio/video sensing.

### 2.2.3 Topology Control

The energy required by a transmission operation is proportional to the transmission distance. The transmission energy used thus determines the communication topology of the WSN. Controlling the topology of the network through adapting the transmission energy can save energy while maintaining the connectivity. Topology control approaches are classified based on (Santi, 2005). Figure 2.4 shows the classification and each of the approaches are described in the following paragraphs.

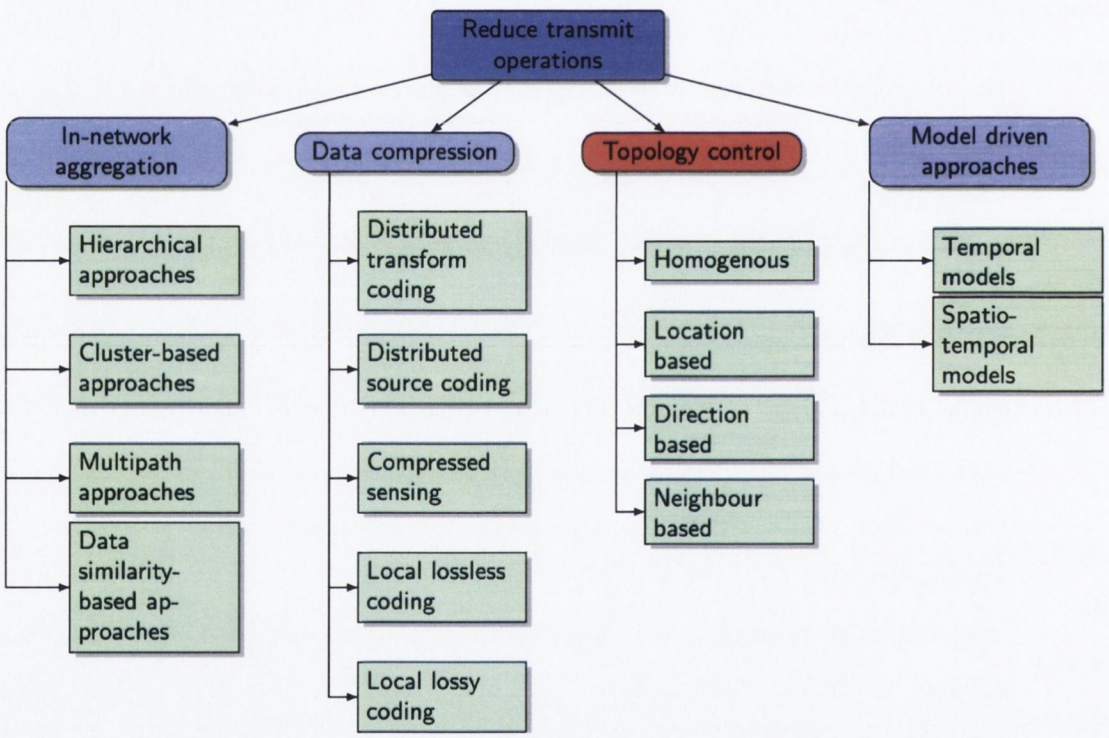



**Fig. 2.4**: Classification of topology control approaches

#### 2.2.3.1 Homogenous

Some transceivers might not allow the transmission distance to be adjusted dynamically. So the problem then comes down to finding an optimal homogenous transmission distance for all the sensor nodes that can maintain connectivity while saving energy. In the case where the physical placement of the sensor nodes is know in advance, then the transmitting distance is the longest distance between two nodes in a minimum spanning tree connecting the sensor nodes (Sánchez et al., 1999). In the case where the node placement is not known a priori, but is uniformly distributed in a dense network, Santi (2005) explains that for a three dimensional network the transmission distance for connectivity with high probability is $d = \sqrt[3]{\frac{logn - loglogn}{n\Pi} + \frac{3}{2} \cdot \frac{1.41 + g(n)}{\Pi n}}$, where $n$ is the number of nodes and $g(n)$ is an arbitrary function such that $\lim_{n \to \infty} g(n) = +\infty$. Given the assumption of dense networks, this result is not applicable for sparse networks. Combining the results of Santi and Blough (2002) and Santi and Blough (2003) proves that for $n$ nodes distributed uniformly at random then the transmission distance for connectivity with high probability is: $d = l\sqrt[3]{c\frac{logl}{n}}$, where $c$ is a constant greater than 0 and $l$ is the side length of the field of the WSN deployment. Instead of using a uniform distribution, Penrose (1999) proposes to identify the transmission distance, where the nodes are distributed according to an arbitrary probability distribution functions.

#### 2.2.3.2 Location-based

In location based approaches, the exact sensor node locations in the WSNs are known and the transmission distance can be adjusted dynamically. Using the location information, the central base station computes different transmission power levels required for each node so that the network connectivity between the sensor nodes is strong while keeping the energy cost minimum. Some of the approaches that use this technique are (Li and Hou, 2004; Li et al., 2003; Ramanathan and Rosales-Hain, 2000).

#### 2.2.3.3 Direction-based

Direction-based approaches are used in cases where the location of the sensor nodes might not be known, but an estimate of the direction of their neighbours might be known. Wattenhofer et al. (2001) propose a technique where a node $u$ transmits at a minimum power such that there is at least one neighbour at an angle $\rho$ from its centre. Wattenhofer et al. (2001) shows that a setting of $\rho \leq \frac{2\pi}{3}$ is a sufficient condition to maintain connectivity. Huang et al. (2002) use a similar method proposed by Wattenhofer et al. (2001) but using directional antennas. Borbash and Jennings (2002) propose a distributed approach to identify the minimum transmit power required by each sensor node in a WSN to maintain connectivity with other sensor nodes, when the locations of sensor nodes are not known, but an estimate of the direction of their neighbour might be known.

#### 2.2.3.4 Neighbour-based

Neighbour based topology control is based on the simple idea of connecting each node to its $k$-closest neighbours. Liu and Li (2002) propose to increase or decrease the transmission power based on the number of neighbours in a node's vicinity, until the number of neighbours is in the desired range. However, no guarantee on the connectivity is provided. While Xue and Kumar (2004) derive a condition that enables to maintain the connectivity with a high probability.

#### 2.2.3.5 Summary

The more information that is available about the node placement within the network, the better the connectivity between sensor nodes that can be maintained whilst saving energy. Most of the approaches mentioned assume either the presence or absence of network connectivity between the nodes. In reality connectivity might appear and disappear between the nodes. A probabilistic approach towards network connectivity is

likely to be more appropriate. If the WSN is planned a priori an optimal transmission range can be calculated during the design of the network, eliminating the need for the above approaches. These approaches are needed when the sensor nodes in the WSN are randomly distributed in a given area.

### 2.2.4 Model-driven sensor reading prediction

In these approaches, sensor readings of sensor nodes are predicted using a prediction model. The prediction models used can be classified as temporal or spatio-temporal models as shown in Figure 2.5. Both of these types of model can be used to reduce both the number of transmission and sensing operations. Approaches using models to reduce the number of sensing operations are discussed in Section 2.3. In this section, approaches that reduce the number of transmission operations are discussed. These approaches use a dual prediction approach to reduce transmission operations. In a dual prediction approach, the prediction model exists both in the base station and the sensor node. In this way sensor nodes can compare a sensed reading to the predicted reading and transmit the reading only if the predicted estimate of the reading is not within the accuracy requirements specified by the application or the user.

#### 2.2.4.1 Temporal models

Approaches using a temporal model can be further classified into algorithmic approaches, autoregressive models, trend-based models and use of multiple models. Each of these approaches are described in the following paragraphs.

**2.2.4.1.1 Algorithmic approaches:** One of the earliest approaches was PREMON, proposed by Goel and Imielinski (2001), for use in an event-based motion detection application. Event-based applications are not the focus of this thesis, nevertheless it is one of the earliest approaches that uses a dual prediction approach and therefore deserves a mention. PREMON represents the sensor nodes in the network as pixels in an MPEG

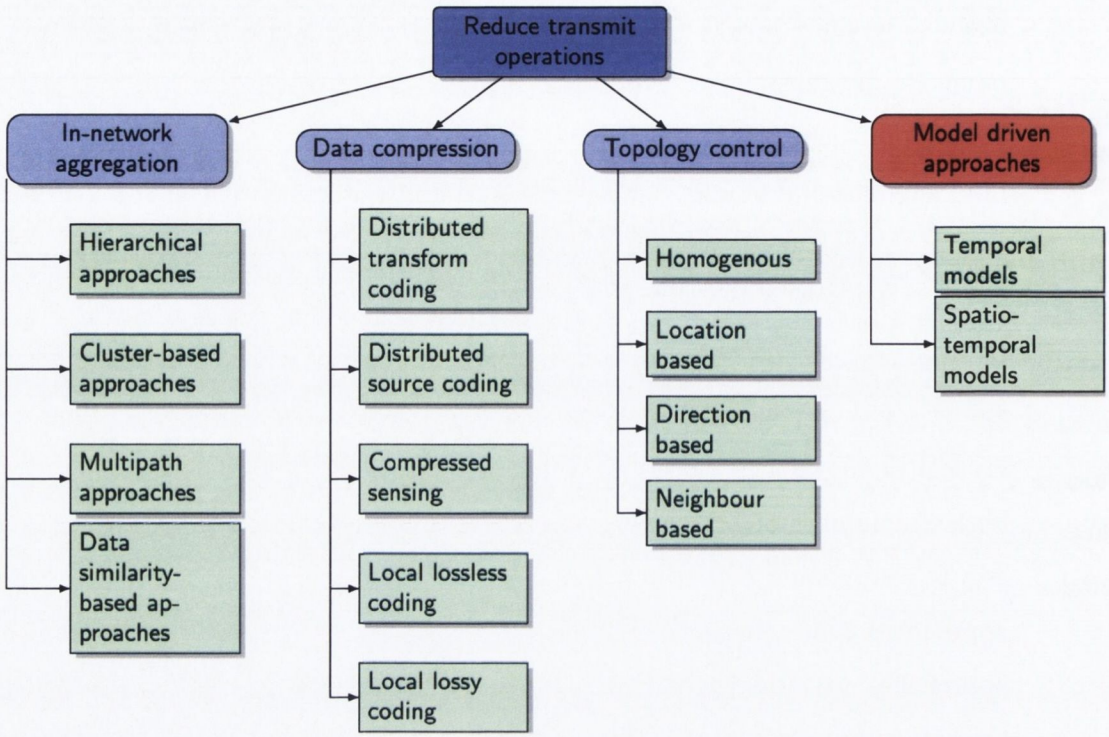



**Fig. 2.5**: Classification of model-driven sensor reading prediction approaches

video frame. The block-matching algorithm used in MPEG, is used as a prediction model to detect motion (Watkinson, 1999). The authors of PREMON assume that the locations of the sensor nodes are available. As a WSN does not necessarily have sensor nodes placed uniformly like the pixels in the MPEG, the block-matching algorithm cannot be applied directly. In order to correct this problem, they use the sensor readings to interpolate the readings at regular grid points. The sensors initially send their readings to the base station and the prediction model is computed and transmitted to the sensor nodes. Once the prediction model is received by the sensor nodes, the sensor nodes only transmit the sensor readings if the prediction estimates of its model is wrong. Another approach to track moving objects is proposed by Jain et al. (2004) using a generalised Kalman filter based prediction model. Kalman filters are versatile in being able to represent different processes. Jain et al. (2004) prove this by applying a Kalman filter to track a moving

object (event-based application), monitoring zonal electric load and monitoring HTTP traffic.

**2.2.4.1.2 Autoregressive models:** Different from PREMON, Tulone and Madden (2006b) proposed PAQ that trains a low-order autoregressive (AR) based prediction model at the sensor nodes instead of the sink. Once the AR model is trained its parameters are communicated to the sink enabling the sink to predict the sensor readings. A low-order AR model is used with the focus on reducing computation in the sensor nodes. PAQ eliminates the requirement for large training data to train the prediction model and instead proposes a strategy to retrain the model when the model does not predict the sensor readings within the accuracy requirements. When a reading is not predicted accurately, it is either marked as an outlier and communicated to the sink, or the retraining of the model is triggered. Retraining of the model is triggered depending on the proportion of poor predicitons within a time window. To prevent transmission of the model parameters from all the sensor nodes, PAQ further organises the sensor nodes into clusters based on data similarity and communication distance. Each cluster consists of a cluster-head that communicates its prediction model parameters to the sink, thus improving energy saving. The performance of PAQ was improved further in SAF (Tulone and Madden, 2006a). SAF introduces a linear trend component in conjunction with an AR model to improve the predictability of sensor readings in the presence of rapid changes in the environmental phenomena being sensed. SAF also does data similarity based clustering where only one node in the cluster needs to transmit the sensor reading (if its predictions estimates are inaccurate) as the rest of the sensor nodes sense similar values. The clustering is done centrally based on the predicted sensor readings of the prediction models learned for each sensor node. Since the clusters are based on the predicted sensor readings adapting to changes in clusters requires only checking the prediction estimates of the readings of each node within the cluster, making it quick to react to changes in the cluster without any overheads.

Unlike the PAQ and SAF approaches, where small training data is used to identify linear trends in the sensor readings, Li et al. (2009) propose an approach called PRESTO which tries to model both short-term and long-term relationships between sensor readings of a sensor node by using a large training dataset. PRESTO achieves that by using the seasonal autoregressive integrated moving average (SARIMA) prediction model. In the experiments conducted by Li et al. (2009), the order of the SARIMA model is chosen based on a series of tests performed on the auto-correlation and partial auto-correlation of the first two days of sensor readings obtained from a sensor node. The SARIMA model is also trained for the identified model order using the first two days of the sensor readings at the sink node and the model parameters are transmitted to the relevant sensor nodes. Retraining of the SARIMA model is done periodically. The major difference between retraining and initial training of the model is that for retraining, the predicted values of the readings are used in conjunction with any other transmitted sensor readings during prediction inaccuracies. In their experiment retraining is done at the end of each day.

**2.2.4.1.3 Trend-based models:** Derivative-Based prediction (DBP) proposed by Raza et al. (2012), is a recent approach that is similar to PAQ and SAF. DBP suggests that the trends of the sensor readings from a sensor node in short and medium time intervals can be approximated using a linear based prediction model. The DBP model is trained with an initial training data set. Using the first and last $l$ points of the training data, a linear trend is identified. The DBP model is retrained when the readings continuously deviate from the original value for some period of time.

**2.2.4.1.4 Multiple models:** All the above approaches assume that a single class of prediction model can give good prediction estimates of the sensor readings. The dual prediction scheme (DPS) proposed by Borgne et al. (2007) tries to find the best model within a class of temporal prediction models. The current best model to be used for

the prediction of sensor readings is identified based on the number of updates that the model has sent (predictions that are not within the accuracy requirements) in the past in conjunction with the number of parameters required to represent the model. This best model is identified by the node and communicated to the base station to be used at every sampling instant. Since holding multiple models in the sensor nodes can be memory intensive, the authors of DPS suggest to eliminate poorly performing models. To eliminate poorly performing models, the number of updates sent in the past by all the models is compared against the best performing model using Hoeffding bound (Hoeffding, 1963). If there is a considerable difference in the number of updates the poorly performing models are removed.

#### 2.2.4.2 Spatio-temporal models

So far, we have discussed approaches using temporal models. KEN, proposed by Chu et al. (2006), on the other hand proposes to use a spatio-temporal model. Using a spatio-temporal model in a dual prediction approach consumes more energy as checking the model's prediction against the sensed value requires sensor readings from other nodes to be brought to one single location. To mitigate this issue, KEN proposes a *disjoint clique* organisation of WSN's communication network. Based on the identified disjoint cliques, a prediction model is trained with all the sensor readings from the nodes in the clique. In this way, only the nodes part of the disjoint clique need to transmit their sensor readings to nodes where predictions are taking place. A multivariate normal distribution is used as the prediction model in KEN.

#### 2.2.4.3 Discussion of model-based approaches

All of the above dual prediction approaches reduce the number of sensor readings transmitted considerably whilst always guaranteeing the accuracy of the predicted sensor readings. This guarantee comes at the cost of the sensing operations. In these approaches, sensing needs to be on at every sampling instant. Thus these approaches are

not suitable for sensing operations whose energy consumption is not negligible and the approaches mentioned in section 2.3 should be used instead to conserve energy.

Temporal models are appropriate for smaller networks and in general when the spatial density of the data is low (Santini, 2009). In networks where the spatial density is high there might be neighbouring nodes transmitting redundant sensor readings and approaches employing spatio-temporal models might be more appropriate. The prediction models used in approaches using temporal models vary, but most of them employ a linear-based model (Tulone and Madden, 2006a,b; Raza et al., 2012; Santini, 2009). These linear-based model approaches have advantages over other approaches:

1. A small amount of training data is required, thus providing the flexibility to adapt to relationship changes of the sensor readings quickly

2. The model complexity is relatively low, and thus can be programmed on resource-constrained sensor nodes.

Different prediction models are used in all the approaches. Certain prediction models are likely to perform better for certain sensor reading relationships, but these sensor reading relationships might change over the lifetime of the sensor network. To address this issue the DPS (Borgne et al., 2007) approach uses a combination of prediction models to predict the sensor readings. The best possible model amongst a set of models is always chosen to be actively predicting the readings and communicating to the base station. One of the drawbacks of the DPS approach is the memory required to hold a set of prediction models. This issue is addressed by removing poorly performing models. This however is problematic if the model removed might be a good model at some future time.

### 2.2.5 Discussion of approaches reducing transmit operations

It should be noted that all the above-mentioned categories of approaches could possibly all be implemented together in a WSN, depending on the application requirements.

Applications requiring timely information might not want to employ data aggregation approaches as it might introduce latency. Latency is introduced in data aggregation approaches because readings need to be gathered and aggregated at various sensor nodes within the WSN. Similarly topology control approaches might not be applicable for applications requiring timely information as multi-hop transmissions are generally used to reduce the transmit power. Multi-hop transmissions might introduce latency as nodes might not be awake or available to receive the sensor readings. Data compression is more useful in applications that are data heavy, therefore simple monitoring applications using sensors such as temperature might not benefit hugely from data compression techniques. Model driven approaches may not be applicable to applications that require one hundred percent accuracy of information from the sensors as a small loss in accuracy is exchanged for energy savings. It should be noted that model-driven sensor reading prediction approaches are the only approaches where sensor readings from all the sensor nodes are not transmitted, with the exception of approaches that might be similar to CAG (Yoon and Shahabi, 2007). In all the other approaches mentioned, all the sensor nodes are actively transmitting their sensor readings either reducing the amount of data transmitted or the transmission distance to improve energy efficiency.

In the next section, we will discuss approaches that reduce the sensing operations.

## 2.3 Energy saving in sensing operations

Depending on the physical phenomena to be sensed by the WSN application, different sensors might be needed. These different sensors might consume significant energy because of the following factors (Anastasi et al., 2009):

1. Power-hungry transducers: Some sensors such as chemical and biological sensors intrinsically require high power resources to observe and sample the physical phenomena.

2. Power-hungry analog to digital converters: Sensors like acoustic and seismic transducers generally require high resolution analog to digital converters leading to significant energy consumption.

3. Active sensors: Sensors such as sonar, radar or laser sense the physical phenomena using active transducers.

4. Long acquisition time: Some sensors require acquisition time in the order of hundreds of milliseconds or even seconds. For example, humidity and temperature sensors require about a second per sample (Deshpande et al., 2004).

Thus whilst trying to save energy in sensor nodes the number of sensing operations might also need to be reduced. It should also be noted that by reducing the number of sensing operations, a reduction in both computing operations and the number of sensor readings to be transmitted are observed. There are several approaches in the literature that reduce the sensing operations for continuous long-term data collection applications in WSNs, which are classified and shown in Figure 2.6. Each of these approaches is discussed in the following paragraphs.

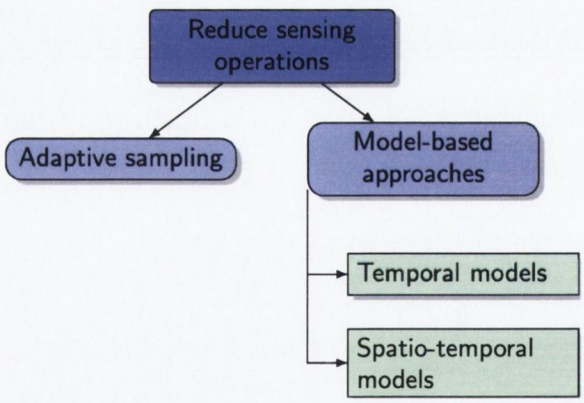



**Fig. 2.6**: Classification of reducing sensing operation approaches

### 2.3.1 Adaptive sampling

In the literature, sensor nodes have been used to adaptively sample the phenomenon according to some end user requirement. Usually, this requirement is a combination of being able to represent the physical phenomena with good accuracy whilst decreasing the energy requirement of activating the sensor nodes to sample.

Given a sensor network collecting environmental conditions in a local area, Willett et al. (2004) propose that sensor nodes need not sample the environment uniformly as there can be regions with high and low variations of the physical phenomenon in the sensor network. A hierarchical approach is taken where a WSN composed of $n$ sensor nodes is partitioned into squares and sub-squares until a resolution of $1/\sqrt{n}$ is obtained. Each square is a cluster and one of the sensors in each square serves as a cluster-head assimilating information from other sensors in the square. Since only one sensor node is needed to be be active in each square, the smaller the squares the more nodes are active in the WSN. Willett and Nowak (2003)'s approach is used to describe the phenomenon for each identified square. In an initial preview phase, some sensor nodes in the network sense the physical phenomena giving a coarse estimate of the phenomenon. A refinement phase then takes place if more sensor nodes are needed to sense the physical phenomena more accurately by making the squares smaller. Then, the smaller the size of the square, the higher the variation of the phenomenon.

Similar to Willett et al. (2004), Lin et al. (2008) propose region sampling where the sensor nodes do not need to monitor the environment uniformly. In region sampling, Lin et al. (2008) identify regions of sensor nodes that do not need to be active if the aggregation result as requested by the user can be estimated within some accuracy bounds. A WSN with $n$ sensor nodes is divided into $n$ regions, with at least one sensor node in each region. After collecting some initial data, regions are merged when their distance and their data variance are small. Each region then chooses a region-head that is responsible for computing the in-region aggregate result. The region heads incur a

greater energy cost and thus are randomly rotated to balance the energy across the WSN. When a user initiates a continuous aggregate query, each region head collects training data from the sensors in its region to compute the sample statistics of the training data and send it to the query node. The query node then computes the initial sampling plan, identifying the nodes required to sample and transmit sensor readings while the rest of the nodes are sleeping. The sampling plan is updated every time a new round of query execution is issued.

Alippi et al. (2007) propose to reconstruct the physical phenomena by identifying the minimum sampling frequency required, using the Nyquist sampling theorem. The frequency of the physical phenomena ($F_{max}$) needs to be identified to apply the Nyquist theorem. Once this frequency is identified, it can change over time and hence the sampling frequency needs to be adapted accordingly. $F_{max}$ is estimated initially by fast Fourier transform on the first $W$ samples. Alippi et al. (2007) establish two thresholds $F_{up}$ and $F_{down}$ based on the identified $F_{max}$ and keep identifying the current frequency $F_{current}$ as new samples are received. To asses a change in $F_{max}$ the CUSUM test proposed by Grigg et al. (2003) is used. Alippi et al. (2007) further define $th_{up} = \frac{F_{max}+F_{up}}{2}$ and $th_{down} = \frac{F_{max}-F_{down}}{2}$. A CUSUM rule for change is then defined as, $F_{current} > th_{up}$ or $F_{current} < th_{down}$ for $h$ consecutive samples, where $h$ is assumed to be known a-priori by the domain experts. Alippi et al. (2010) further improve this approach by identifying the value of $h$ dynamically by counting the maximum number of subsequent false positives that occur in the training data.

#### 2.3.1.1 Summary

Willett et al. (2004) and Lin et al. (2008)'s approaches reduce the sensor sampling rate, improving the energy efficiency of sensor nodes with a single sensor modality. These approaches are however not best suited for sensor nodes that have more than one sensor modality. This is because the regions/squares identified in these approaches appropriately are based on the phenomena sensed by the same sensors. Adding more

sensors will require identification of multiple regions/squares for each phenomenon being sensed which might not necessarily overlap with each other. Thus in the worst case scenario, different regions/squares for each phenomenon being sensed are identified such that all sensor nodes are required to be turned on. Alippi et al. (2007)'s approach on the other hand though not assessed with multiple sensing modalities could be extended to do so. This can be done by identifying different sampling frequencies per sensor using their proposed algorithm.

The approach proposed by Willett et al. (2004) always requires a minimum number of sensor nodes ($n^{\frac{3}{4}}$, where $n$ is the total number of nodes) to be active and extra sensor nodes are made active, if necessary, to identify sharply varying behaviours of the sensed phenomenon. Thus regardless of the behaviour of the phenomenon, a defined minimum amount of energy is expended. This energy could potentially be saved by having less nodes active depending on the behaviour of the phenomenon. In region sampling (Lin et al., 2008), energy is reduced only in queries that expect an aggregated reading from the sensor nodes. Region sampling is thus not applicable for applications that require raw sensor data. The approaches of Alippi et al. (2007) and Alippi et al. (2010) experience delay in adapting to the changes of the phenomenon which is proportional to the parameter $h$. Alippi et al. (2007) use priori information to identify the value of h and is user-defined, Alippi et al. (2010) extend this to identify the value of $h$ based on initial training data. This value of $h$ identified by Alippi et al. (2010) on the initial training data is used for the entire lifetime of the WSN, but in reality it might have to be changed as changes take place in the environment.

### 2.3.2 Model-based approaches

In model-based approaches, a prediction model is built that can represent the phenomena sensed. These approaches can be broadly classified based on the type of relationship modelled by the prediction models, as temporal or spatio-temporal models as shown in Figure 2.7. In the next few paragraphs these approaches are analysed.

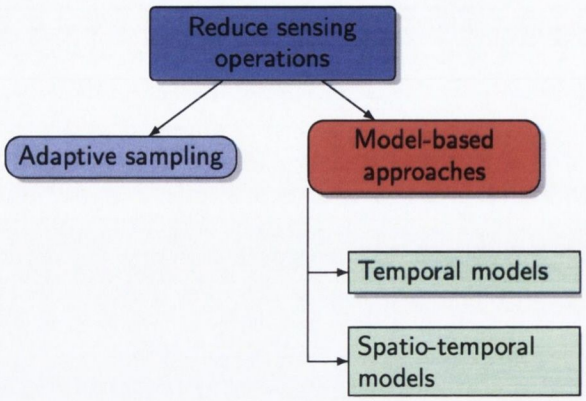



**Fig. 2.7**: Classification of model-based approaches

#### 2.3.2.1 Temporal models

These approaches can generally be classified further into approaches that 1) find the optimal sampling frequency to best represent the physical phenomena or 2) find how many samples can be skipped given the frequency of sampling required by the user.

**2.3.2.1.1 Identifying the sampling frequency** In these approaches, the sampling frequency is increased or decreased to sample the most important reading required to best represent the phenomena. The importance of a reading is described quantitively as the value of a reading. The value of a reading is calculated using different methods in different approaches. Padhy et al. (2006) use a linear regression model to model and predict estimates of future readings of the physical phenomena. This model is updated when a sensor reading is sampled from the sensor nodes. The smaller the confidence interval of the prediction model, the more accurate the predicted readings. The sensors are sampled at appropriate times to keep the confidence interval of the model within the user-specified bounds of the confidence interval. The value of a sensor reading sampled is then given as $V(reading(t_s)) = |ci(t_s - 1) - ci(t_s)|$, where $ci(t_s - 1)$ is the confidence interval before the data was sampled at time $t_s$ and $ci(t_s)$ is the confidence interval at time $t_s$ calculated using $reading(t_s)$. Padhy et al. (2006) suggest skipping future

samples when the sampled sensor reading is within the confidence interval bounds as specified by the user, otherwise sampling the sensor node at the possibly maximum sampling frequency. Similar to the previous work, Padhy et al. (2010) try to improve the value of the sensor readings sampled by using a bayesian linear regression as a prediction model trained with some initial sample of readings collected by the sensor node. Every time a new sensor readings is received, the model is updated and the value of the new information that the sensor reading provides is calculated using the Kullback-Leibler divergence measure (Kullback and Leibler, 1951). If the value of the prediction estimate of the future sensor reading as calculated by the model and the Kullback-Leibler divergence measure, is less than a threshold $V_{threshold}$ then the sample at that point in time is skipped, otherwise sampled. $V_{threshold}$ is the average value of all the readings used in creating the prediction model apart from the first two readings. The first two readings used in training the model are ignored because their value is very high and could set an unreasonably high threshold.

Kho et al. (2009) deploy a gaussian probability distribution function in each sensor node to model the physical phenomena. By defining a covariance function that describes the relationship between sensor readings at different times, the gaussian process can predict the physical phenomena. When the sensor readings are closely related, the predicted values will have low variance and high information value. This information value is derived using the Fisher information, which is a function inversely proportional to the variance of the predicted reading. Given a limited number of samples that a sensor node can sample, Kho et al. (2009) try to find the sampling points where the Fisher information is maximised. A naive approach of trying to find all the combinations of sample points to maximise the Fisher information is computationally very expensive. Kho et al. (2009) thus propose some alternative greedy algorithms. These greedy algorithms are still computationally expensive, Kho et al. (2009) propose to value information heuristically rather than computing the information value using the Fisher information. To do this they use a linear regression where the uncertainty is represented

using the confidence intervals. The total deviation is then derived by calculating the area between the confidence intervals for a set of data points. Total deviation represents the uncertainty over these data points. Using total deviation it is possible to tell whether one set of observations is more valid than the other. Given this total deviation, it is possible to derive the increase in information value when different sampling decisions are made. An action from a given set of actions is chosen to reduce the total deviation given some constraints such as the maximum number of possible samples using a binary integer programming.

**2.3.2.1.2 Skipping samples** In these approaches, the sampling frequency is predefined by the user/application and a prediction estimate of the sensor reading is calculated for each sample that is skipped. Chatterjea and Havinga (2008) propose to use ARMA (Auto Regressive Moving Average) as their prediction model. Each sensor node in the network trains an ARMA model. When the sensor is sampled, at a frequency predefined by the user, the predicted value from the ARMA model can be compared against the sensor reading sampled. If the predicted sensor reading is within a user-specified error threshold then an assumption is made by Chatterjea and Havinga (2008) that the prediction accuracy will continue to hold in the near future and skip the next sample. Each time the model predicts the sensor reading within the error threshold, the number of samples to be skipped increases until a specified maximum sample skipping limit. If inaccurate predictions of the sensor readings are identified, the sensors start to sample at regular intervals as specified by the user. Chatterjea and Havinga (2008) also propose to identify the maximum sample skipping limit based on the sensing coverage of the neighbouring sensor nodes. More the neighbouring nodes, the higher the maximum sample skipping limit. This is because there is less chance of missing an event when there are more neighbours in the vicinity. Law et al. (2009) provide some heuristic improvements to Chatterjea and Havinga (2008)'s work, by skipping samples if the predicted reading's confidence interval is less than twice the user-specified error. This could be an issue

when the sensor reading variations are high causing, more unnecessary sampling than required. This issue is rectified by increasing the tolerance for uncertainty by halving the user-specified error only, if the first predicted value's confidence limit is larger than twice the user-specified error. When the sensor node is not sampling spikes in the predicted readings might appear because of wrong predictions, to rectify this interpolation is used. The forecasted readings are overridden with values interpolated using the sensor reading immediately before the sample is skipped and after the sample is skipped.

**2.3.2.1.3 Summary of temporal model approaches** All the above approaches have an inherent absolute trust in the fact that the model can completely represent the physical phenomena. There needs to be a measure of how well the model can estimate the relationship of sensor readings and this needs to be taken into account when adaptively identifying to sample or to identify when a sample needs to be skipped.

At every possible sampling interval, Padhy et al. (2006) or Padhy et al. (2010) take a decision whether to sample or not, while Kho et al. (2009) identify a priori the best sampling points within a given energy budget. As a result the approach taken by Kho et al. (2009) is computationally more expensive. Padhy et al. (2006)'s approach requires choosing the level of confidence bound which might not be known a priori. Kho et al. (2009) eliminate this need when using gaussian probability distribution as the prediction model. Padhy et al. (2010) also improve over Padhy et al. (2006) by eliminating the need of choosing the confidence level.

Chatterjea and Havinga (2008) and Law et al. (2009) use some form of threshold that will take the decision to monitor or not. A sensor reading value very close to the threshold might pass the condition, leading to skipping samples, where in reality those skipped samples' readings are quite likely to contain prediction errors above the user-specified threshold. This highlights the rigidity of this approach. A finer grained approach is thus necessary that assesses the performance of the prediction model and performs the sampling accordingly. Chatterjea and Havinga (2008) and Law et al. (2009)

use a naive approach to skip sampling, where the number of samples to be skipped slowly increases by one and rapidly decreases to sampling at every possible instance when certain conditions identified by the approach are not met. Usually the trained models perform well at the start until the relationship of sensor readings changes slowly, deteriorating the performance of the prediction model. Thus a faster increase in skipping samples as soon as the model is trained could save more energy whilst keeping the sensing fidelity high.

#### 2.3.2.2 Spatio-temporal models

The BBQ approach proposed by Deshpande et al. (2004) uses a multivariate time-varying gaussian probability distribution function (PDF) as the prediction model. The PDF estimates the relationships between the various physical phenomena that the sensors in the WSN can send to the sink node. The time-varying evolution of the physical phenomena is modelled as a Markovian process. The PDF is used to answer user queries to the WSN, given an acceptable error tolerance with a corresponding confidence bound for the error. If the PDF is not able to answer the user queries within the confidence bounds of the requested error tolerance, an observation plan is created that queries sensor nodes and updates the model which can then be used answer queries answers with specified error tolerance and confidence intervals. The observation plan takes into account the cost of the sampling and communication to query the appropriate sensor nodes. For example, BBQ samples the voltage sensor in order to answer queries regarding temperature as it is cheaper to sample voltage sensors. Identifying the optimal observation plan has exponential complexity and hence the authors propose a polynomial-time heuristic which is effective to find practical solutions.

