DOCTOR OF PHILOSOPHY

Model-based transmission reduction and virtual sensing in Wireless Sensor Networks

Goldsmith, Daniel

Award date:
2013

Awarding institution:
Coventry University

Model-based Transmission Reduction and Virtual Sensing in Wireless Sensor Networks

Daniel Goldsmith

A thesis submitted in partial fulfilment
of the University's requirements for the degree of
Doctor of Philosophy

April 2013

Coventry University
Faculty of Engineering and Computing

Dedicated to the memory of R.F. Cox.
Who gave me a great start in the world of 'puters

Abstract

This thesis examines the use of modelling approaches in Wireless Sensor Networks (WSNs) at node and sink to: reduce the amount of data that needs to be transmitted by each node and estimate sensor readings for locations where no data is available.

First, to contextualise the contributions in this thesis, a framework for WSN monitoring applications (FieldMAP) is proposed. FieldMAP provides a structure for developing monitoring applications that advocates the use of modelling to improve the informational output of WSNs and goes beyond the *sense-and-send* approach commonly found in current, fielded WSN applications. Rather than report raw sensor readings, FieldMAP advocates the use of a *state vector* to encapsulate the state of the phenomena sensed by the node.

Second, the Spanish Inquisition Protocol (SIP) is presented. SIP reduces the amount of data that a sensor node must transmit by combining model-based filtering with Dual-Prediction approaches. SIP makes use of the state vector component of FieldMAP to form a simple predictive model that allows the sink to estimate sensor readings without requiring regular updates from the node. Transmissions are only made when the node detects that the predictive model no longer matches the evolving data stream. SIP is shown to produce up to a 99% reduction in the number of samples that require transmission on certain data sets using a simple linear approach and consistently outperforms comparable algorithms when used to compress the same data streams. Furthermore, the relationship between the user-specified error threshold and number of transmissions required to reconstruct a data set is explored, and a method to estimate the number of transmissions required to reconstruct the data stream at a given error threshold is proposed. When multiple parameters are sensed by a node, SIP allows them to be combined into a single state vector. This is demonstrated to further reduce the number of model updates required compared to processing each sensor stream individually.

Third, a sink-based, on-line mechanism to impute missing sensor values and predict future readings from sensor nodes is developed and evaluated in the context of an on-line monitoring system for a Water Distribution System (WDS). The mechanism is based on a machine learning approach called Gaussian Process Regression (GPR), and is implemented such that it can exploit correlations between nodes in the network to improve predictions. An on-line windowing algorithm deals with data arriving out of order and provides a feedback mechanism to predict values when data is not received in a timely manner.

A novel approach to create *virtual sensors* that allows a data stream to be predicted where no physical sensor is permanently deployed is developed from the on-line GPR mechanism.

The use of correlation in prediction is shown to improve the accuracy of predicted data from 1.55 Pounds per Square Inch (PSI) Root Mean Squared Error (RMSE) to 0.01 PSI RMSE. In-situ evaluation of the Virtual Sensors approach over 36 days showed that an accuracy of ±0.75 PSI was maintained.

The protocols developed in this thesis present an opportunity to improve the output of environmental monitoring applications. By improving energy consumption, long-lived networks that collect detailed data are made possible. Furthermore, the utility of the data collected by these networks is increased by using it to improve coverage over areas where measurements are not taken or available.

Acknowledgements

I would like to begin by thanking my Director of Studies, Dr James Brusey, and supervisors Professor Elena Gaura and Dr James Shuttleworth for the support, feedback and encouragement they have given me.

As my Director of Studies, James has been the one to ask the "hard" questions and has helped me focus on the research questions. James' feedback on the structure of the thesis and his hard work in proofreading has been invaluable.

I would especially like to thank Elena for giving me the opportunity to get into research and for building such a great team within the Cogent Computing Applied Research Centre. The high quality of research by the team at Cogent is a constant inspiration and a true reflection of the value of your leadership.

During the course of my research I have had the opportunity to work with two groups of wonderful people: My fellow research students at Cogent and the guys at SMART in Singapore. I don't have the space to name you all, but be assured that the good times we have shared and the discussions on a range of topics (not all research based) have taught me a lot and helped me get though the tough times. A special mention to Dr Mike Allen for making me feel at home in Singapore and for your help in proofreading the final Thesis.

Finally, I would like thank my wife Claire for all her support, love and kindness. You have put up with my moods when the work got me down and shared in my joy when experimentation worked well. Looking forward to not studying for the first time since we got married all those years ago.

Contents

List of Figures

List of Tables

Acronyms

WSN	Wireless Sensor Network	115
RMSE	Root Mean Squared Error	103
SQL	Structured Query Language	24
RAM	Random Access Memory	15
PDF	Probability Density Function	25
BSN	Body Sensor Network	44
ALT	Accuracy, Lifetime and Timeliness	19
ADC	Analog to Digital Converter	59
SIP	Spanish Inquisition Protocol	115
KF	Kalman Filter	60
EWMA	Exponentially Weighted Moving Average	55
LMS	Least Mean Squares Adaptive Filter	117
NLMS	Normalised Least Mean Squares Adaptive Filter	59
DPS	Dual Prediction System	116
DKF	Dual Kalman Filter	64
MCU	Microcontroller Unit	3
RLE	Run Length Encoding	26
PMC	Poor Man's Compression	62
PCA	Piece-wise Constant Approximation	52
PLA	Piece-wise Linear Approximation	117
EKF	Extended Kalman Filter	60
NDBC	National Data Buoy Centre	64
USAC	Utility based Sensing And Communication protocol	32
PAQ	Probabilistic Adaptable Query system	29
SAF	Similarity-based Adaptive Framework	78
CUSUM	Cumulative Sum	31
GPR	Gaussian Process Regression	115
GP	Gaussian Process	91
WDS	Water Distribution System	115

SMART	Singapore MIT Alliance for Research and Technology	88
NTU	Nanyang Technological University	88
PUB	Singapore Public Utilities Board	88
PSI	Pounds per Square Inch	90
ARIMA	Autoregressive Integrated Moving Average	116
AR	Autoregressive Models	29

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) present a unique opportunity to gather detailed information on environmental phenomena and improve knowledge of the world around us. However, a key factor when designing a sensor network system is the limited energy budget and the trade-off between the coverage of the data and the lifetime of the network.

This thesis focuses on the use of data modelling techniques at both node and sink to improve sensor network lifetime and the resolution of the data collected.

A generalised framework for WSN monitoring applications, *FieldMAP*, is developed to aid in the WSN design process and to situate the contributions in the rest of this thesis. FieldMAP advocates the use of in-network processing of raw sensor data into contextually relevant information. A key concept in the framework is the use of a *state vector* that encapsulates the sensed data into a representation of the state of the phenomena. By capturing such information, the state vector begins the transformation from raw *sensor data* to *user-relevant information* and thus increases the informational output of the WSN

The Spanish Inquisition Protocol (SIP) is an instantiation of the model component of FieldMAP is described and evaluated. SIP improves on prior work that makes use of a predictive model to reduce the amount of data that needs to be transmitted through the network. The state vector provides a simple predictive model that allows readings to be estimated while supporting event detection and maintaining the timeliness of the data. This model allows the user to make a trade off between the accuracy of the reported data and the number of samples that require transmission. The algorithm is shown to produce up to a 99% reduction in the number of packets transmitted in exchange for a small reduction in accuracy in the reported data stream (for example, 0.5 ℃ in temperature data). SIP has been shown to consistently outperform existing work when used to evaluate environmental data.

While SIP makes use of simple linear models on the node, a second approach makes use of a more complex machine learning technique that exploits the sink's global view of the data. *Virtual Sensors* aim to reduce the number of missing samples in the data reported at the node by estimating missing values in the data stream. The accuracy of this approach is increased by exploiting correlations between other nodes in the deployment, and is shown to reduce the Root Mean Squared Error (RMSE) of estimated sensor readings in data collected by a Water Distribution System (WDS) from 1.55 Pounds per Square Inch (PSI) to 0.01 PSI, when using well matched data sets. Given that a robust prediction mechanism can allow missing sensor readings to be estimated for an extended period of time, this approach is extended to create virtual sensors that can estimate readings at points where no permanent node is deployed. Virtual Sensors were evaluated in a real world deployment in a WDS in Singapore. After 36 days without input, the virtual sensor had maintained an accuracy of 0.75 PSI, compared to the actual sensor reading.

The rest of this chapter is organised as follows: First, the motivation for the work is presented along with the benefits that could be obtained by using such modelling approaches in WSN applications. Next, the research questions addressed by this thesis are stated. The approach to research is discussed, followed by a list of the contributions to knowledge. Next, A list of publications that have resulted from the work in this thesis is given. Finally the thesis structure is presented and contributed work is acknowledged.

1.1 Motivation

Measurement of environmental parameters has been used to help understand the world around us. Early measurement techniques involved manual observations and simple mechanical instrumentation. These techniques were often labour intensive and allowed manual measurement for a limited range of phenomena. Improving technology has allowed a wider range of phenomena to be measured over a wider area, in a less labour intensive way and at a higher resolution. One such technological advance has been the development of WSNs.

A WSN consists of miniature, self powered, computing devices (*nodes*) with sensing and communication capabilities that can autonomously gather data about a phenomena. Each node will periodically gather a sample from its attached sensors and transmit this data back to a central point (or *sink*) for further analysis and processing. While networks that make use of multiple sinks have been proposed, such approaches are rare in real world environmental monitoring applications.

The ability to deploy a robust sensing system without requiring a large amount of infrastructure makes WSN an ideal tool for collecting environmental data. Such sensing systems have made it possible to collect data from environments as diverse as volcanoes [105], forests [99] and glaciers [68].

However, a potential problem with the use of WSN for environmental monitoring is that their resources are strictly limited. Electronic devices require energy to operate and unless power harvesting approaches are used, this is typically provided by batteries, which can only supply a finite amount of energy. Often, long term operation will require battery replacement, which may not be possible, or where it is possible can be costly and time consuming.

An efficient sensor network will use the minimum amount of energy to collect and disseminate an accurate representation of the phenomena it is being used to measure. The need to provide a timely, accurate representation of the data while simultaneously meeting the demands of an application requires careful management of the energy budget. Often a trade-off is made between the accuracy, timeliness and resolution of the data collected and the lifetime of the node.

All components attached to a node (such as the Microcontroller Unit (MCU) or attached sensors) will consume part of the energy budget. Usually the radio unit has the highest energy demand, in some cases requiring more than ten times the energy to operate the MCU. The Shimmer devices, for example, typically consume 0.1 mA for the CPU, 1 mA during ADC conversions, and up to 20 mA for the 802.15.4 radio [90]. This means that while collecting and processing data is relatively cheap, transmitting this raw data back to a central *sink* for processing will quickly consume a node's energy resources and reduce the effective lifetime of the network. There have been many approaches proposed to address power consumption in WSNs including hardware optimisation, MAC layer and routing protocols, and application level data driven algorithms. Algorithmic approaches such as the work proposed here are appealing as they can complement other approaches and give the potential for further energy saving. Therefore, use of a modelling component to reduce the amount of data that needs transmission is advantageous. If the system is showing *normal* behaviour then the model will be able to adequately estimate the sensor readings and no transmission will be made. However, in the case of unexpected data, the model will not be able to accurately predict these values triggering a transmission to the sink. Thus any data stream events due to a change in behaviour will be captured and transmitted to the sink.

In some situations, failures in the network can mean that samples are missing from the collected data. These missing samples can make analysis of the monitored phenomena difficult, as a full picture of the is required for an accurate understanding of the data. Furthermore, the output of a sensor network system may be used as input to other applications to assist in making maintenance decisions. If these higher level applications are sensitive to missing sensor values, then this can have a detrimental effect on any decision made.

Regression analysis can be used to estimate these missing values and increase the resolution of the data stream. However, using traditional prediction techniques, it can sometimes be difficult to determine the underlying model for complex data sets. Machine learning algorithms are often used to estimate missing values in these situations. However, these techniques are still limited in their ability to correctly estimate readings when the trends in the underlying data change. By exploiting its global view of the collected data and correlations between sensor nodes, the sink can improve the output of the prediction mechanism.

1.2 Methodology

Much of the work in this Thesis is described with reference to a conceptual WSN software architecture called FieldMAP. The FieldMAP architecture, presented in Chapter 3, provides a generic structure that supports a common data flow within WSN, and provides a basis within which the contributions to knowledge is evaluated. FieldMAP was developed through applying software design techniques to the WSN domain.

Two components proposed in the FieldMAP framework were instantiated. Each of these components was evaluated using an iterative *develop* - *experiment* - *evaluate* approach. The algorithms were evaluated against real world data sets and in the case of virtual sensors, in-situ deployments.

In the case of SIP, the data sets used were those reported against in prior work. Where possible, the same data traces and pre-processing techniques were use to prepare the data. If details of the exact data stream and pre-processing were omitted from the results reported in the literature then comparable data sets were used. For example, in the case of the National Data Buoy Centre (NDBC) data no details of the exact buoys used were given, so traces that measured and reported the same phenomena were used. Additional environmental monitoring data sets from prior deployments undertaken by Cogent Computing

ARC, Coventry University, were also used to evaluate the approach. The algorithm was developed and tested on a desktop PC, in a laboratory based experimental approach. It was further tested in a trial deployment on TelosB nodes within a single-hop network.

In the case of virtual sensors, historical data-traces from a deployment on a WDS were used to evaluate the effectiveness of the prediction mechanism for the initial laboratory based experimentation. The concept was also evaluated using a real world deployment in the WDS.

1.3 Research questions

The thesis attempts to answer the following research questions:

1. Is transmission reduction beyond the current state of the art possible by combining model-based filtering with dual prediction approaches in environmental monitoring applications?
2. When using a dual prediction based approach is there a consistent relationship between the number of transmissions required and the error budget for a given data set?
3. Can combining multiple sensors readings into a single predictive model allow a greater reduction in the number of packets transmitted, than when compressing each stream individually?

Concerning virtual sensing, the following research questions are explored:

1. Can a prediction mechanism that exploits previously learnt temporal correlations between nodes be used to estimate a sensor's readings where no value is currently available?
2. Can this prediction mechanism be used to used to estimate a sensor's readings where there is no sensor deployed, assuming a short period of training data?

1.4 Contributions to knowledge

The contributions to knowledge from the work presented in this thesis are:

1. A model-based transmission suppression algorithm (SIP) that reduces a node's transmissions by up to a factor of 100 in selected applications and which still has wide applicability.

2. A scheme for applying the SIP compression algorithm to multiple streams of data, further reducing the number of transmissions required.

3. A windowing scheme for input to the Virtual Sensors algorithm that makes use of *feedback* to maintain a constant set of training data.

4. A scheme for increasing a network's resolution through a system of virtual sensors, which predict missing data points and generate virtual sensor data.

1.5 Publications

The work described in this thesis has led to the following publications

Journal articles

- J. Brusey, E. Gaura, D. Goldsmith and J. Shuttleworth. 'FieldMAP: A Spatiotemporal Field Monitoring Application Prototyping Framework'. In: *Sensors Journal, IEEE* 9.11 (Sept. 2009), pp. 1378 –1390

- E. I. Gaura, J. Brusey, M. Allen, R. Wilkins, D. Goldsmith and R. Rednic. 'Edge mining the Internet of Things'. In: *IEEE Sensors* (2013). (submitted)

Conference proceedings

- D. Goldsmith, E. Gaura, J. Brusey, J. Shuttleworth, R. Hazelden and M Langley. 'Wireless Sensor Networks for Aerospace Application - Thermal Monitoring for a Gas Turbine Engine'. In: *Proceedings of 2009 NSTI Nanotechnology Conference*. Vol. 1. NSTI Nanotech. May 2009, pp. 507–512

- D. Goldsmith and J. Brusey. 'The Spanish Inquisition Protocol — Model based transmission reduction for wireless sensor networks'. In: *Sensors, 2010 IEEE*. Nov. Pp. 2043–2048

- D. Goldsmith, A. Preis, M. Allen and A. J. Whittle. 'Virtual Sensors to Improve On-line Hydraulic Model Calibration'. In: *Proceedings of the 12th annual Water Distribution Systems Analysis (WSDA) conference*. Aug. 2010

- A. Whittle, L. Girod, A. Preis, M. Allen, H. Lim, M. Iqbal, C. Srirangarajan S. Fu, K. J Wong and D. Goldsmith. 'WaterWiSe@SG: a Testbed for Continuous monitoring of the Water Distribution System in Singapore'. In: *Proceedings of the 12th annual Water Distribution Systems Analysis (WSDA) conference.* Aug. 2010

Patents and disclosures

The work described in Chapter 5 forms part of the following IP disclosure and Patent application:

- A. Preis, M. Allen, D. Goldsmith, 'System and method for water distribution modelling', (PCT/SG2011/000310)

1.6 Thesis structure

This chapter has presented an introduction to the thesis, including the motivation for the work, the research approach adopted, and the research questions and contributions to knowledge.

The rest of this thesis is organised as follows:

In Chapter 2, literature relevant to the topics discussed in the thesis is reviewed. A review of power consumption in WSNs, along with techniques to reduce energy use is made. Model-based approaches to data modelling and energy reduction are discussed, as well as machine learning techniques for estimating sensor values.

Chapter 3 introduces *FieldMAP*, a conceptual framework for WSN applications that forms a base for the data modelling approaches presented in the remainder of the thesis.

Chapter 4 presents the *Spanish Inquisition Protocol (SIP)*, a model-based transmission suppression algorithm and investigates its performance over several real world data sets, and an trial deployment in a single hop sensor network.

Chapter 5 explores the use of Gaussian Process Regression (GPR) for imputation and prediction of sensor values in an on-line WDS monitoring application. The concept is extended to allow long term prediction of sensor values without requiring a deployed sensor node (via a *virtual sensor*), using a short period of training data and correlated readings from other nodes in the network. The algorithm is evaluated within a WDS monitoring application. Finally, Chapter 6 presents the conclusions and discusses possible directions for future work.

1.7 Acknowledgement of contributed work

This section details the contribution made by other researchers that have aided the work presented in this thesis.

The HomeREACT data sets used to evaluate the SIP in Chapter 4 were provided by Tessa Daniel and Ross Wilkins fellow PhD students in the Cogent Computing Applied Research Centre. Ross Wilkins also provided the hardware and assisted in the implementation of the TelosB based deployment of SIP.

Analysis of the virtual sensors concept in Chapter 5 was made possible using data collected by the WaterWiSE@SG project. In particular Dr Mike Allen provided the testing data and assisted in integrating the virtual sensors algorithm into the on-line model.

The WaterWiSe@SG project is a collaboration between the Center for Environmental Sensing and Modeling (CENSAM), part of the Singapore–MIT Alliance for Research and Technology; the Singapore Public Utilities Board (PUB); and the Intelligent Systems Center (IntelliSys) at the Nanyang Technological University (NTU). It has been supported by the National Research Foundation of Singapore (NRF) and the Singapore–MIT Alliance for Research and Technology (SMART) through the Center for Environmental Modeling and Sensing.

The research team at SMART-CENSAM were also instrumental in deployment of the in-situ evaluation of the virtual sensors concept.

Chapter 2

Literature survey

This chapter reviews literature relevant to the work presented in this thesis.