BBQ computes the PDF and the observation plan centrally whilst Guestrin et al. (2004) propose to compute a prediction model in a distributed fashion. A parametric approach is used to model the physical phenomena where a set of basis functions are given whose functions are solved using linear regression. Solving the basis functions in a

distributed fashion is expensive, hence the authors propose to solve these basis functions with partial network data. They propose to accomplish solving the basis functions by dividing the WSN into a number of overlapping regions, with all sensor nodes expected to lie in at least one of the regions. A kernel function maps each sensor node to a non-negative number that represents the degree to how much that particular node is associated with each of the region it lies in. Based on this additional kernel information the basis functions are solved with partial information.

ASAP (adaptive sampling approach) (Gedik et al., 2007) on the other hand demonstrate that a purely centralised approach, where all the correlations are identified and the predictions done at the sink node, or a localised approach, where the cluster-heads do the prediction and where correlations are identified by the sensor nodes, have high transmission costs. ASAP instead show that a hybrid hierarchical approach (correlations are identified by the cluster-heads and predictions are done at the sink) reduce the transmission costs compared to both the centralised and the localised approaches (Gedik et al., 2007). In ASAP, the WSN is first organised into clusters based on data similarity, with at least one sensor node selected as a cluster-head per cluster. All the nodes in a cluster are forced to sample frequently at a specified sampling period. Using data collected during the forced sampling period, each cluster-head builds a prediction model similar to the model used by Deshpande et al. (2004) and organises their clusters further into sub-clusters such that the sensor readings within the sub-clusters are highly correlated. The cluster-head then identifies a fraction of nodes in the sub-clusters to be samplers. Their sensor readings are used in conjunction with the prediction model to predict the sensor readings of the non-sampler nodes thus saving energy. Both clusters and sub-clusters are periodically recomputed.

Koushanfar et al. (2006) propose a non-parametric prediction model called the combinatorial isotonic regression (CIR) model. The authors show that the spatial correlation between the sensor reading of sensor nodes is not proportional to distance and use an isotonicity metric instead to capture the correlations effectively. Isotonicity is defined

as follows: *"If two sensor nodes $s_x$ and $s_y$ sense a physical phenomena at time $t_1$ and if the sensor reading value of $s_x$ at time $t_2$ increases then the sensor reading value of $s_y$ at time $t_2$ will also increase"*. This approach is split into a training, validating and model utilisation phases. In the training phase, the CIR model is trained with initial sensor readings and finds a mapping of values between the explanatory variable and the predictor variable and fits piecewise linear components at the mapping points. In the validation phase, more sensor readings are collected from all the nodes. Then a directed graph $G$ is created with all the nodes as vertices $v \in V$, and an edge $e \in E$ is created between node pairs $u, v \in V$, if a trained CIR model can predict estimates of all the sensor readings of $v$ using $u$'s readings collected in the validation phase, within the user-specified error bounds. In the model utilisation phase, dominating sets are identified in the graph $G$ which are scheduled to transmit sensor readings at every sampling interval while the rest of the nodes are sleeping. The trained CIR model is used to predict estimates of the sleeping nodes' sensor readings. Monitoring of the sleeping nodes are done at various times to ensure that the CIR model is predicting the sensor readings within the user-specified error bounds. If an error above the user-specified error threshold is observed during the monitoring, the CIR model starts collecting sensor readings from all the sensor nodes to train the model until a few sensor readings are simultaneously predicted correctly.

**2.3.2.2.1 Discussion of spatio-temporal model approaches** The above mentioned approaches enable a portion of sensor nodes to sleep within the WSN thus saving sensing, computation and transmission operations. BBQ uses lengthy training samples to train the prediction models. Guestrin et al. (2004)'s approach requires expert domain knowledge to map the regions and kernel functions to each sensor node in the region. Location information is also a prerequisite which might require additional energy for sensor node localisation protocols. ASAP requires a lot of communication to maintain the validity of the clusters, sub-clusters and the models identified. In the experiment

conducted by Gedik et al. (2007) using ASAP, clusters are checked periodically at every hour and sub-clusters every 15 minutes. Instead of using a fixed period for checking the validity of the clusters, the checking of clusters needs to be done more frequently when cluster changes are suspected to happen in order to improve sensing fidelity and save energy, otherwise the checking of clusters can be done less frequently to save energy. The sub-clustering only ensures high correlation between nodes that are close to each other, network-wide correlation which might be present is not exploited. Similar to the BBQ approach, the approach proposed by Koushanfar et al. (2006) requires lengthy training data. Koushanfar et al. (2006)'s approach fails to take advantage of the varying predictability of sensor readings between the sensor nodes at different times of the day. In this approach, monitoring an error above the user-specified error threshold triggers the collection of data from all nodes to retrain all models, making it energy expensive.

Changes can take place at any time in the physical phenomena and since the time of these changes are not known a priori, the model based approaches should use adaptable techniques to deal with these changes. Instead all the above model-based approaches use a non-adaptable approach to deal with changes taking place in the physical phenomena: BBQ requires domain experts to explicitly specify the time variation of the Markovian process, Guestrin et al. (2004) propose that different basis functions might be required depending on the time of the day, ASAP requires frequent checking at fixed periods to validate the clusters and the models and Koushanfar et al. (2006)'s approach monitor the sleeping nodes for errors based on the periods where high errors are observed in the validation phase. The monitoring pattern used by Koushanfar et al. (2006) might not be suitable to identify appropriate monitoring times in future days as the models might get worse in estimating the changing physical phenomena. All these issues point to a requirement for an approach that can adaptively monitor the performance of the prediction model. This can be done by estimating the performance of the prediction model in the future given the current and past knowledge of the prediction model's performance.

#### 2.3.2.3 Discussion of model-based approaches

Model-based approaches using either temporal or spatio-temporal models reduce the sensing operations. Reducing the sensing operations can result in a reduction in sensing fidelity. Most of these approaches use some form of monitoring of the idle nodes to prevent the reduction of sensing fidelity. This monitoring of idle nodes consumes energy and thus should be done intelligently to improve energy savings whilst keeping the sensing fidelity high.

One of the major differences between model-based approaches that do not reduce sensing operations (explained in section 2.2.4) to model-based approaches that reduce sensing operations (section 2.3.2) is that by not reducing sensing operations hard guarantees in accuracy can always be maintained. In order to do the same whilst reducing sensing operations efficient approaches for monitoring the performance of the prediction models used are required.

### 2.3.3 Discussion of approaches to reducing sensing operations

Both, adaptive sampling and model-based approaches, try to capture a representation of the physical phenomena under observation. Based on this representation, sets of sensor nodes are turned off saving energy. In model-based approaches, a prediction model is trained to represent the physical phenomena under observation and retraining of the model is usually done when the performance of the prediction model is low. This training and retraining used in the model-based approaches, makes it more generally applicable compared to adaptive sampling approaches because:

1. Willett et al. (2004) require the distribution of sensor nodes to be uniform.

2. Lin et al. (2008) reduce energy expenditure only for queries involving some form of aggregation.

3. Alippi et al. (2007) require the value of $h$ ($h$, is a counter that counts the number of times the CUSUM rule is not valid, different actions are taken based on its value.) to be set based on a priori information about the process and cannot be generalised. In (Alippi et al., 2010), an improvement is made by identifying $h$ dynamically based on the maximum number of false positives that can occur in the training data; this still does not guarantee the $h$ value will remain good in the future as changes happen in the environment.

Though model-based approaches are more generally applicable, they come with their own set of issues such as:

1. Appropriate models must be chosen and trained efficiently. Using long training data might be energy expensive especially when the trained model needs to be retrained as a result of changes happening in the environment.

2. Efficient techniques for monitoring the performance of the prediction model are required. Approaches used in the current literature either use always on sensing operations or an interval based approach bringing its own set of issues explained in the previous Section 2.3.2.2.1.

3. Predictability of the readings between the sensor nodes change, turning on/off the sensing operations in sensor nodes needs to be done appropriately to the changes taking place.

In the next section we will discuss approaches that reduce the receive operations.

## 2.4 Energy saving in receive operations

Sensor node transceivers need to be in receive mode to receive information from other sensor nodes or the base station. Energy consumption will be the highest when the sensor node's transceiver is always on, to be able to receive messages at anytime. In

some transceivers such as Chipcon (2013), where the receive mode requires more energy than the transmit mode, extra care must be taken. The node needs to be awake when it is ready to receive. This can either be achieved through a schedule, based on an asynchronous event or woken up on demand. These different approaches are classified in this thesis as shown in Figure 2.8.

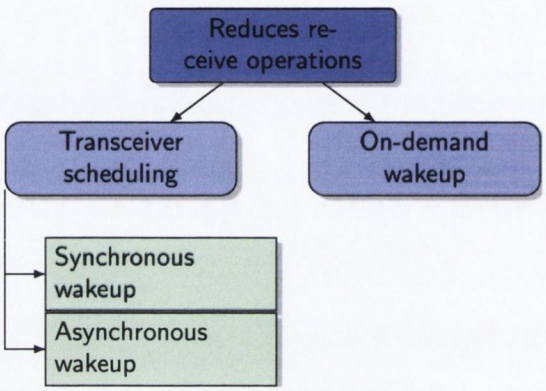



**Fig. 2.8**: Classification of model-based approaches

### 2.4.1 Transceiver scheduling

In these approaches, transceivers within the WSN are left turned off most of the time. Based on a schedule, the transceivers of these nodes are turned on for a short duration to receive messages. This schedule is called the duty cycle. Duty cycling can increase message delivery latency if the transceiver of the recipient node is off, when a message is to be sent to it. The goal of these approaches is thus to reduce the overall latency whilst also conserving energy. It should also be noted that most of the MAC protocols implemented for WSNs provide some sort of scheduling which will be discussed in the following sections. The scheduling approaches can be divided into synchronous and asynchronous wakeup approaches as shown in Figure 2.9.

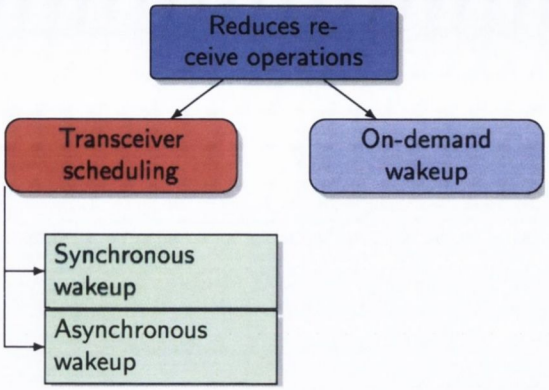



**Fig. 2.9**: Classification of model-based approaches

#### 2.4.1.1 Synchronous wakeup

In synchronous approaches, the scheduling decisions are made taking into account all nodes present in the WSN. The simplest synchronous wakeup schedule is the fully synchronised pattern (Keshavarzian et al., 2006), where the transceivers of all sensor nodes are switched on at the same time $t$ for an active time $t_{active}$. It is a very simple approach and is used in MAC protocols such as S-MAC (Ye et al., 2004) and T-MAC (Van Dam and Langendoen, 2003). The disadvantage of this simple approach is that a high number of wireless packet collisions might occur when many sensor nodes are trying to transmit messages at the same time. To solve this problem, some approaches take advantage of the networks hierarchical layers and propose staggered wakeup approaches (Lu et al., 2004; Cao et al., 2005; Li et al., 2005). In the staggered wakeup approach, the sensor nodes in adjacent levels wakeup with a time difference of $\tau$. This way not all the nodes are on at the same schedule, making wireless packet collisions less likely. A staggered wakeup approach is used in the DMAC protocol (Lu et al., 2004).

Time division multiple access (TDMA) based MAC protocols enable nodes to transmit/receive at dedicated time slots whilst sleeping the rest of the time, leading to collision-free data delivery and predictable data transfer delays (Suriyachai et al., 2012). This approach is adopted by Arisha et al. (2002); Chintalapudi and Venkatraman (2008);

Ergen and Varaiya (2006). Using a small active period in TDMA approaches has a high probability to improve energy saving but can result in high message delivery latency. To deal with this, an adaptive wakeup and active time scheme, based on the actual traffic needs, is proposed by Anastasi et al. (2006) and AI-LMAC (Chatterjea et al., 2004).

IEEE 802.15.4 is another standard MAC layer proposed by IEEE (2007) and specifies the physical layer and media access control for low-rate wireless personal area networks (PAN). In 802.15.4, the beacon-enabled mode saves energy by enabling a duty cycle. In the beacon-enabled energy saving mode, exists a super frame structure between beacons, and this super frame structure can be further divided into active and inactive periods. The active period is further divided into a contention access period (CAP) and a contention free period (CFP). During the CAP, a CSMA/CA protocol is used to contend for transmitting a message to that node. In the CFP, a number of guaranteed time slots can be reserved by individual nodes.

#### 2.4.1.2 Asynchronous wakeup

Asynchronous approaches allow each sensor node to switch the transceiver on independently of the other nodes with a guarantee that the sender and receiver node's transceiver are both on within a specified time (Anastasi et al., 2009). Zheng et al. (2003) develop an asynchronous wakeup mechanism by associating each node with a wakeup function that generates wakeup schedules for them. The wakeup function guarantees communication of each node to its neighbours in a finite time, while being resilient to packet collisions and variations in the network topology. Paruchuri et al. (2004) present a random asynchronous wakeup approach (RAW), leveraging the fact that most WSNs are dense deployments. In this approach, each node wakesup randomly once every $T$ and remains active for a predefined time of $T_a$ before going to sleep. Everytime the nodes are awake a neighbour discovery protocol is initiated to look for active neighbours. The probability that at least one of the node's neighbours is awake is given by $P = 1 - (1 - \frac{2.T_a}{T})^m$, where $m$ is the number of neighbours for the node.

There are approaches that try to predict when sensor nodes wakeup, so messages can be sent to those nodes without any synchronisation. These predictive wakeup approaches are discussed next.

**2.4.1.2.1 Predictive wakeup** During the transmission of data between a sender node and a receiver node, the optimal condition for energy saving in the receive operations is that the transceivers of both the sender and the receiver are on at the same time. Tang et al. (2011) achieve a near optimal condition for energy saving by using a pseudo-random wakeup schedule for the receivers, which the senders are able to predict. A pseudo-random wakeup is used to avoid wireless packet collisions as a result of the possibility of neighbouring nodes consistently waking up at the same time. An on-demand prediction error correction mechanism is also used in this approach, that takes into account unpredictable hardware and operating system delays to correctly predict the waking up of the receiver nodes.

### 2.4.1.3 Discussion of transceiver scheduling approaches

The synchronous wakeup approaches generally might require additional overhead to send sync messages and synchronise amongst each other to identify the appropriate schedule (Anastasi et al., 2009). Asynchronous approaches reduce this additional overhead of identifying a global schedule making it easier to implement (Anastasi et al., 2009). This reduction in overhead comes under the price of energy efficiency as the asynchronous approaches need to wakeup more frequently (Anastasi et al., 2009).

## 2.4.2 On-demand wakeup

These approaches notify the sensor node to switch on its transceiver only when it has to receive a packet from other sensor nodes or the base station. Usually these approaches use multiple low-energy radios to inform the nodes to switch on its transceiver. The specialised hardware used by most approaches is usually a receiver and in some ap-

proaches (Van Langevelde et al., 2009) a complete transceiver is used. Unlike traditional requirements of transceivers such as data rate and spectral efficiency, the specialised receiver's/transceiver's requirements are power efficiency at the node and the network level (Jelicic et al., 2012). In the case of approaches using extra receivers, these receivers monitor the communication channel continuously and wakes up all the neighbouring sensor nodes once a special message is received. If the message received is not intended for itself, the node goes back to sleep. To prevent waking up all the neighbouring nodes, some approaches use an extra hardware with the receiver that can decode the address in the message sent and wake the node up only if the message is intended for it (Shih et al., 2011; Hambeck et al., 2011). In the case where the recipient is not in the neighbourhood, then the neighbouring nodes need to wake up and use their main transceiver to transmit the message to the intended node. Van Langevelde et al. (2009) use an extra low power transmitter for these multi-hop scenarios.

#### 2.4.2.1 Discussion of on-demand wakeup approaches

The on-demand wakeup approaches offer really good mechanisms to reduce the energy in the receive operations whilst also minimising the latency of message delivery. The drawback with the approaches using specialised hardware is the need for extra hardware components in the sensor node and the possible difference in transmit or receive range from the main transceiver giving rise to a coverage mismatch (Anastasi et al., 2009).

### 2.4.3 Discussion of approaches reducing the receive operations

In the approaches that reduce the receive operations, the on-demand wakeup approaches achieve low latency and low energy expenditure. They do however require an additional radio component and there are no real world applications that have implemented this approach to date (Jelicic et al., 2012). The transceiver scheduling approaches are more widely applicable, as there is no requirement for an extra hardware.

In the next section we will discuss which combination of all the above mentioned

approaches improve energy efficiency the most and discuss some of the issues in model based approaches.

## 2.5 Overall analysis

Maximum energy savings in environmental monitoring applications using WSNs can be obtained by reducing both the transmit and the receive operations. Certain sensors might consume significant energy because of the factors identified in section 2.3. Thus depending on the type of sensors used, reducing the number of sensing operations is crucial in conserving energy of a WSN. For example, even simple sensors such as temperature and humidity sensors can consume more energy than the transmission and receive operations (Deshpande et al., 2004). This is because, Deshpande et al. (2004) assume that the senders and receivers are well synchronised, i.e., the receiving node switches on its transceiver exactly when the sender node transmits sensor readings. This is generally not the case and an appropriate approach that can synchronise the senders and receivers as mentioned in section 2.4 is needed. Currently, in this thesis the focus is on reducing the sensing operations and an assumption of good synchronisation between sender and receiver is assumed. It should also be noted that by reducing the sensing operations the transmission operations are reduced as well.

In reducing the sensing operations, model-based approaches provide more general solutions than their adaptive sampling counterparts as explained in section 2.3.3. The model-based approaches proposed in the literature can be further improved to save more energy and in improving sensing fidelity, if solutions to the following issues are identified:

### 2.5.1 Issues of model-based approaches

Some of the issues in model-based approaches are summarised below:

**Prediction models and their training** An appropriate representation of the physical phenomenon by the model is needed to obtain good prediction estimates of

sensor readings. As changes take place in the environment, the representation of the models might need to be changed and their parameters need to be identified by retraining the models. The training/retraining of these models require sensor readings, which in turn requires energy for sensing the phenomena and possibly transmitting these sensor readings. Thus the issues are: 1) to be able to find a good prediction model that can represent the current phenomena and 2) training/retraining these models need to be done by obtaining minimum number of sensor readings from sensor nodes to reduce energy expenditure whilst getting good prediction estimates of sensor readings.

**Tracking the performance of a prediction model** In approaches were the sensors are always on, such as the approaches mentioned in Section 2.2.4, changes in the performance of the prediction model can be identified by checking the predicted sensor reading against the real sensor reading. In model-based approaches that reduce sensing operations, sensing operations cannot be on all the time to track the performance of the prediction model continuously. Sampling the sensors at various intervals is one of the options to assess the performance of the prediction models. Frequent sampling of sensors helps to track the accuracy better but reduces energy efficiency. The issue is to find the correct frequency of sampling sensors and change this frequency as the performance of the model changes. It should be noted that there is a paradoxical issue as finding the most efficient frequency of sampling will require sampling of the sensor nodes.

**Changes in the predictability of sensor readings** Different sensor nodes' readings are predictable by different nodes' readings or by its past readings, depending on the behaviour of the phenomenon. As changes take place in the environment affecting the phenomenon, the predictability between readings of different nodes or itself can change. A mechanism that can react quickly to take advantage of the changes in predictability between readings to enable energy conservation and

maintaining the sensing fidelity as specified by the user/application, is needed.

The effect of the issues mentioned will be shown in relative terms on some of the relevant model-based approaches in the next section.

### 2.5.2 Relative comparison of model-based approaches

Based on the designs of some of the related approaches a relative comparison of them is presented in Figure 2.10, showing the relative number of sensing operations between the approaches and its corresponding sensing fidelity. The number of sensing operations and the sensing fidelity of the approaches are compared against each other based on the published results of experiments performed on the data collected by (Peter et al., 2004). An explanation of the relative placement of the approaches in Figure 2.10, is given below:

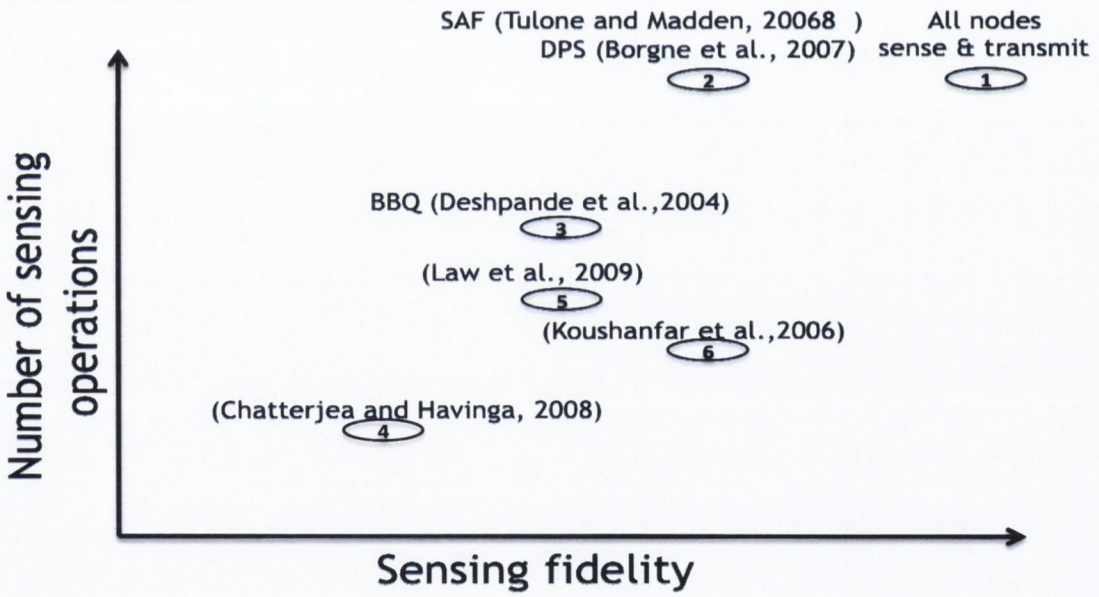



**Fig. 2.10**: Relative comparison of model-based approaches who have experimented on the Intel lab dataset (Peter et al., 2004)

1. When all nodes sense and transmit, there is no loss in sensing fidelity, assuming that the sensors are sensing frequently, accurately and not faulty. Since all nodes sense and transmit, there is no reduction in sensing operations.

2. Tulone and Madden (2006a) and Borgne et al. (2007) propose approaches wherein the sensing operation of the nodes are always on at every sampling frequency as specified by the user/application thus they are always able to check and maintain the accuracy of the predicted sensor reading estimates within some user-specified threshold. There is very little loss in the sensing fidelity as a result.

3. In the BBQ approach proposed by Deshpande et al. (2004), a very large training data set (6 days, for the intel lab dataset experiment and tested with following 2 days of data) is used. They also assume the changes taking place in the environment at a fixed interval of one hour. Given the large training set required the sensing operations are higher than approaches proposed by Chatterjea and Havinga (2008), Law et al. (2009) and Koushanfar et al. (2006). Hence it is placed relatively higher than the rest of the model-based approaches in terms of sensing operations. The error specification in BBQ requires specifying an error threshold and a confidence level for it. The confidence level specifies the limit for the total percentage of the occurrence of prediction errors above the error threshold. Given the extra requirement for this confidence limit and the occasional result where the percentage of occurrence of prediction errors exceeds the confidence limit, the sensing fidelity of BBQ is placed slightly to the left of (Tulone and Madden, 2006a) and (Borgne et al., 2007).

4. In the approach proposed by Chatterjea and Havinga (2008), the number of sensing operations to skip increases if it lies within an error threshold specified by the user. Thus, a sensor reading value very close to the threshold might pass the condition, leading to skipping samples, where in reality those skipped samples readings are quite likely to contain prediction errors above the user-specified threshold. Thus, in

the experiments carried out by Law et al. (2009) using the approach of Chatterjea and Havinga (2008), it shows that the prediction errors exceed the user-specified threshold. The experiment also shows that only about 20% of the sensing operations are used for a user-specified threshold of $0.3°C$, which is much lower than all the other approaches being compared against.

5. The approach proposed by Law et al. (2009), provides heuristic improvements over the approach proposed by Chatterjea and Havinga (2008). In the experiments carried out by Law et al. (2009), it shows that the root mean square of the prediction errors is lower than (Chatterjea and Havinga, 2008), but not all the prediction errors are within the user-specified bounds. The experiments also show that, the number of sensing operations used are higher than (Chatterjea and Havinga, 2008) and very close to (Koushanfar et al., 2006).

6. In the approach proposed by Koushanfar et al. (2006), monitoring the sleeping nodes for prediction errors is based on patterns of errors observed in the validation phase. These patterns hold throughout the lifetime of their experiment (10 days) but might not hold as the models get retrained and in the future might get worse in modelling the changing physical phenomenon. In their experiments it is shown that the prediction errors do not exceed the user-specified threshold and thus in Figure 2.10, this approach lies exactly below the approaches proposed by Tulone and Madden (2006a) and Borgne et al. (2007). Training, monitoring and collecting sensor readings from all nodes during retraining of the model increases the number of sensing operations and is approximately around the approach proposed by Law et al. (2009) and hence in Figure 2.10, these approaches lie close to each other.

From the above discussion, we can compare the differences between the approaches on six different criteria as shown in Table 2.1. These six different criteria are used in comparing the effectiveness of the approaches for the following reasons:

1. Reduces transmit operation : Reducing transmit operations can save energy (Raghunathan et al., 2002).

2. Reduces sensing operation : Reducing sensing operations can save energy (Raghunathan et al., 2006).

3. Small training/re-training data set : As explained in Section 2.5.1, training/retraining prediction models need to be done by obtaining minimum number of sensor readings from sensor nodes to reduce sensing and transmission operations whilst getting good prediction estimates of sensor readings.

4. Non-fixed periodic monitoring: Fixed periodic monitoring of the sensing fidelity is flawed because it can either decrease the sensing and transmission operations or improve sensing fidelity but not both. Thus, this criterion highlights approaches that use a different scheme of monitoring compared to the fixed periodic monitoring.

5. Detect changes in predictability : As explained in Section 2.5.1, reacting to changes in predictability is necessary to improve sensing fidelity and reducing sensing and transmission operations.

6. Error within user-specified bounds : This criterion shows the approaches that achieve the desired sensing fidelity.

Table 2.1 shows that our closest competitor is the work proposed by Koushanfar et al. (2006). This is because this is the only approach that always predicts sensor reading estimates within some user-specified threshold and manages to reduce both the sensing and the transmission operations. The proposed approach by Koushanfar et al. (2006) also does not perform fixed periodic monitoring but instead proposes to monitor based on the time, in which it is expected to observe errors above the user-specified threshold.

| Approaches/Criteria | Reduces transmit operation | Reduces sensing operation | Small training/re-training data set | Non-fixed periodic monitoring | Detect changes in pre-dictability | Error within user-specified bounds |
|---|---|---|---|---|---|---|
| Tulone and Madden (2006a) & Borgne et al. (2007) | ✓ | | ✓ | NA | NA | ✓ |
| Deshpande et al. (2004) | ✓ | ✓ | | | | |
| Law et al. (2009) | ✓ | ✓ | ✓ | ✓ | NA | |
| Koushanfar et al. (2006) | ✓ | ✓ | | ✓ | | ✓ |
| Chatterjea and Havinga (2008) | ✓ | ✓ | ✓ | ✓ | NA | |

**Table 2.1**: Comparison of approaches

## 2.6 Summary

In this chapter, the different approaches to reducing the number of transmit, sensing and/or the receive operations for energy saving in the WSNs were discussed. Most of the approaches to reducing the number of transmit operations can be applied independently on top of the approaches reducing the sensing or the receive operations. The appropriate approach for reducing transmission operations must be chosen based on the application requirements. Sensing operations can be energy consuming depending on the sensors used. Reducing the sensing operations also reduces the transmit operations. Amongst these approaches, model-based approaches provide more general solutions. These model-based approaches have some issues and needs to be fixed to improve the sensing fidelity and to reduce the number of sensing operations. The approaches reducing the receive operations are classified into transmitter scheduling and on-demand approaches. The on-demand approaches are promising as they might perform better than scheduling approaches. These on-demand approaches have not been used in real deployments and needs to be implemented for a step towards improving the reduction in receive operations. Currently, in this thesis, the focus is to reduce the sensing operations by using model-based approaches.

# Chapter 3

# The Less Battery Algorithm

In the previous chapter, the various approaches to reducing the number of transmission, reception and sensing operations to save energy in a WSN are discussed. In this thesis, the focus is to reduce the number of sensing and transmission operations in a WSN to potentially reduce its energy consumption. If an environmental monitoring application can tolerate a loss in accuracy, model-based approaches offer good general solutions to reduce the number of sensing operations whilst keeping the loss in sensing fidelity low. This thesis provides contributions to address the issues in the model-based approaches identified in the previous chapter, and is the focus of the remainder of the thesis.

The chapter begins with a summary of the issues in model-based approaches in reducing the number of sensing operations, then the design of the Less Battery (BLESS) algorithm, which addresses the issues identified, is proposed. An extension of the BLESS algorithm for sensor nodes containing multiple sensors is proposed followed by an explanation of a typical working scenario of the BLESS algorithm, then the implementation details of the BLESS algorithm are described. Finally, a summary of this chapter is presented.

## 3.1 Issues of model-based approaches

Model-based approaches can be further divided into approaches that reduce the number of transmit operations and approaches that reduce the number of sensing operations. In response to reducing the number of sensing operations, a reduction in the number of transmit and the number of computing operations is also achieved. Depending on the type of sensors used the energy consumed by the sensing operations might not be negligible due to the following factors:

- high energy consumption within transducers;
- high energy consumption within analog to digital converters;
- active sensors; and
- long acquisition time.

For all such sensors, model-based approaches that are limited to reducing the transmit operations are not as effective as model-based approaches reducing the sensing operations. This thesis proposes a model-based approach that can reduce the sensing operations and thus is particularly suited for sensors with the energy consumption characteristics as described above. Further reduction in sensing and transmission operations and achievable sensing fidelity can be observed in model-based approaches that reduce the number of sensing operations, when the following issues as identified in Chapter 2, can be resolved:

1. *Prediction model:* In model-based approaches, a prediction model is trained to represent the physical phenomenon. A good representation of the physical phenomenon needs to be identified by the prediction model. Changes can take place in the environment requiring the representation of the environment to be changed and therefore re-training of the model. Training and re-training requires sensor readings from all the sensor nodes involved in modelling the prediction models.

The training and re-training overheads need to be minimised to reduce the sensing and transmission operations.

2. *Performance of prediction model:* The performance of the prediction model cannot be checked when the sensing operations are off. Turning the sensing operations on frequently to check accuracy will increase energy consumption. A smarter energy-efficient alternative to waking up the sensor nodes to maintain the user-specified accuracy is needed.

3. *Changes in predictability:* Different sensor nodes' readings are predictable by different nodes depending on the state of the environment. As changes take place in the environment, the predictability of readings between the nodes changes. A mechanism that will re-train the models to adapt to these changes is required to maintain the user-specified accuracy and possibly improve the energy efficiency.

## 3.2 The BLESS algorithm design

The BLESS algorithm is designed to reduce sensing and transmission operations of WSNs used for environmental monitoring applications requiring continuous data collection. WSNs used in continuous environmental monitoring applications collect information about a phenomenon or a set of phenomena within an environment by collecting sensor readings from its sensor nodes. Thus, a WSN used in continuous environmental monitoring applications is structured such that there is one or more entities called the sink node or also referred to as the base station that collects and/or stores the sensor readings sensed by its sensor nodes. The sensor nodes in the WSN need to sense the phenomenon and transmit these sensed readings to the sink nodes. The sink node runs the BLESS algorithm, which uses a model-based approach to save energy in WSNs by instructing certain sensor nodes to go to sleep while predicting estimates of their sensor readings, there by reducing the number of sensing and transmission operations in the

WSN. The BLESS algorithm is designed to address the issues of model-based approaches identified in the previous section. The design of BLESS can be broken down into three main components: prediction model, node duty-cycling and identifying changes. First a brief overview of these components in the BLESS algorithm is given, followed by how these components interact with each other to reduce sensing and transmission operations and finally the design of each of these components are discussed.

### 3.2.1 BLESS overview

For the moment, let us assume that the sensor nodes are directly connected to a single sink node, i.e., a single-hop network to a sink node, each of these sensor nodes contains only a single sensor, capable of sensing only one phenomenon.

1. *Prediction model:* Prediction models are used in BLESS to predict estimates of the sensor readings within a user-specified error bound ($u_{error}$). A prediction model exists for each ordered pair of sensor nodes within the WSN. Thus, for a WSN with $n$ sensor nodes, there exists $n(n-1)$ number of prediction models. These prediction models are present in the sink node, and are used to predict estimates of the sensor readings of some sensor nodes, instead of turning the sensing and transmission operations on those nodes on, thus saving energy. For a prediction model associated with an ordered node pair $(a, b)$, where $a$ and $b$ represent sensor nodes in the WSN, prediction estimates of the sensor readings of node $b$ are obtained by using the sensor readings of node $a$. Assuming no changes happen in the environment affecting the performance of the prediction models, each of these prediction models undergoes three phases: training, validation and exploitation. In the training phase, $\alpha$ sensor readings from all the sensor nodes are collected to identify the parameters of the prediction models which determines the sensor reading relationships between the $n(n-1)$ ordered pair of sensor nodes in the WSN. In the validating phase, $\beta$ sensor readings from all the sensor nodes are collected to

validate all the trained prediction models. The prediction model is then said to be trusted, if the prediction error of all $\beta$ sensor readings are within the user-specified error bounds and the prediction model is expected to predict estimates of sensor readings within the user-specified error bounds in the near future. The exploitation phase of the prediction models starts after a node duty-cycle is identified.