The chapter is organised as follows: Section 2.1 gives a brief description of approaches to Wireless Sensor Network (WSN) design and terminology relevant to Chapter 3. Section 2.2 examines the trade-offs between lifetime of nodes in a WSN and the accuracy and timeliness of the data gathered. Section 2.3 examines the power consumption of WSNs nodes and hardware components and highlights areas where energy performance gains can be made. Section 2.4 presents an overview of energy saving approaches that are applied in WSN. Following this, Section 2.5 reviews methods of reducing energy consumption in WSN deployments, with a focus on model based approaches, and informs the author's contributions in Chapter 4. Section 2.6 examines methods for estimating missing sensor readings and informs the contributions in Chapter 5. A summary of the chapter can be found in Section 2.7.

2.1 Approaches to WSN design

The purpose of this section is to define approaches to WSN design and to introduce several key design terms that will be used to describe WSN applications.

Over the past decade, many WSN-based monitoring applications have been proposed. Some of them have been implemented, but only a few have been deployed. Examples here are VoxNet [5], Brimon [17], GridStix [42, 41], ZebraNet [52], Macroscope in the redwoods [99] and Manhattan story mashup [102]. Common to all deployments is their application specific implementation and the designers' / developers' steep learning curve towards the production of a successful system.

While there has been much discussion within the WSN research community on the need for a common WSN architecture to aid in reaching the goals of interoperability and code-reuse [92, 93], it appears from the literature that each of the systems cited above was developed and deployed as an one-off system, responding and optimising against, a specific set of constraints, mainly dictated by the application at hand. For example, an audio-analysis WSN application, such as VoxNet [5], focuses on the timeliness of the data rather than node-size, battery life, or scalability. Consequently, the outline WSN definition for VoxNet is driven by in-network processing, which was essential to satisfy its timeliness constraints. A body sensor application, such as CodeBlue [67], focuses on miniaturisation and reducing power consumption. A comprehensive set of low power medical sensing units have been developed in CodeBlue. However, the information transport segment of the application is based on a *publish-subscribe* paradigm, and ignores the question of how best to transport data through the network. These are two examples but many more have been reported in the literature with the common trait of limiting the design space according to the application at hand. This one-off approach to development makes it difficult to evaluate any particular WSN component against past application related work or achievements.

While the tight coupling of components commonly found in WSN applications such as the ones mentioned previously can lead to an efficient design, it can hinder code reuse, since it becomes difficult to separate components and their functionality from the underlying applications. If software design is separated into distinct components, it leads to a greater opportunity for code reuse, reducing the length of design, implementation and testing cycles for new applications.

Software architectures or design patterns are useful because rather than describe a specific implementation they can be seen as a general description of functionality. They document high-level processing elements, data structures and their interactions, providing a reference design for system implementation. By providing an overall structure for the design of a system, an architecture defines how that system should behave instead of the mechanics of individual components. This abstraction of system functionality into high-level components can allow a better understanding of the data and information flow.

An architecture also provides a common language and reference point for design and development teams. Questions raised during the design of new components can be explored, without the need for a specific implementation, which can lead to an improved understanding of requirements. An architecture can also encourage a holistic view of the design rather than focusing on individual components. There is a clear link between an overall software architecture and specific *design patterns*. Design patterns are commonly used in software design, to provide reusable templates to common problems. The use of such patterns can speed up software design. The work by Gamma et al. [27] suggests several classes of design pattern, along with detailed descriptions, and is often used as a reference text when developing traditional desktop applications. The division of a system's design into components makes it possible to strictly define their interactions. This makes it easier to adjust the functionality of each component.

In Chapter 3, FieldMAP defines a sensor network specific architecture and components that are instantiated in Chapters 4,5. The approach taken by FieldMAP has also been further taken up by Brusey, Gaura and Hazelden [12] [13].

2.2 The accuracy, lifetime, timeliness trade-off

The motivation for the work presented in this thesis is that WSNs are deployed to investigate phenomena and learn something new about the environment being studied. With this view, one key requirement of an end-to-end WSN system is the ability to determine the occurrence of an unexpected event in the data stream against a background pattern. Event detection motivates a data collection strategy where the sampling frequency must meet or exceed the Nyquist rate, although much of the time it is likely to contain a high amount of redundant information. Transmitting this information back to the sink will consume a high proportion of the energy used in a WSN [87].

Given the limited energy available to nodes within a WSN, the need to balance meeting application requirements against energy consumption presents a trade-off between three parameters: sensor node lifetime, data accuracy and timeliness of data updates (also referred to as the power-distortion-latency trade-off by Bajwa et al. [8]). An efficient sensor network could be described as one that uses a minimal amount of energy to collect and disseminate an accurate representation of the phenomena being observed. Environmental monitoring WSN applications require extended periods of timely, accurate data acquisition to achieve their goal [22]. However, collecting and transmitting this data consumes sensor nodes' energy, thus restricting the lifetime of a sensor network.

Figure 2.1 shows the lifetime—accuracy—timeliness space. How the designer will choose to optimise within the lifetime–accuracy–timeliness space will depend on the specifics of the application. For example,

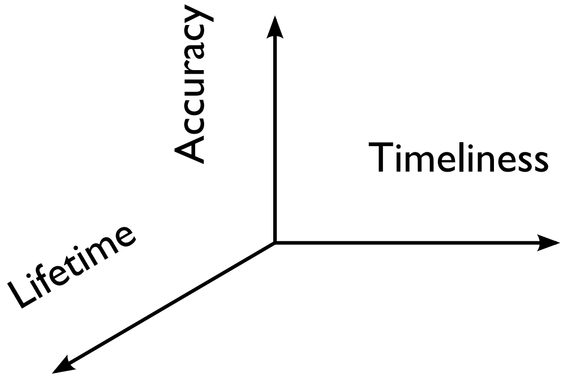


Figure 2.1: Accuracy-Lifetime-Timeliness triangle

an application that has a long lifetime requirement will make have to make a trade-off in either accuracy or increase the amount of energy available. Given the assumption that radio transmission is the most energy-consuming activity of a sensor node, a WSN designer could consider one or more of the following on-node strategies to address the trade-off:

- Optimise node lifetime by reducing the sensing rate, thus collecting and transmitting less data. Using this approach, it is possible to gain an increase in node lifetime, at the expense of accuracy and timeliness.
- Optimise node lifetime by caching data locally to transmit an aggregated packet of samples. This reduces the number of transmissions at the expense of timeliness.
- Optimise accuracy by sampling and transmitting at a high frequency. This leads to a more accurate and timely representation of the phenomena but requires more energy, shortening node lifetime.

A better optimisation of this space would be to maintain the sampling rate while reducing the amount of transmissions without aggregating data locally (in order to maintain timeliness). This optimisation implies that data would be processed locally. Pottie and Kaiser [85] have demonstrated that the energy cost of transmitting a single bit of data is approximately the same as performing a thousand processing operations. Thus, the case for performing processing on-node (or in-network) is that local computation is cheap and transforming data to information reduces the number of bits and thus packets that must be transmitted thereby reducing the node's energy consumption.

Processing locally goes beyond the basic *sense and send* approach commonly applied to real-world WSN deployments which push data to a central node for processing.

2.3 Node level power consumption

Typically nodes in a WSN are battery powered, with a limited energy supply. Thus a key factor in the design of any WSN to achieve a long node lifetime is careful management of the energy available. Understanding how and where energy is used can give insight into which components have the highest power requirements and this provides scope for optimisation. This section reviews sensor node power consumption with reference to several well known and frequently used sensor node platforms in WSN research: the Mica2, MicaZ and TelosB [84].

2.3.1 Power consumption of sensor node components

A typical sensor node used in WSN deployments consists of a number of common subsystems and each of these will have a role in the node's energy consumption:

Sensing Includes one or more sensors along with supporting hardware (such as the Analog to Digital Converter (ADC)) used for gathering data.

Processing Consists of a processor, generally a microcontroller such as the MSP430 [44] or ATmega328 [62], and associated hardware (such as clocks) used for data processing and transformation.

Radio Used for communication with the sink or neighbouring nodes, the most common radio unit is the CC2420 [48] which provides low power communication capabilities in the 2.4 GHz band.

Storage Is used to retain data and information at the node, and may consist of RAM or flash memory.

Several studies have examined node power usage. For example, Table 2.1 shows the energy consumption of common WSN motes as reported by Polastre, Szewczyk and Culler [84]. Table 2.2 shows the results of a similar study of the Mica2 nodes by Mainwaring et al. [66]. For all platforms considered in Tables 2.1 and 2.2 the energy required to operate the radio is significantly more than any other component. It should also be noted that both transmitting and receiving require similar amounts of energy,

Mode	Telos	Mica2	MicaZ
Year	2004	2002	2004
Minimum Voltage	1.8 V	2.7 V	2.7 V
Standby	5.1 μA	19.0 μA	27.0 μA
MCU Idle	54.5 μA	3.2 mA	3.2 mA
MCU active	1.8 mA	8.0 mA	8.0 mA
MCU + RX	21.8 mA	15.1 mA	23.3 mA
MCU + TX	19.5 mA	25.4 mA	21.0 mA
MCU + Flash Read	4.1 mA	9.4 mA	9.4 mA
MCU + Flash Write	15.1 mA	21.6 mA	21.6mA

Table 2.1: Energy consumption of common WSN nodes given by Polastre, Szewczyk and Culler [84]

Operation	nAh
Transmit a packet	20.0
Receive a packet	8.0
Radio Listen	1.250
Analog sample	1.080
Digital sample	0.347
Read from ADC	0.011
Flash Read Data	1.111
Flash Write / Erase	83.333

Table 2.2: Power required by various Mica2 operations [66]

due to the need to have the radio transceiver activated. Therefore just listening for radio transmissions requires a considerable amount of power.

The energy consumption of the MCU can be split between the power used for computation when software instructions are executed and that used for background operations when no computation is being performed. Background energy consumption occurs even when the MCU is in its low power sleep state and represents the minimal power consumption of a system. In the studies described above, the MCU has minimal energy requirements compared to all other components. Notably the energy requirements in low power sleep mode are around one thousand times lower than when in idle or active modes. This trend is also seen in other microcontrollers such as the ATmega328 [62]. Hence, a common method to maximise the lifetime of a WSN deployment is to keep the MCU in a sleep state for as much of the time as possible.

While the energy requirements in processing, idle and sleep modes have decreased significantly as mote class devices have evolved, the cost of transmitting and receiving data has remained static. For example, in the TelosB, Mica2 and MicaZ nodes reviewed above, the energy cost of when the MCU is active has decreased by more than a factor of four between generations of motes, from 8 mA to 1.8 mA. This means that the relative energy cost of using the radio has actually increased. This is partly due to the change in radio from a proprietary narrow band approach to the standardised IEEE 802.15.4 operating in the 2.4 GHz band [84]. While this change has resulted in greater throughput and has provided other advantages in the way data is transmitted, it also means that the cost of communicating has become more significant compared to the base cost of running the node hardware. Therefore, approaches that limit radio use will have a greater impact on the lifetime of the node.

In terms of storage, small amounts of data can be stored in Random Access Memory (RAM). However, the amount of data that can be stored is limited (approximately 10 kB on the TelosB [84]) and this form of storage is vulnerable to power loss. With larger amounts of data, or in applications where the data needs to be permanently stored on the node, flash memory is used. Tables 2.1 and 2.2 show that the while the energy cost of reading from flash are low. The energy required to write to flash memory is close to that of using the radio. The energy cost of flash memory has also been explored by Nguyen et al. [72], who demonstrated that using a flash based caching and aggregation approach was less efficient than transmitting each sample separately. These results suggest that caching approaches that write to flash memory may not be appropriate for energy reduction.

According to Raghunathan et al. [87] sensors can be classified into two types: *active* and *passive*. Passive sensing, where the sensor simply takes a reading from the surrounding environment, is the most common type and includes sensors such as thermistors, humidity sensors, accelerometers and light meters. These sensors typically have a low power requirement. Active sensing on the other hand, relies on the sensors interacting with the environment to take a sample. Active sensors include radar, sonar and gas sensing devices.

In the general case, when using passive sensors, an assumption that the power required to sense data will be minimal holds true. For example, a study of the sensors used in the Duck Island [66] deployment, stated that the sensor current requirements were lower than those required to maintain the MCU in an Idle state (Table 2.3).

While the assumption that sensors will have low power requirements holds true for many passive sensing systems, active sensors tend to have a higher current draw. For example, a CO_2 sensing device such as the B530 [43] has a peak current draw of 130 mA which is over five times greater than the radio current draw on a Telos mote. In this case, the power consumption of the sensors rather than the radio

Sensor	Current Draw
Photoresistor	1.235 mA
I^2C Temperature	0.150 mA
Barometric Pressure	0.010 mA
Barometric Pressure Temp	0.010 mA
Humidity	0.775 mA
Thermopile	0.170 mA
Thermistor	0.126 mA

Table 2.3: Current consumption characteristics of sensors on the Mica2 weather board [66]

becomes the main limiting factor in the lifetime of any deployment.

Summary

This section reviewed the energy consumption of WSN nodes, and the role individual node components play in the overall energy consumption of a node. While different nodes will have different energy consumption profiles, the following observations are generally true.

The power supply component must provide all the energy a sensor node requires to function. In the general case the power available to a node is supplied through batteries, which provide a limited energy supply; replenishing batteries can be a time consuming and expensive process. Therefore, to achieve a long system lifetime, careful management of the power resource is required. In some cases energy harvesting techniques may be able to alleviate this problem. Although nodes can be connected to a permanent power supply, this tends to reduce the advantages of using wireless systems.

Depending on the specific application, the energy requirements of the sensors attached to the node may be greater that that of the radio. In this situation it is important that the energy consumption of the sensors is reduced.

However, is clear that in majority of cases the radio unit has the highest energy requirements, with the transmission of a single packet requiring the same energy as taking 60 digital sensor readings [66]. This suggests that a significant increase in node lifetime can be obtained by reducing the energy consumption of the radio unit by trading transmission for computation. A discussion of software driven approaches to energy consumption is given in Section 2.5. The next section provides an overview of energy saving approaches that can be applied to wireless sensor nodes.

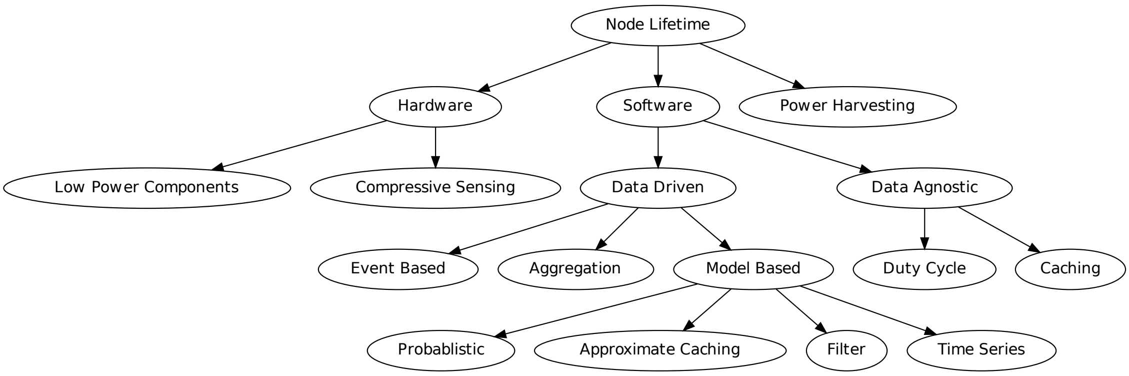


Figure 2.2: Energy Saving Approaches in WSN

2.4 Taxonomy of energy saving approaches

Generally speaking, energy saving approaches in WSN fall into four categories as shown in Figure 2.2:

Hardware optimisation focuses on reducing the amount of energy used by the node hardware. This may include the use of energy aware operating systems such as Pixie [61], or designing new hardware systems that optimise energy consumption for a particular application such as in Code Blue [67].

While the use of low power components can provide significant energy savings it relies on the hardware (and the design expertise to use it) being available and therefore this approach may not be appropriate in all cases. For example, the study of nodes in Section 2.3.1 has demonstrated that while the energy consumption of the MCU was reduced between generations of three commonly used sensor nodes (Mica 2, MicaZ and TelosB), the power requirements of the radio have remained relatively static.

Network optimisation addresses the energy used by the radio, by modifying how and when nodes are able to communicate. Examples here are MAC layer optimisation (such as B-Mac [83]), and routing protocols (such as Leach [39]) . The topic of routing algorithms and MAC layer optimisation has been an active research area within the area of WSNs. (For a survey of routing protocols see Al-Karaki and Kamal [3]). However, while many routing protocols have been proposed, only a few have been used in real-world deployments [30].

Algorithmic approaches focus on the use of software to reduce the energy consumption of sensor nodes. For example, reducing the amount or frequency of transmission through aggregation or suppression. Algorithmic approaches can offer a high level of energy saving. However, as these approaches rely on exploiting certain data characteristics, the energy savings can be dependent on the nature of the phenomena being sensed.

Power harvesting while not strictly an energy saving approach, aims to supplement or replace the normal power supply and thus increase the length of time a network can be deployed for.

Each of these approaches can be used individually. For example, deploying an algorithm on new hardware is likely to reduce the overall power consumption. However, in most cases a combination of techniques will be used.

By reducing the amount of data that is transmitted through a network, algorithmic approaches can reduce the amount of energy consumed by a sensor node. These energy savings are independent of those from other optimisations, and thus algorithmic approaches can be used to complement other energy reduction techniques.

The following section focuses on algorithmic approaches to reducing energy consumption of sensor nodes.

2.5 Software driven transmission reduction

Software driven methods for reducing WSN energy consumption can be divided into data agnostic and data-driven approaches.

Data agnostic approaches provide general techniques for reducing the energy consumption of the node. These methods do not rely on any prior knowledge of the phenomena to be sampled, and thus can be applied in more general situations. A discussion of data agnostic approaches is given in Section 2.5.1

Data driven approaches rely on some knowledge of the data being observed to reduce the energy consumption of a sensor node. In general these techniques aim to reduce energy consumption of the radio by reducing the amount of data that is transmitted by a node. For example, transmission suppression reduces the number of samples transmitted, whilst maintaining a suitable level of accuracy for the application. Other approaches aim to reduce the energy consumption of other node subsystems. For example, compressive sensing aims to reduce the amount of samples required to represent a data stream.

Data-driven approaches to energy reduction are discussed in Section 2.5.2.

2.5.1 Data agnostic approaches

This section describes some general approaches to reducing energy consumption that can be applied without requiring detailed knowledge of the data under consideration.

Duty cycling

Given that the *sleep* mode power consumption of WSN devices is extremely low, having the shortest duty cycle possible is a simple and effective method of reducing power consumption. In terms of the Accuracy, Lifetime and Timeliness (ALT) trade off described in Section 2.2, reducing the duty cycle represents a reduction in both accuracy and timeliness in exchange for an increase in network lifetime. Assuming that the whole node (radio, MCU and sensors) enters sleep node, no sampling takes place and thus the accuracy of the data stream may suffer. If just the radio unit is turned off and sampling continues, some form of data aggregation or event triggered delivery will be required.

Sleeping for much of the time between taking samples is essential to achieving long-lived deployments, with duty cycles of 4–5 seconds every 5 minutes (1.3%) being common [66, 99]. The duty cycle can be calculated given a target network lifetime.

While reducing the duty cycle is essential to lowering node power consumption when used alone (such as in the case of sense-and-send) it will still transmit each sample that is gathered. Therefore, duty cycling can be combined with data driven approaches such as those discussed in Section 2.5.2 to further reduce the energy consumption of a node by reducing the amount of redundant information transmitted.