2. *Node duty-cycling:* Based on the validation phase of prediction models, this component identifies the minimum number of nodes required to actively sense and transmit their sensor readings to the sink node in order to predict sensor readings of all other sensor nodes who are put to sleep. To do this, first a directed graph $G = \{V, E\}$ is created in the sink node, where each sensor node is represented by $v \in V$, and a directed edge $\{u, v\} \in E$ represents that the sensor readings of node $v$ are predictable using the readings from node $u$. Edges between an ordered pair of sensor nodes are added, if their associated prediction model is classified as trusted in the validation phase of the prediction model. Since $G$ represents which nodes' readings can be used to predict sensor readings of other nodes, it is referred to as the *predictability graph.* The minimum dominating sets (MDS) of $G$ are then identified at the sink node, which gives minimal sets of nodes required to predict the sensor readings of all the other sensor nodes in the WSN. The sensor nodes from one of the sets in MDS can then be used to estimate the sensor readings of all other nodes. Using the same MDS set of sensor nodes constantly to actively sense and transmit sensor readings will drain the energy of those nodes. In order to balance the energy expenditure across the WSN, each set of nodes in the MDS is duty-cycled by actively sensing and transmitting for a duration of $\theta$, while the rest of the sensor nodes in the WSN sleep. After this time $\theta$ another set of active nodes is used. Once the set of active nodes is identified, the exploitation phase of the prediction models begins, where prediction estimates of the sensor readings of the sleeping nodes are calculated using the sensor readings of the active nodes and

the associated prediction models.

3. *Identifying changes:* So far, an assumption has been made that no changes in the environment affecting the performance of the prediction models happen. In reality, this is not the case; changes continually happen within the environment of the WSN, that can change the relationships between the sensor readings modelled by the prediction models. This change in relationship can affect the predictability of the models: 1) a prediction model might not be able to predict estimates of certain sensor readings within the bounds of $u_{error}$ any more, thus compromising sensing fidelity and 2) predicting the estimates of certain sensor readings within the bounds of $u_{error}$ might become possible, which can be used to improve energy efficiency. In order to deal with these changes, first these changes need to be identified and then secondly appropriate action needs to be taken to conserve energy efficiency and maintain the required sensing fidelity. Assuming that changes happen sporadically, the exploitation phase of the prediction models can still be used to reduce sensing and transmission operations in the WSN. Since some of the sensor nodes are sleeping in the exploitation phase, changes happening cannot be detected unless the sensing and transmitting operations are switched on in these nodes. If the changes happening is known a priori, the sleeping sensor nodes can turn on the sensing and transmitting operations in those instances. Using these sensor readings, the prediction models can be re-trained and then exploited to reduce sensing and transmission operations in the WSN. In reality the changes happening are not known a priori and one way of checking for changes is to monitor the sleeping sensor nodes by turning on their sensing and transmitting operations. Frequent monitoring of the sleeping sensor nodes can identify changes quickly but also expends more energy compared to monitoring the sleeping nodes occasionally, which can potentially delay the identification of any change taking place. To overcome this issue, this thesis proposes to monitor the sleeping sensor nodes

more frequently when the respective performance of the prediction model used in predicting estimates of its sensor readings is low or expected to be low within the near future. The performance of a prediction model currently and in the near future is represented as $Trust$ in the prediction model. The value of $Trust$ can be between 1 and 0; 1 representing that the model is expected to predict estimates of the sensor readings within the user-specified error bounds and 0 representing that the model cannot be expected to predict estimates of the sensor readings within the user-specified error bounds. This value of $Trust$ is first estimated in each of the trained models by identifying a trend for the prediction errors using the $\beta$ readings collected in each of the models respective validation phase. Using this trend, it is possible to estimate if the prediction errors of the next $K$ prediction estimates of a prediction model are going to be within the bounds of $u_{error}$. If no prediction error above $u_{error}$ is expected for a model, then the corresponding sleeping node whose readings are predicted by the model is monitored at the $K_{th}$ reading, otherwise the trust is lowered and the monitoring of the sleeping node is done before the $K_{th}$ reading. Thus, clearly there exists a relationship between monitoring and trust which is given as: $t_{Monitor} = round(K \times Trust)$. The value of $K$ represents how frequently to monitor the sleeping nodes whose readings are predicted by a model with a $Trust$ value of 1. Every time the monitoring of the sleeping sensor nodes take place, the value of $Trust$ in the model used to predict its readings is updated. Once $(\alpha + \beta)$ sensor readings are monitored, they can be used to re-train the prediction model. Since monitoring happens more frequently when the value of the $Trust$ in a model is low, re-training of that model happens quickly to react to any possible changes happening in the environment. Then based on the value of $Trust$ in the re-trained model, it is possible to deal with changes in the predictability by appropriately deleting or adding an edge in $G$. Changes to the predictability graph $G$ will require re-identifying the MDS and the node duty-cycle appropriately.

In this section we introduced the three main components of BLESS: prediction model, node duty-cycling and identifying changes. In the next section we explain how these three components work in harmony to reduce sensing and transmission operations of sensor nodes.

### 3.2.2 Interaction between the three main components

The three main components of the BLESS algorithm: prediction model, node duty-cycling and identifying changes follows the following steps to reduce the sensing and transmission operation of sensor nodes:

1. Collect $(\alpha + \beta)$ number of sensor readings from all the sensor nodes in the WSN at every user-defined sampling interval at the sink (assuming only a single sink exists)

2. Train the prediction models between each ordered pair of sensor nodes with $\alpha$ number of sensor readings collected at the sink node. This way each ordered pair of sensor nodes can use their respective model to predict the sensor readings of other sensor nodes

3. Validate all the trained prediction models with $\beta$ number of sensor readings at the sink node. If the prediction errors are within the user-specified error bound $u_{error}$ an edge is added between the nodes represented by the prediction model in the predictability graph $G = \{V, E\}$, and the value of the trust in model is estimated. Else a trust value of 0 is given to the prediction model.

4. Identify representative node sets at the sink node by identifying the minimum dominating sets in $G$ at the sink node. Then identify a duty-cycle for the representative sensor node set by identifying a representative node set with the highest average energy as the set of active sensor nodes for a duration of $\theta$. Also the monitoring period $t_{Monitor}$ is calculated at the sink based on the trust for each of

the prediction models calculated. This duty-cycle and monitoring information is communicated to the sensor nodes in the WSN by the sink node so that certain nodes actively sense and transmit sensor readings while the rest of the nodes are idle and are only switched on at their respective $t_{Monitor}$

5. Exploit the prediction models to predict the sensor readings of the idle nodes

6. Certain idle sensor nodes turn themselves on to monitor and transmit the sensor readings to the sink based on the $t_{Monitor}$ communicated to it previously. After the monitoring the trust in the prediction models are updated at the sink node. As a result the $t_{Monitor}$ is updated at the sink and communicated to the appropriate sensor nodes

7. Re-training of the model between two sensor nodes is triggered once $(\alpha+\beta)$ readings are collected by the nodes. These readings are collected only when the nodes are active or in the monitoring phase. If the trust of the re-trained model is 1 and no edge exists between the sensor node pair that the model represents, a directed edge is added connecting the sensor nodes

8. Delete edges between sensor nodes if the trust identified forces monitoring of the idle node continuously at every successive sample for a period greater than $\tau$ readings. The value of $\tau$ is set to $(\alpha + \beta)$ in BLESS as re-training of the model can be done before deleting the edge.

9. Go to step 4 if any changes to $G$ takes place that is greater than $X\%$ or go to step 5 if it is the end of the duty-cycle.

These steps are also shown in Figure 3.1. In the above scenario, it is assumed that a single hop network with only one sensor type sensing the phenomenon per sensor node and transmitting these sensor readings to one central sink node. To extend the above scenario for multi-hop networks only Step 4 needs to be changed so that the

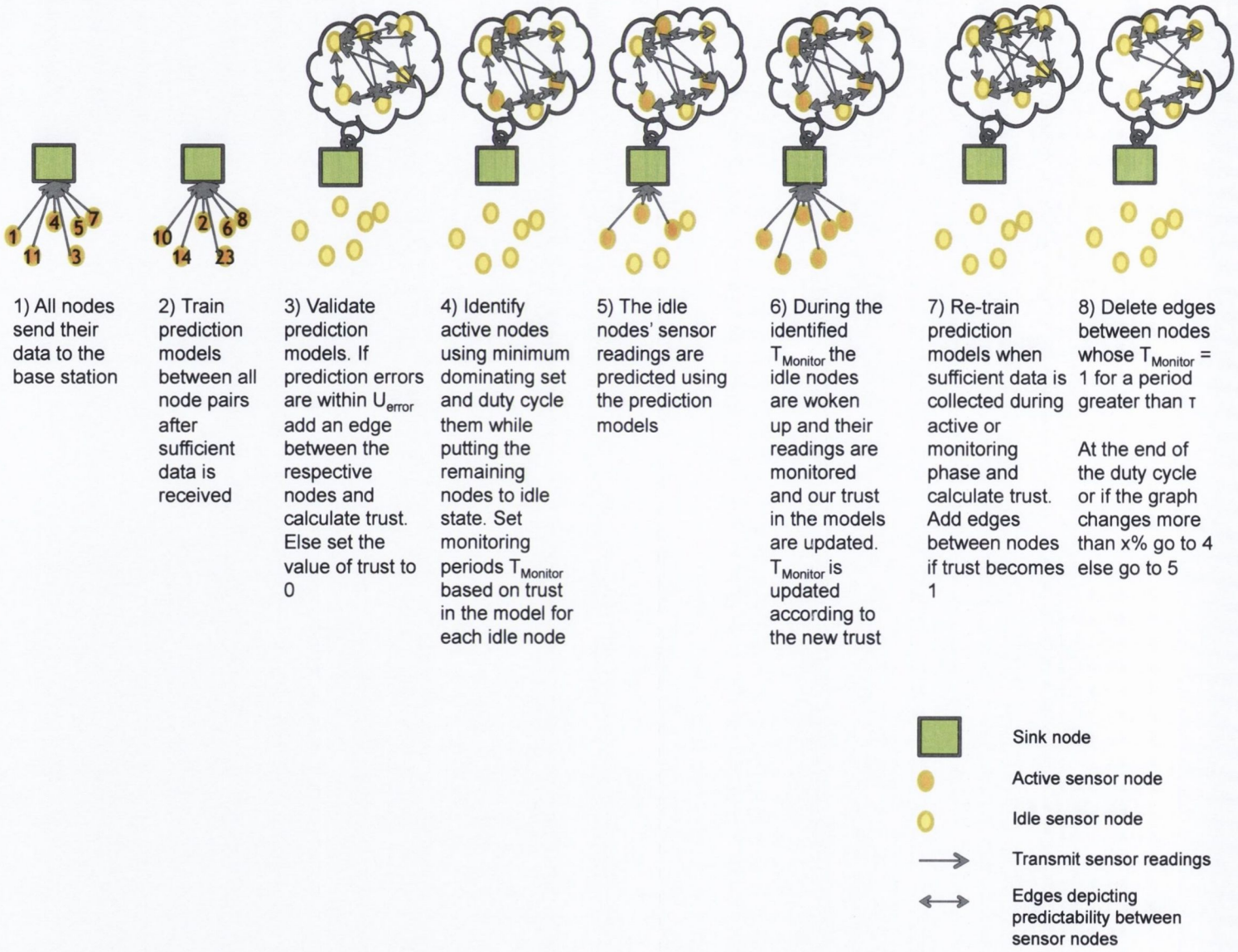



**Fig. 3.1**: Interaction between the three main components

representative node sets include the required intermediary nodes as explained in Section 3.2.4.3. To extend the above scenario with sensor nodes containing multiple sensor types per node whose energy requirements are much higher than the transmission of sensor readings, Step 4 needs to be replaced with the solution presented in Section 3.2.7. If multiple sinks exist then we assume that each of these sink nodes have indefinite energy. These sink nodes can thus communicate all the received sensor readings to one of the sink nodes where all the mentioned steps take place.

So far an overview and the interactions of the three main components in BLESS: prediction model, node duty-cycling and identifying changes components to reduce the sensing and transmission operations are discussed. The next sections details the design choices of these components.

### 3.2.3 Prediction model

In WSNs, prediction models are used to predict estimates of the sensor readings of some sensor nodes, reducing the loss of sensing fidelity while those sensor nodes are sleeping. The different design decisions needed to identify an appropriate prediction model are as follows:

#### 3.2.3.1 Sensor reading relationship

As explained in Section 1.3, prediction models identify relationships between sensor readings over time, or over space and time, and are referred to as temporal models or spatio-temporal models respectively. Temporal models are appropriate for smaller networks where the spatial density of the data is low (Santini, 2009). For larger and denser networks, spatio-temporal models might be more appropriate. In this case, a sensor reading sensed at a node is used to predict sensor readings of other nodes at the same time. Sensor readings of multiple sensor nodes could be used to predict the sensor readings of one or more other nodes. Using sensor readings of multiple nodes expends more energy than using readings from a single sensor node as more sensor readings

are needed to predict the readings of the other nodes increasing the number of sensing operations of the sensor nodes. For these reasons BLESS uses spatio-temporal models to predict estimates of sensor readings of one or more nodes using sensor reading from one node. This requires a prediction model to represent the relationship of each ordered pair of sensor nodes within the WSN. For example, given an ordered node pair $(a, b)$, where $a$ and $b$ represent sensor nodes in the WSN, prediction estimates of the sensor readings of node $b$ can be obtained by using the sensor reading of node $a$ and the trained prediction model for the ordered node pair $(a, b)$. Thus for a WSN with $n$ sensor nodes, there exists $(n^2 - n)$ number of prediction models. This $(n^2 - n)$ number of prediction models are present in the sink node where the sensor readings are either predicted or collected. In the next section the choice of the type of prediction model to be used is discussed.

#### 3.2.3.2 Complex or simple prediction model?

In this thesis, a complex prediction model is defined as a model that tries to capture and model all the possible relationships and the relationship changes between sensor readings of nodes. Usually, a complex model requires a large training data set to try and capture all the possible relationships and the relationship changes between sensor readings. Changes taking place in the environment not captured by the model might render the relationships identified by the prediction model useless. Invalid models need to be re-trained to maintain the sensing fidelity of the WSN. A large training data set might also be required for re-training a complex prediction model. Approaches proposed by Guestrin et al. (2004) and Koushanfar et al. (2006) use complex prediction models. Conversely to complex prediction models, in this thesis simple prediction models is defined as models that do not try to capture all the possible relationships and relationship changes between sensor readings. Simple prediction models instead capture relationships that might not last long and require relatively small training data sets for training the model compared to complex models. It appears that complex prediction models

might perform prediction of the sensor readings better than simpler prediction models. Our hypothesis however is, that simple models when trained with a small dataset and re-trained when possible would perform better in terms of prediction accuracy and reduction in sensing and transmission operations (see section 4.12 for the evaluation). This is because a simple model is able to capture relationships lasting for a short time with only small overhead required for training/re-training. Approaches proposed by Tulone and Madden (2006b,a); Raza et al. (2012); Borgne et al. (2007) use simple models, reducing only the transmit operations. Since the sensing operations are always on in these approaches, it is feasible to check the predicted reading to identify outliers or changes in the phenomenon that requires re-training of the prediction model. Similar to these approaches a prediction model needs to be chosen such that it is a good representation of the physical phenomenon and consumes less overheads when training/re-training. In the next section choosing such a simple prediction model is discussed.

#### 3.2.3.3 Simple prediction models

As explained in Section 1.3, prediction models can be further divided into parametric and non-parametric models. Non-parametric models as used by Koushanfar et al. (2006), requires large training data hence classified as complex models. Parametric models on the other hand need to be fully described using a finite set of parameters. Given the changes taking place in the environment, these set of parameters can keep changing. To simplify choosing the set of parameters, a model describing a linear function is used. The motivation behind this, is that a segment of a continuous function can be approximated to a line, and as the size of the segment approaches zero the approximation error tends to zero (see Appendix A, for further details). Thus, given a small enough time window, the relationship between sensor readings of an ordered pair of nodes can be approximated using a linear function. Simple linear regression (SLR) models can be effectively used to represent linear relationships (Mullins, 2003). An SLR model of order one as shown in Equation 3.1 is used as the prediction model, to identify model parameters Equation

3.2 is used and in order to predict an estimate of sensor reading, Equation 3.3 is used.

$$y_t = A + Bx_t + \epsilon_t \tag{3.1}$$

$$\begin{bmatrix} \hat{A} \\ \hat{B} \end{bmatrix} = \arg\min(\sum_{t=0}^{n}(y_t - \hat{y}_t)^2) \tag{3.2}$$

$$\hat{y}_t = \hat{A} + \hat{B}x_t \tag{3.3}$$

where $A, B$ are parameters of the SLR model that are estimated by applying ordinary least squares (OLS) on the training data of the sensor nodes X and Y. In principle, OLS involves calculating the vertical distances from the observed data points, $y_t$, to the corresponding points, $\hat{y}_t$, and finding the sum of the squares of these distances. Then the values of $\hat{A}, \hat{B}$ that minimises this sum are the parameter values of the SLR model as shown in Equation 3.2 (Mullins, 2003). Using the calculated value of $\hat{A}$ and $\hat{B}$, a prediction estimate $\hat{y}_t$ of the predicted sensor reading for the sensor node Y at time $t$ can be calculated applying Equation 3.3, $x_t$ is the sensor reading observed by the sensor node X at time t, $\epsilon_t$ is the model error. The residual/prediction error $e_t$ at time $t$ of the prediction estimate can then be defined as in Equation 3.4.

$$e_t = y_t - (\hat{A} + \hat{B}x_t) \tag{3.4}$$

To use the SLR model to predict the sensor readings of node $Y$, the model needs to be trained. The training data from sensor nodes X and Y must be time synchronised to identify the appropriate relationships between the sensor readings of these sensor nodes. In time-varying data a relationship between the residual/prediction error terms $E_t = \{e_0 \ldots e_{CurrentTime}\}$ exists (Chib and Greenberg, 1994). Since WSNs' sensor readings are time-varying, there exists a relationship over time between $e_t$ values. This property is taken advantage of to identify the trust in the model explained further in Section 3.2.5.1.

The relationship between sensor readings of an ordered pair of nodes can be approximated to a linear function for a time window $Tw$. The SLR model is then trained with a small proportion of the readings occurring during the time window $Tw$ and is exploited to predict the sensor readings for the remaining time of $Tw$. The $Tw$ value is not a constant and it varies as changes take place in the environment under observation. Identifying this value is important to maintaining the sensing fidelity and will be discussed further in Section 3.2.5.2. For now we assume that $Tw$ is long enough for us to train and exploit the SLR model.

The relationship between sensor readings of a certain ordered pair of sensor nodes might have a weak or no relationships and thus their readings can not be predicted using the prediction model and given the sensor readings of one of the nodes within $u_{error}$. Approaches such as PAQ (Tulone and Madden, 2006b), SAF (Tulone and Madden, 2006a), DBP (Raza et al., 2012) and AMS (Borgne et al., 2007) have their sensing operations on all the time and can recognise weak or absent relationships. Since the sensing operations of some nodes are not going to be on during the exploitation phase of the prediction model, some form of pre-validation becomes a prerequisite.

The validation of the SLR model is done before exploitation, by collecting $\beta$ number of sensor readings. During the validation phase, the prediction errors are checked if they are greater than the user-specified error and then calculates trust in the model as explained in Section 3.2.5.1. If there are no errors greater than the user-specified error threshold, i.e., $\forall e_{validation} \in E_{validation} : |e_{validation}| \leq u_{error}$ and the model is trusted then the exploitation phase can be started. Figure 3.2 shows the training, validation and exploitation phases of a prediction model within the time window $Tw$. Based on the validation phase of all the prediction models, the minimum number of nodes required to actively sense and transmit its sensor readings to the sink node in order to predict sensor readings of all the other sensor nodes is identified

In the next section, we describe the method in identifying and duty-cycling the minimum number of nodes required to actively sense and transmit its sensor readings to

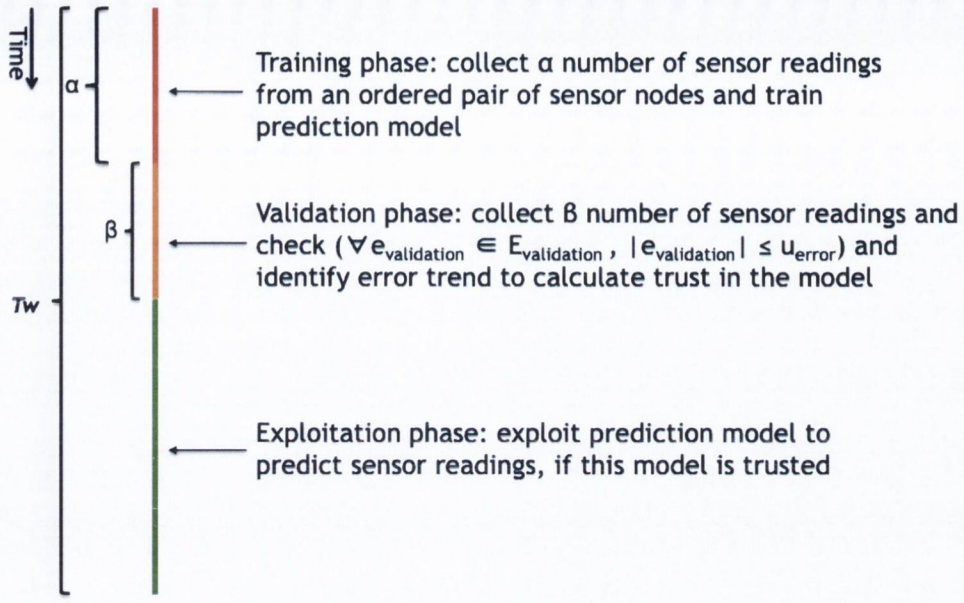



**Fig. 3.2**: Different phases of a prediction model within $Tw$ for an ordered sensor node pair

the sink node in order to predict sensor readings of all the other sensor nodes who are put to sleep, conserving energy.

### 3.2.4 Node duty-cycling

After training and validation of the prediction model, it is possible to tell which sensor nodes' readings are predictable within the user-specified error threshold, given the sensor readings of some other nodes. The performance of the prediction models for each ordered node pair is termed predictability in this thesis. Based on this predictability, subsets of sensor nodes needs to be identified, such that given only the sensor readings from sensor nodes in one of the subsets, the sensor readings of all the other nodes in the WSN can be accurately predicted. These subsets are referred to as representative node sets. Reduction in sensing and transmission operation is then achieved by having one representative node set actively sense and transmit the sensor readings while the rest of

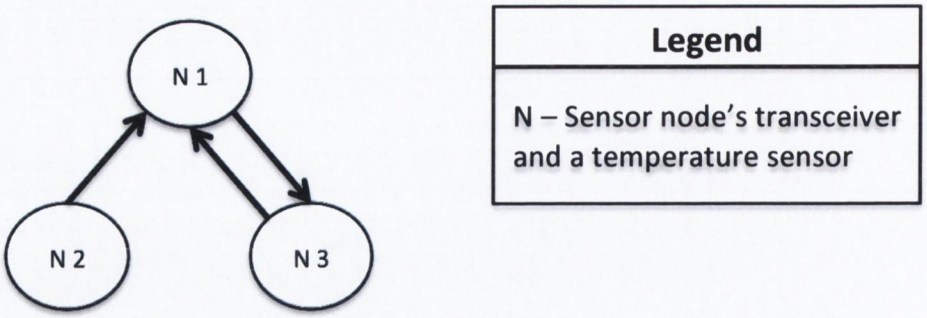



**Fig. 3.3**: Predictability graph with three sensor nodes sensing temperature

the nodes are sleeping. These sensor nodes are referred to as active nodes. The active nodes consume energy to sense and transmit: the fewer the active nodes, the higher the reduction in sensing and transmission operations are. Using the same set of representative nodes to be active will thus drain energy of these nodes. These representative nodes need to be duty-cycled to balance energy expenditure across the WSN. In the next sections we first discuss how to identify the representative sensor nodes followed by how active nodes are duty-cycled to balance energy across the WSN.

#### 3.2.4.1 Identifying representative sensor nodes

To find the sets of representative nodes, the WSN is represented as a directed graph $G = \{V, E\}$ where each sensor node is represented by $v \in V$ and an edge $\{u, v\} \in E$ represents the fact that the sensor readings of node $v$ are predictable using the readings from node $u$. In general, the prediction accuracy is not symmetric and hence directed edges are used instead of undirected edges. Since the graph $G$ represents the predictability between the sensor readings of nodes, it is referred to as the predictability graph in the rest of the thesis. For example Figure 3.3, shows a predictability graph with three sensor nodes sensing temperature in a WSN. As the predictability of the models for an ordered sensor node pair starts varying, the structure of the graph G will start to change. In this section we assume that the structure of the graph is a constant and explain how to identify the

subset of representative nodes. Identifying the variations in the graph $G$'s structure and dealing with these variations will be discussed later in Section 3.2.5.3.

As mentioned earlier, the fewer the number of nodes in the representative sets, the higher the reduction in sensing and transmission operations are. Therefore, representative node sets containing the minimum number of nodes that directly connect all the other nodes in the WSN through an edge $e \in E$ need to be identified. The concept of dominating set used in graph theory can be used to identify such representative node sets. A dominating set is a set in which every vertex not in the set is connected to at least one vertex in the set (Gibbons, 1985). Then a minimum dominating set is a dominating set from which no vertex can be removed without destroying the dominance property (Deo, 2004). Thus, the problem of finding representative node sets narrows down to finding the minimum dominating sets (MDSs) in graph G. Finding minimum dominating sets is a well-known NP-complete problem. Though it is an NP-complete problem, it is still possible to find MDS in a reasonable time with a worst-case estimate of $O(3^{n/3})$, where n is the number of nodes. In order to find MDSs within a worst-case estimate of $O(3^{n/3})$, the problem is first reduced into a clique problem and solved by using the Bron-Kerbosch algorithm developed by (Samudrala and Moult, 1998). Sensor nodes of one of the representative node sets identified need to be active at any given time to predict an estimate of the sensor readings of the rest of the sensor nodes. In the next section, an approach for choosing these active nodes from the identified representative node sets such that energy is balanced across the WSN is discussed.

#### 3.2.4.2 Balancing energy

Having the sensor nodes of the same representative node set active all the time increases the energy usage of these nodes. These sensor nodes might thus die sooner than the rest of the WSN, which might lead to a reduction in sensing fidelity and might also cause network partition. To prevent this from happening, the representative sensor node sets need to be duty-cycled, i.e., sensor nodes of different representative node sets

need to be activated periodically. To do this an an energy-aware duty-cycling mechanism is proposed. In this mechanism, the residual energy level information from the sensor nodes is piggy backed during the transmission of sensor readings. In this way the residual energy levels of the sensor nodes are known at the sink node and appropriate action can be taken by the sink node based on this information. The representative node set with the highest average energy compared to the other representative node sets transmits sensor readings to the sink node for a cycle of duration $\theta$. After every $\theta$, the sensor nodes from another representative node set with the highest average energy among its nodes is made active.

In the previous sections, we have assumed a single hop network. In the next section strategies to work with multi-hop networks are explained.

#### 3.2.4.3 Multi-hop network

The active sensor nodes chosen from the identified representative node set in the WSN are actively transmitting sensor readings to the base station. If these sensor nodes are not in the one hop range of the base station, the assistance of intermediary nodes is required. These intermediary nodes, bridge the gap between representative sensor nodes and the base station. This is done by receiving the sensor readings from nodes not in the one hop vicinity of the base station and transmitting them to the base station or another intermediary node. There might be many such intermediary nodes present in the WSN. Usually a routing algorithm identifies which intermediary nodes to choose and specifies a route from a sensor node to the base station. An assumption is made, that the routing algorithm used in the WSN provides the sink node with all the routes that each node needs to take to deliver messages to the sink node. Based on these routes the intermediary nodes required for the representative node sets are identified at the sink node. The sink node can then add the appropriate intermediary nodes to the representative node set. Thus the energy levels of the intermediary nodes are also considered when a representative node set is chosen to be active.

### 3.2.5 Identifying changes

As the environment changes, the following issues have to be dealt with to maintain sensing fidelity and to reduce sensing and transmission operations:

- Detect the performance of prediction models
- Detect variations in predictability
- Re-identify and schedule sets of representative nodes accordingly

BLESS proposes a solution to deal with these issues in an energy efficient manner by assessing the trust in the model . The next section describes how trust is identified and the rest of the section describes how this identified trust can deal with the issues above in an energy-efficient manner.

#### 3.2.5.1 Trust in the prediction model

The performance of the prediction model is defined to be good if the prediction estimate is within the user-specified error threshold. Then in BLESS, the trust in the prediction model is an expectation of the performance of the prediction model. The value of $Trust$ can be between 1 and 0. The closer the value of trust is to 1, the higher the expectation that the performance of the model is good, i.e., a trust value of 1 represents that the model is expected to predict estimates of the sensor readings within the user-specified error bounds and a value of 0 represents that the model cannot be expected to predict estimates of the sensor readings within the user-specified error bounds. The value of trust is estimated based on current and past knowledge of the performance of the prediction model. Based on the value of trust estimated, actions can be taken that can potentially reduce energy usage and improve accuracy, as follows:

1. *Monitoring sleeping nodes:* As time progresses changes might happen in the environment changing the performance of prediction models. An approach to monitor

the change in performance of a prediction model is to turn on the sensing operations of the sleeping nodes at a specified time interval. The interval monitoring has the issue of spending unnecessary energy just for checking whilst there are no changes, if the interval is small and a possibility of a delayed reaction to changes if the interval is large. By representing the monitoring frequency as a function of trust the sleeping nodes are monitored more frequently when the performance of the prediction model is expected to be poor and less frequently when the performance of the prediction model is expected to be good.

2. *Changing the predictability graph:* The predictability between sensor readings of sensor nodes changes in the WSN, which might result in edges being added or deleted in the graph $G$. This addition or deletion of edges needs to be identified to reduce sensing and transmission operations and maintain the sensing fidelity. In BLESS the trust value is used to identify addition or deletion of edges in graph $G$.

All the above actions are discussed in further detail in Sections 3.2.5.2 and 3.2.5.3. For now we focus on trust and how to identify its value. To identify the value of trust in a prediction model, first a trend based on the error terms $E_t$ of the regression model is identified. Using this trend, a value of trust is assigned to the prediction model and is re-estimated every time monitoring of the idle node happens. Each of these parts is explained in more detail below:

1. *Error trend:* A value for the trust is identified using the error terms $E_t$ of the regression model. As explained in Section 3.2.3.3, relationships exists between the error terms of the regression model $E_t$. Making use of this relationship, a trend of these error terms for all the prediction models between the ordered node pairs is identified to estimate the prediction errors of each of these prediction models in the future. An SLR model is used to identify this trend. The newest error terms provide the most information concerning future error terms. To give higher importance to newest error terms, weighted least squares (WLS) is used to calculate

the parameters of the SLR model as shown in Equation 3.5. Since newest error terms provide more information, the weights of $W \in w_0 \dots w_t$ is increased as time $t$ increases. In BLESS an exponential weighing scheme is used.

$$\begin{bmatrix} \hat{A}_{error} \\ \hat{B}_{error} \end{bmatrix} = \arg\min(w_t \times \sum_{t=0}^{n} (e_t - \hat{e_t})^2) \tag{3.5}$$

2. *Estimating value of trust:* During the validation phase, $\beta$ number of sensor readings are obtained from the sensor nodes. The prediction error values for the validation phase $E_{Validation} = \{e_0 \dots e_\beta\}$ can be calculated by comparing the sensor readings obtained with the estimates of the predicted sensor readings. If no error values in the set $E_{Validation}$ exceeds $u_{error}$, a trust value of 1 is initially assumed else a trust value of 0 is given. If the assumed trust value is 1, using this value the next monitoring time for the sleeping node $t_{monitor}$ is identified. Next an error trend is estimated using the validation error terms in $E_{Validation}$ as explained previously in point *Error trend.* Using this error trend estimates of the future errors till the next monitoring time $t_{monitor}$ is estimated, $E_{\hat{Monitor}} = \{e_{\hat{\beta+1}} \dots e_{\hat{Monitor}}\}$. If the estimated error values in the set of the estimated errors $E_{\hat{Monitor}}$ does not contain any value greater than $u_{error}$, the trust remains unchanged with a value of 1, otherwise the time $t_{error}$ of the first occurrence of error values in $E_{Validation}$ exceeding the $u_{error}$ value is noted. A trust value is then given such that monitoring occurs $\phi$ samples before the $t_{error}$ value.

3. *Re-estimating value of trust during monitoring:* During the monitoring of the sleeping nodes, $\omega$ number of sensor readings are obtained. The prediction errors $E_{Monitor} = \{e_0 \dots e_\omega\}$ can then be calculated by comparing all monitored sensor readings against predicted sensor readings. If any of the values in the set $E_{Monitor}$ exceeds the user-specified error $u_{error}$, a trust value of 0 is given. Otherwise, the error trend's accuracy in predicting future errors is checked, to prevent taking

actions based on wrong values of trust calculated using the inaccurate error trend identified. This accuracy is checked by verifying if all the real error values $E_{Monitor}$ obtained during the monitoring of sleeping nodes lie within the prediction interval of the prediction estimates of $E_{\hat{Monitor}}$. If all the error values do not lie within the prediction interval of $E_{\hat{Monitor}}$, more monitoring needs to be enabled, which will in turn improve the estimates of the value of trust and thus enables to take better actions. In the case that all the error values lie within the prediction interval of $E_{\hat{Monitor}}$, the estimate of trust in the model is good and more monitoring might not be required to improve estimate of the value of trust. Increasing the frequency of monitoring for identifying better estimates of the trust value, increases the energy consumption but by taking appropriate actions energy conservation and the sensing fidelity can be improved as a result. Since the frequency of monitoring is going to be a function of trust (see point *Monitoring sleeping nodes*), the value of trust can be decreased or increased to either increase monitoring or decrease monitoring respectively. The reduction in the value of trust needs to be higher when the error trend is expected to be inaccurate compared to increasing the value of trust when the error trend is expected to be accurate. This is because actions taken during inaccurate error trends can cause a loss in sensing fidelity and need to be improved as soon as possible. On the other hand increasing the value of trust less than decreasing its value can make BLESS more cautious when the error trend is expected to be accurate which can also improve sensing fidelity. For this reason the value of trust is decreased and increased by $|e_{Monitor} - e_{\hat{Monitor}}|$. It should be noted again that the trust value cannot be greater than 1 or less than 0, so when increasing or decreasing the trust values are capped between 1 and 0. Error trends could also be inaccurate if the older error values in $E_t$ are skewing the trend line. Though this is reduced by the exponential weighing scheme is used, to further prevent this skewing, all the old error values are removed apart from the newest two values $e_{CurrentTime-1}$ and $e_{CurrentTime}$. The removal of the old error values is

done if the trust is decreased twice in a succession. The newest two values are kept because a minimum of two values is necessary to identify the error trend. After increasing/decreasing the value of trust appropriately, using the current value of trust the next monitoring time for the sleeping node $t_{monitor}$ is identified. The error trend is refitted including the $E_{Monitor}$ values and the new value of the trust in the model is evaluated based on this trend. As explained previously in point *Estimating value of trust*, future errors till the next monitoring time $t_{monitor}$ is estimated and then the new value of trust is appropriately estimated.