Caching

Another approach to reducing the amount of energy consumed is to optimise how and when data is sent (or to remove transmission entirely). Energy savings can be made though the use of a caching approach (buffering the readings and transmitting a packet with combined or aggregated data) as the total number of packets transmitted is decreased. This solution is efficient for applications that only require offline data analysis. Applications that require the user to respond to events may be adversely affected by the caching approach.

Caching can also be used where communication between a node and the sink is difficult or intermittent. For example, to increase data yield Macroscope in the Redwoods [99] buffered samples during periods during network failures and transmitted the cached data when a connection was available. Other approaches have used caching to reduce the frequency that data is sent to the sink by storing samples and only transmitting at a specified time [68].

Other forms of caching include the use of amnesic models [79]. By degrading the data stored in the cache over time, these approaches allow large quantities of data to be retained in a limited memory footprint.

An extreme example of caching is where a *Sneakernet* is formed [54]. Data can be manually collected from individual nodes and transferred to the sink at a later date. Depending on how the data is transferred (for instance by serial connection or swapping memory sticks), this can completely remove wireless communication costs from the node. The approach is costly in terms of time, though, as human intervention is needed to collect the data.

In terms of the ALT trade-off described in Section 2.3, caching primarily affects the timeliness of the data, as samples are stored on-node before transmission. If aggregation is used (for example, only transmitting summary statistics) then the accuracy will also be reduced. Depending on the amount of data that is cached and the method used to store the cached data, this approach may have little benefit in terms of energy savings. While small amounts of data can be stored and forwarded using the memory available on the MCU, large amounts of data will require storing in flash memory. As noted previously, studies have shown that the energy used to write to flash can be comparable to using the radio [72].

Summary

Data agnostic approaches to energy reduction can be used without prior knowledge of the data. To achieve a reasonable lifetime, duty cycling will be used in most WSN applications. However, duty cycling focuses only on reducing the amount of time the node is active, and takes no account of the data collected. Thus, there is scope to further increase the energy savings by combining data-driven approaches and duty cycling to remove redundant information from the collected data. Caching approaches allow a node to reduce the number of transmissions by storing data locally before transmission. If small amounts of data are stored in RAM these approaches can be effective at reducing the number of transmissions, but will affect the timeliness of the data. If a user needs to respond to events in the network, or real-time display of information is required then caching may not be an appropriate approach. Where larger amounts of data need to be stored, or more permanent storage is required, the energy costs associated with using the flash memory may exceed that of transmitting the data to the sink.

2.5.2 Data-driven approaches to reducing energy consumption

Data-driven approaches focus on reducing the amount of data that is transmitted. In many cases, this allows a high sampling rate to be maintained, increasing the accuracy of the data, while reducing the number of redundant samples that are transmitted. Given that the radio has the highest energy draw, reducing the amount of data transmitted can be expected to significantly extend the lifetime of the node.

Event based transmission

One approach to reducing the amount of raw data transmitted is to use an event-based transmission strategy, where messages are transmitted only when a predefined event is detected by a node. The approach is well suited to applications where events are both sparse and easily detectable (such as sniper localisation [97] and intrusion detection [7]).

One difficulty with event-triggered delivery is in specifying event thresholds [53]. While smoke and high temperatures may indicate a fire to a human, what levels of each need to be present before an event is triggered? Since a system's state tends to evolve over time (e.g., due to seasonal variations), predefined triggers may lose relevance to real events in the underlying phenomena. This can lead to missing events that are of interest but which are outside of the parameters originally envisaged. For example, while temperature levels below 0 ℃ may be commonplace in the winter, they would be a significant event during the summer.

In terms of the ALT trade-off, depending on the frequency of events, this approach can provide a significant increase in lifetime as the number of transmissions will be proportional the number of events. However, in its simple form, event-based transmission has the disadvantage of only transmitting that an event has occurred and not transmitting information about the period of data leading up to this event. This goes against the view that WSN are employed as a exploratory tool to learn about environmental data. Having a complete picture of the data leading to any event is important in increasing our understanding of how and why a particular event occurred.

The approach taken by Lance [105] attempts to address this problem. Rather than transmit the full data window, only a summary of the sample in the window are transmitted, while the raw data stream is stored locally on the node. If the sink determines that the summary is of interest, the complete data window for this summary is requested.

Aggregation

In-network data aggregation, such as returning average, minimum or maximum readings over a cluster of nodes or combining readings from several nodes into one packet, is also a commonly proposed approach to long lived networks [26]. However, while many cluster based aggregation approaches have been proposed in the literature, such approaches are rarely seen in sensor network deployments [30]. Given that in-network aggregation will require nodes to communicate, many aggregation protocols have been designed with a particular form of routing protocol in mind. Examples of these protocols include Directed Diffusion[47, 46] and LEACH [39].

With Directed Diffusion, *interests* are created by the user. Interests are queries that describe the event the user is interested in, the area in which this event is expected, and the interval (data rate) required. These queries are propagated through the network, forming *gradients* though which data matching the event will be drawn back through the network to the sink. As events are detected and transmitted back to the sink, these gradients are reinforced, creating a low latency route between the node detecting the event, and the sink. Aggregation is supported through the use of filters, that can manipulate messages as they travel down a gradient. One simple filter suppresses matching reports from neighbouring nodes, which was reported to save up to five times the energy compared to when suppression was not used [47].

Cluster based approaches such as LEACH focus on reducing the amount of communication by aggregating data between neighbouring nodes. Clusters of nodes are formed, with one node randomly selected as the cluster head. Nodes in the cluster transmit readings to the cluster head. When the cluster head has received readings from all nodes, they are aggregated into a single packet and forwarded to the sink.

Where aggregation is used to produce summary statistics (such as minimum, maximum or average) algorithms such as TAG [64] can reduce the number of transmission made, by combining sensor readings into aggregated summaries, at the expense of accuracy. However, where data is aggregated into summary packets, deployment experience with the Sonoma Redwoods [99], shows that domain scientists tend to be resistant to any form of in-network aggregation which may reduce the quality or interfere with the gathered data.

Aggregation needs to be carefully managed to achieve energy savings. A principle advantage of aggregation schemes is that they reduce the number of *hops* required to transmit data to the sink in large networks. However, aggregation performance is affected by the density and topology of the network [45]. In smaller networks, for example, there may not be enough nodes to gain any benefit from aggregation. Retransmitting data in large multi-hop networks requires that the node is not only actively retransmitting packets but also has its radio in receive mode. Since the cost of listening for data can be as high as receiving or transmitting, this could take up a significant proportion of the available energy budget. Additionally, if the number of transmissions required to aggregate data is greater than that required to send a raw sample to the sink, the local cost of aggregation can outweigh the costs involved for a simple sense and send strategy. These limitations can be overcome through the use of the model based techniques in the following section.

2.5.3 Model Based Transmission Reduction

Given that much of the time, the data gathered by sensor networks (such as from temperature sensors) follows slow moving trends, a large proportion of the data gathered is redundant, either repeating a previous reading, or reporting only small changes in value. As communication energy is a major factor in the overall energy requirements of a sensor node, transmitting redundant information can take up a significant proportion the the energy budget.

Model based transmission reduction approaches attempt to encode information collected at a node into a model. These models are then transmitted to the sink for use in future queries. This approach can reduce communication costs since if the model can produce an accurate representation of the sensor value, then no transmission is required. Figure 2.3 gives an overview of model based transmission reduction.

Dual prediction schemes are model driven energy reduction strategies that aim to transmit only parts of the data stream that add to the users knowledge of the monitored phenomena. A predictive model is shared between node and sink. At each sampling interval, the node makes a prediction of the sensor reading and compares it to the actual sensor value. If the predictive model is capable of adequately estimating the sensor reading, then the packet is dropped. Otherwise the model is out of date and an update is transmitted to the sink. The sink simply uses its most recent copy of the predictive model to

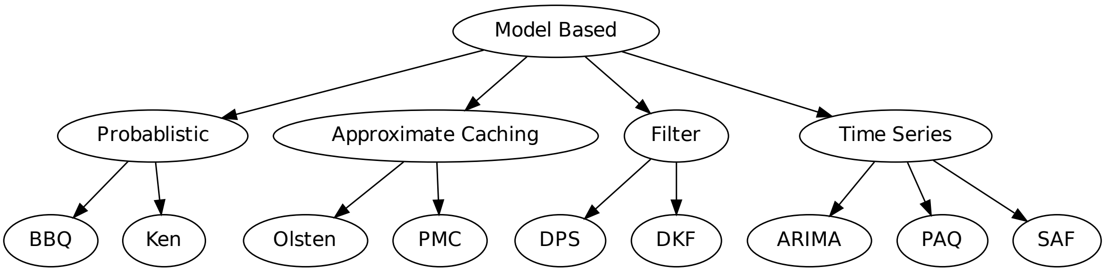


Figure 2.3: Overview of model based transmission reduction

estimate sensor readings in response to queries.

To further reduce the number of transmissions, these approaches exploit a trade off between the accuracy of the data, and the number of transmissions made. By accepting values from the predictive model within some error budget, a significant saving in the number of transmissions can be made.

An advantage of dual prediction schemes is that they are well suited to applications that deal with event or anomaly detection. While frequent sensor readings are required to detect these events, in most cases, the state of the sampled data will be *normal* and therefore no transmission is needed. If an unusual event occurs, the values estimated by the model will be incorrect, and the data is pushed to the base station for processing. While the use of an error budget can introduce some uncertainty to the readings reported, these values will still represent the true value of the sensor within some confidence interval. In comparison to aggregation through summary statistics, this means that while some accuracy is lost, much of the detail within the data stream is maintained.

Probabilistic Models

One approach to gathering data from WSNs used by algorithms such as TinyDB [65] and Cougar [111] is to treat the sensor network as a database, gathering the required data via Structured Query Language (SQL) like queries. However, this approach can require a significant amount of communication, as queries are generally performed using a request-response approach. Secondly, to answer each query as much data as possible is gathered from the network and this can require a large amount of communication, and possibly the transmission of a large amount of redundant data.

Probabilistic model driven approaches to querying such as BBQ [24, 25] aim to optimise the amount of data gathered by applying probabilistic models to the sensed phenomena and using these to answer queries. The Probability Density Function (PDF) for each model is built during an initialisation phase. Responses to queries are generated by a planning component, that makes use of the probabilistic model to calculate the most efficient way to fetch the required sensor reading within a given confidence threshold. The data is then requested from the selected sensors using a *pull* approach. The planning component of BBQ can also make use of correlations between nodes and sensors. For instance, if the temperature readings on two nodes are highly correlated, the value for one node may be inferred based on this relationship. A significant disadvantage of BBQ is that the pull based approach is not able to respond to events within the data stream. As transmission of a sample is initiated by the sink, no data is collected unless a sample has been requested. Additionally, while the algorithm aims to supply data within a given confidence interval as calculated by the probabilistic model, there are no guarantees on the actual accuracy of the samples received.

Ken [20] attempts to address the event detection problems found in BBQ. Rather than have the sink issue *pull* queries when a sample is requested, Ken relies on the nodes to initiate communications. Using a Dual Prediction System (DPS) approach an identical pair of probabilistic models are stored at both node and sink. At each sampling period, Ken calculates the accuracy of the estimated sensor reading, and updates the model if it is outside of the specified error bounds. The model used can be modified to take account of spatial correlations between neighbouring nodes. However these correlated models can require a significant amount of intra-node communication. To reduce this overhead, a Disjoint-Cliques model is used, which uses a greedy algorithm to discover an optimal solution to the clustering problem. Similarly Wang and Deshpande [103] proposed a comparable algorithm using a *decomposable* model to form the PDF. To reduce the number of direct intra-node communications required, the broadcast nature of WSN was used to aid in model formation. While both algorithms were able to provide a reasonable reduction in the number of transmissions required, both approaches relied heavily on highly correlated data to achieve these results. For example, over the Intel data set Ken only transmitted 35% of packets with large cluster sizes, however without correlation this increased to 65% of samples (roughly equivalent in performance to Piece-wise Constant Approximation (PCA) described in Section 2.5.3) . Additionally, the approaches based on predictive models described above require a period of training data ranging from 4 to 15 days on the Intel data set (approx 11520 samples), the transmission of this training data would require a large amount of energy. Additionally when evaluating the performance of Ken, the training data was excluded when calculating the training percentage of data transmitted [20], meaning that the actual number of transmissions required to represent a signal may be significantly higher.

Approximate Caching

Run Length Encoding (RLE) is a algorithm that provides a simple and popular method of loss-less compression that is used in many applications and protocols (such as in image files (TIFF) and Adobe PDF documents). The concept behind RLE is that sequences of the same value can be replaced with shorter segments each containing a single value and count. Where there is a large amount of repetition, the level of compression offered by RLE can be significant. However, in its standard form RLE reduces the timeliness of the data as the algorithm will only transmit the summarised segment when the value changes. Additionally, in the case where the input data has frequent variations (such as sensor noise), the level of compression offered is low.

Variants of RLE for WSNs have been proposed that aim to address these problems. By allowing values within a given error threshold (the *caching width*) to be contained in one segment, these approaches (termed approximate caching by Olston, Loo and Widom [74]) mean the variability of the data has less impact on the number of transmissions.

K-RLE [15] addresses the problem of sensor noise allowing the user to specify an error threshold K for the data. After the first value for each segment is stored all subsequent sensor readings will be included in that segment if they are less than this error threshold from the first value. By introducing an element of loss to the algorithm the number of packets required to transmit a stream of temperature data was reduced by an average of 56% when using a 2.0 ℃ error threshold. However, the K-RLE still requires a complete segment to be gathered before transmitting a value, and thus will increase the latency of the application.

Poor Man's Compression (PMC) [56, 82] is another such approximate caching scheme. Two methods of generating segments were proposed:

1) PMC-MR (or midrange) monitors the range of the input and while this is less than twice the error budget 2ε the range is updated if needed and the sample dropped. When the range exceeds the error threshold, a segment with a value equal to the midpoint of the range ($c = \max + \min/2$) and a count of samples since the last PMC midpoint is generated. While this approach can be shown to be optimal for

algorithms that use a PCA representation of the data, the mean error can be large and often approaches the maximum error value.

2) An alternative approach to segment generation is PMC-MEAN; rather than use the midpoint of the sample, the mean value within the segment is used. While this approach reduces the mean error of the reconstructed data it has been shown to produce a higher number of segments than the PMC-MR approach, thus reducing the compression ratio [56].

To allow for on-line operation, and reduce the latency associated with the caching approach a prediction mechanism based on transmitted segments is used. An estimate of a reading at the current time can be made using prior segment summaries transmitted to the sink. As a new segment is received, the model used to estimate these readings is updated and the new information used in future readings. The PMC approach offers compression ratios of around 50%, however the reported values suffer from quantisation, so much of the short term detail within the data is lost. Additionally the greater reduction in transmissions by the PMC-MR approach comes at the cost of a higher mean error. While this will not violate the specified accuracy bounds, it can affect the accuracy of the estimated samples. This approach also takes no account of trends within the data when setting the error bounds. For example, while static data will produce a high compression ratio, when the data is rising or falling, it can quickly move outside of the error bounds triggering a delivery.

The Approximate caching [74] algorithm, takes a similar approach in the context of a query based TinyDB like abstraction [65]. SQL like queries (over both individual and groups of nodes) are registered with a central stream processor located at the sink, along with the maximum error permissible for that query δ. These queries request aggregate data from a node or group of nodes. To answer these queries a *bounded approximate answer* is formed, where the sensor value is guaranteed to lie within a user specified precision interval. Each bounded approximate answer is a pair of real values L and H defining an interval $[L, H]$, within which the real value is guaranteed to lie. The precision constraint δ for each query, is a user specified value stipulating the maximum width of these bounds. $0 \leq H - L \leq \delta$. Each data source maintains a its own set of bounds centred around the most recent update to the sink. If when a sample is taken it falls outside of these bounds (and therefore the specified error threshold), an update of the sensor reading is transmitted. One interesting aspect to this work is the use of the central stream processor to manage each sensor's precision interval in an attempt to optimise the communication overhead [73]. The approximate caching algorithm does not provide any latency guarantees, and therefore may not be suitable for situations where a rapid response to any data stream events is required. Additionally, the use of data aggregation into summary statistics means that the original data stream will not be available, which may affect any analysis of the data gathered.

Filter Based approaches

Rather than use probabilistic models, or approximate caching approaches, Jain and Chang [49] [50] introduce the concept of using a linear filter as a predictive mechanism for transmission reduction.

Rather than the bounded approximations used in prior approaches this approach made use of a Kalman filter [104] to estimate sensor readings. Like previous dual prediction system approaches, the Dual Kalman Filter (DKF) makes use of identical models at the node and sink. The *estimation* stage of the Kalman filter is used to predict the next sensor reading. If the estimated reading is outside of the accuracy bounds, the parameters required to update the filter are transmitted to the sink. One potential issue that is presented by the use of a Kalman filter is that for the filter to produce output relevant to the data stream, the parameters of the filter need to be well specified in advance. Thus the output of the Kalman filter is impaired without *a priori* knowledge of the data stream. This may make the algorithm less well suited for a generic approach. Additionally, while the computation required for the Kalman filter is not high compared to some approaches (such as Gaussian Process Regression (GPR)), it is still significantly higher than other filter based approached such as the DPS (discussed below), and has been shown to induce significant latency sensor network applications [11].

In an attempt to reduce the *a priori* knowledge of the data that a Kalman filter based approach requires, Santini and Römer [95] propose the Dual Prediction System. This approach makes use of the Least Mean Squares Adaptive Filter (LMS) [37] for the filtering and prediction of sensor values. Adaptive filters are commonly used in environments where some property of the signal (for instance the noise levels) are not known in advance. Thus the use of the LMS is aimed at reducing the amount of prior knowledge required to model the data when using complex filters such as the Kalman Filter (KF). This approach has several drawbacks. First, the LMS filter is sensitive to the choice of step-size parameter μ. While the approach proposed by Santini and Römer [95] to calculate the value of step-size will ensure efficient operation, there is a period of *training data* required to calculate this parameter, and this can affect the output of the algorithm. Secondly, the LMS is known to suffer from a convergence period when the reported values do not match the underlying data stream [107]. During this period the output of the filter will result in a greater error in the predicted sensor values, or a greater number of transmissions.

For both filter based approaches, the predictions are created on a *one step ahead* basis. This means that estimations can only take place at set interval. This will affect the output of the model if a query is made outside of these steps. It is also unclear what effect clock drift will have on keeping the two models synchronised. However, both filter based approaches offer attractive levels of compression over other data driven approaches (approximately 90% in the case of the LMS).

Time series approaches

Time series modelling techniques such as those based on Autoregressive Integrated Moving Average (ARIMA) [60] have also been used to reduce the amount of data transmitted. While these approaches can support spatial-temporal correlations between nodes, they require a long training phase, which can make them impractical for use in monitoring applications, and require intensive computation that may not be appropriate for WSN.

Probabilistic Adaptable Query system (PAQ) [101] and Similarity-based Adaptive Framework (SAF) [100] are designed for use in TinyDB [65] like abstractions of sensor networks and address the high computation requirements of general ARIMA by using simpler Autoregressive Models (AR). SAF improves upon PAQ by adding a trend component (based on linear interpolation) to the AR model, and reducing the training set required. In terms of transmission reduction performance SAF shows similar characteristics to the Kalman filter based approach [50], when used to compress a preprocessed version of the Intel lab data set [100].