This whole process is explained as a flow chart in Figure 3.4.

In the next section the relationship between the monitoring time $t_{Monitor}$ and the trust calculated in the prediction model is explained.

#### 3.2.5.2 Monitoring sleeping sensor nodes

As explained earlier, changes might happen in the environment as time progresses affecting the performance of the prediction models. One of the approaches to check for changes is to monitor sleeping nodes periodically. Periodic monitoring spends a lot of energy if the period is small and takes a lot of time to react to changes if the period is large. Monitoring of the sleeping nodes needs to be done within small intervals when their respective prediction model's performance is expected to be low. When the performance of a prediction model is expected to be high, monitoring of their respective sleeping nodes still needs to be done to ensure its performance but can be done using larger periodic intervals between monitoring of sleeping nodes. Since the trust calculated for the prediction models is an indicator of its current and near future performance, it is used in identifying the monitoring interval $t_{Monitor}$ based on the Equation 3.6.

$$t_{Monitor} = round(K \times Trust) \tag{3.6}$$

Where $K$ is the monitoring constant that represents the maximum $t_{Monitor}$ interval

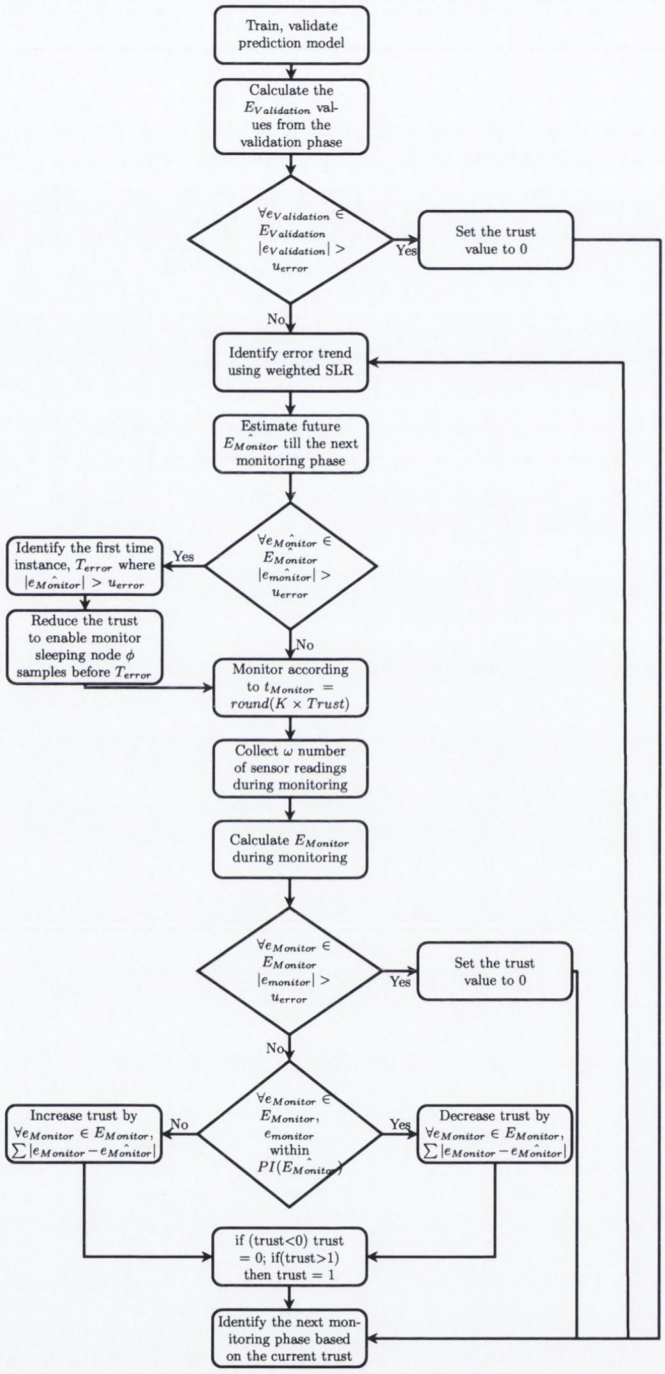



**Fig. 3.4**: Flowchart showing the calculation of the trust in the model

that is allowed when the prediction model is completely trusted. As the trust in the prediction model is lowered, the $t_{Monitor}$ interval becomes smaller, which leads to monitoring the sleeping sensor nodes more frequently, reducing the possible occurrences of $e_t > U_{error}$. These monitored readings can then be used to re-train the model to quickly react to any changes in the performance of the prediction model. Details of re-training the prediction models are explained further in Section 3.2.6. It should be noted that different values for $t_{Monitor}$ can exist for different prediction models, thus the monitoring of all the sleeping sensor nodes does not have to take place at the same time. Every time monitoring of the sleeping sensor nodes take place at the specified $t_{Monitor}$, $\omega$ number of sensor readings are collected. It is more beneficial to monitor sensor readings of sleeping nodes frequently than collect a lot of sensor readings during monitoring, i.e., smaller value of $t_{Monitor}$ is more beneficial than having larger values of $\omega$ (see evaluation in Section 4.7). This is because more new information about the interaction of the prediction model with the environment can be obtained, leading to better calculation of trust in the prediction model. Hence the values of $\omega$ is usually smaller than values of $(\alpha + \beta)$.

If a priori information, related to the variation[1] between sensor readings of different sensor nodes is available better decisions can be taken on choosing a value for $K$. Larger values of $K$ can be used, if the variation between the sensor readings of different sensor nodes is low. The lower the variation between the sensor readings of different sensor nodes slower are the changes happening in the environment. Information such as time of the day and distance between sensor nodes might play a crucial role in choosing the value of $K$. If this information is available it can be used in choosing the value of $K$:

1. *Time of day:* In certain environments, different times of day might show different variations between the sensor readings of sensor nodes. For e.g., in the Intel lab deployment (Peter et al., 2004) it can be observed that during the night time (6PM to 6AM) the variations between sensor readings are low. Thus during the night time, larger values of $K$ can be used and during the day time smaller values of $K$

[1]Variation here denotes the amount of change in the values of sensor readings

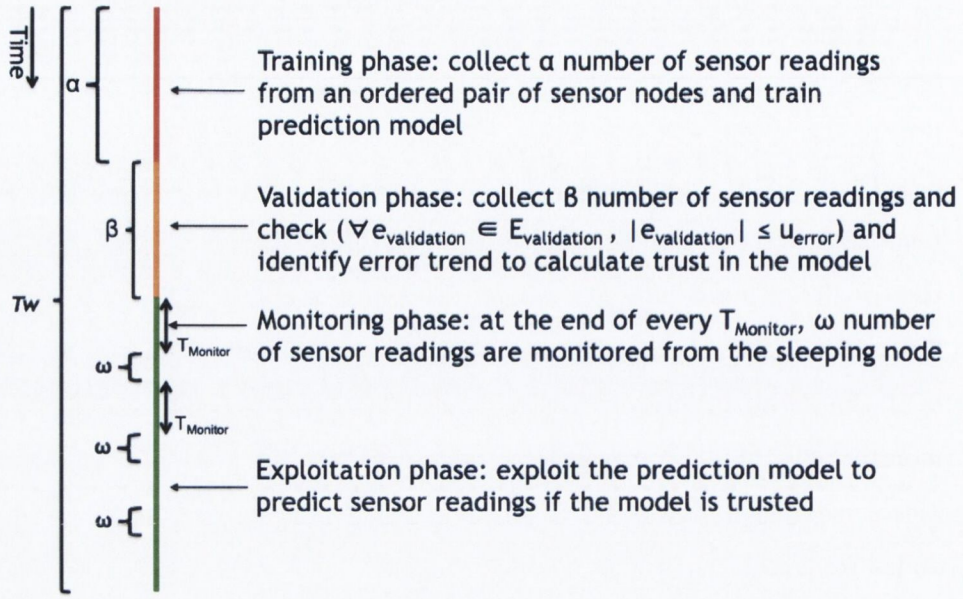



**Fig. 3.5**: Different phases of a prediction model within $Tw$ for an ordered sensor node pair

can be used.

2. *Distance between sensor nodes:* Similar to time of the day, when the sensor nodes are close the sensor reading variation between them might be lower. The closer the sensor nodes, the larger the values of $K$ that can be used.

Thus by having some extra information better choices for the value of $K$ can be made. These effects are further evaluated and discussed in Section 4.6.1 and 4.6.2. Figure 3.2 is modified appropriately to include the monitoring of the sensor nodes and is shown in Figure 3.5.

In the next section details of how to identify and take appropriate actions to changes in the predictability graph $G$, as there are changes in the environment is described.

#### 3.2.5.3 Adjusting changes in the predictability graph

The predictability graph $G$ might undergo changes through out the lifetime of the sensor network, as a result of:

- Node failures, which can happen as a result of hardware failure, damages and due to lack of energy. This will cause some vertices $v \in V$ in graph $G$ to be removed.
- Extra nodes, which can be added in the sensor network to improve sensing fidelity or coverage of the phenomenon under observation. This will cause vertices $v \in V$ to be added in $G$.
- Change in the predictability of sensor readings between the sensor nodes might happen as the environment changes. This will cause some edges $e \in E$ to be added or deleted.

Changes in $G$ will affect the set of the representative nodes. Thus, changes in $G$ should trigger identification of representative sensor nodes and rescheduling the identified sets appropriately. If a sensor node $V$ is supposed to transmit sensor readings and it does not for $\Theta$ number of sensor readings, it is assumed dead. It should be noted that node failures can be detected only when the node is active and transmitting sensor readings or during the monitoring phase. The addition of new sensor nodes in the WSN can be detected when they start to transmit sensor readings. Nodes that do not have connecting edges in $G$ are re-trained continuously as $\alpha + \beta$ number of sensor readings become available. When the value of the trust in the re-trained model becomes 1, an edge between these sensor nodes in $G$ is added. Re-training is further discussed in Section 3.2.6. An edge $e$ in $G$ is deleted, if the trust identified forces monitoring of the sleeping node continuously at every successive sample for a period of $\tau$. The period $\tau$ is chosen such that $(\alpha + \beta)$ number of sensor readings can be obtained. When $(\alpha + \beta)$ sensor readings can be obtained, the model can be re-trained and if the resulting model has a trust of 0, the edge can be deleted.

Certain sensor nodes might be faulty, giving wrong information or could potentially be hacked to give wrong information. These nodes are referred to as rouge nodes in this thesis. Rogue nodes that might exist in the WSN can cause deletion and addition of edges/nodes continually. This can be a hindrance as representative nodes and the appropriate duty-cycle need to be identified every time deletion and addition of edges/nodes happen. To prevent this from happening, a mechanism can be employed such that the representative nodes and the appropriate duty-cycle is calculated only when $G$ undergoes changes above $X\%$. Whenever an edge or node is added or deleted in $G$ a counter is incremented at the sink node. The percentage of change is then a ratio of the number of changes in $G$ to the total number of vertices and edges in $G$ before change. A small value of $X\%$ leads to faster reaction when changes occur. If it is believed there are not many rouge nodes in the WSN, small values of $X\%$ can be used as the threshold to recalculate the representative nodes and the appropriate duty-cycle.

A delay in reacting to changes in $G$ can also be caused when using transceiver scheduling approaches that reduce the receive operations of sensor nodes. Usually in WSNs the scheduling of transceivers with respect to receive operations is implemented in the MAC layer. Some of these approaches are discussed in Section 2.4.1. Depending on the MAC layer used there might be some delay in receiving the new duty-cycles when changes occurring in $G$ are identified.

Re-training of the prediction model has been mentioned previously, the next section presents details of how and when re-training is done in BLESS.

### 3.2.6 Re-training prediction models

In BLESS, re-training of the prediction model that models a pair of sensor nodes is done every time $(\alpha + \beta)$ number of training data are received from both the nodes. The training data is collected from the sensor nodes either when they are active, or during the monitoring phase. The predicted readings of sensor nodes are ignored for training the model, even if they are predicted by a prediction model that is trusted. The value of

trust is dependent on the user-specified error $u_{error}$. If the $u_{error}$ value specified by the user is large, then the trust in the model will still be high though the predicted sensor readings are not close to the actual sensor reading. The tolerated errors in predicted readings can cause inaccurate modelling of the sensor readings when used in training the prediction model and thus are not used. It should be noted that the sensor readings obtained from the sensor nodes that can be used for training must be time synchronised. The earliest $\alpha$ number of sensor readings in the training set are used to train the model while the remaining $\beta$ sensor readings obtained at the later time are used for validation.

In the next section a solution for sensor nodes in a WSN using multiple sensors is described.

### 3.2.7 Multiple sensor modalities

The solution described so far is for sensor nodes to which just one sensor is attached. In reality that might not be the case. For example, in the Intel lab WSN deployed by Peter et al. (2004), temperature, light and humidity sensors are embedded in each sensor node. In the case of multiple sensors embedded in one sensor node, the BLESS algorithm can be applied by changing only the approaches for representing the sensor nodes and its sensors in the predictability graph $G$, and for identifying the representative nodes, while keeping all the other parts of the BLESS approach the same. Depending on the energy consumption characteristics of the sensing and transmitting operations, the designs of two alternate solutions are proposed. The first solution assumes that the transmit operations are energy inexpensive and trade-offs transmit operations for a less computationally expensive algorithm. The second solution provides a heuristic approach which is more computationally expensive than the first solution, to reduce both the transmit and the sensing operations of sensor nodes. These solutions are discussed below:

#### 3.2.7.1 Energy of sensing greater than transmit operations

If multiple sensors exist in sensor nodes and the energy cost of the transmission operations are negligible compared to the sensing operations of each of the sensors, then the WSN is represented as a weighted directed graph $G_{sensors} = \{V, E\}$, where each sensor in the nodes of the WSN is represented by $v \in V$ and an edge $\{u, v\} \in E$ represents the fact that sensor readings of the sensor $v$ are predictable using the sensor readings of $u$. For example Figure 3.6, shows the modified predictability graph $G_{sensors}$ for a WSN containing three sensor nodes with temperature and humidity sensors in each node. The representative sensors can then be identified by identifying the minimum dominating sets in $G_{sensors}$. Next, in order to identify the appropriate duty-cycle, 1) a representative sensor set is chosen to be active ($active_{sensors}$) such that its corresponding nodes have the highest average energy among all the other representative sensor sets' corresponding nodes, 2) the corresponding transceivers ($active_{tr}$) required to transmit sensor readings for each $active_{sensors}$ is also chosen to be active. Then the $active_{sensors}$ represent the minimum number of active sensors needed to be switched on for a duration of $\theta$. The $active_{tr}$, then represents the number of transceivers needed which is not necessarily the minimum transceiver required to be switched on for a duration of $\theta$. To conserve energy within the WSN using BLESS, the same approaches described earlier can then be applied to the $G_{sensors}$ for training or re-training the prediction models, and dealing with changes in the environment.

#### 3.2.7.2 Energy of transmit greater than sensing operations

In the case that, the energy cost of transmission operations is not negligible and consumes the same or a higher amount of energy as the sensing operations then a heuristic solution is proposed to identify the representative nodes. It should be noted that whenever any one of the sensors in the sensor node performs a sensing operation, that node needs to use the transmit operation to transmit the sensed reading. Since energy consumed by

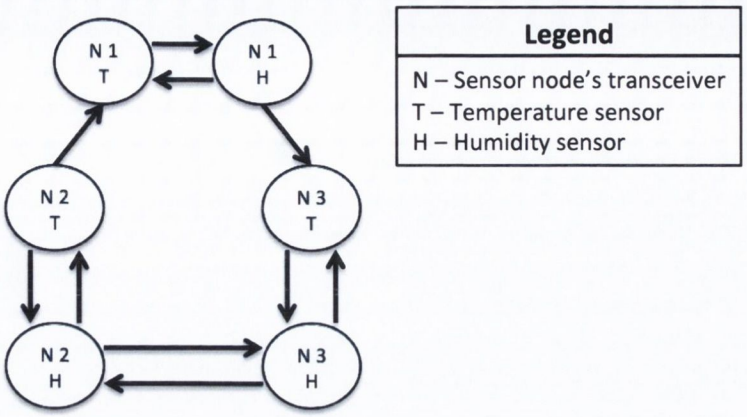



**Fig. 3.6**: Predictability graph $G$ for multiple sensors (sensing energy > transmit energy)

transmit operations are not negligible, the total number of transmit operations performed needs to be kept to a minimum. After this has been achieved the total number of sensing operations then needs to be kept to a minimum. Keeping both the transmission and sensing operations to a minimum will increase the energy savings. To do this a heuristic solution is proposed which is divided into two steps. The first step involves identifying the minimum number of transceivers required in transmitting the sensor readings, thus reducing the number of transmit operations. Based on the calculated minimum number of transceivers required, the second step involves identifies the minimum number of sensors required to sense the phenomena. These steps are shown in Figure 3.7 and 3.8, for a WSN containing three sensor nodes, with temperature and humidity sensors and is also described below:

#### 3.2.7.2.1 Step 1: Identifying minimum transceivers required in WSN

1. A weighted directed graph $G_{tr} = \{V, E\}$ is created, where $V = \{TR, S\}$, $TR = \{tr_i \ldots tr_n\}$ represents transceiver of each node $i$, $n$ represents the total number of transceivers present in the WSN, $S = \{s_i \ldots s_{ns}\}$ represents each sensor $i$ in the WSN, $ns$ represents the total number of all the sensors present in the WSN, an edge $\{tr_i, s_i\} \in E$ represents the fact that $s_i$ requires $tr_i$ to transmit the sensed

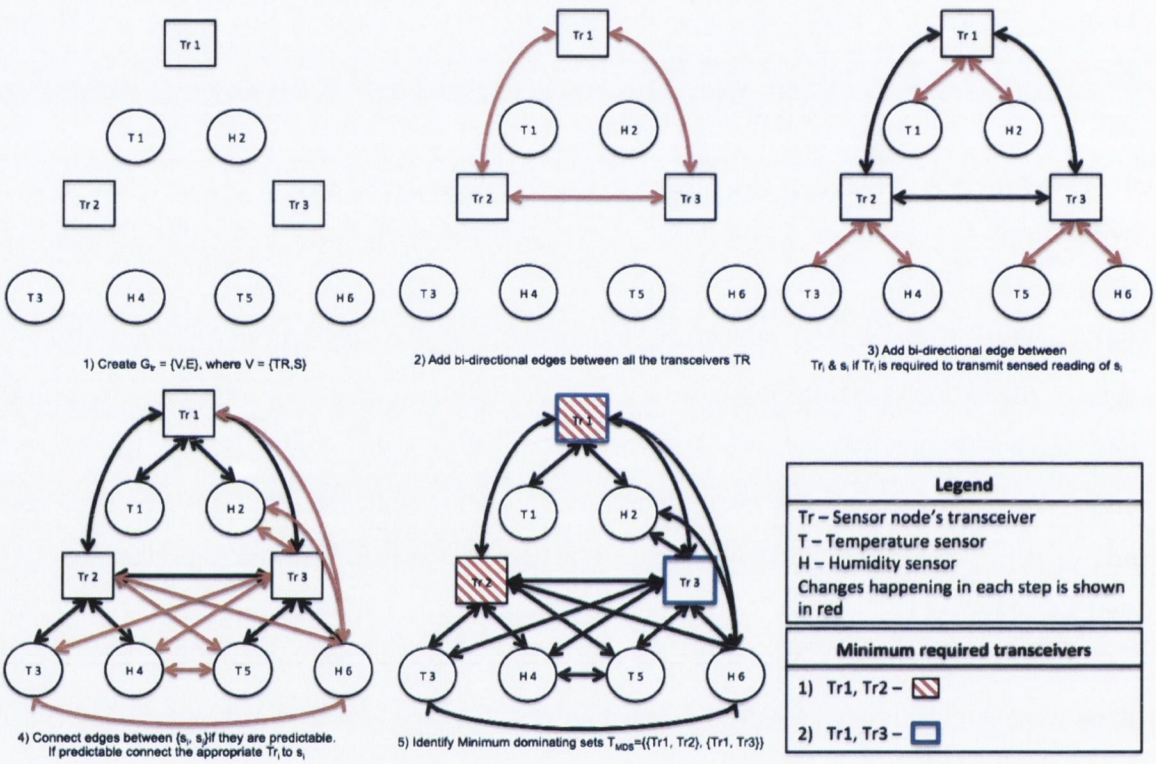



**Fig. 3.7**: Step 1: Identifying minimum transceivers required in WSN (for multiple sensors, sensing energy < transmit energy)

readings, an edge between two different sensors $\{s_i, s_j\} \in E$ represent the fact that the sensor readings of sensor $s_j$ are predictable using the readings from sensor $s_i$.

2. Since, the minimum number of transceivers need to be switched on to reduce the transmit operations, bi-directional edges are added between all the transceivers $TR$ in the WSN.

3. If sensor $s_i$ uses $tr_i$ to transmit sensor readings a bi-directional edge is added between these vertices

4. If the sensor readings of $s_i$, can predict the sensor readings of $s_j$ then an edge is added between these two sensors. In the case the sensors $s_i$ and $s_j$ use different transceivers $tr_i$ and $tr_j$ respectively to transmit sensor readings, a bi-directional edge is also added from $s_j$ to $tr_i$. The bi-directional edge between $s_j$ and $tr_i$ represent that $tr_j$ is not required to transmit the sensor readings of $s_j$ as its readings can be predicted using $s_i$'s readings.

5. Next in order to find the minimum transceivers required the minimum dominating sets $TR_{MDS}$ is identified in $G_{tr}$, the resulting minimum dominating set containing only the set of $TR$ vertices are retained, i.e., $TR_{MDS} \subseteq TR$.

#### 3.2.7.2.2 Step 2: Identifying minimum sensors required in WSN

1. The minimum number of transceivers required are identified, the next step is to find the minimum number of sensors required. Given a directed graph $G_{phy} = \{V_{phy}, E_{phy}\}$, where $V_{phy} = \{TR_{phy}, S_{phy}\}$, $TR_{phy} = \{tr_i \dots tr_n\}$ represents transceiver of each node $i$, $S_{phy} = \{s_i \dots s_n\}$ represents all the sensors in the WSN, a bi-directional edge can exist only between $\{tr_i, s_i\} \in E_{phy}$ representing a physical requirement of $tr_i$ to transmit sensor readings sensed by $s_i$. A new graph $G_{sensors}$ is then identified using $G_{phy}$ and the previously identified $TR_{MDS}$ (see Step 1,

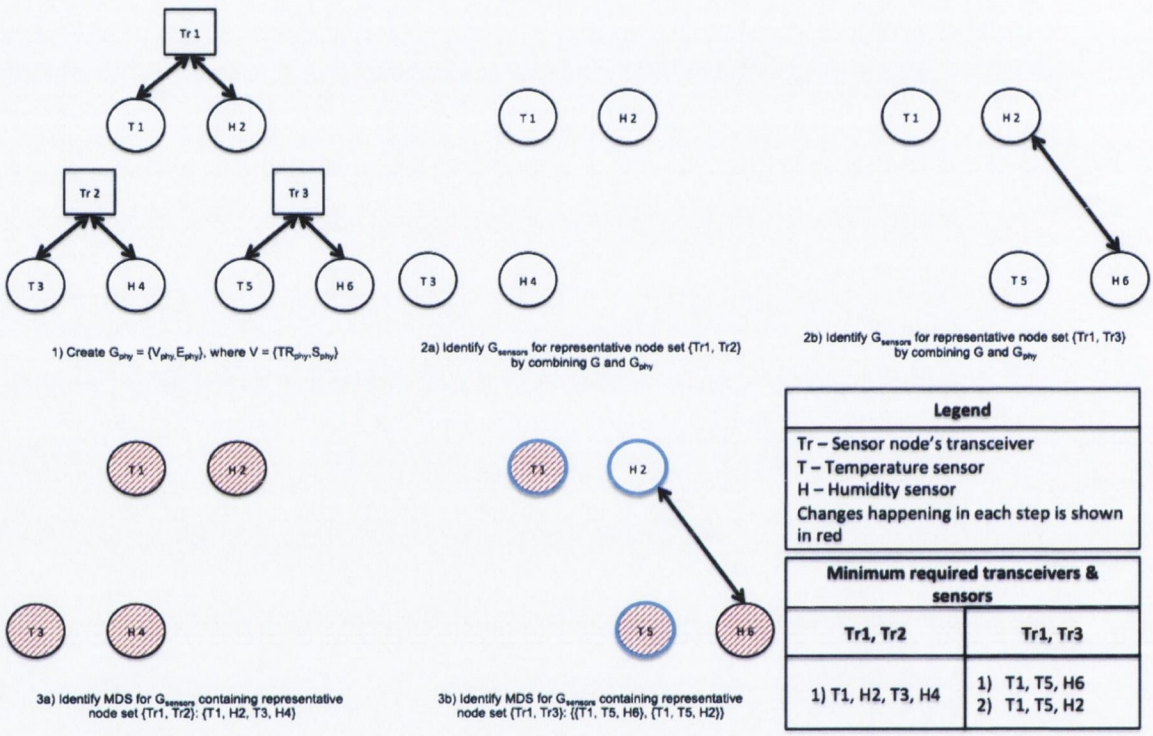



**Fig. 3.8**: Step 2: Identifying minimum sensors required in WSN (for multiple sensors, sensing energy < transmit energy)

point 5). By identifying minimum dominating set in $G_{sensors}$, the minimum sensors required can be obtained.

2. A directed graph $G_{sensors} = \{V_{sensor}, E_{sensor}\}$ is created for each $tr_{mds} \in TR_{MDS}$, such that $V_{sensor} \subseteq S_{phy}$ containing only those sensors $s_i \in S_{phy}$ with edges to $tr_i \in tr_{mds}$ in $G_{phy}$, and an edge $E_{sensors}$ exists only between $\{s_i, s_j\}$ in $G_{sensor}$ only if an edge exists between $\{s_i, s_j\} \in E$ in graph $G_{tr}$.

3. The minimum dominating set $S_{MDS}$ for each of the $G_{Sensors}$ identified, which gives the minimum set of sensors that need to be on.

Next, in order to identify the appropriate duty-cycle, 1) a $tr_{mds} \in TR_{MDS}$ set is chosen to be active ($active_{tr}$) such that its corresponding nodes have the highest average energy among all the other $tr_{mds} \in TR_{MDS}$ sets' corresponding nodes, 2) the corresponding $s_{MDS} \in S_{MDS}$ for the ($active_{tr}$) required to sense the phenomena is chosen to be active ($active_{sensors}$) such that its corresponding nodes have the highest average energy among all the other $s_{mds} \in S_{MDS}$ sets' corresponding nodes. Then the $active_{tr}$ and $active_{sensors}$ represents the minimum number of sensors and transceivers that need to be switched on for a duration of $\theta$.

So far the training and re-training of prediction models, duty-cycles and trust calculations are all done at the central base station. This might limit the size of the WSN to be deployed, in the next section some potential solutions to scale the size of the WSN is discussed.

### 3.2.8 Scalability

In a centralised architecture as described above, all the sensor nodes need to transmit sensor readings to the central base station where the computations of prediction models, duty-cycles and trust calculations are done. Every time a model needs to be re-trained, a new duty-cycle needs to be identified, or recalculation of trust is required the appropriate sensor readings need to be transmitted to the central sink node where these computations

take place and the calculated new information about duty-cycle and monitoring phases must be propagated to the sensor nodes respectively. Sending and receiving such information from the sink node to the sensor nodes incurs energy expenditure. The upside of the centralised approach is that an advantage can be taken of all possible relationships existing between the sensor readings of all nodes, to reduce the sensing and the transmission operations. In a completely distributed approach sending and receiving updated information from the base station is eliminated but to identify all possible relationships existing between the sensor readings of all nodes requires communication between all the sensor nodes. For example if a WSN consists of N sensor nodes with one central sink node and one sensor modality, then to calculate all the possible relationships existing between the sensor readings of all nodes in the centralised approach requires $N \times (\alpha + \beta)$ number of communications and for the distributed approach $N(N-1) \times (\alpha + \beta)$ number of communications. Thus making the completely distributed approach more energy expensive than the centralised approach if all the possible relationships existing between the sensor readings of all nodes needs to be calculated.

Gedik et al. (2007) suggests that a hierarchical approach can be a better solution than a completely distributed or centralised approach to save energy. Thus, we implement a hierarchical approach where a group of sensor nodes communicate to a cluster head which in turn might communicate to another cluster head or a central base station. The architecture is organised such that each level in the hierarchy is managed by at least one cluster head. Each cluster head can then perform training and re-training of prediction models, identify duty-cycles for the group of sensor nodes and calculate trust in the prediction models. The prediction models trained and the readings of active nodes are sent to the higher levels. The cluster heads at the higher layers in turn try to identify correlations amongst the active nodes and train prediction models, identify duty-cycles and identify trust in the models for the active nodes. Since the prediction models are propagated to all the higher layers the base station will receive all the prediction models trained and thus are able to predict and store sensor readings from all the sensor nodes.

In the hierarchical approach, the need for updating information from the sink node to the sensor node is reduced as a result. The hierarchical approach is useful in larger scale sensor networks where multi-hop communication is required. The centralised solution is a special case of the hierarchical approach if the sensor nodes are one hop away from the base station.

In the hierarchical approach, the cluster heads need to be powerful enough to train and re-train the prediction models, identify duty-cycles and calculate trust in the prediction models. The prediction model used by BLESS is a simple model and could potentially be trained in resource constrained sensor nodes as demonstrated by Borgne et al. (2007). The cluster heads ability to train the prediction models, identify duty-cycles and calculate trust depends on the number of sensor nodes that they are allowed to manage. Higher sensor nodes requires more memory and computational power for storing and training the simple models. Higher computation power is also required for calculating the minimum dominating sets for identifying the representative sensor nodes.

In the next section an approach for dealing with missing sensor readings is discussed.

### 3.2.9 Missing sensor readings

Since wireless communication is not fully reliable we could potentially miss receiving some sensor readings. To counteract this we employ linear interpolation techniques to interpolate missing readings De Boor et al. (1978). Since the user defines the sampling interval it is exactly known which sensor reading is missing. Using the sensor readings before and after the missing sensor readings, linear interpolation is applied. The downside of such an approach is that the effectiveness of the interpolation techniques is low when there are lot of missing sensor readings.

In the next section the implementation of all the above mentioned approaches in BLESS is described.

## 3.3 Implementation of the BLESS algorithm

The BLESS algorithm is implemented in Java and consists of various components as shown in Figure 3.9. The implementation of each of these components is detailed below:

1. *WSN simulator:* The BLESS algorithm is evaluated by a WSN simulator which uses data collected from previously deployed WSN testbeds. The simulator simulates energy consumption by counting the number of sensor readings sensed and transmitted, sensing and sending/receiving by sensor nodes. The sensor readings for each sensor node obtained during the WSN deployment are stored in a CSV file with timestamps of when the sensor readings were obtained or sensed. Using the timestamps the sensor reading for a given time can be retrieved.

2. *Initialisation:* This component initialises all the sensor nodes to be used by the simulator and all the parameter values of BLESS ($\alpha$, $\beta$, $\omega$, $\theta$, $K$, $u_{error}$), collects initial sensor readings from the simulator to train the prediction model between each ordered node pair and finally creates the initial predictability graph.

3. *Representative node set identification:* Given a predictability graph $G$, sets of representative nodes are identified. This is done by identifying minimum dominating sets (MDS) in $G$. Finding MDS is an NP-complete problem and a heuristic solution is used. The MDS problem is converted into a clique problem and solved by using the Bron-Kerbosch algorithm developed by Samudrala and Moult (1998). The worst-case estimate in finding the clique using the Bron-Kerbosch algorithm is $O(3^{n/3})$, where n is the number of nodes.

4. *Scheduling:* The energy-aware scheduler described in Section 3.2.4.2 is implemented in Java to schedule the sets of identified representative nodes. This in-turn instructs the simulator to keep certain sensor nodes active while the rest are sleeping.

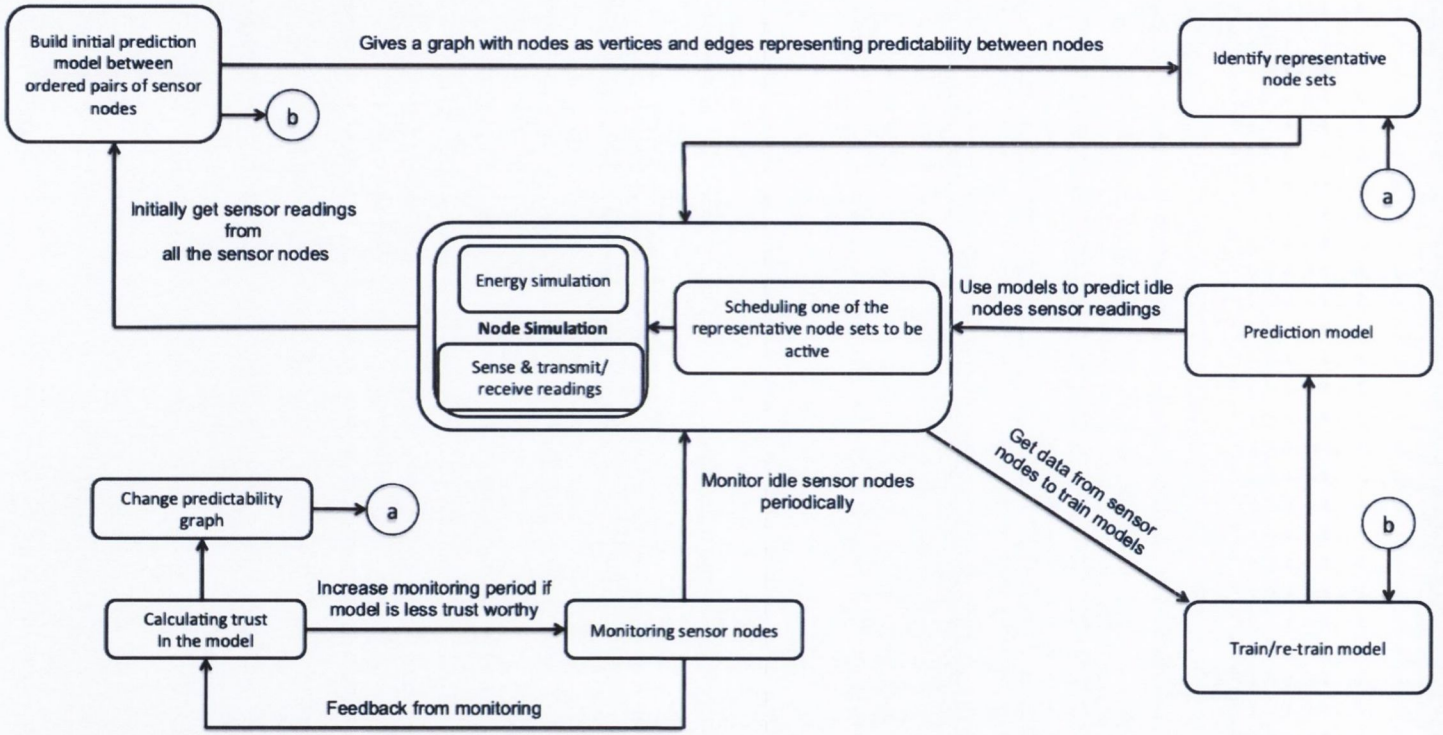



**Fig. 3.9**: Components of BLESS algorithm

5. *Prediction model:* The SLR model is used in BLESS to predict an estimate of the sensor readings of idle nodes and is implemented using the lm command provided in R (Hornik, 2013). Interfacing between R and Java is provided by the rJava libraries (RForge, 2013).