Le Borgne, Santini and Bontempi [57] address the problem of model selection when using AR techniques through the use of adaptive model selection. Specifically, discovering the order of the AR required to best represent the data during prediction. During the initial deployment phase, the node simultaneously estimates sensor readings using several AR models. At each sampling interval, a *racing* algorithm was used to discard poorly performing models, meaning that after some time (approximately 1000 samples during experimentation) the most appropriate model is used for future predictions.

During experimentation the racing algorithm was shown to outperform an approximate caching approach (see Section 2.5.3) in 9 out of 14 tests. The average reduction in the number of transmissions made to represent these data set ranged between 50% of data with an error threshold of 1% of the sensor range, to a reduction of 96% for a rough estimate of the data (20% of the range of the data).

Approach	Data set	Error threshold	% Data Transmitted
PMC [56]	NDBC [71] Sea Surface Temperature	1% Range of Data (ε=0.06)	50 %
PMC	NDBC Salinity	1% Range of Data (ε=0.0187)	45 %
PMC	NDBC Shortwave Radiation	1% Range of Data (ε=13.513)	45 %
PCA [74] (Uniform)	Local Network [81]	2 pkts/s	33%
PCA (Dynamic)	Local Network	2 pkts/s	22%
DKF [95]	Electric Power Load [6]	ε=150 MW	45%
DKF	Network Monitoring Dataset [55]	ε=5 pkts	5%
DPS	Intel Node 13 [63]	ε=0.5 ℃	10%
Ken [20] (Average)	Intel Lab [63]	ε=0.5℃	65%
Ken (Disjoint Clique 6)	Intel Lab	ε=0.5℃	35%
PCA (From [20]))	Intel Lab	ε=0.5℃	65%
SAF [100]	Intel Lab (Adjusted to 1 min Samples)	ε=0.5℃	150-200 Packets / Week (2.6%)
DKF (From [100])	Intel Lab (Adjusted to 1 min Samples)	ε=0.5℃	150-200 Packets / Week (2.6%)
Racing	Temperature (Vineyard)	1% Range of Data (unspecified)	44%
Racing	NDBC [71] Temperature	1% Range of Data (unspecified)	36%

Table 2.4: Summary of prior work

While not strictly a transmission suppression algorithm in its own right, the racing algorithm can aid model selection when using AR models. However, there is no clear information on the number or type of models that should be used in the initial set. If a large number of models are required to converge on an optimal solution, then the overhead of running such models may outweigh the benefits. Additionally, the approach still requires a period of training while the racing algorithm selects the most appropriate sample.

Model Based Approaches: Performance Summary

The approaches described in this section reduce the number of transmissions required to represent a data stream by representing the data as a model. In all cases, these approaches allow the user to trade accuracy in the data for a further reduction in transmissions.

The transmission reduction performance of selected model based approaches is given in Table 2.4. It is clear that the model based approaches reviewed can provide significant savings in the number of packets that are transmitted with transmission ratios ranging between 65% and 5%.

Many of the experiments have focused on reporting Temperature data from the Intel lab data set at an error threshold of 0.5 °C. Of these, it is clear that the SAF and DKF approaches yield the best transmission suppression performance with a stated 150–200 packets required per week to represent the data [100]. However these results were achieved after pre-processing had taken place (estimating missing readings via linear interpolation, and re-sampling the data to one sample per minute). The effect that such pre-processing would have on the transmission suppression performance is unclear. Another algorithm that shows impressive performance over the Intel Lab data is the DPS, which required only 10% of the gathered samples to be transmitted. As pre-processing of data is impossible in real world situations, this figure of 10% is considered the benchmark for transmission suppression algorithms for the work in this thesis.

2.5.4 Reducing the cost of sensing

While it is generally assumed that the radio is the most significant consumer of energy, in some cases the sensors themselves may consume a significant amount of power. In this situation data driven approaches to transmission reduction may not provide the greatest possible energy savings.

A sensing orientated approach to energy reduction, will focus on reducing the number of samples taken, rather than focus on reducing communication. Of course, reducing the amount of data collected, will decrease the number of samples that have to be transmitted, and thus the energy cost of using the radio.

This section provides a brief overview of sensor orientated energy reduction techniques.

Adaptive sampling

Rather than sample at a fixed rate, adaptive sampling algorithms attempt to collect data at an optimal rate. However, the optimal sampling frequency for a particular sensor can be difficult to determine as it requires *a priori* knowledge of the characteristics of the data.

The adaptive sampling approach put forward by Alippi et al. [4] attempts to calculate the Nyquist frequency required to correctly sample the data stream through a modified Cumulative Sum (CUSUM) test. Given that the calculations required are computationally intensive the sink calculates the optimal rate for each sensor and transmits this to the node. The adaptive sampling algorithm was shown to reduce the number of samples required by between 18–27%, compared to a static sampling frequency.

A similar centralised approach to this problem using the output of a Kalman filter is proposed by Jain and Chang [49]. The above approaches are centralised and require the sink to calculate the sampling frequency. The need for a centralised algorithm to coordinate when samples are taken can will lead to an increased network overhead due to control messages, and will affect the adaptive sampling process if the sink is not available.

One decentralised approach to adaptive sampling is Utility based Sensing And Communication protocol (USAC) [78]. The node uses a linear regression model to estimate the value of the next reading. If this value falls outside of a given confidence interval, then it is likely that a sudden variation in the data will occur, and the sampling rate is increased. Otherwise, the algorithm concludes that the data has remained stable and the sample rate is decreased until a minimum sample rate is reached.

Compressive Sensing

Compressive Sensing (CS) is a technique to compress data as part of the sampling process [14]. In the context of a WSN, a sensor is sampled at pseudo random intervals and the signal is reconstructed using ℓ_1 optimisation at the sink. A key problem is knowing beforehand how sparsely to sample the signal. Chen and Wassell [19] proposed an adaptive approach to adjusting the sampling rate in order to maintain a particular reconstruction quality by feeding back to the sensor node during operation.

Although Chen, Rodrigues and Wassell [18] have examined CS performance in the context of the Intel Lab data set, the aim of their work was not to minimise transmissions, and the results were not in those terms. Thus, it is difficult to know how it will compare with model-based compression algorithms. Nevertheless, it is a promising approach that has a key advantage that it reduces sensing energy as well as transmission cost. This should make it suitable for sensors that require significant energy to operate.

2.5.5 Reducing Node Power Consumption: Summary

This section has reviewed techniques for reducing sensor node power consumption. While hardware and network optimisations can offer scope for reducing the energy used by a sensor node, they can be specific to a given hardware platform or network topology.

Algorithmic approaches to energy reduction focus on the use of software to reduce the amount of energy used by sensor nodes. An advantage of algorithmic approaches is that they can be independent of the hardware or network topology, and can complement optimisations at this level.

Data-driven approaches make use of the characteristics of the sensed phenomena to reduce the amount of redundant information collected or transmitted by a sensor network. Data driven approaches to energy reduction in WSN can provide a significant increase in node lifetime. While approaches such as approximate caching are simple to implement, and the parameters are easy to tune, they do not provide the level of transmission reduction offered by more complex models. More complex approaches that make use of ARIMA models or linear filters offer better compression performance, but can require a significant amount of training or tuning to achieve optimal performance.

It is clear that there is a need for a simple DPS algorithm that can provide increase the lifetime of a sensor node while providing a timely, accurate representation of the data collected by a sensor network. The Spanish Inquisition Protocol (SIP) algorithm presented in Chapter 4 addresses this gap.

2.6 Dealing With Missing Data and Predicting Future Values

The previous section discussed model based approaches to reducing the amount of data transmitted by a sensor node. Sometimes nodes may not be able to successfully transmit data in a timely manner, leading to missing samples in the data stream received at the sink. This means that it is important to have a mechanism to fill in these missing values (imputation); additionally, applications may need to estimate future sensor values (prediction).

This section describes imputation (Section 2.6.1) and prediction (Section 2.6.2) techniques as well as combined imputation and prediction (Section 2.6.3).

2.6.1 Imputation

Missing values in the data collected by WSNs are common. For example, the sensor data yield in the commonly used Intel Lab data set [63], is between 0% and 66% with average yields of around 43% (Figure 2.4). There are several reasons for these missing samples such as: node failure, unreliable sensors, packet collisions, communication failure or lack of power.

One method of dealing with missing values in WSNs is to have the node retransmit the missing packet. However, given that the cost of transmitting a packet through the network is high, this approach can quickly deplete the available energy. Making use of model based approaches to transmission reduction (Section 2.5.3) can reduce the impact of these difficulties. Once the sink has received a model from

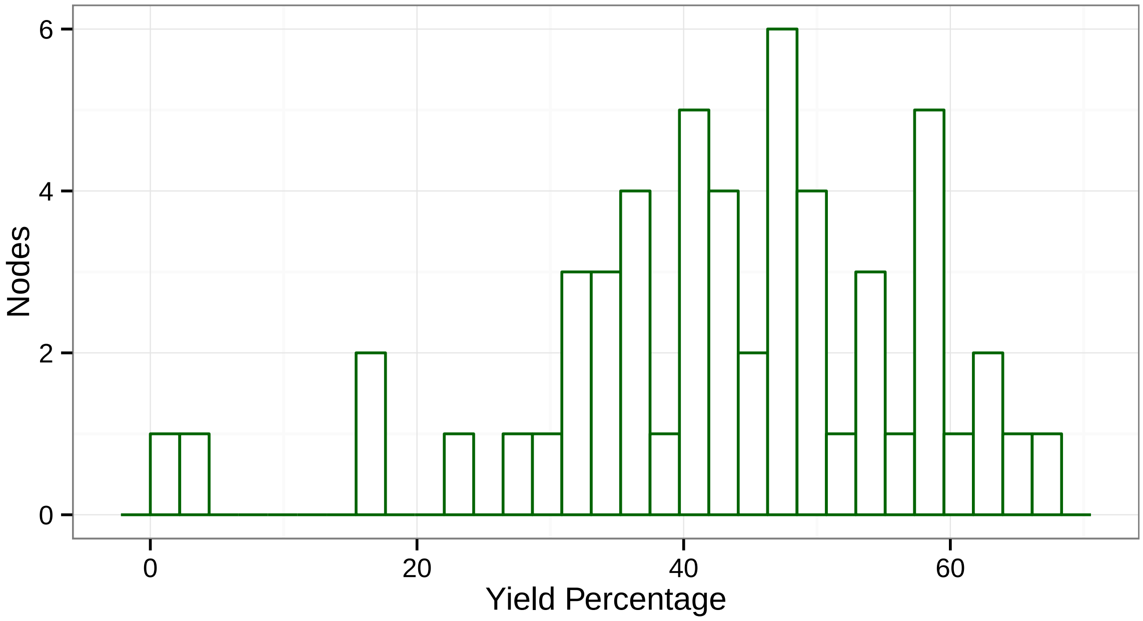


Figure 2.4: Histogram of node yields for Intel lab data set

the node, this model is used to estimate future sensor readings. These schemes reduce the amount of data a node needs to transmit which leads to an increase in available bandwidth in the network for acknowledgements to be sent.