6. *Train/re-train model:* This component is used to collect and store sensor readings from the simulator which is used to appropriately train/re-train the SLR model.

7. *Trust:* The exponentially weighted regression model is used in calculating trust and the same lm command in R is used with exponential weights. The rest of the algorithm described in Section 3.2.5.1 is implemented in Java.

8. *Monitoring sleeping nodes:* This component instructs the simulator, based on the value of trust in the models to monitor sleeping sensor nodes and updates the value of trust as explained in Section 3.2.5.2.

9. *Change predictability graph:* Graph $G$ is modified depending on the trust in the models as explained in Section 3.2.5.3. Once there is a change in $G$, the representative nodes are re-identified and an appropriate new schedule is identified.

## 3.4 Chapter summary

This chapter described the design and implementation of the Less Battery (BLESS) algorithm, a model-based approach that reduces the sensing operations of sensor nodes in a WSN to save energy. It should be noted that by reducing sensing operations the transmit operations are also reduced. BLESS is designed to reduce energy consumption of WSNs, where sensor nodes sense and report about a phenomenon or a set of phenomena at a sampling rate $S_t$ within the deployed environment. The higher the value of $S_t$, the better the approximation of the relationship between sensor readings to a linear function. An SLR model is then used to approximate these linear relationships, which can in turn be used to predict the sensor readings of some nodes given the sensor readings of other

nodes within the WSN. The sensor readings of those nodes that can be predicted, can be put to sleep to reduce sensing and transmission operations. A predictability graph $G$ is identified, which describes which node's sensor readings can be used to predict sensor readings of an other node. To get maximum energy savings, a minimum number of sensor nodes are required to sense and transmit sensor readings. This problem then translates into identifying a minimum dominating set in $G$, a heuristic solution using the Bron-Kerbosch algorithm developed by Samudrala and Moult (1998) is used. This approach for reducing sensing and transmission operations works well if changes taking place in the environment do not affect the performance of the prediction model.

The main contribution of this thesis is designing the BLESS algorithm to be versatile to changes affecting the performance of the prediction model. The issues of model-based approaches when changes in the environment affect the performance of the prediction model were identified in Chapter 2. The following list summarises the solutions provided by BLESS to deal with the identified problems:

1. *Prediction model training*

   Problem: Changes taking place in the environment might require re-training of the trained prediction models. Training and re-training overheads need to be minimised to further reduce sensing and transmission operations.

   Solution: Simple models are used in BLESS thus keeping the training data required to train or re-train the prediction model to a minimum. Thus, when changes happen less energy is required to re-train the model.

2. *Predicted reading accuracy*

   Problem: To check the performance of the prediction model, the sleeping sensor nodes need to be woken up frequently. Waking up frequently will increase the energy consumption.

   Solution: Wake up the sleeping sensor nodes more frequently only when the per-

formance of the prediction model is expected to be low. Trust in the model is calculated to identify the expected performance of the prediction model in the near future. Based on the trust in the model, the sleeping nodes are woken up.

3. *Changes in predictability*

   Problem: Changes taking place in the environment might change the predictability of sensor readings between sensor nodes. Identifying this immediately is crucial to reduce sensing and transmission operations and improve sensing fidelity.

   Solution: The trust in the prediction model identifies the expected near future performance of the prediction model. It can be used to identify if a prediction model is performing badly for a period of time, meaning that the predictability of the sensor reading being predicted has changed. Lower performance of the model triggers frequent monitoring. Thus $(\alpha + \beta)$ readings are available faster to re-train the prediction model, enabling to adapt to changes quickly. By also re-training whenever possible, changes in predictability between sensor readings which were not predictable before can be detected.

In the next Chapter the evaluation of the effectiveness of these solutions is presented.

# Chapter 4

# Evaluation

In this chapter, the BLESS algorithm proposed in the previous chapter is evaluated to show that BLESS can reduce sensing and transmission operations with very little loss in sensing fidelity. To evaluate BLESS, different datasets from real WSN deployments are used, and BLESS is evaluated using five evaluation metrics as follows:

1. *Number of readings sensed and transmitted ($N$):* The number of sensor readings sensed and transmitted corresponds to the number of sensing and transmitting operations performed by the sensor nodes in the WSN. It is assumed that whenever a sensor node senses a reading, it is also transmitted. The lower the value of $N$, the lower is the energy spent.

2. *Ratio of sensing and transmission ($R_t$):* This is the ratio of the number of readings sensed and transmitted to the number of readings the WSN would have sensed and transmitted without any energy saving algorithm, i.e, if all the sensor nodes are sensing and transmitting. An assumption is made here that whenever a sensor senses the phenomenon, those sensed readings are transmitted. Thus, the number of sensing operations is equal to the number of transmit operations. Since communication and sensing operations consume the highest energy compared to other operations (Raghunathan et al., 2006, 2002), thus more energy is conserved when

the value of $R_t$ is lower.

3. *Absolute error count ($\epsilon$):* The absolute error is the absolute value of the difference between the predicted reading and the actual reading. The absolute error count is the number of times that the absolute error exceeds the user specified error $u_{error}$. There is an assumption that the actual readings capture the ground truth, thus $\epsilon$ can directly asses the sensing fidelity of the WSN. The lower the value of $\epsilon$, the better the sensing fidelity.

4. *Ratio of errors propagated ($R_{error}$):* This metric assesses the total errors propagated within the WSN and is the ratio of the absolute error count to the total number of readings the WSN would have transmitted without using any energy saving algorithm. The lower the value of $R_{error}$, the higher the sensing fidelity.

5. *Ratio of inaccurate predictions ($R_\epsilon$):* This is the ratio of the absolute error count to the total number of sensor readings predicted. This metric assesses the predictive accuracy of the trained prediction models. $R_\epsilon$ differs from $R_{error}$ as the predictive capacity of the models are checked. The lower the value of $R_\epsilon$, the higher the sensing fidelity.

The chapter is organised as follows, first the nature of the different data sets being used is explained, then the different parameter settings of BLESS are evaluated and discussed using one of the datasets. Based on this parameter analysis, BLESS is evaluated on other datasets to show its general applicability, then an evaluation of a multiple sensor scenario is presented and, finally, BLESS is compared against the work proposed by Koushanfar et al. (2006).

## 4.1 Data sets

The three data sets used for evaluating BLESS are derived from environmental monitoring applications that require continuous sensing and transmission of the sensor readings.

These data sets are chosen to represent an indoor, outdoor and a semi-indoor environment. Semi-indoor environments have some sensors that are deployed indoors and the rest are deployed outdoors. These datasets are chosen in particular to show that the BLESS algorithm can work in wide variety of environments. The three data sets are:

1. *Intel Lab (Peter et al., 2004):* This is an indoor deployment made inside the Intel Berkeley Research lab where data was collected from a 54-sensor node deployment between February 28th and April 5th, 2004. Each of these sensor nodes sensed and transmited temperature, humidity, light and voltage readings every 31 seconds to the base station.

2. *Sensor Scope (Guillermo et al., 2006):* An outdoor deployment of 10 sensor nodes was made in Patrouille des Glaciers. These 10 sensor nodes collected ambient temperature, surface temperature, solar radiation, relative humidity, soil moisture, watermark, rain meter, wind speed and wind direction. The sensor nodes sense and communicate the sensor readings every 2 minutes.

3. *Tunnel data (Ceriotti et al., 2011):* A semi-indoor deployment of 40 sensor nodes was made in a road Tunnel in Italy. These sensor node are deployed to adjust the lighting control hence light readings are sensed and transmitted every 30 seconds.

### 4.1.1 Dealing with irregularities in the datasets

The real-world deployment conditions of WSNs are harsh and as a result there might be readings that are inconsistent and not realistic. Some of the observed irregularities and the methods for dealing with them are:

- *Incorrect readings:* Errors in the sensors might give sensor readings that are not realistic such as a temperature of $100°C$. These unrealistic sensor readings are filtered from the dataset.

- *Readings arriving at different times:* Sensor readings from different nodes might arrive at different times at the base station. This could be as a result of wireless communication or device faults. In BLESS, the sensor readings from the nodes need to be time synchronised for training of the prediction model. Timestamps of sensor readings are checked to identify whether the sensor readings are time synchronised. Time stamps of sensor readings were taken in the three data sets used for evaluation, when sensor reading were sensed or received at the base station. We assume that the sensor readings are immediately sent to the base station when sensed and so the time stamp at the time of monitoring and receiving the sensor readings at the base station should be approximately the same. Sensor readings that are not time synchronised are removed from the dataset.

- *Missing readings:* Sensor readings might be missing as a result of wireless communication issues or faulty sensors. For example, the soil moisture, watermark, rain meter sensors were not operational in the deployment by Guillermo et al. (2006) and thus no sensor readings are collected from these sensors . If a small percentage of readings is missing from the data sets, then the technique described in Section 3.2.9 is used.

## 4.2 BLESS parameters

In BLESS, there are seven parameters whose values can be adjusted depending on the scenario to get the best results. These six parameters and their function are re-iterated here:

1. The training data size ($\alpha$) specifies the number of consecutive sensor readings used for training the prediction model.

2. The validation data size ($\beta$) specifies the number of readings used to validate the performance of the trained prediction model.

3. The scheduling time ($\theta$) specifies when the active nodes should be rescheduled.

4. The monitoring constant ($K$) specifies the frequency at which the sensor readings of idle nodes need to be monitored when the trust in the model is 100%.

5. The sampling size ($\omega$) specifies how many readings are obtained during the monitoring of sensor nodes.

6. The error threshold ($u_{error}$) specifies the accuracy requirements of the application.

7. The graph change percent ($X\%$) specifies that the representative nodes need to be recalculated if the graph changes over $X\%$.

Another important factor that can affect the performance of BLESS is the sampling rate of the sensors as requested by the user ($S_T$). The value of $S_T$ cannot be specified in BLESS and it completely depends on the application specifications and requirements. In the following sections, experiments evaluating the effects of the above mentioned parameters on the performance of BLESS are analysed and discussed. In order to analyse the performance of the BLESS algorithm for the various parameters, the parameter under analysis is changed while the rest of the parameters are set to default values (unless otherwise specified). The default values are identified based on some initial experiments and their values are as follows:

1. $\alpha = 10$ readings

2. $\beta = 5$ readings

3. $\theta = 4$ hours

4. $K =$ every 10 readings

5. $\omega = 1$ reading

6. $U_{Error} = 1°C$

7. The default value of $S_T$ depends on the requirements of the application and is different for all the three datasets used for evaluating BLESS. In the Intel Lab dataset, $S_T$ is set to 31 seconds, in the Sensor Scope dataset, $S_T$ is set to 120 seconds and in the Tunnel dataset $S_T$ is set to 30 seconds.

8. In our experiments, we assume that there are no rogue nodes and hence the $X\%$ value is always set to 0%.

Most of the experiments are carried out on the temperature sensor readings of the Intel Lab dataset (Peter et al., 2004), as it was observed to have one of the least irregularities compared to the other sensors and datasets. To reduce the effect of irregularities further in the Intel Lab dataset, 10 sensor nodes of Intel Lab dataset that have the least irregularities in their readings are used (Nodes: 45, 46, 47, 48, 21, 22, 23 ,24, 25, 29).

In the next section, we evaluate how the $\alpha$ values affect the performance of BLESS.

## 4.3 Effect of the size of the training data ($\alpha$)

Training the prediction model requires sensor readings from the sensor nodes resulting in energy expenditure. Thus, less training data $\alpha$ used to train the prediction models will result in conserving more energy. In experiments relating to the effects of the value of $\alpha$, first a comparison is made between large $\alpha$ values with no re-training and small $\alpha$ values with frequent re-training against the BLESS approach. This experiment demonstrates that small $\alpha$ values give less errors and sense and transmit less sensor readings. The next experiment investigates how small should $\alpha$ values be to get good results. This experiment is repeated for varying values of $S_T$.

### 4.3.1 Large or small training data size

Approaches proposed by Deshpande et al. (2004) and Koushanfar et al. (2006) use use complex prediction models to predict estimates of sensor readings of sensor nodes. In

complex models the value of $\alpha$ tends to be large, which is in the order of days. Contrary to the use of complex models, we hypothesise that when using simple prediction models such as simple linear regression models, the linearity assumption might not hold for large $\alpha$ values, and that the model's performance can be improved when using small $\alpha$ values and by re-training the model appropriately. To evaluate this hypothesis, results from three different approaches are compared: 1) large training and validation data without re-training, 2) small training data with frequent re-training and 3) the adaptive monitoring approach of BLESS. In all these three approaches, simple linear regression models are used as the prediction models. All these three approaches use a validation phase where the sensor readings are checked against the predicted readings. If the readings are within $u_{error}$ then an edge is added between this sensor node pair to create a graph $G = \{V, E\}$ (similar to the approach proposed by Koushanfar et al. (2006)). Finally the minimum dominating sets are identified and duty cycled appropriately as explained in Section 3.2.4.

1. Large training and validation data with no re-training: In this approach, $\alpha$ values used are 1440 and 2880 and the corresponding values of $\beta$ used are 1440 and 2880. These $\alpha$ values are chosen to represent the large $\alpha$ values used by complex models, as 1440 represents half a day's worth of collected sensor readings and 2880 represents a day's worth of collected sensor readings.

2. Small training data with frequent re-training: The default $\alpha$ values and $\beta$ are set to 10 and 5 respectively. The prediction models are frequently re-trained at a fixed interval of every hour or two hours. Re-training for every hour or two is chosen because in the literature changes are monitored hourly. For example, Deshpande et al. (2004) propose to model changes in sensor readings every hour.

3. BLESS approach: In this approach, the default $\alpha$ values and $\beta$ are set to 10 and 5 respectively and the adaptive monitoring approach proposed in Section 3.2 is used for the evaluation of the hypothesis.

It should also be noted in the above scenarios that if there is no mention of a parameter value, the default parameter value specified is used. The results of the experiments are shown in Figure 4.1 and 4.2. From these figures, it can be seen that when the prediction model is trained using a large training data set, the predicted readings in the validation phase do not lie within the $u_{error}$ and thus there are no edges in the graph G. This in turn does not reduce the sensing and transmission operations, as all the sensors are sensing and transmitting. Improvements in terms of energy are seen when the model is re-trained frequently. The more frequent the re-training the more energy is expended and fewer errors are propagated. On the contrary, using the BLESS approach, sensing and transmitting operations are switched on when the prediction model's performance is expected to be low, thus the number of these operations are slightly higher than the frequent re-training approach. The improvement of BLESS with regards to the frequent re-training approach is seen where the propagated errors drops from 9.5 % (hourly re-training) to 0.27 %. The error bars in the Figure 4.1 and 4.2 depict a 95% confidence interval (CI). From the figures it is seen that none of the 95% CI error bars overlap each other and hence the values shown are statistically different from each other.

From these results it can be said that using small $\alpha$ values and by re-training the model appropriately, better predictions estimates of sensor readings can be achieved which can be used to reduce sensing and transmission operations. Compared to all the three approaches, BLESS approach of using small $\alpha$ values and re-training based on the readings sensed and transmitted by the nodes based on the performance of prediction models, gives the best balance of reducing sensing and transmission operations and errors transmitted.

The next section presents an experiment to determines the size of the training data.

### 4.3.2 How small should the size of the training data be?

To determine $\alpha$, an experiment is conducted varying the $\alpha$ values whilst keeping all the other variables as the default values. The results of this experiment are shown in Figures

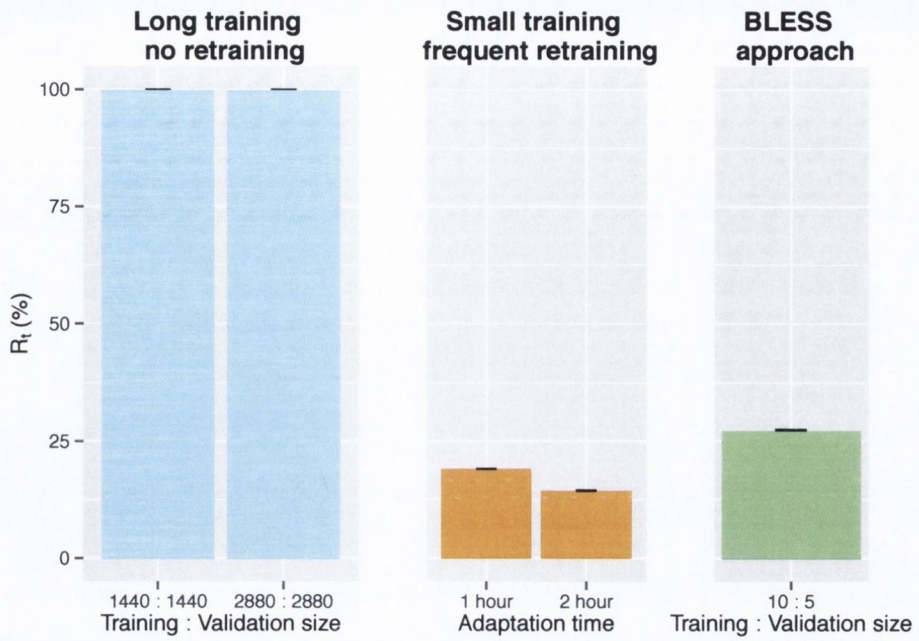



**Fig. 4.1**: The values of $R_t$ for different approaches

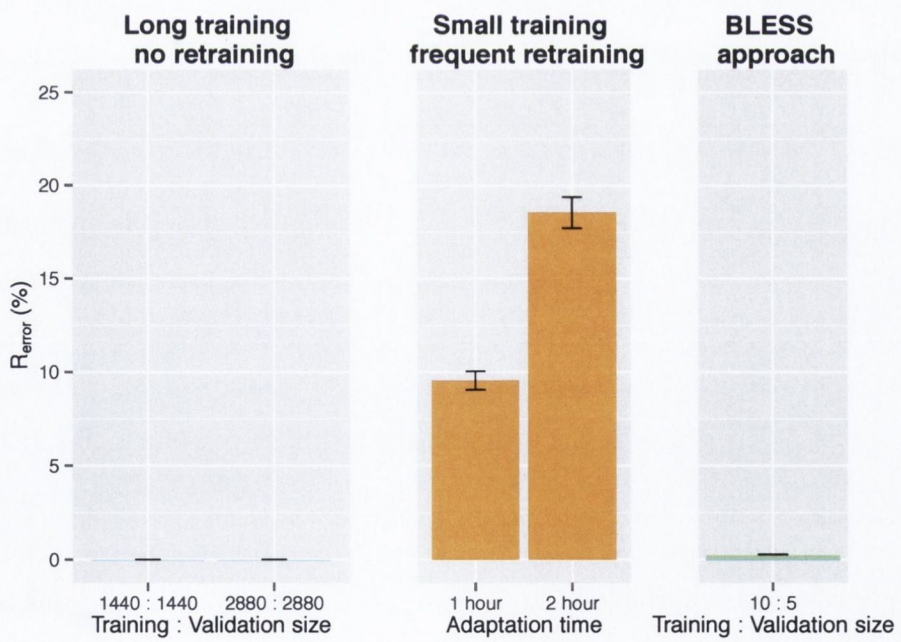



**Fig. 4.2**: The values of $R_{error}$ for different approaches

4.3 and 4.4. From these figures it can be inferred that the ratio of readings sensed and transmitted gets higher as the $\alpha$ values gets higher. The $R_t$ values increase as more sensor readings are required to train and re-train the prediction model. This increase is statistically valid as none of the 95% CI error bars overlap except for $\alpha$ values of 5 and 10. The statistical significance of the results when using 5 and 10 as $\alpha$ values are assessed through honestly significant difference test (HSD) (Mullins, 2003). In Figure 4.4 as the value of $\alpha$ increases, the $R_\epsilon$ values also increase, as the linearity assumptions do not hold for higher $\alpha$ values. The significance test is done again using HSD with 95% confidence and shows that the increase in $R_\epsilon$ for smaller $\alpha$ values (5 to 50) are not statistically significant but the $R_\epsilon$ values between these smaller $\alpha$ values against larger values (300 and 500) are statistically significant. The ratio of the number of packets sensed and transmitted increases statistically as the $\alpha$ values increase(except for $\alpha$ values of 5 and 10), and the errors are low with no statistically significant differences for $\alpha$ values between 5 and 50. Thus we can say that a small $\alpha$ value of 5 or 10 gives the best performance to reduce sensing and transmission operations and improve the sensing fidelity compared to the other $\alpha$ values.

### 4.3.3 Training data size in relation to sampling rate of sensors

In this thesis, the value of $S_T$ is assumed to be specified by the application user. When high value of $S_T$ is used, it might not be possible to approximate sensor readings' relationship to a linear relationship. In this section we explore whether it is possible to minimise approximation errors in sensor reading prediction estimates caused because of $S_T$ using different $\alpha$ values. The effect of $\alpha$ for different sampling rates is thus analysed. It should be noted that currently in the Intel Lab data, the application sets the value of $S_T$ to 31 seconds. To simulate higher sampling rates only values of sensor readings at every consecutive $S_T$ is taken into account. For example, if the value of $S_T$ is set to 1 min, the sensor readings used for the evaluation will only take into account the readings at every minute, discarding the rest of the sensed reading from the dataset. The results

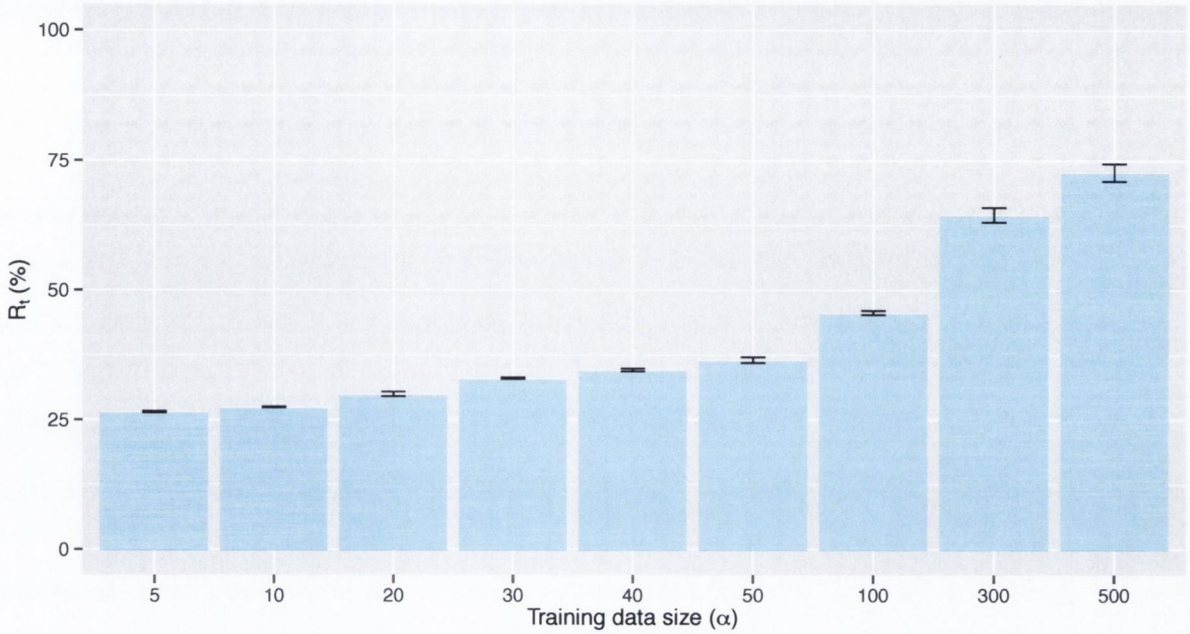



**Fig. 4.3**: The ratio of readings sensed and transmitted for varying $\alpha$ values

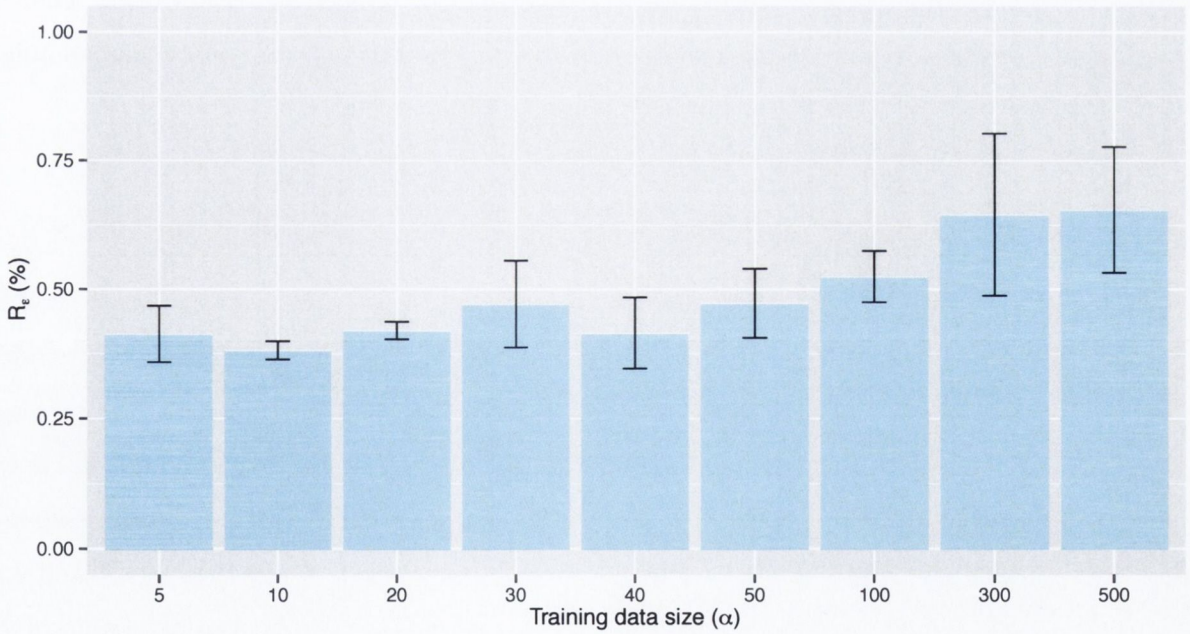



**Fig. 4.4**: The ratio of prediction errors over the network for varying $\alpha$ values

of this experiment are shown in Figures 4.5 and 4.6.

From Figure 4.5, it can be concluded that increasing either $S_T$ or $\alpha$ increases the $R_t$ values considerably. This considerable difference between the $R_t$ values for the various $S_T$ values starts to decrease and become statistically the same as the value of $\alpha$ increases. With increasing $\alpha$ values, the $R_t$ values increase as more sensor readings are required to train and re-train the prediction model. Increasing $S_T$ values causes an increase in $R_t$ values because of the poor performance of the prediction model given that the linearity assumptions do not hold. This poor performance can trigger the monitoring of the idle nodes more often, increasing the $R_t$ values. Similarly in Figure 4.6 increasing the value of $S_T$ increases the value of $R_\epsilon$ while increasing the value of $\alpha$ decreases the $R_\epsilon$. This can be explained by the fact that increasing the $S_T$ values makes the sensor readings' relationship deviate from linearity, thus giving higher prediction errors. Increasing the $\alpha$ values gives an impression of decreasing the value of $R_\epsilon$, but in reality the $R_\epsilon$ value is small because the number of sensor readings predicted using the prediction model is small. This can be confirmed by the high values of $R_t$ for higher $\alpha$ values. Thus it can be concluded that varying $\alpha$ values in BLESS for higher values of $S_T$ cannot minimise approximation errors in sensor reading prediction estimates.

In the next section, the effect of $\beta$ values are assessed.

## 4.4 Effect of validation data size ($\beta$)

For small sampling times, it is identified that small values of 10 or 5 for $\alpha$ offer the best tradeoff for reducing the number of packets transmitted and the errors propagated. Now we assess how much data is best to validate the model trained with small $\alpha$ values. Keeping the default values for all the other variables of BLESS, the experiments are run varying the $\beta$ values. The result of this experiment is shown in Figures 4.7 and 4.8.

In Figure 4.7, the general trend is as the value of $\beta$ increases the value of $R_t$ increases. There is no statistically significant difference for the ratio of number of readings

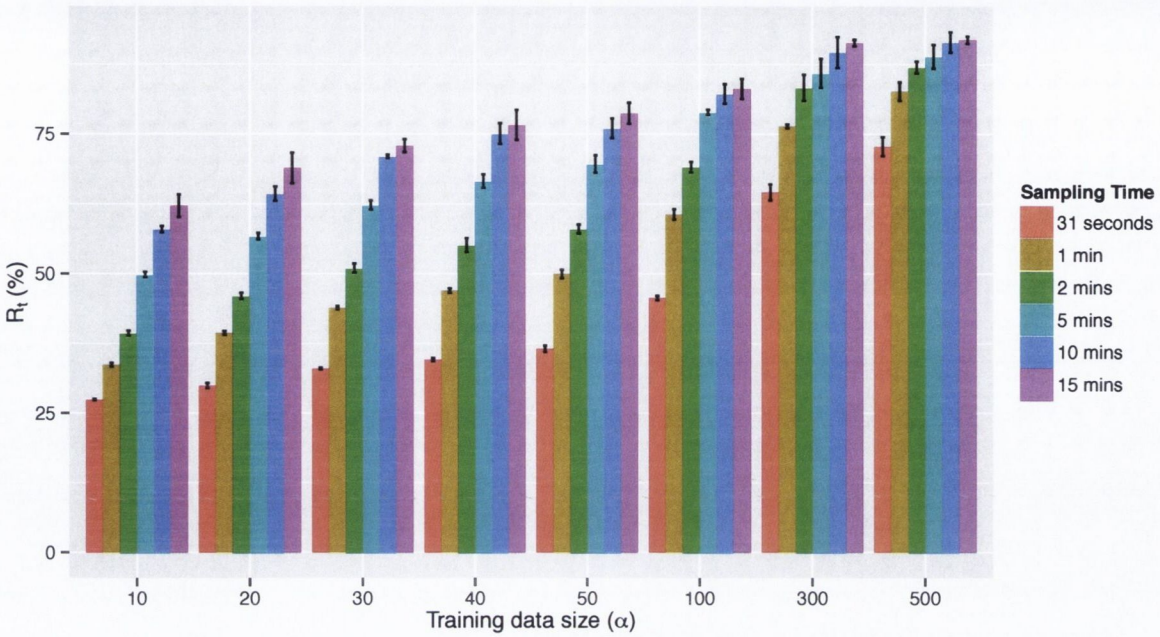



**Fig. 4.5**: The ratio of readings sensed and transmitted for varying $\alpha$ values and $S_T$

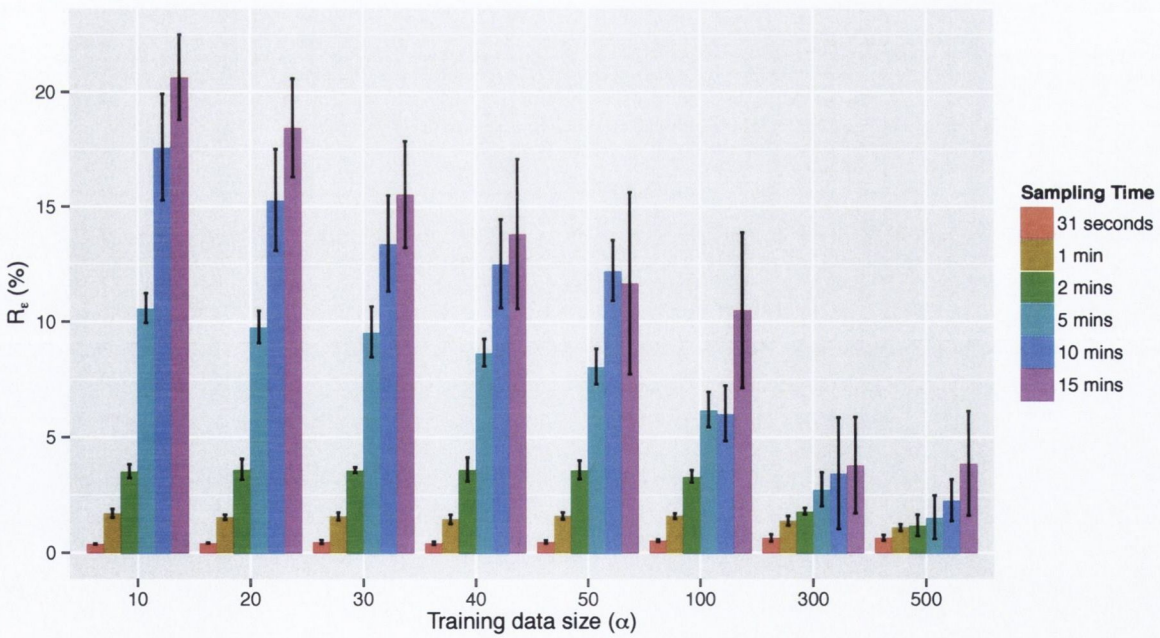



**Fig. 4.6**: The ratio of prediction errors over the network for varying $\alpha$ values and $S_T$

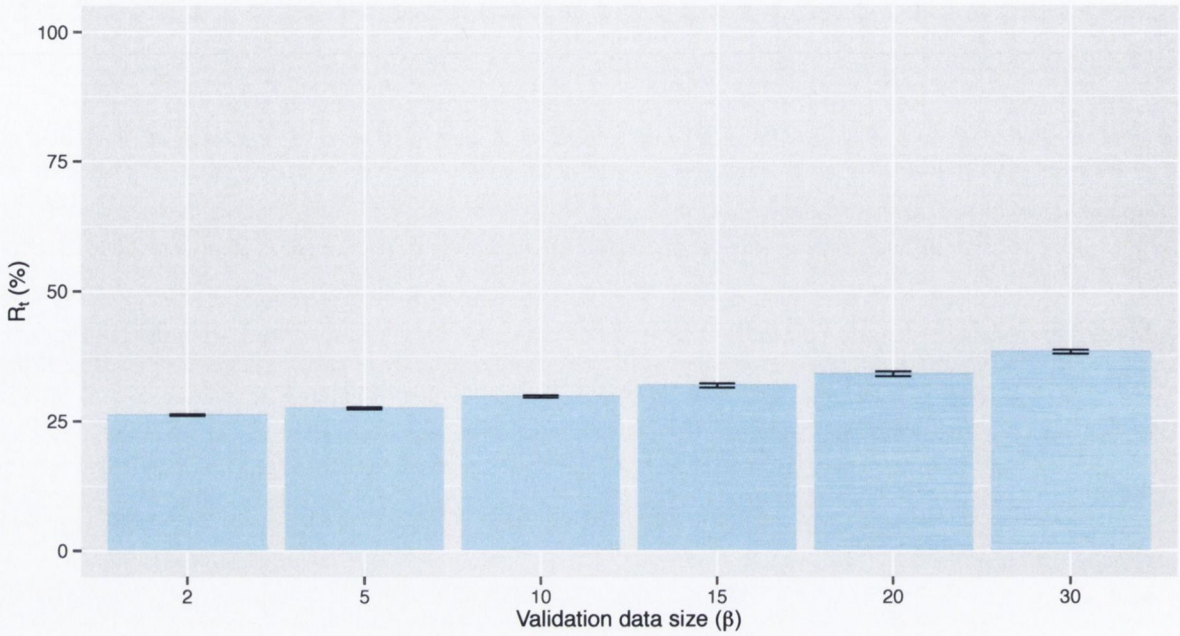



**Fig. 4.7**: The ratio of readings sensed and transmitted for varying values of $\beta$

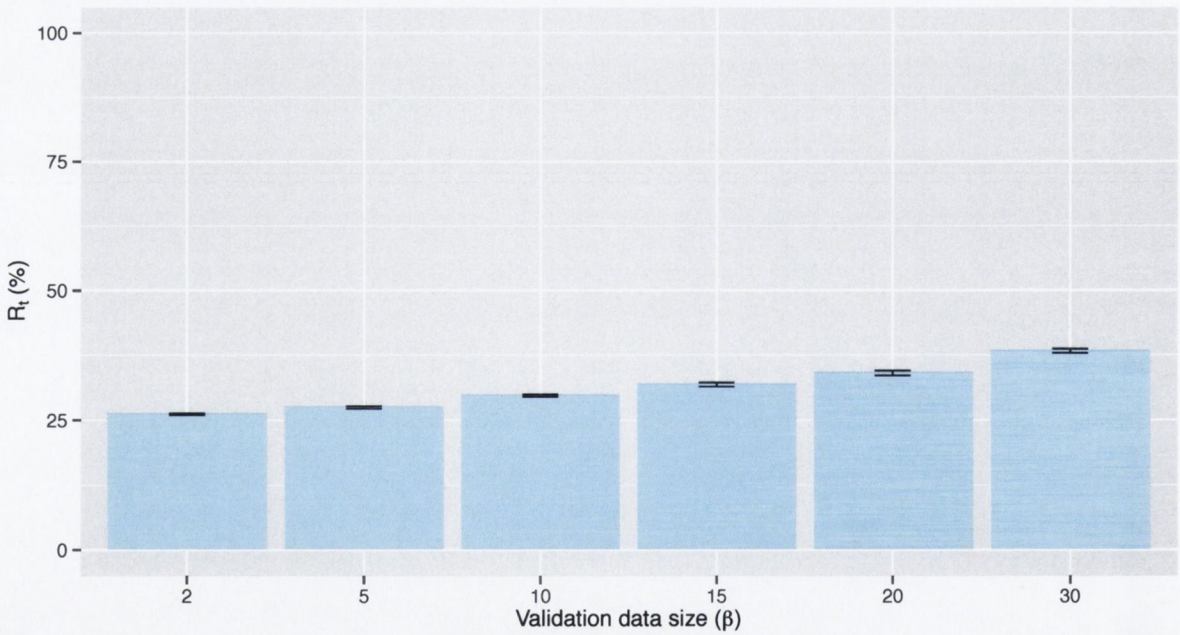



**Fig. 4.8**: The ratio of prediction errors over the network for varying values of $\beta$

transmitted between $\beta$ values of (2,5,10) and between (15,20), but there are statistically significant differences between the rest of the values. In Figure 4.8, a similar trend exists, as the value of $\beta$ increases the ratio of prediction errors increases. There is no statistically significant difference for the ratio of the number of errors propagated between $\beta$ values of (2,5,10) and between (15,20), but there are statistical significant differences between the rest of the values. Lower $R_t$ and value of $R_\epsilon$ are obtained with smaller $\beta$ value of 2,5 or 10. This is because the prediction models mostly perform significantly well in predicting the sensor readings immediately after their training therfore not many sensor readings are required to validate its performance. Whereas over a long time the trained prediction models might not perform so well and thus approaches such as monitoring of the sleeping nodes are used.

In the next section the effects of the $\theta$ value are assessed.

## 4.5 Effect of scheduling time ($\theta$)

Changes to the schedule of the sets of identified representative nodes happen in two cases: 1) Any change in the predictability graph $G$ will trigger the identification of the new sets of representative nodes and thus a new schedule for these nodes, 2) At the end of the time specified by the $\theta$ value. Changes in the predictability graph cannot be controlled, whereas the $\theta$ value can be controlled to determine how long one of the identified representative node sets will be chosen as the set of active nodes to predict the sensor readings of idle nodes. Different prediction models are likely to be used when different representative node sets are chosen to be active, thus the $\theta$ values has an effect on the usage of trained prediction models, which can influence the prediction estimates of sensor readings. The effect of the $\theta$ values are shown in Figures 4.9 & 4.10.

In Figure 4.9 it can be seen that as the $\theta$ value becomes large, less packets are transmitted. This is due to the fact that when new representative node sets are scheduled to be active, monitoring is triggered to ensure that the new active nodes can use their

prediction model to predict accurately. The smaller the duration of scheduling the more often monitoring takes place, leading to higher packet counts. HSD shows that the $R_t$ values are not statistically different for $\theta$ values between 6, 4 & 5 and 10, 12, 24 & 30, whereas the $R_t$ values between the rest of the values are statistically significant. Figure 4.10 shows a trend of error values decreasing as the $\theta$ value increases. This is because, when a large $\theta$ value is used, the prediction models between the chosen set of active nodes and the idle nodes are used for a long time and the trust in these prediction models are updated every time monitoring of the idle nodes happen. Thus, the performance of the prediction model is closely monitored, giving rise to smaller errors. It should be noted that the trust in the prediction models between the rest of the sensor nodes are updated only after re-training the models and this re-training happens only when sufficient monitored data is available from the idle nodes. Sufficient monitored data is not available before new sets of active and idle nodes are chosen at the end of $\theta$, when the $\theta$ value is small. Thus the models between the new set of active and idle nodes chosen might actually be performing poorly, increasing the value of $R_\epsilon$. This can be clearly seen with a big drop in errors when values of 1 hour and 2 hours are used for $\theta$. To re-train, 15 sensor readings (default $\alpha + \beta$ values) are required and cannot be obtained before new set of active and idle nodes are chosen, when the $\theta$ value is set to 1 hour. Through HSD with a confidence level of 95% we evaluate that statistically the ratio of error values between the $\theta$ values of (3, 4, 5, 6) and (10, 12, 24, 30) are the same and are statistically different for all other values.

The main purpose of scheduling different representative nodes is to maintain energy balance across the nodes in the WSN. To show how well different $\theta$ values balance energy across the WSN, the average number of sensor readings sensed and transmitted by the different nodes for different $\theta$ values are shown in Figure 4.11. The straighter the line, the better the balance in energy as all nodes are sensing and transmitting equal numbers of packets. It should be noted that the simulation data used is for 10 consecutive days and thus smaller $\theta$ values enable scheduling all the possible representative nodes identified

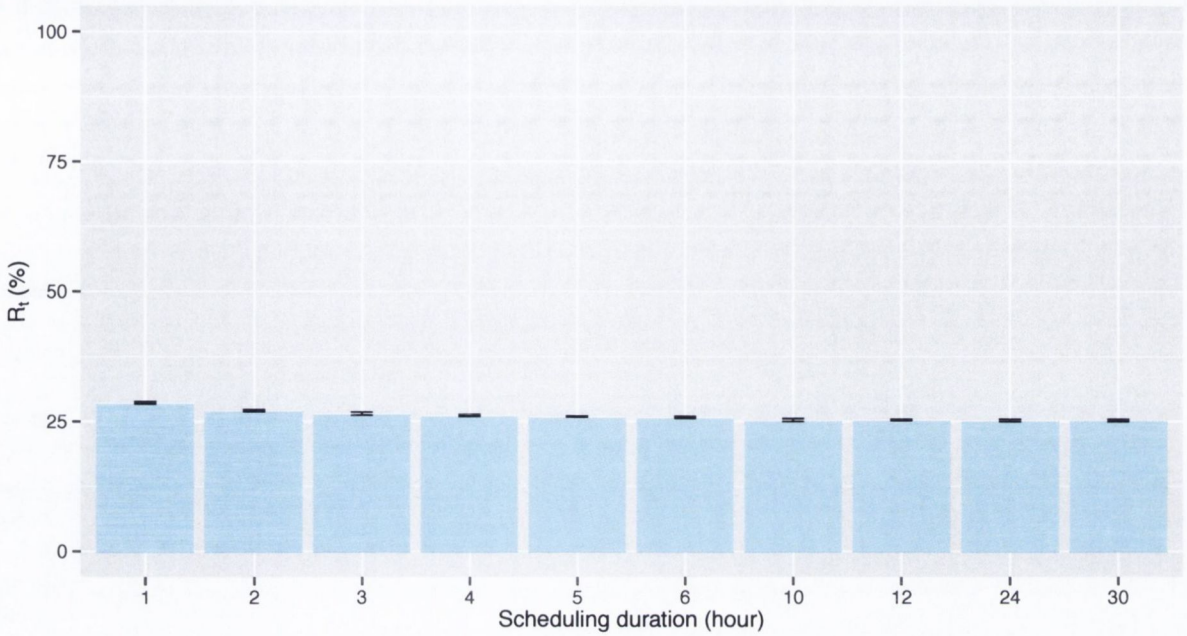



**Fig. 4.9**: The ratio of readings sensed and transmitted for varying $\theta$ values

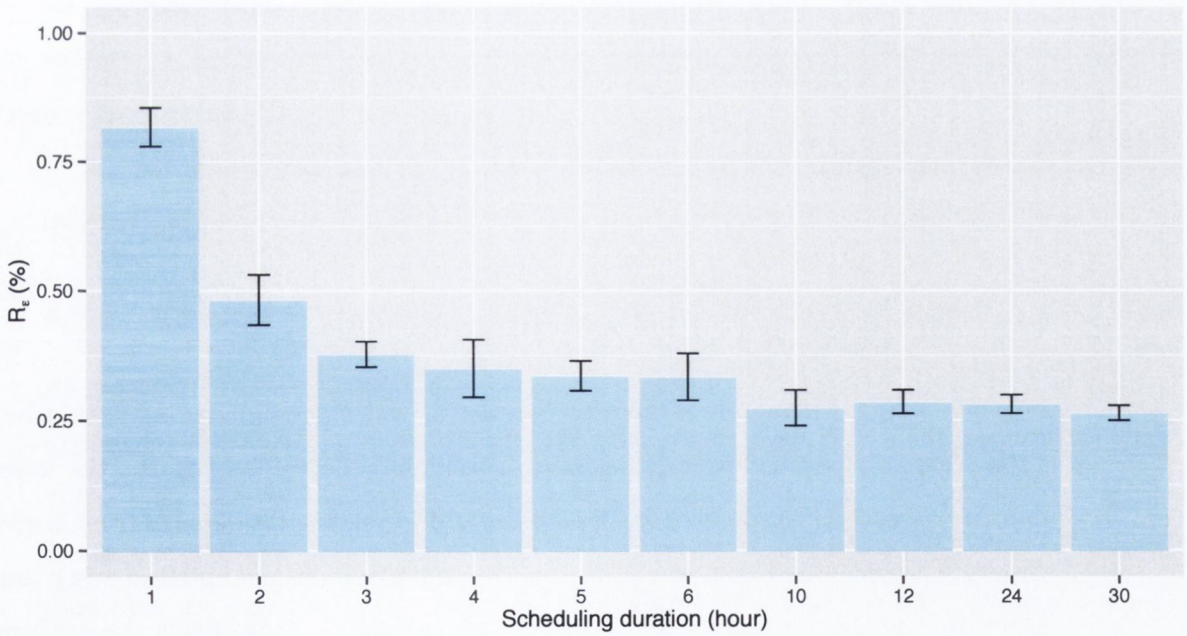



**Fig. 4.10**: The ratio of prediction errors over the network for varying $\theta$ values

giving a better balance in energy across the WSN. The longer the deployment of WSN, the bigger the $\theta$ values that can potentially be used to balance energy evenly across the sensor nodes.

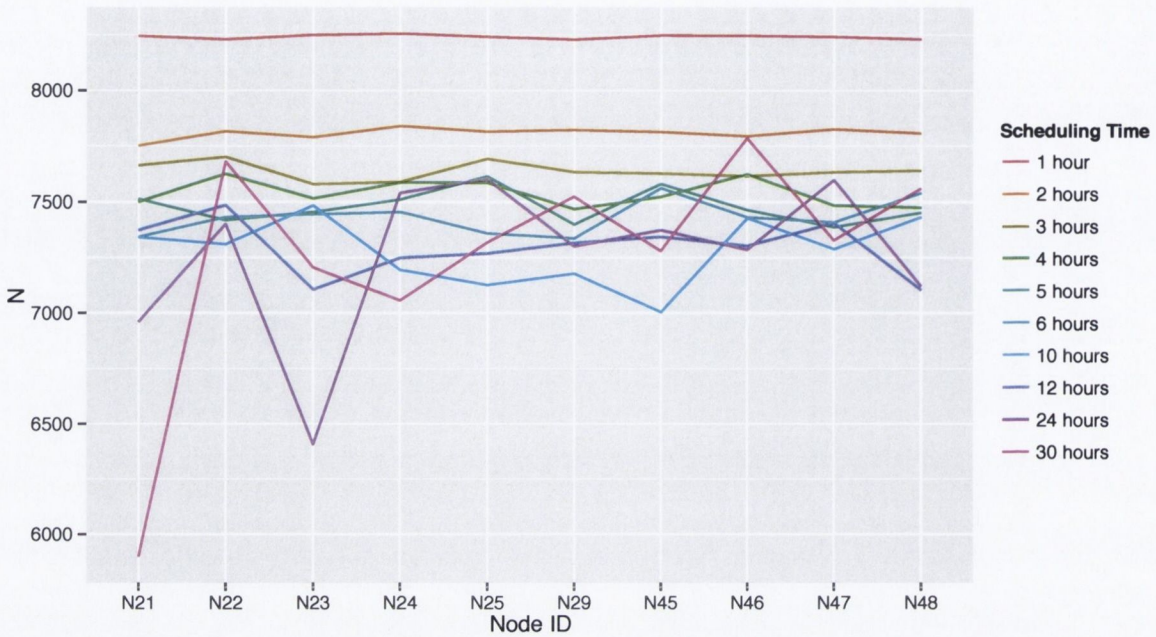



**Fig. 4.11**: The average number of readings sensed and transmitted by each sensor node for varying $\theta$ values

## 4.6 Effect of monitoring constant ($K$)

The monitoring constant $K$ represents the rate at which the monitoring should be done if the trust in the prediction model is 1. The higher the value of $K$ is , the lower the number of monitoring operations taking place and the slower the value of trust is updated (trust in the prediction models between active and idle nodes are updated every time the monitoring of the sleeping nodes is done). The value of $K$ thus effects the performance of BLESS and hence an experiment is conducted to evaluate the effect of $K$. The result

of this experiment is shown in Figures 4.12 & 4.13. As expected, for lower values of $K$ the sensor readings of the idle nodes are monitored more often thus increasing the total number of packets sensed and transmitted, this is shown in Figure 4.12. An interesting result is that the difference in the values of $R_t$ obtained when using the lowest $K$ value and the highest $K$ value is only around 4%. The reason for such a small difference is that when using high values of $K$, the trust identified in the prediction model during monitoring is most likely to go down resulting in more frequent monitoring. This is also evident with only a small decrease in $R_t$ with increasing values of $K$.

Figure 4.12 shows an increase in error with higher $K$ values. This is also confirmed statistically with a confidence of 95% using the HSD method where $R_\epsilon$ is significantly different for all values of $K$, except (25, 30) and (45, 50). The more frequently the error trends are updated for the prediction models, the better the calculated value of trust can estimate possible errors in the prediction estimates of sensor readings in the future. Since the error trends are updated more frequently with smaller values of $K$ the $R_\epsilon$ decreases as the value of $K$ is lowered, giving better error estimates. Choosing smaller values of $K$ is the best approach forward as the reduction of $R_\epsilon$ is substantial compared to the increase of $R_t$.

In the next section, the effect of changing the value of $K$ depending on some known information of the environment is discussed.

### 4.6.1 Changing values of monitoring constant based on time

As explained in Section 3.2.5.2, better results can be obtained if a priori conditions of variation are known about the environment in which WSNs are deployed. As in the case of the Intel Lab data set, it is obvious that during night time there is very little change in the environment compared to the day time when people are present in the lab. To show this, an experiment is done where the value of $K$ is halved during the time from 6 AM to 6 PM representing day time and during the rest of the time (night time) the value of $K$ is doubled. The results are shown in Figure 4.14 & 4.15. In Figure 4.14

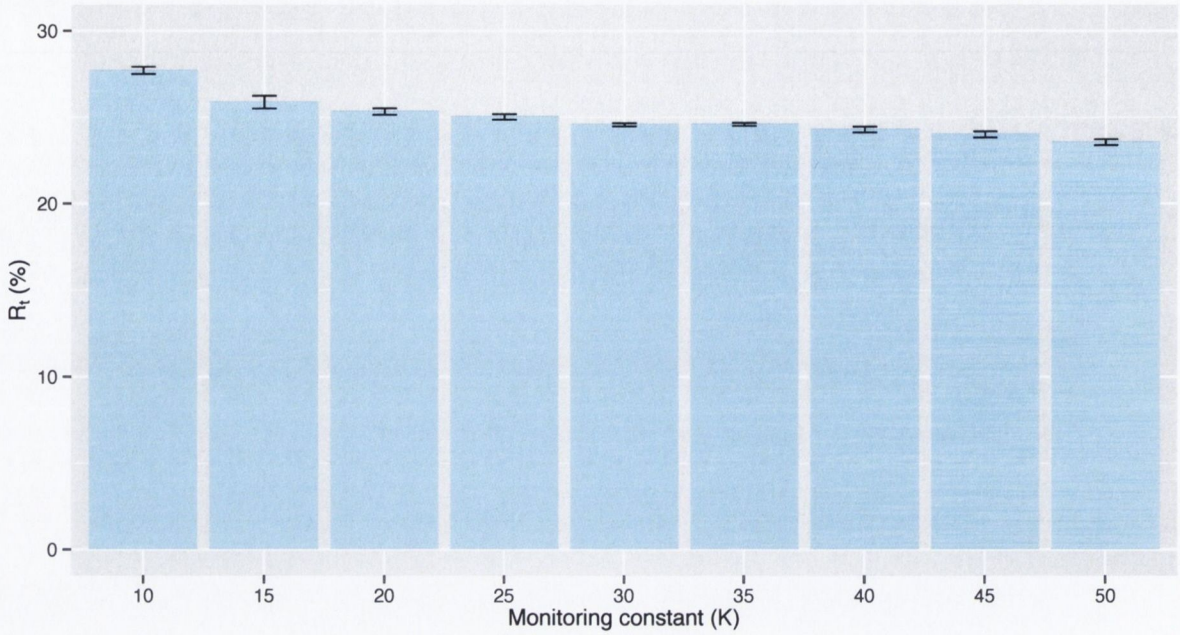



**Fig. 4.12**: The ratio of readings sensed and transmitted for varying $K$ values

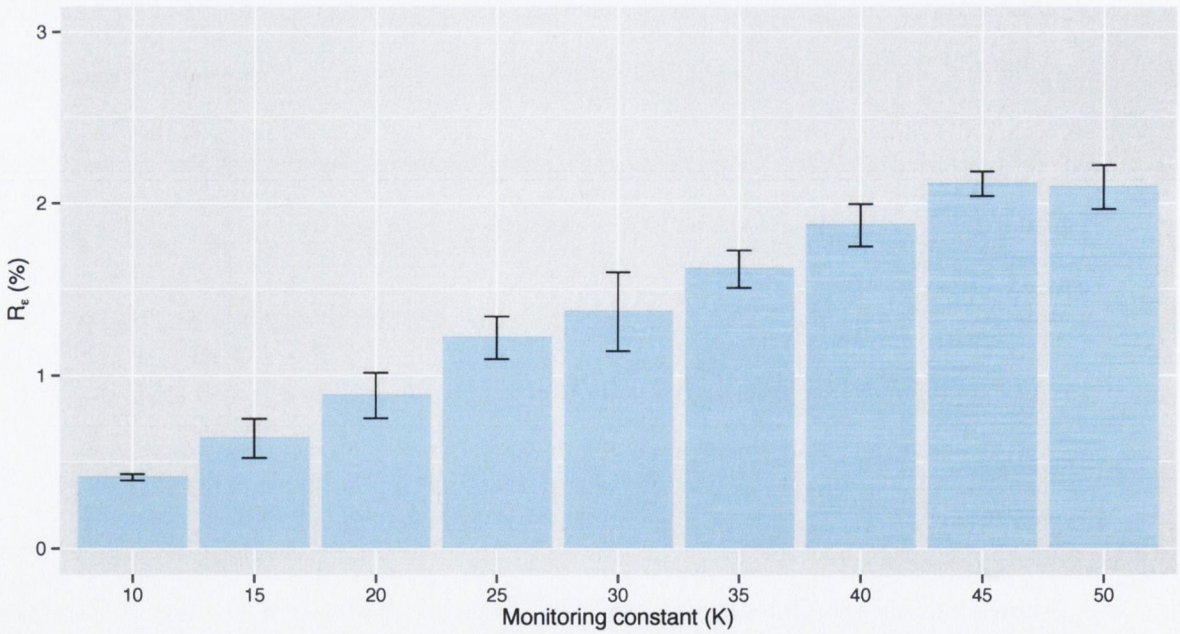



**Fig. 4.13**: The ratio of prediction errors over the network for varying $K$ values

it can be seen that the value of $R_t$ decreases as the time-varying values of $K$ increase, similar to fixed values of $K$. There is statistically no difference in the $R_t$ values for fixed and time-varying values of $K$. This is because the amount of monitored readings reduced during night time is equally increased during the day time thus having very little difference with the $R_t$ values when a fixed value of $K$ is used.

Figure 4.15 shows that the value of $R_\epsilon$ increases as the value of $K$ is increased both for the fixed and the time-varying values of $K$. The value of $R_\epsilon$ is lowered for the time-varying values of $K$ compared to the fixed values of $K$. The reduction in value of $R_\epsilon$ is statistically valid with a confidence of 95% as none of the CI bars overlap. This happens because the variation of change in the sensor data is higher during morning time when people present interact with the environment, such as opening and closing windows. Thus having a lower value of $K$ during the morning time enables better estimates of the error trends, leading to better monitoring of the performance of the prediction model, hence improving the value of $R_\epsilon$.

### 4.6.2 Changing values of monitoring constant based on distance

Similar to changing the values of $K$ based on time, the value of $K$ changes based on the distance between the sensor nodes, as discussed in Section 3.2.5.2. For this purpose, we classify the distance between nodes as 'close', 'far' and 'very far'. The nodes that are 'close' to each other will use a monitoring constant of double the $K$ value ($K_{close}$), for 'far' the monitoring constant used will be $K$ ($K_{far}$) and for nodes that are 'very far' the monitoring constant used will be half the $K$ value ($K_{veryFar}$). To determine whether the nodes are 'close', 'far' or 'very far' a coordinate system is used to specify the value of x and y in meters. Any two nodes that are within a distance of (5m, 12m) of each other are classified as 'close', any nodes within a distance of (10m, 24m) of each other are classified as 'far' and any two nodes that are further than (10m, 24m) of each other are classified as 'very far' in this experiment. The results of this experiment are shown in Figures 4.14 & 4.15.

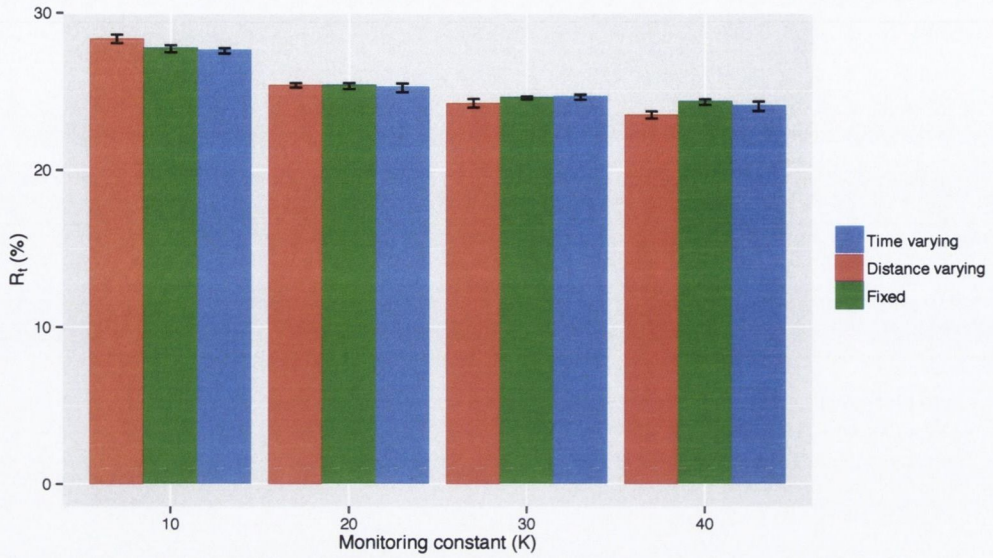



**Fig. 4.14**: The ratio of readings sensed and transmitted for fixed and changing $K$ values over time and distance

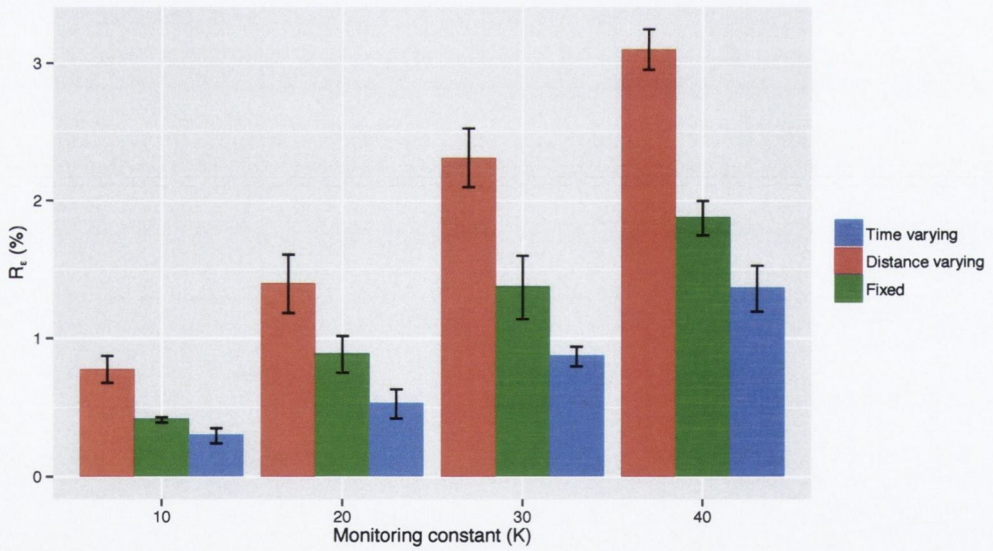



**Fig. 4.15**: The ratio of prediction errors over the network for fixed and changing $K$ values over time and distance

The nodes in the Intel Lab data set are stationary, the distance between the sensor nodes always stays the same. Thus the values $K_{close}$, $K_{far}$ and $K_{veryFar}$ assigned to the prediction models as a proportion of $K$ value depending on the distance between nodes stay the same for a given $K$ value throughout the lifetime of the WSN. As the $K$ value increases the $K_{close}$, $K_{far}$ and $K_{veryFar}$ values increases proportionally compared to the fixed and time-varying approaches. Hence with increasing $K$ values the $R_t$ value decreases more compared to the fixed and time-varying approaches as shown in Figure 4.14.

Figure 4.15 shows that the error values for changing the $K$ values based on distance are statistically higher than both the fixed and time-varying $K$ values. The likely reason is that distance is not a good metric to observe the variations in the Intel Lab data. Koushanfar et al. (2006) also pointed out that the relationship between the sensor readings of the sensor nodes does not depend on the distance between the sensor nodes in the Intel Lab dataset. From this experiment it can be concluded that varying $K$ values based on distance in the Intel Lab datset does not yield better results and should be avoided.

## 4.7 Effect of the sampling size ($\omega$)

The values of $\omega$ specifies how many readings will be collected during the monitoring phase. By changing the value of $\omega$ for different $K$ values we can answer the following question: Can collecting more sensor readings give better error trend estimates than monitoring frequently ? The results of the experiment varying the $\omega$ values and the $K$ values are shown in Figure 4.16 & 4.17.

Figure 4.16 shows that as the value of $K$ increases, the value of $R_t$ drops. The reason behind this is the same as presented in section 4.6. As expected, Figure 4.16 shows that more readings are obtained during the monitoring phase and the value of $R_t$ increases as the value of $\omega$ increases. In Figure 4.17 it can be seen that the lowest value of $R_\epsilon$

are obtained when the value of $K$ is low. Increasing $\omega$ values for small values of $K$ does not give any statistically different value of $R_\epsilon$. For higher values of $K$, a value of 1 for $\omega$ gives a statistically significant smallest value of $R_\epsilon$ compared to all other values of $\omega$. This is due to the fact that frequent new information is obtained when the value of $K$ is low and not much new information is obtained by getting more sensor readings

Most of the values of $R_\epsilon$ for different $\omega$ and the same values of $K$ show no statistical difference, in the case where the value of $K$ is 30 or 40 then an $\omega$ value of 1 is statistically different to the rest of the values of $R_\epsilon$. This value of 1 for $\omega$ results in the lowest $R_\epsilon$ value. Increasing the $\omega$ values does not improve the estimation of the error trends as not much new information about the trends of errors is obtained in consecutive readings. New information for estimating the error trends better is obtained by frequent monitoring (small $K$ values) rather than increasing the $\omega$ values. By getting frequent new information the error trends estimate future errors better, giving better estimates of trust in the models and hence lowering the value of $R_\epsilon$.

## 4.8 Effect of error threshold ($u_{error}$)

Different user requirements of the error thresholds have an impact on the overall efficiency of BLESS. An experiment with different error thresholds for temperature sensors is run with varying $u_{error}$ values. The result of this experiment is shown in Figures 4.18 & 4.19. As expected the lower the value of $u_{error}$ the higher are the values of $R_t$ and $R_\epsilon$. All the values of $R_t$ are statistically different with a 95% confidence, as none of the CI bars overlap each other. Also all the $R_\epsilon$ except (1.5, 2) are statistically different with a 95% confidence as analysed using the HSD technique. The reason why $R_t$ values decrease as the $u_{error}$ values increase is because the tolerable error by which the performance of the prediction model is measured is wider thus triggering the monitoring of idle sensor nodes less often. Similarly the wider the constraints of the prediction model the better its performance and hence the smaller the $R_\epsilon$ as the value of $u_{error}$ increases.

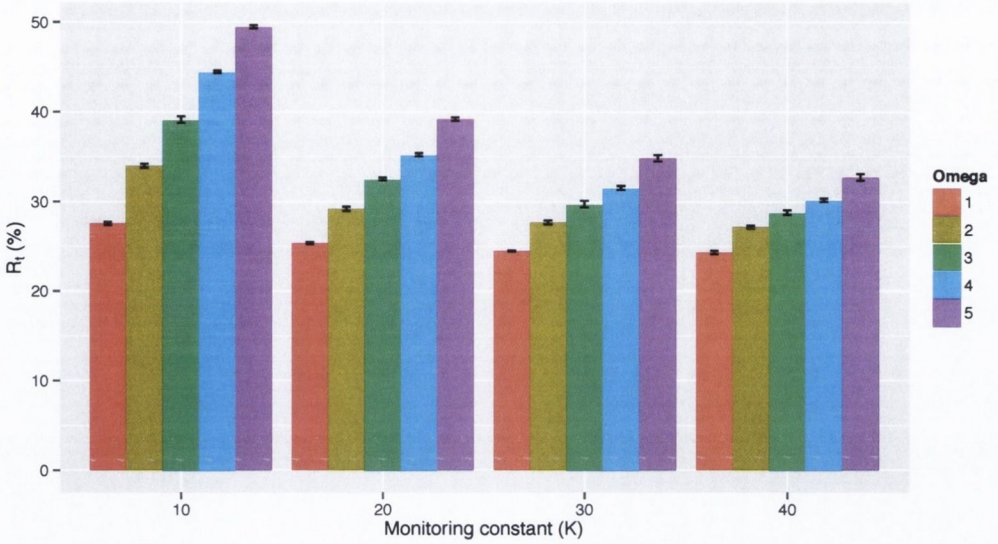



**Fig. 4.16**: The ratio of number of readings sensed and transmitted over the network for fixed and changing $\omega$ & $K$ values

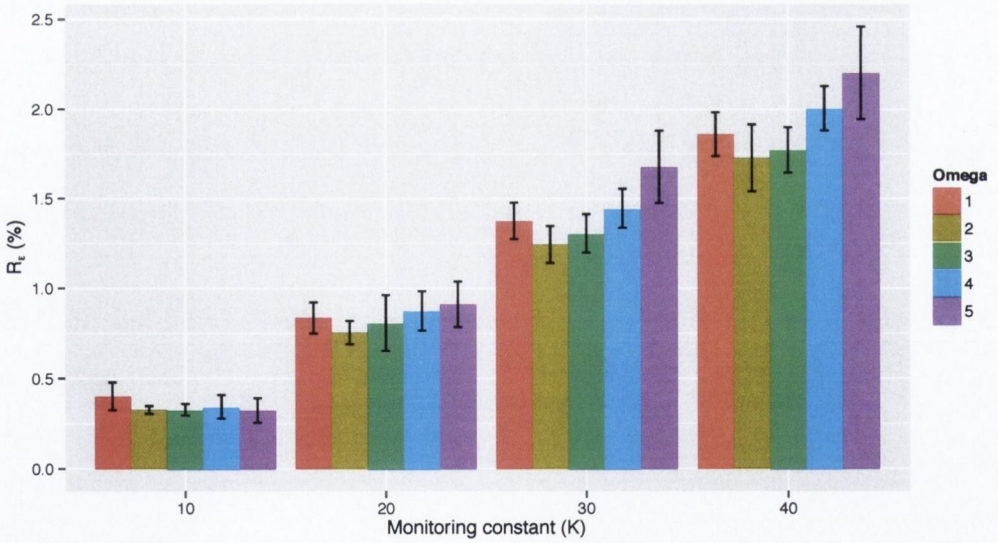



**Fig. 4.17**: The ratio of prediction errors over the network for fixed and changing $\omega$ & $K$ values

For higher value of $u_{error}$ it might be beneficial to use bigger values of $K$ for monitoring the sleeping nodes.

In the next section a discussion of all the parameter analysis is presented.

## 4.9 Parameter analysis discussion

The various parameters of the BLESS algorithm and their effects in saving energy and improving sensing fidelity on the temperature sensors of the Intel Lab dataset was evaluated. A summary of these effects is discussed below:

1. *Effect of the size of training data ($\alpha$):* From the results in Section 4.3, it can be concluded that training SLR models with a small training dataset and re-training them whenever possible can increase energy savings and sensing fidelity of WSN applications using a high sampling rate. The higher the sampling rate the better is the linear approximation leading to lower loss in sensing fidelity. The best results from using this prediction model are obtained when this model is trained using small $\alpha$ values of 5 or 10. These trained prediction models perform the best when they are re-trained frequently as the linearity relationship may not hold in the long run. BLESS takes the approach of re-training the prediction model between two sensor nodes whenever enough training data is collected either through monitoring of the sleeping nodes or through the sensor readings sensed and transmitted by the active nodes. It should be noted that when the performance of the prediction model is expected to be low (i.e., the trust is low), the sleeping nodes monitoring frequency increases, enabling quicker re-training of the prediction models. Finally it is shown that the lower the sampling rates, the lower the performance of BLESS is in predicting the sensor readings of sleeping nodes as the assumption of linearity does not hold well.

2. *Effect of the size of the validation data ($\beta$):* Changes in the relationship between sensor readings in most cases do not happen immediately after training and hence

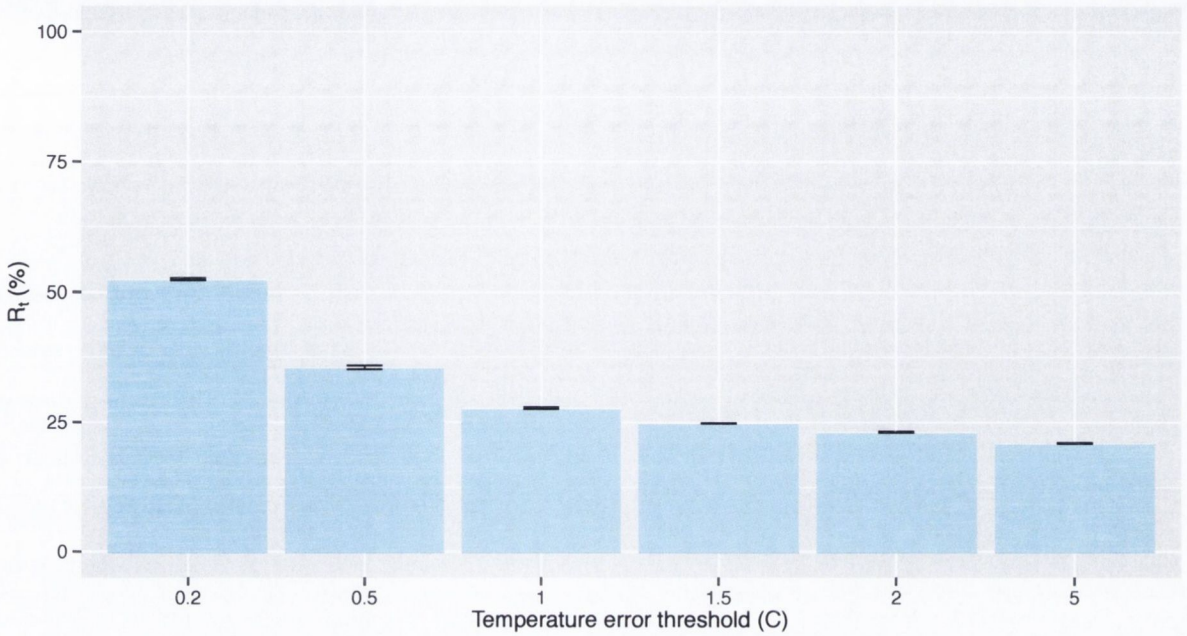



**Fig. 4.18**: The ratio of number of readings sensed and transmitted for varying $u_{error}$ values

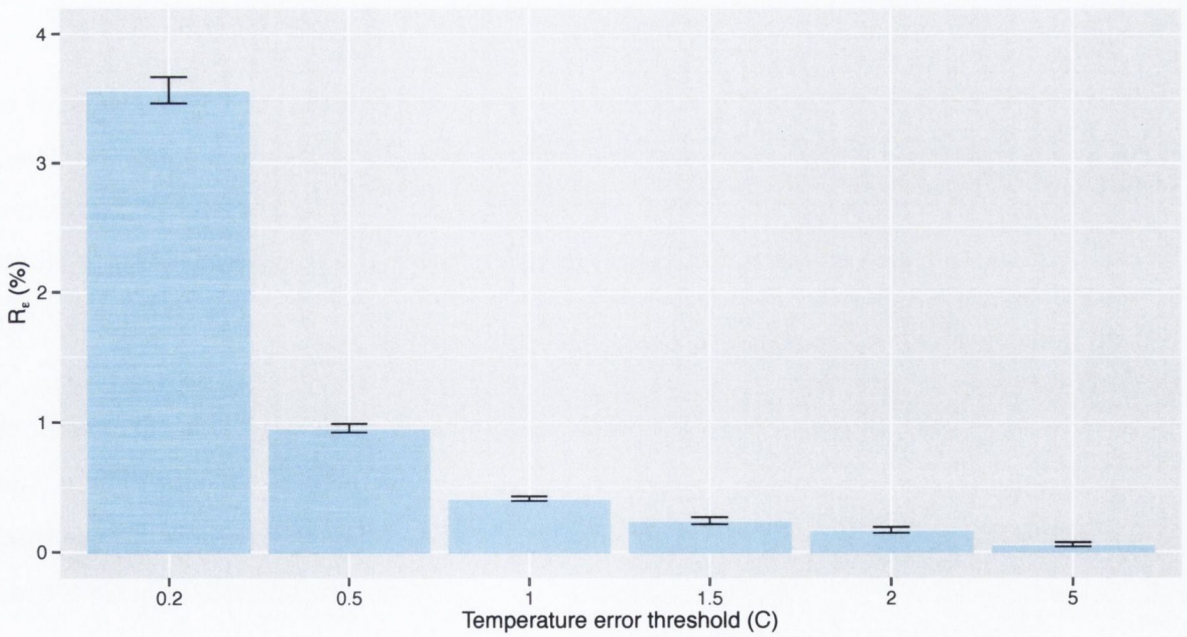



**Fig. 4.19**: The ratio of prediction errors over the network for varying $u_{error}$ values

using large validation datasets only consumes more energy. Thus small values of $\beta$ in the order of 2, 5 or 10, give the best results.

3. *Effect of the scheduling time ($\theta$):* To balance energy across the entire network, the $\theta$ value to use depends on the length of the deployment of the WSN. In general, small $\theta$ values do not give good performance because the trust of the sleeping nodes is updated only during re-training of the models and this re-training of the models happens only when sufficient monitored data is available. Sufficient monitored data is not available when a short time frame of $\theta$ is used. Thus when new sets of nodes are made active at the end of $\theta$ the prediction models used by these active nodes to predict the sleeping nodes might perform badly. Hence larger $\theta$ values are desirable. When the WSN deployment is short-term, large $\theta$ values will not be able to evenly balance out energy consumption between all sensor nodes. Thus $\theta$ values need to be chosen depending on the length of the WSN deployment. Longer the length the higher the $\theta$ values can be used, given that the active sensor nodes have enough energy to sense and transmit sensor readings for the assigned duration of $\theta$.

4. *Effect of the monitoring constant ($K$):* In general the smaller the value of $K$, the more frequently the sensor readings of sleeping nodes are monitored. This gives a better prediction estimate of the future errors when identifying the trend of the errors, thus giving a better estimate of the trust in the prediction model. The difference in values of $R_t$ when using small values of $K$ and large values of $K$ is not too high (about 4%) but there is a considerable difference in the value of $R_\epsilon$ (about 1.5% more errors are present in the predicted readings). This value of $K$ can be changed depending on a priori information on the deployments. These changing values of $K$ are evaluated for different times of day and distance for the Intel Lab data. For this data better performance is obtained when changing the value of $K$ for different times of the day.

5. *Effect of the sampling size ($\omega$):* The effect of $\omega$ is quite similar to the effect of $\beta$. Good estimates of trust are obtained when getting samples more frequently rather than collecting more samples during monitoring. Thus using smaller values of $\omega$ in the order of 1, saves energy without compromising the sensing fidelity.

6. *Effect of the error threshold ($u_{error}$):* The $u_{error}$ value varies depending on the application requirements. In general, the higher the values of $u_{error}$, the better the performance in saving more energy and reducing the $R_\epsilon$, because the prediction estimates of the error trends or the prediction models does not need to be very accurate.

It can thus be seen that the default parameter values assigned in Section 4.2, give the best results in saving energy and improving sensing fidelity on the Intel Lab dataset. So far all the experiments carried out on BLESS have been on the Intel Lab data set. The next section then evaluates the BLESS algorithm for improving energy savings and sensing fidelity on other WSN datasets, to show that BLESS can be applicable in other environmental monitoring WSN application involving continuous data collection.

## 4.10 BLESS and other datasets

To show that the BLESS algorithm can be applied to other environmental monitoring WSN applications for saving energy, BLESS is evaluated using datasets from the Sensor Scope dataset deployed by Guillermo et al. (2006) and the Tunnel dataset deployed by Ceriotti et al. (2011). To evaluate the BLESS algorithm, its parameters need to be chosen for these datasets. In the previous section the effect of the parameters on the Intel Lab dataset was discussed. It should be noted that these effects do not depend on the nature of the WSN deployment except for $\theta$ values. Assuming that the changes taking place in the environment are sporadic, using the same parameter values (except $\theta$) as defined in Section 4.2 for the BLESS algorithm should generally be able to improve

energy savings with little loss in sensing fidelity. Regarding the $\theta$ value, the default value of 4 hours used previously can be used since the deployment lasts only couple of days. Thus using these identified parameter values the BLESS algorithm is evaluated on the Sensor Scope and the Tunnel datasets and its results are explained below.

### 4.10.1 Sensor Scope dataset

The Sensor Scope data set consists of about 5 days of data. BLESS is evaluated on this 5 days of data for the ambient temperature sensor by varying the $u_{error}$ values. The results from these experiments are shown in Figures 4.20 and 4.21. In Figure 4.20, it can be seen that in the worst case about 60% of the sensing and transmission operations are reduced. In Figure, 4.21 the values of $R_\epsilon$ are higher than the evaluation of BLESS against the Intel Lab data set for the same sensor type. This is possibly because the value of $S_T$ used is 31 seconds in the Intel Lab deployment, which is much lower than the value of 2 minutes used as $S_T$ in the Sensor Scope deployment. The BLESS algorithm was previously evaluated on the Intel Lab dataset using an $S_T$ value of 2 minutes in Section 4.3.3 and those results are shown in Figures 4.5 and 4.6. By comparing these results with the Sensor Scope data it can be seen that for both the datasets the $R_\epsilon$ when using an $u_{error}$ value of $1°C$ is about 4%.

### 4.10.2 Tunnel dataset

BLESS is evaluated on the Tunnel data set collected for the first 3 days from all the sensor nodes. The reason why only 3 days of data was used is that there was no time synchronisation algorithm implemented in this application during its deployment time. During the initial days of deployment there is only a little loss of synchronisation, the maximum difference in time synchronisation between nodes in these initial 3 days of deployment is about 10 minutes. The difference in time lag between the sensor data collected gets higher as the days progress. To asses the effectiveness of BLESS, the $R_t$ and value of $R_\epsilon$ are measured for varying error threshold values while keeping the rest

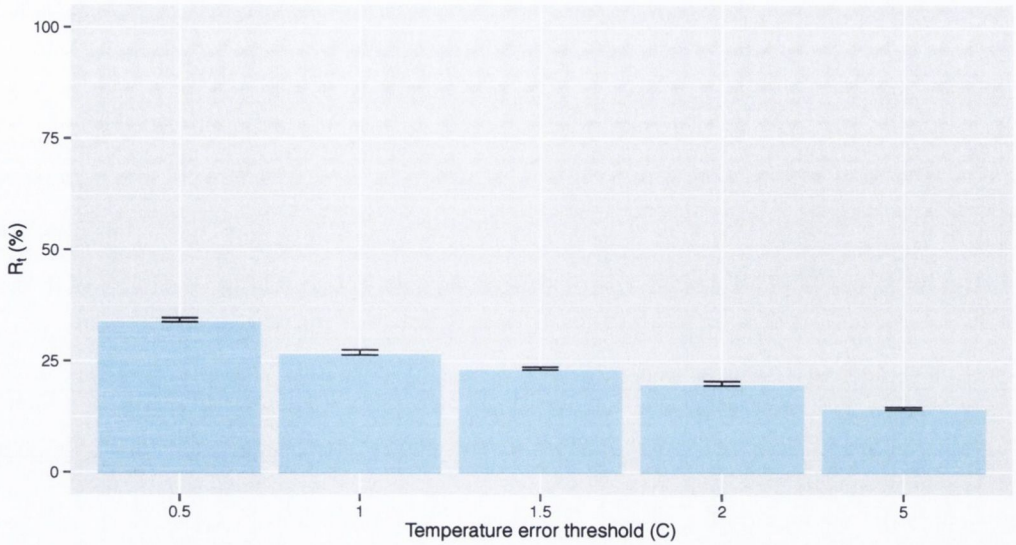



**Fig. 4.20**: The ratio of number of readings sensed and transmitted for the ambient temperature sensor in Sensor Scope data by varying $u_{error}$ values

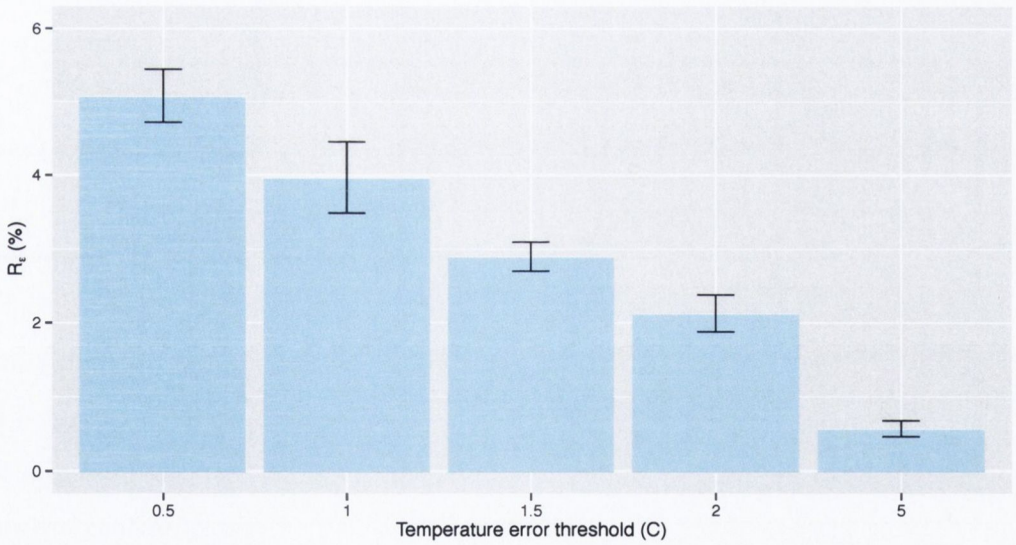



**Fig. 4.21**: The ratio of prediction errors for the ambient temperature sensor in Sensor Scope data with varying $u_{error}$ values

of the parameters values at the default values as explained in Section 4.2. It can be seen from Figure 4.22 that about 70% to 75% of transmission and sensing operations are saved thus saving energy. In Figure 4.23, the value of $R_\epsilon$ reduces as the tolerable error threshold values increase. The highest $R_\epsilon$ value is about 2%, and it is expected that the errors can be further lowered if the sensor readings of the nodes were time synchronised.

From evaluating the BLESS algorithm with other datasets, deployed by Guillermo et al. (2006) and Ceriotti et al. (2011), it can be concluded that the BLESS algorithm can be used to save energy in other WSN environmental monitoring applications with very little loss in sensing fidelity. So far, it is assumed that only a single sensor is present in a sensor node, in the next section the BLESS approach, for multiple sensors present in a single sensor node is evaluated.

## 4.11 Dealing with multiple sensors in a node

Section 3.2.7 describes two solutions for dealing with multiple sensors in nodes. In the following experiment, an assumption is made that sensing consumes more energy than transmission and hence the solution pertaining to this assumption is analysed. The experiment is conducted on 52 sensor nodes of Intel Lab data (2 sensors were faulty and are excluded) with only temperature and humidity sensors and varying the $u_{error}$ values. The light sensors in the Intel Lab dataset are omitted because there are lot of missing sensor readings.

The results obtained for the 52 sensor nodes with temperature and humidity sensors are shown in Figures 4.24 & 4.25. More than 75% energy saving is obtained for all of the $u_{error}$ values considered. Comparing this with using the temperature sensor for 10 sensor nodes shown in Figure 4.24, it can be seen that higher energy savings are achieved, this is because more sensor readings were being predicted using BLESS. In Figure 4.25, it can be seen that the values of $R_\epsilon$ are slightly higher when compared to the experiment results with temperature sensors for 10 sensor nodes in Figure 4.19. This shows that as

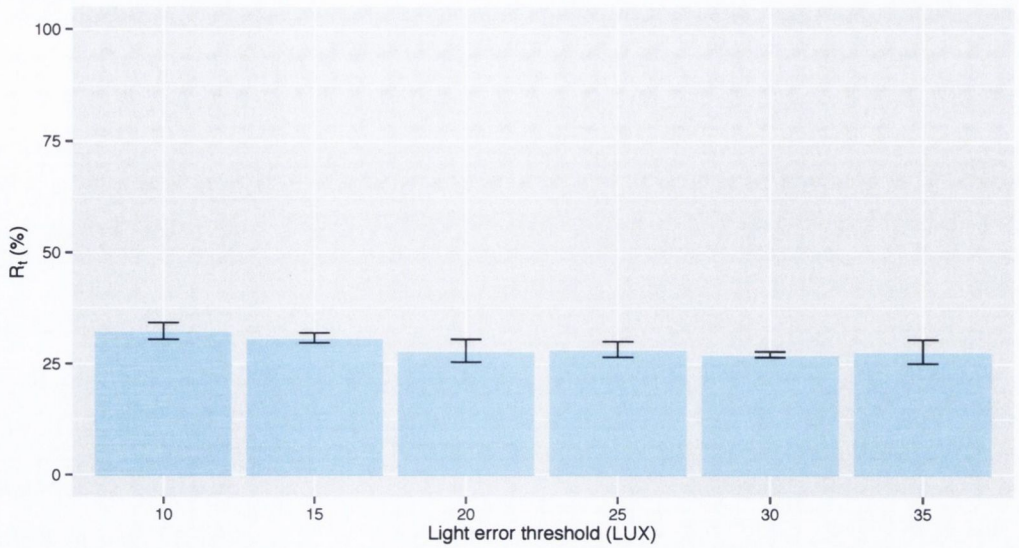



**Fig. 4.22**: The ratio of number of readings sensed and transmitted for the ambient temperature sensor in Sensor Scope data by varying $u_{error}$ values

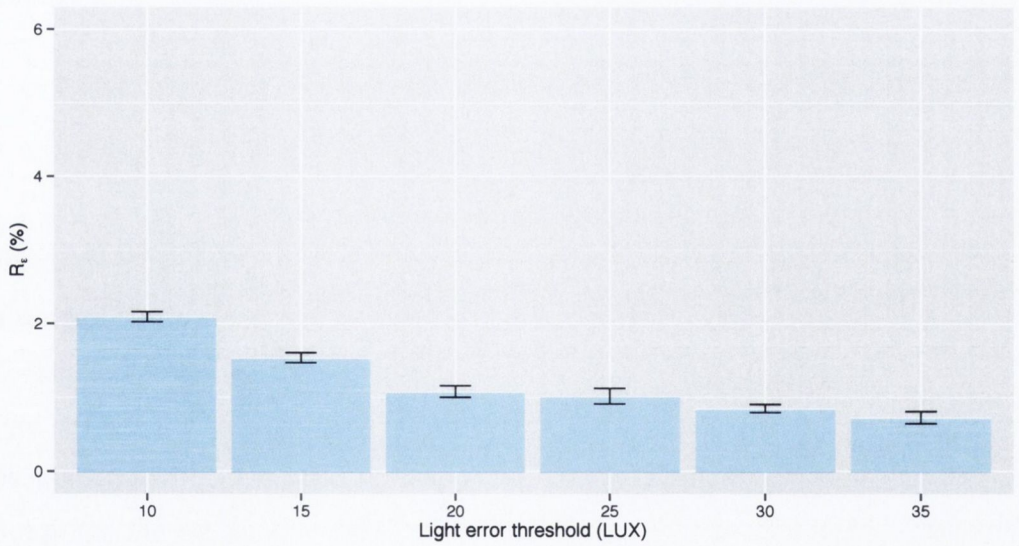



**Fig. 4.23**: The ratio of prediction errors for the ambient temperature sensor in Sensor Scope data with varying $u_{error}$ values

the number of sensor nodes or the number of sensors in the nodes increases, the more sensor readings can possibly be predicted. This enables putting more sensors to sleep thus decreasing the sensing and transmission operations considerably but with only a little increase in the number of predicted errors.

## 4.12 Comparing BLESS with (Koushanfar et al., 2006)

The closest competitor for BLESS is the work done by Koushanfar et al. (2006) as explained in Section 2.5.2, and their published results are compared in this section. Koushanfar et al. (2006) evaluate their algorithm with temperature and humidity sensors on the Intel Lab dataset with 10 days worth of sensor readings. In order to compare the BLESS algorithm with Koushanfar et al. (2006) results, the BLESS algorithm is applied on 10 days worth of readings collected using temperature and humidity sensor data in the Intel Lab dataset. Since multiple sensors are used, the algorithm proposed in Section 3.2.7.2, is used with an assumption that transmission operations consume less energy than the sensing operations. The results comparison can be further split into comparing the $R_T$ and $R_\epsilon$ values.

### 4.12.1 Comparing $R_T$ values

To initially train and validate the prediction models, Koushanfar et al. (2006) use an $(\alpha + \beta)$ value of 5760 (two days worth of sensor readings); depending on the value of the user-specified error threshold ($u_{error}$) the number of sensor readings collected for monitoring and re-training the prediction models vary. The lower the value of $u_{error}$ used, the higher the number of sensor readings collected by Koushanfar et al. (2006). The total number of all these readings is the overhead incurred in their approach. Apart from this overhead, the active node sets identified need to transmit their sensor readings at an interval of 31 seconds. In this section, it is shown that the BLESS algorithm outperforms the approach proposed by Koushanfar et al. (2006) in terms of reducing

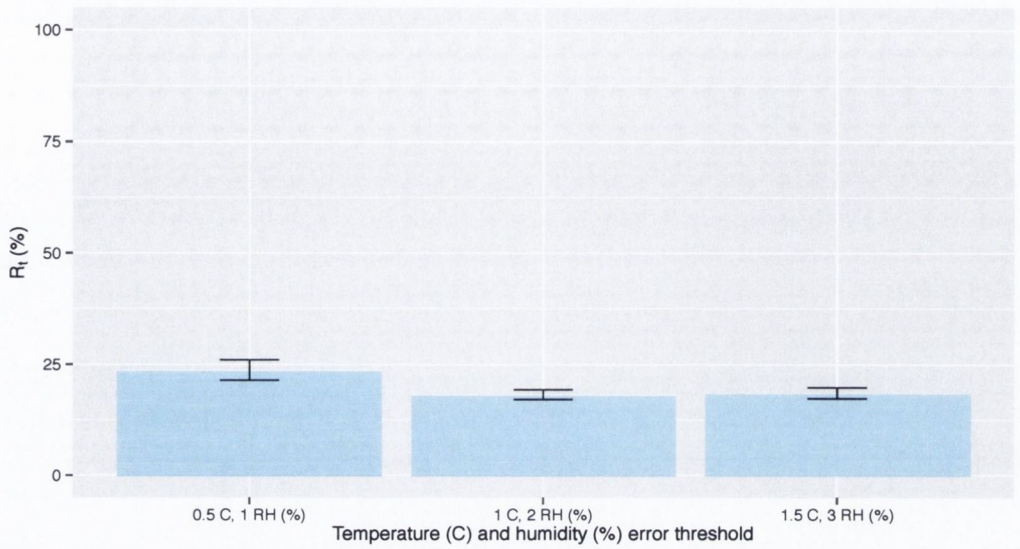



**Fig. 4.24**: The ratio of number of readings sensed and transmitted over the 52 sensor node network for relative humidity(RH) and temperature sensors varying $u_{error}$ values

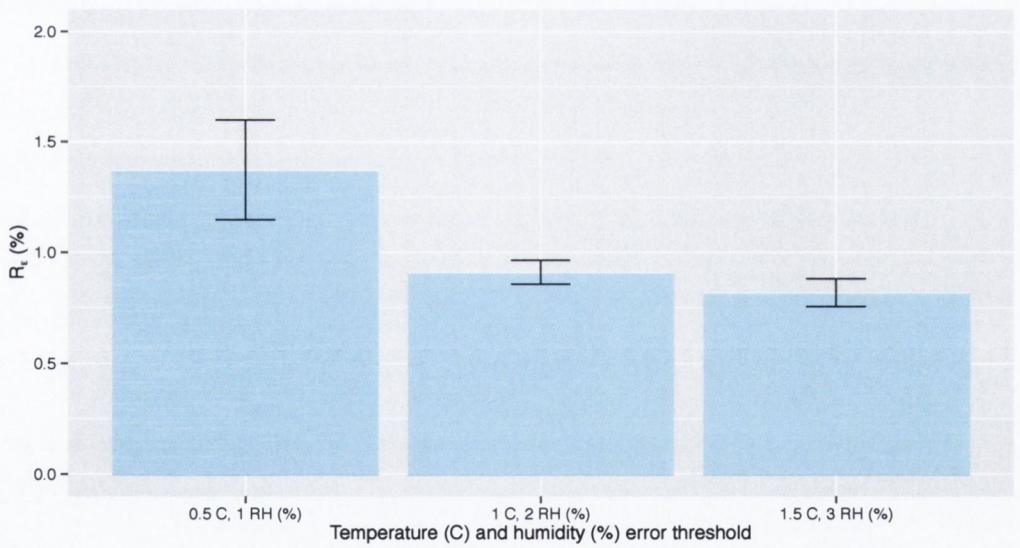



**Fig. 4.25**: The ratio of prediction errors over the 52 sensor network for relative humidity (RH) and temperature sensors varying $u_{error}$ values

the total sensing and transmitting operations. This is done by comparing the total sensed and transmitted readings in BLESS against the overhead incurred in the approach proposed by Koushanfar et al. (2006) as shown in Figure 4.26. BLESS performs better than the approach proposed by Koushanfar et al. (2006) because:

1. The size of the training/re-training dataset used in BLESS is very small compared to the large training dataset of 5760 sensor readings. The prediction models trained had to be re-trained again at day 9 showing that changes in the relationships happen. BLESS is built on the principle that changes in the environment affecting the performance of the predication models are inevitable and thus caters to changes more energy efficiently than Koushanfar et al. (2006).

2. Predicability between sensor readings is not a constant and changes over time. For example, in the case of the Intel Lab dataset, more sensor readings can be estimated during night time when less variations happen compared to morning time. Koushanfar et al. (2006) do not deal with changes in predictability until the prediction estimates of one of the sensor readings is above $u_{error}$, when sensor readings from all nodes are collected to re-train the prediction models, leading to higher values of $R_T$. The BLESS algorithm on the other hand adapts to these changes in predictability and turns on/off the sensor nodes in the WSN giving higher energy savings than (Koushanfar et al., 2006).

### 4.12.2 Comparing $R_{error}$ values

In Figure 4.27, the number of errors propagated in both these approaches are compared. In the approach proposed by Koushanfar et al. (2006) no errors occur. This is because the monitoring of the sensor readings is done only at times when high prediction errors have occurred in the validation dataset. This is not a generalised solution as it is not known for how long the prediction models will follow a similar mechanism of the one identified during the validation phase. In their case, it has worked for their 10 days

of tested data, it is unclear whether it will continue to hold true after re-training of prediction models that takes place on the 9th day and in the future. Comparing BLESS against the work of Koushanfar et al. (2006), the total errors propagated is still only about 1.3% for the lowest $u_{error}$ value. This could be further improved by lowering the value of monitoring constant $K$ but this will increase the $R_t$ value.

## 4.13 Summary

This chapter provides a comprehensive evaluation of the BLESS algorithm using three real-world WSN datasets. First the various parameters of the BLESS algorithm are evaluated on the Intel Lab dataset. Based on the evaluation of the various parameters, appropriate parameter values are chosen to run on the other two data sets, namely Sensor Scope deployed by Guillermo et al. (2006) and the Tunnel dataset deployed by Ceriotti et al. (2011). The results of this evaluation shows that savings of about 60% in sensing and transmission operations with a $R_\epsilon$ value of about 4% in the worst case scenario is observed ($u_{error}$ values of $0.2°C$ and 10 LUX are used for Sensor Scope and the Tunnel dataset respectively). These experiments also show that the Sensor Scope dataset yields slightly higher values of $R_\epsilon$ compared to the Intel Lab dataset, this is because the sampling rate of sensors for the Sensor Scope deployment is higher. Experimenting with the Tunnel dataset also yields slightly higher values of $R_\epsilon$ and it is expected to be lower if a time synchronisation algorithm was used in their WSN deployment. These evaluations shows that the BLESS algorithm can be applied to other environmental monitoring WSN applications to save energy with a little loss in sensing fidelity.

An experiment with multiple sensor nodes was also carried out with an assumption that the sensing operations are more energy consuming than the transmit operations (Deshpande et al., 2004). This experiment is evaluated on the data collected by the temperature and the humidity sensors from the Intel Lab deployment, also showing about 75% reduction in sensing and transmission operations with only a $R_\epsilon$ value of

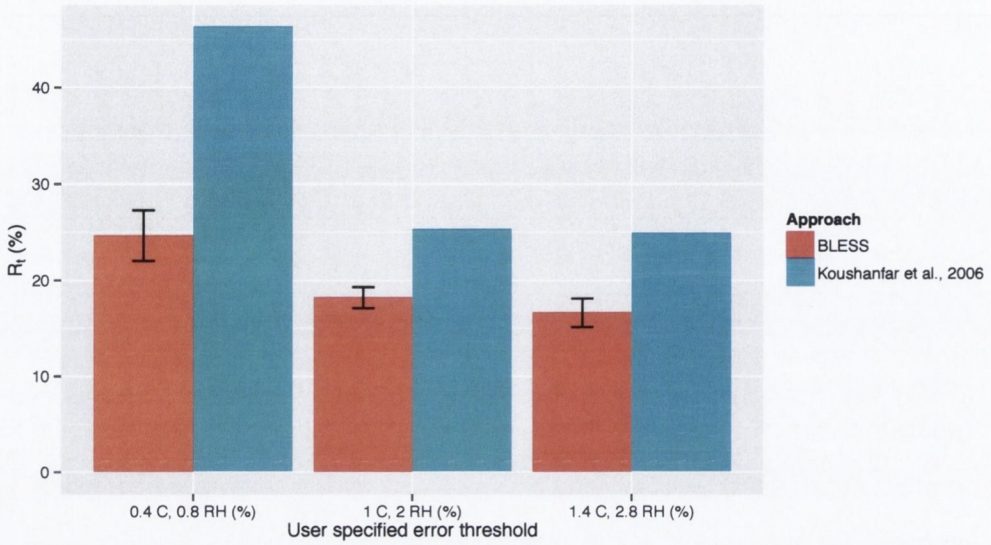



**Fig. 4.26**: The ratio of number of readings sensed and transmitted for the BLESS and Koushanfar et al. (2006)'s approaches, under varying $u_{error}$ values

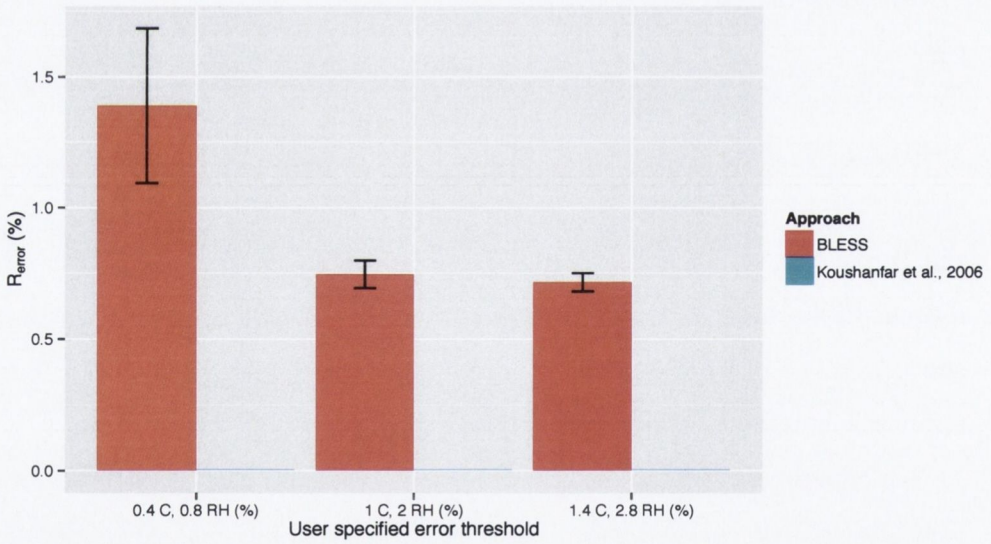



**Fig. 4.27**: The ratio of errors propagated for the BLESS and Koushanfar et al. (2006)'s approaches, under varying $u_{error}$ values

about 1.5% in the worst case scenario ($u_{error}$ values for temperature: $0.5°C$, humidity: 1%). Finally, BLESS is compared against the work of Koushanfar et al. (2006) since Section 2.5.2 shows that the work of Koushanfar et al. (2006) is our closest competitor. The evaluation shows that the work of Koushanfar et al. (2006) performs slightly better in terms of $R_{error}$ by about 1.4% in the worst case ($u_{error}$ values for temperature: $0.4°C$, humidity: 0.8%). However, BLESS outperforms the work of Koushanfar et al. (2006) in terms of the number of packets sensed and transmitted by more than 25% in the worst case scenario ($u_{error}$ values for temperature: $0.4°C$, humidity: 0.8%). This is done by comparing just the training and monitoring overhead of the approach proposed by Koushanfar et al. (2006) against the BLESS algorithm. In the work of Koushanfar et al. (2006), overhead includes for all the sensing and transmitting operations in training, monitoring and re-training the model in their 10 days of deployment. It should be noted that in the overhead, the count for sensing and transmission operations required by the active nodes is not included.

From these evaluations, it is shown that by providing a solution that addresses the issues mentioned in section 2.5.1, sensing and transmission operations can be reduced with no significant loss in sensing fidelity, in environmental monitoring WSN applications. It should be noted that as discussed in Section 1.2, reduction in sensing and transmission operations can result in saving energy in WSNs. In Figure 2.10 of Chapter 2, we expected BLESS to reduce the number of sensing operations ($R_t$ values) with few prediction errors that are not within the user-specified threshold ($R_{error}$ values). From the evaluations, it can be confirmed that $R_{error}$ value is about 3% (note the figure in 4.19 does not show $R_{error}$ values) and the reduction in the sensing operation is about 50% in the worst case scenario for temperature sensors in the Intel lab data with $u_{error}$ values of $0.2°C$. When comparing these results with the approaches in the Figure 2.10, it can be said that the number of sensing operations is less than Koushanfar et al. (2006)'s approach but greater than Chatterjea and Havinga (2008)'s approach with a small $R_{error}$ value. Thus, the expected results of BLESS are very similar to Figure 2.10, except for

a bit higher sensing operations and thus we re-draw the relative comparison chart in Figure 4.28.

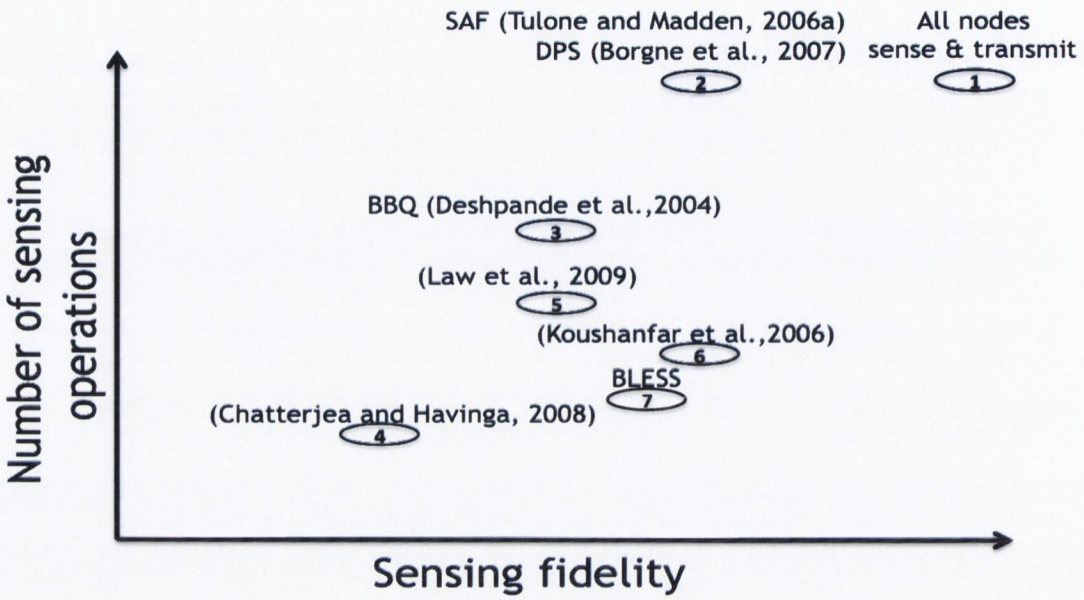



**Fig. 4.28**: Relative comparison of model-based approaches who have experimented on the Intel lab dataset (Peter et al., 2004)

# Chapter 5

# Conclusion and Future Work

In this chapter a summary of the thesis and its contributions are provided along with a discussion of possible future work.

## 5.1 Thesis summary and contributions

This thesis presents the Less Battery (BLESS) algorithm, for saving energy in a WSN used for environmental monitoring applications requiring frequent and continuous sampling of environmental phenomena. The BLESS algorithm saves energy by putting a set of sensor nodes to sleep while predicting estimates of their sensor readings using prediction models and the sensor readings of a set of active nodes.

Chapter 1 provided an overview of WSNs and environmental monitoring applications of WSNs. It then described prediction models and how they can be used to save energy in WSNs. Issues around changes happening in the environment affecting the performance of the prediction model were discussed. The chapter then motivated the need for a better approach to deal with changes affecting the performance of the prediction model. A brief description of the BLESS algorithm, which can effectively deal with monitoring and changes affecting the performance of the prediction model by identifying trust in the prediction model, was provided. Finally, the contributions of the thesis were presented.

In Chapter 2, the various approaches that can reduce energy consumption in WSNs were discussed by classifying them into approaches that reduce transmit, sensing and receive operations of sensor nodes. It can be seen that reducing all these operations reduces energy consumption, but Deshpande et al. (2004) show that even simple sensors such as temperature and humidity sensors can consume more energy than transmission and receive operations. Hence, this thesis focused on reducing the sensing operations. It should be noted that by reducing the sensing operations, the transmission operations are also reduced. This chapter identified that model-based approaches for reducing sensing operations provide a general solution compared to other approaches to reducing sensing operations. Though the model-based approaches provide general solutions, some issues arise when changes in the environment affect the performance of the prediction model, these issues were identified and are:

1. Choosing an appropriate model representation, and training/re-training this model with minimum number of sensing operations to minimise energy consumption as a result.

2. The performance of the models need to be tracked by monitoring the sleeping sensor nodes adaptively to improve energy saving and sensing fidelity.

3. Different sensor nodes' readings are predictable by different nodes' readings and this can change as the environment undergoes changes. This change in predictability between sensor readings of nodes need to be identified to improve energy saving and sensing fidelity.

Chapter 3 proposed the design of the BLESS algorithm that overcomes the issues identified in Chapter 2. The solutions to these issues are the contributions of this thesis, and they are as follows:

1. *Model representation and training:* BLESS uses simple models with small training data and frequent re-training to reduce the number of sensing operations whilst also

trying to maintain the user-specified accuracy. In more detail, BLESS is designed to use simple linear regression (SLR) models as the prediction model, with an assumption that the relationship between sensor readings can be approximated by linear functions when the sampling rate of the sensors is high. As changes in the environment happen continually, the SLR model might be valid only for a small time window to predict estimates of sensor readings within an user-specified threshold and so the SLR model is trained/re-trained with a small training dataset so that the rest of the time window can be used to exploit the trained model to save energy and reduce the energy consumption required to train and re-train the model.

2. *Tracking performance of the models adaptively:* As mentioned in the previous point, changes in the environment continually happen affecting the validity of the SLR models to accurately predict estimates of sensor readings within the user-specified threshold. In order to identify the validity of the SLR models, the performance of the models need to be tracked to improve the sensing fidelity. BLESS proposed to track the performance of the models by monitoring the sleeping nodes whose readings are predicted by the SLR models. If the frequency of monitoring the sleeping nodes is high, the number of sensing operation increases. If the frequency of monitoring the sleeping nodes is low, the prediction estimates of sensor readings might not be within the user-specified threshold. To prevent this issue, the monitoring of the sleeping nodes is done based on the trust identified in the prediction models the lower the trust, the more frequently sleeping nodes are monitored. This trust in the model estimates the near future performance of the prediction model by identifying a trend in the errors of the SLR models. The trust is updated every time monitoring of the sleeping nodes take place. Thus the sleeping nodes are monitored more frequently only when the performance of the models is expected to be low.

3. *Dealing with changes in predictability between nodes:* Changes in the predictability between the nodes can happen as changes take place in the environment. Using the trust identified in the models the changes in predictability between nodes can be identified. If the trust in the model is low for a period of $\tau$ then it can be said that the predictability of the nodes represented by the model is changed and their readings are no longer predictable. Re-training of the models is done every time the nodes representing the model can collect $(\alpha + \beta)$ number of sensor readings either during the active or monitoring phase. By re-training it can be identified if predictability between nodes has changed such that is their readings are predictable now. When the trust in the model is low, re-training happens more quickly to identify changes in the predictability of the sensor readings. Thus BLESS can adapt quickly to identify changes in predictability between nodes.

In Chapter 4, the BLESS algorithm designed in Chapter 3 was evaluated using data collected from real-world WSN deployments used for environmental monitoring. First, the effects of the various parameters of the BLESS algorithm were evaluated on one of the WSN datasets. Based on the analysis of these effects, parameter values are chosen for the BLESS algorithms to be evaluated on the remaining datasets. The results of this evaluation shows that savings of about 60% in sensing and transmission operations with a $R_\epsilon$ value of about 4% in the worst case scenario is observed ($u_{error}$ values of $0.2°C$ and 10 LUX are used for Sensor Scope and the Tunnel dataset respectively). Next the BLESS algorithm was evaluated for sensor nodes containing multiple sensors with an assumption that the sensing operations consume more energy than transmit operations. Finally, the BLESS algorithm was compared against the approach proposed by Koushanfar et al. (2006) to show that Koushanfar et al. (2006) performs slightly better in terms of $R_{error}$ by about 1.4% in the worst case ($u_{error}$ values for temperature: $0.4°C$, humidity: 0.8%). However BLESS outperforms the work of Koushanfar et al. (2006) in terms of the number of packets sensed and transmitted by more than 25% in

the worst case scenario. These evaluations show that:

1. When using SLR models, using small training data with frequent re-training gives better savings in energy whilst improving sensing fidelity, compared to using large training/re-traning data.

2. Monitoring the sleeping sensor nodes more frequently when the performance of the prediction model is expected to be low, improves energy savings and sensing fidelity.

3. Identifying changes in predictability using the trust identified in the models and re-training the models whenever possible improves energy savings and sensing fidelity. Re-training more frequently during low trust values in the prediction models enables quick adaptation to changes in the models further improving energy savings and sensing fidelity.

## 5.2 Future work

When designing and evaluating BLESS, we have identified several areas where BLESS's performance and applicability could be extended and identified a number of areas for potential future research. We outline these below.

One of the assumptions of BLESS is that the sampling rate of the sensors need to be high, in order to use SLR as the prediction model. There are applications that use low sampling rates such as the Glacsweb deployment by Martinez et al. (2004), where the sensors are sampled every 4 hours. Identification of suitable prediction models for such applications are needed. Currently, we propose to use weighted SLR models for identifying the error trends to estimate future prediction errors. Possible next steps to improve the accuracy of this weighted SLR model in predicting estimates of future errors are: 1) Different weighing schemes need to be evaluated and identified for different application scenarios and 2) an evaluation of identifying the trends using different prediction models,

to identify suitable predictions models for different application scenarios. In this thesis two solutions are provided in BLESS for dealing with multiple sensor nodes, only one of these solutions has been implemented. The other approach will be implemented in the future. In the case that these multiple sensors have different energy characteristics, these two solutions provided need to be extended to include this information.

The classic idea of having static sensor nodes in a WSN is being replaced by mobile sensor nodes and an extension of BLESS that works in combination with mobile devices is a necessary step forward for saving energy for future WSN deployments. In the BLESS approach, if the sensor readings of an active node are missed, the prediction estimates of the sleeping sensor nodes using the prediction model cannot be obtained. An interesting solution could be is use the dual prediction scheme to predict estimates of the active nodes sensor readings, thereby also decreasing the transmission operations of the active nodes. Currently, the sending and the receiving node are assumed to be tightly time synchronised, but this is not the case. The on-demand approaches seem a promising solution to improve synchronisation between sensor nodes. So, a possible step forward could be is to deploy a real WSN using on-demand approaches for reducing the number of receive operations.

# Appendix A

# Linear approximation

We wish to formally demonstrate that a twice differentiable continuous function $f(x)$ can be approximated with a number of line segments, and as the length of each segment reduces, the approximate error approaches to zero.

Firstly, for any interval $[a, b]$ in the domain of $f(x)$, we can draw a line segment $p(x)$ connecting points $(a, f(a))$ and $(b, f(b))$. The following results have been established (Parnell, 2013):

For any $x \in [a, b]$, there exists $\mu_x \in [a, b]$ such that

$$f(x) - p(x) = \frac{1}{2}(x - a)(x - b)f''(\mu_x),$$

Indeed, this can be viewed as the error term if we use the line segment $p(x)$ to approximate $f(x)$. The maximum absolute error within interval $[a, b]$ can be written as

$$\begin{aligned} |f(x) - p(x)| &\leq \frac{1}{2}|x - b||x - a| \max_{\mu \in [a,b]} f''(\mu) \\ &\leq \frac{(b-a)^2}{4} \max_{\mu \in [a,b]} f''(\mu) \end{aligned}$$

The overall cumulative error for all $x \in [a, b]$ is

$$\int_a^b |f(x) - p(x)|dx \leq \int_a^b \frac{(b-a)^2}{4} \max_{\mu \in [a,b]} f''(\mu)dx = \frac{(b-a)^3}{4} \max_{\mu \in [a,b]} f''(\mu)$$

That means, as the interval reduces i.e. $h = b - a \to 0$, the cumulative error of the linear approximation within the interval diminishes.

Secondly, we approximate $f(x)$ with uniformly spaced segments with interval $h = b - a$. Without loss of generality, let $[X_s, X_e]$ denote the domain of $f(x)$. The overall error of the approximation is upper bounded by

$$\sum_{i=1}^{\lceil (X_e - X_s)/h \rceil} \frac{h^3}{4} \max_{\mu \in [h(i-1), hi]} f''(\mu) = \frac{h^2}{4} \max_{\mu \in [X_s, X_e]} f''(\mu)$$

Apparently, as $h \to 0$, the overall error of the approximate approaches zero. Hence the general continuous function $f(x)$ can be represented by line segments, which motivates us to adopt linear regression model over nonlinear models.

# Bibliography

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422.

Akyildiz, I. F., Melodia, T., and Chowdhury, K. R. (2007). A survey on wireless multimedia sensor networks. *Computer Networks*, 51(4):921 – 960.

Alippi, C., Anastasi, G., Di Francesco, M., and Roveri, M. (2010). An adaptive sampling algorithm for effective energy management in wireless sensor networks with energy-hungry sensors. *IEEE Transactions on Instrumentation and Measurement*, 59(2):335–344.

Alippi, C., Anastasi, G., Galperti, C., Mancini, F., and Roveri, M. (2007). Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, 2007. MASS 2007*, pages 1 –6.

Amar, A., Leshem, A., and Gastpar, M. (2010a). A greedy approach to the distributed karhunen-loève transform. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2970–2973.

Amar, A., Leshem, A., and Gastpar, M. (2010b). Recursive implementation of the distributed karhunen-loève transform. *IEEE Transactions on Signal Processing*, 58(10):5320–5330.

Anastasi, G., Conti, M., Di Francesco, M., and Passarella, A. (2006). An adaptive and low-latency power management protocol for wireless sensor networks. In *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*, MobiWac '06, pages 67–74, New York, NY, USA. ACM.

Anastasi, G., Conti, M., Di Francesco, M., and Passarella, A. (2009). Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537–568.

Arisha, K., Youssef, M., and Younis, M. (2002). Energy-aware tdma-based mac for sensor networks. In Karri, R. and Goodman, D., editors, *System-Level Power Optimization for Wireless Multimedia Communication*, pages 21–40. Springer US.

Bogena, H., Herbst, M., Huisman, J., Rosenbaum, U., Weuthen, A., and Vereecken, H. (2010). Potential of wireless sensor networks for measuring soil water content variability. *Vadose Zone J.*, 9(4):1002–1013.

Borbash, S. and Jennings, E. (2002). Distributed topology control algorithm for multihop wireless networks. In *Proceedings of the 2002 International Joint Conference on Neural Networks, 2002.*, volume 1, pages 355–360.

Borgne, Y.-A. L., Santini, S., and Bontempi, G. (2007). Adaptive model selection for time series prediction in wireless sensor networks. *Signal Processing*, 87(12):3010 – 3020.

Cao, Q., Abdelzaher, T., He, T., and Stankovic, J. (2005). Towards optimal sleep scheduling in sensor networks for rare-event detection. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, IPSN '05.

Ceriotti, M., Corra, M., D'Orazio, L., Doriguzzi, R., Facchin, D., Guna, S., Jesi, G., Lo Cigno, R., Mottola, L., Murphy, A., Pescalli, M., Picco, G., Pregnolato, D., and Torghele, C. (2011). Is there light at the ends of the tunnel? wireless sensor networks

for adaptive lighting in road tunnels. In *10th International Conference on Information Processing in Sensor Networks (IPSN), 2011*, pages 187–198.

Chatterjea, S. and Havinga, P. (2008). An adaptive and autonomous sensor sampling frequency control scheme for energy-efficient data acquisition in wireless sensor networks. In *Distributed Computing in Sensor Systems*, volume 5067, pages 60–78. Springer Berlin Heidelberg.

Chatterjea, S., Van Hoesel, L. F. W., and Havinga, P. J. M. (2004). Ai-lmac: an adaptive, information-centric and lightweight mac protocol for wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004.*, pages 381–388.

Chib, S. and Greenberg, E. (1994). Bayes inference in regression models with arma (p, q) errors. *Journal of Econometrics*, 64(1-2):183–206.

Chintalapudi, K. K. and Venkatraman, L. (2008). On the design of mac protocols for low-latency hard real-time discrete control applications over 802.15.4 hardware. In *Proceedings of the 7th international conference on Information processing in sensor networks*, IPSN '08, pages 356–367, Washington, DC, USA. IEEE Computer Society.

Chipcon (2013). Cc2420 datasheet. `http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf`.

Chou, J., Petrovic, D., and Ramachandran, K. (2003). A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks. In *Proceedings of INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*, volume 2, pages 1054–1062.

Chu, D., Deshpande, A., Hellerstein, J. M., and Hong, W. (2006). Approximate data collection in sensor networks using probabilistic models. In *Proceedings of the 22nd International Conference on Data Engineering*, pages 48–60.

Ciancio, A. and Ortega, A. (2004). A distributed wavelet compression algorithm for wireless sensor networks using lifting. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004*, volume 4, pages iv–633–iv–636 vol.4.

Ciancio, A. and Ortega, A. (2005). A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005*, volume 4, pages iv/825–iv/828 Vol. 4.

Crossbow (2013). Telosb data sheet. `http://www.willow.co.uk/TelosB_Datasheet.pdf`.

De Boor, C., De Boor, C., De Boor, C., and De Boor, C. (1978). *A practical guide to splines*, volume 27. Springer-Verlag New York.

Deo, N. (2004). *Graph theory with applications to engineering and computer science.* PHI Learning Pvt. Ltd.

Deshpande, A., Guestrin, C., Madden, S. R., Hellerstein, J. M., and Hong, W. (2004). Model-driven data acquisition in sensor networks. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 588–599. VLDB Endowment.

Donoho, D. (2006). Compressed sensing. *IEEE Transactions on Information Theory, 2006*, 52(4):1289–1306.

Ergen, S. and Varaiya, P. (2006). Pedamacs: power efficient and delay aware medium access protocol for sensor networks. *IEEE Transactions on Mobile Computing*, 5(7):920–930.

Fasolo, E., Rossi, M., Widmer, J., and Zorzi, M. (2007). In-network aggregation tech-

niques for wireless sensor networks: a survey. *Wireless Communications, IEEE*, 14(2):70 –87.

Gedik, B., Liu, L., and Yu, P. S. (2007). Asap: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 18:1766–1783.

Gibbons, A. (1985). *Algorithmic graph theory.* Press syndicate of the University of Cambridge.

Goel, S. and Imielinski, T. (2001). Prediction-based monitoring in sensor networks: taking lessons from mpeg. *SIGCOMM Comput. Commun. Rev.*, 31(5):82–98.

Grigg, O., Farewell, V., and Spiegelhalter, D. (2003). Use of risk-adjusted cusum and rsprtcharts for monitoring in medical contexts. *Statistical Methods in Medical Research*, 12(2):147–170.

Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., and Madden, S. (2004). Distributed regression: an efficient framework for modeling sensor network data. In *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 1–10.

Guillermo, B., Olivier, C., Thierry, B., and Davis, D. (2006). Patrouille des glaciers deployment. `http://lcav.epfl.ch/page-86035-en.html`.

Györfi, L. (2002). *A distribution-free theory of nonparametric regression.* Springer.

Hambeck, C., Mahlknecht, S., and Herndl, T. (2011). A 2.4 w wake-up receiver for wireless sensor nodes with -71dbm sensitivity. In *Proceedings of the 2011 IEEE International Symposium on Circuits and Systems*, pages 534–537.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *Linear Methods for Regression.* Springer.

Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Hornik, K. (2013). The R FAQ. `http://CRAN.R-project.org/doc/FAQ/R-FAQ.html`.

Huang, Z., Shen, C.-C., Srisathapornphat, C., and Jaikaeo, C. (2002). Topology control for ad hoc networks with directional antennas. In *Proceedings of the Eleventh International Conference on Computer Communications and Networks, 2002*, pages 16–21.

IEEE (2007). *Approved Draft Amendment to IEEE Standard for Information technology-Telecommunications and information exchange between systems-PART 15.4:Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs): Amendment to add alternate PHY (Amendment of IEEE Std 802.15.4)*.

Jain, A., Chang, E. Y., and Wang, Y.-F. (2004). Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 11–22, New York, NY, USA. ACM.

Jelicic, V., Magno, M., Brunelli, D., Bilas, V., and Benini, L. (2012). Analytic comparison of wake-up receivers for wsns and benefits over the wake-on radio scheme. In *Proceedings of the 7th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, PM2HW2N '12, pages 99–106, New York, NY, USA. ACM.

Kansal, A., Hsu, J., Zahedi, S., and Srivastava, M. B. (2007). Power management in energy harvesting sensor networks. *ACM Trans. Embed. Comput. Syst.*, 6(4):32–es.

Keshavarzian, A., Lee, H., and Venkatraman, L. (2006). Wakeup scheduling in wireless sensor networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, pages 322–333, New York, NY, USA. ACM.

Kho, J., Rogers, A., and Jennings, N. R. (2009). Decentralized control of adaptive sampling in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):19:1–19:35.

Kimura, N. and Latifi, S. (2005). A survey on data compression in wireless sensor networks. In *International Conference on Information Technology: Coding and Computing, 2005*, volume 2, pages 8–13.

Koushanfar, F., Taft, N., and Potkonjak, M. (2006). Sleeping coordination for comprehensive sensing using isotonic regression and domatic partitions. In *Proceedings of the 25th IEEE International Conference on Computer Communications. INFOCOM 2006*, pages 1 –13.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Law, Y. W., Chatterjea, S., Jin, J., Hanselmann, T., and Palaniswami, M. (2009). Energy-efficient data acquisition by adaptive sampling for wireless sensor networks. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, IWCMC '09, pages 1146–1151, New York, NY, USA. ACM.

Li, M., Ganesan, D., and Shenoy, P. (2009). Presto: feedback-driven data management in sensor networks. *IEEE/ACM Transactions on Networking*, 17(4):1256–1269.

Li, N., Hou, J., and Sha, L. (2003). Design and analysis of an mst-based topology control algorithm. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1702–1712.

Li, N. and Hou, J. C. (2004). Flss: a fault-tolerant topology control algorithm for wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, MobiCom '04, pages 275–286, New York, NY, USA. ACM.

Li, Y., Ye, W., and Heidemann, J. (2005). Energy and latency control in low duty cycle mac protocols. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 2, pages 676–682.

Lin, S., Arai, B., Gunopulos, D., and Das, G. (2008). Region sampling: Continuous adaptive sampling on sensor networks. In *IEEE 24th International Conference on Data Engineering, 2008. ICDE 2008*, pages 794–803.

Liu, J. and Li, B. (2002). Mobilegrid: capacity-aware topology control in mobile ad hoc networks. In *Proceedings of the Eleventh International Conference on Computer Communications and Networks, 2002.*, pages 570–574.

Lu, G., Krishnamachari, B., and Raghavendra, C. (2004). An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004.*, pages 224–232.

Luo, C., Wu, F., Sun, J., and Chen, C. W. (2009). Compressive data gathering for large-scale wireless sensor networks. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, MobiCom '09, pages 145–156.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2002). Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97.

Marcelloni, F. and Vecchio, M. (2009). An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. *The Computer Journal*, 52(8):969–987.

Marcelloni, F. and Vecchio, M. (2010). Enabling energy-efficient and lossy-aware data compression in wireless sensor networks by multi-objective evolutionary optimization. *Information Sciences.*, 180(10):1924–1941.

Maróti, M., Kusy, B., Simon, G., and Lédeczi, A. (2004). The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 39–49, New York, NY, USA. ACM.

Martinez, K., Ong, R., and Hart, J. (2004). Glacsweb: a sensor network for hostile environments. In *Proceedings of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004.*, pages 81–87.

Mullins, E. (2003). *Statistics for the quality control chemistry laboratory.* The Royal Society of Chemistry.

Nath, S., Gibbons, P. B., Seshan, S., and Anderson, Z. R. (2004). Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 250–262.

Niyato, D., Hossain, E., Rashid, M., and Bhargava, V. (2007). Wireless sensor networks with energy harvesting technologies: a game-theoretic approach to optimal energy management. *Wireless Communications, IEEE*, 14(4):90–96.

Padhy, P., Dash, R. K., Martinez, K., and Jennings, N. R. (2006). A utility-based sensing and communication model for a glacial sensor network. In *Proceedings of the*

*fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pages 1353–1360.

Padhy, P., Dash, R. K., Martinez, K., and Jennings, N. R. (2010). A utility-based adaptive sensing and multihop communication protocol for wireless sensor networks. *ACM Transactions on Sensor Networks*, 6:1–39.

Parnell, C. (2013). An introduction to the approximation of functions. `http://www-solar.mcs.st-andrews.ac.uk/~clare/Lectures/num-analysis/Numan_chap3.pdf`.

Paruchuri, V., Basavaraju, S., Durresi, A., Kannan, R., and Iyengar, S. (2004). Random asynchronous wakeup protocol for sensor networks. In *Proceedings of the First International Conference on Broadband Networks, 2004.*, pages 710–717.

Penrose, M. D. (1999). A strong law for the largest nearest-neighbour link between random points. *Journal of the London Mathematical Society*, 60:951–960.

Peter, B., Wei, H., Carlos, G., Sam, M., Mark, P., and Romain, T. (2004). Intel berkley research lab deployment. `http://db.csail.mit.edu/labdata/labdata.html`.

Quer, G., Masiero, R., Munaretto, D., Rossi, M., Widmer, J., and Zorzi, M. (2009). On the interplay between routing and signal representation for compressive sensing in wireless sensor networks. In *Information Theory and Applications Workshop, 2009*, pages 206–215.

Raghunathan, V., Ganeriwal, S., and Srivastava, M. (2006). Emerging techniques for long lived wireless sensor networks. *Communications Magazine, IEEE*, 44(4):108 – 114.

Raghunathan, V., Schurgers, C., Park, S., and Srivastava, M. (2002). Energy-aware wireless microsensor networks. *Signal Processing Magazine, IEEE*, 19(2):40 –50.

Ramanathan, R. and Rosales-Hain, R. (2000). Topology control of multihop wireless networks using transmit power adjustment. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2000.*, volume 2, pages 404–413.

Ramaswamy, S., Viswanatha, K., Saxena, A., and Rose, K. (2010). Towards large scale distributed coding. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010*, pages 1326–1329.

Raza, U., Camerra, A., Murphy, A., Palpanas, T., and Picco, G. (2012). What does model-driven data acquisition really achieve in wireless sensor networks? In *IEEE International Conference on Pervasive Computing and Communications (PerCom), 2012*, pages 85–94.

Rezayi, E. and Abolhassani, B. (2009). Multirate distributed source coding in wireless sensor network usind ldpc codes. In *Proceedings of Canadian Conference on Electrical and Computer Engineering, 2009.*, pages 171–174.

RForge (2013). rjava. `http://www.rforge.net/rJava/index.html`.

Roedig, U., Barroso, A., and Sreenan, C. (2004). Determination of aggregation points in wireless sensor networks. In *Euromicro Conference, 2004. Proceedings. 30th*, pages 503–510.

Sadler, C. M. and Martonosi, M. (2006). Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 265–278.

Samudrala, R. and Moult, J. (1998). A graph-theoretic algorithm for comparative modeling of protein structure. *Journal of Molecular Biology*, 279(1):287 – 302.

Sánchez, M., Manzoni, P., and Haas, Z. (1999). Determination of critical transmission range in ad-hoc networks. In Biglieri, E., Fratta, L., and Jabbari, B., editors, *Multiac-*

*cess, Mobility and Teletraffic in Wireless Communications: Volume 4*, pages 293–304. Springer US.

Santi, P. (2005). Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv.*, 37(2):164–194.

Santi, P. and Blough, D. (2002). An evaluation of connectivity in mobile wireless ad hoc networks. In *Proceedings of the International Conference on Dependable Systems and Networks, 2002. DSN 2002.*, pages 89–98.

Santi, P. and Blough, D. (2003). The critical transmitting range for connectivity in sparse wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):25–39.

Santini, S. (2009). *Adaptive sensor selection algorithms for wireless sensor networks.* PhD thesis, ETH Zurich.

Shih, W.-C., Jurdak, R., Lee, B.-H., and Abbott, D. (2011). High sensitivity wake-up radio using spreading codes: design, evaluation, and applications. *EURASIP Journal on Wireless Communications and Networking*, 2011(1):1–14.

Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(1):pp. 1–52.

Slepian, D. and Wolf, J. (1973). Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19(4):471–480.

Srisooksai, T., Keamarungsi, K., Lamsrichan, P., and Araki, K. (2012). Practical data compression in wireless sensor networks : A survey. *Journal of Network and Computer Applications*, 35(1):37 – 59.

Suriyachai, P., Roedig, U., and Scott, A. (2012). A survey of mac protocols for mission-critical applications in wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 14(2):240–264.

Tang, L., Sun, Y., Gurewitz, O., and Johnson, D. (2011). Pw-mac: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communications,2011*, pages 1305–1313.

Tulone, D. and Madden, S. (2006a). An energy-efficient querying framework in sensor networks for detecting node similarities. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, MSWiM '06, pages 191–300, New York, NY, USA. ACM.

Tulone, D. and Madden, S. (2006b). Paq: Time series forecasting for approximate query answering in sensor networks. In Römer, K., Karl, H., and Mattern, F., editors, *Wireless Sensor Networks*, volume 3868 of *Lecture Notes in Computer Science*, pages 21–37. Springer Berlin / Heidelberg.

Van Dam, T. and Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 171–180.

Van Langevelde, R., Van Elzakker, M., van Goor, D., Termeer, H., Moss, J., and Davie, A. J. (2009). An ultra-low-power 868/915 mhz rf transceiver for wireless sensor network applications. In *Radio Frequency Integrated Circuits Symposium, 2009. RFIC 2009. IEEE*, pages 113–116.

Watkinson, J. (1999). *MPEG-2*. Local Press.

Wattenhofer, R., Li, L., Bahl, P., and Wang, Y.-M. (2001). Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proceedings of the IEEE Computer and Communications Societies. Proceedings, 2001.*, volume 3, pages 1388–1397 vol.3.

Willett, R., Martin, A., and Nowak, R. (2004). Backcasting: adaptive sampling for

sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 124–133.

Willett, R. and Nowak, R. (2003). Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging. *IEEE Transactions on Medical Imaging*, 22(3):332–350.

Xue, F. and Kumar, P. R. (2004). The number of neighbors needed for connectivity of wireless networks. *Wirel. Netw.*, 10(2):169–181.

Yao, Y. and Gehrke, J. (2002). The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18.

Ye, W., Heidemann, J., and Estrin, D. (2004). Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330.

Yoon, S. and Shahabi, C. (2007). The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM Trans. Sen. Netw.*, 3:3–42.

Y.Yao and Gehrke, J. (2003). Query processing for sensor networks. *Proceedings of ACM Conference on Innovative Data Systems Research.*

Zheng, R., Hou, J. C., and Sha, L. (2003). Asynchronous wakeup for ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '03, pages 35–45.