These missing values can be problematic when analysing the data collected. Higher level analysis or classification tools may be ill-equipped to deal with these missing values. Therefore, filling in these missing values with an estimate of the sensor reading is a vital tool for preparing data for further analysis. This technique is called *imputation*. There are a variety of imputation techniques some of which are more accurate at predicting the missing values. Techniques that produce more accurate estimates tend to use complex models or computationally demanding algorithms.

Imputation has been widely used as a technique of statistical analysis, with techniques such as regression [10] common. An advantage of WSN when it comes to imputation is that in dense networks the sensor readings are likely to be correlated. Therefore, the output of any data imputation technique can be improved by exploiting correlations between nodes.

The Applying K-nearest neighbour Estimation (AKE) approach presented by Pan and Li [80] makes use of correlations between nodes to estimate sensor readings. A linear regression model is used to describe the spatial correlation between nodes, based on periods where sensor data is available. This model is then used to calculate the estimated reading for a missing data point based on a weighted average of the regression model based values of its nearest neighbours.

The authors of AKE suggest several refinements to improve imputation (Li and Parker [58][59]). These strategies include: doing nothing, using a fixed constant, using a moving average of past readings, using the last seen sensor reading, and an estimation maximisation approach. The approaches involving moving averages, and last seen sensor readings can be viewed as a form of the approximate caching approach reviewed in Section 2.5.3. These refinements are compared to a nearest neighbour based approach that makes use of the most common reading from neighbouring nodes.

Imputation is used to fill in missing data but to predict future values a forecasting techniques is required as discussed in the next section.

2.6.2 Forecasting

Forecasting is closely linked to imputation. However, rather than estimate missing values between existing readings, forecasting is concerned with the prediction of future values from historical data.

Traditional time series forecasting techniques such as regression, Holt-Winters or Box-Jenkins approaches [10] have been shown to be effective when forecasting time series data. However, in the general case they do not make use of the correlations between data streams that are commonly found in WSNs. One key assumption that the time series forecasting methods mentioned previously make is that the pattern identified in the historical data will continue into the future [10]. While this may not present much of a problem for short term *one step ahead* predictions, the accuracy of longer term predictions can be affected by unexpected trends or by accumulated model error [98]. Long term forecasting will take one of two approaches. *Recursive* forecasts perform one step ahead predictions until the desired time frame is reached. *Direct* approaches, such as the methodology put forward by Sorjamaa et al. [98] tend to use multiple prediction models for each time step. While the recursive approach is simple, it can suffer from forecast errors being accumulated between each time step. On the other hand, a direct approach requires multiple models to be trained and is therefore computationally more expensive.

Exploiting correlations between data streams using multivariate models has been shown to improve long term forecast performance. However, these approaches are not without limitations. For example, ARIMA models require a complete data set with fixed sampling intervals [16] to produce a forecast; similarly, a multivariate Holt-Winters approach proposed by Corberàn-Vallet, Bermùdez and Vercher [21] relies on each stream having statistically similar properties, which is not always the case.

Forecasting and imputation techniques tend to be applied in isolation either to predict a future data point, or fill in gaps in incomplete data sets. The data collected by WSNs tends to have missing values due to the lossy nature of communication. The analysis applied to this data may expect complete data streams therefore imputation is important; additionally, analysis may benefit from prediction of future values.

2.6.3 Gaussian Process Regression

The imputation and forecasting techniques presented in the previous section rely on an appropriate model being used to estimate sensor values. However, with complex data it can be difficult to determine the underlying model. Machine learning algorithms such as neural networks have historically been used for such multivariate regression problems. In an attempt to address practical difficulties and simplify design decisions (such as choice of learning rate) associated with supervised neural networks [108], Bayesian techniques such as Gaussian Process Regression (GPR) [88] have been proposed. GPR (also known as Kriging) [32, 51], has been widely used in the geostatistics domain but is only recently being applied to other fields, such as sensor networks [76, 75]. As GPR can be applied to both imputation and prediction, it provides an ideal method for dealing with the missing data and forecast problems tackled in this thesis.

While GPR is established for imputation and short-term forecasting (for examples, see Williams [108] and Rasmussen and Williams [88]), applying this approach to long term forecasts is still an active research area. Methods of improving multiple step ahead forecasts include incorporating the prediction uncertainty in the model [33], and exploiting correlations between data streams [16]. A study by Yan, Qiu and Xue [110] examined both recursive and direct approaches to forecasting using a single input GPR. During experimentation it was shown that the most appropriate approach differed depending on the data set. While in the general case the direct approach did outperform recursive forecasting, the difference between each model was statistically insignificant. However, this approach only evaluated single input GPR, where no correlation between data streams was considered.

With regard to using GPR to evaluate data collected by WSNs, Osborne et al. [77] present a computationally efficient GPR, motivated by the need for autonomous, real-time processing of data collected by WSNs. Correlations between sensor are included as part of the input to the Gaussian Process (GP), and have been shown to improve the output of the algorithm. The approach supports both regression and short-term prediction. However, long-term prediction has not been considered. The approach put forward by Osborne et al. [77] improves on conventional approaches to GPR by taking an iterative approach to calculating the covariance matrix, that reduces the computational complexity of an update from $O(N^3)$ to $O(N^2)$. Rather than have the GPR consider each prediction as a separate occurrence, this approach considers streams of ordered sensor data. As each update arrives only a few items in the input are changed, and thus much of the work used to calculate the covariance matrix can be reused. The iterative approach is also able to bound the size of the input by using a *downdating* function that removes "stale" (in this case the oldest) data from a fixed-size window, under the assumption that older data is less useful for prediction. Furthermore, by examining the uncertainty of the prediction, the downdate mechanism can be used to select when a an update is needed for the input data set. Furthermore, the update mechanism can be used for *active selection* to determine when a particular sensor should take a sample, which may reduce the cost associated with sensing and transmission.

The approach was evaluated against several real world data sets, as well as in an trial deployment. When used to estimate values for historical data, the use of correlated readings was demonstrated to significantly outperform both independent GP and other prediction techniques such as Kalman filters. The use of correlation also made it possible to accurately predict a sensor's readings that did not follow established patterns. The active selection approach was also able to exploit correlated readings to significantly reduce the number of samples required to maintain the prediction.

While the iterative update approach has been shown to be computationally efficient, it relies on ordered samples arriving in the data stream. In some cases samples may arrive out of order, and thus the iterative approach may not be feasible. Further, while one step ahead forecasting has been evaluated, the appropriateness of the downdate mechanism when performing long term forecasting is not clear.

GPR Algorithm

GPR uses GPs to provide a probabilistic framework for regression. The observations in a given data set can be seen as a sample from some multivariate Gaussian distribution. The relationship between these observations is defined by the covariance function $k(x, x')$, representing the correlation between samples.

The regression process begins by encoding the relationship between observations using the covariance function $k(x, x')$ for all combinations of inputs x into the covariance matrix K,

$$K = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{pmatrix} \tag{2.1}$$

At each regression step, the covariance between training data and the point of interest x_* (the point to be predicted) is calculated, giving rise to two covariance matrices representing the relationship between observed data and the point of interest.

$$K_* = \begin{bmatrix} k(x_*, x_1) & k(x_*, x_2) & \cdots & k(x_*, x_n) \end{bmatrix} \tag{2.2}$$

$$K_{**} = k(x_*, x_*) \tag{2.3}$$

Using the above notation, the input data is defined as a sample from a Gaussian distribution, as follows,

$$\begin{bmatrix} y \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right) \tag{2.4}$$

For prediction, the aim is to determine conditional probability of test values y_* given the observed data y. This is itself a Gaussian with probability,

$$P(y_*|y) \sim \mathcal{N}(K_* K^{-1} y, K_{**} - K_* K^{-1} K_*^T) \tag{2.5}$$

From this, the value at x_* can be inferred by taking the mean of this distribution,

$$\bar{x}_* = K_* K^{-1} y \tag{2.6}$$

Rather than directly invert the matrix as in the above equations, a practical implementation of Gaussian Process Regression makes use of a computationally faster approach, such as Cholesky decomposition. A more detailed explanation of the theory of Gaussian Process Regression is provided by Rasmussen and

Williams [88].

Covariance Functions

A key part of using GPR to generate a prediction y_* is the design of the covariance function $k(x, x')$ used to create the covariance matrix K. The reliability of the regression is dependent on a well-specified covariance that allows the complexities of the input data set to be captured.

The covariance function relates one observation to another. Since observations made at points close to x are likely to be well correlated it can be assumed that the output for any target value x_* will also be correlated to nearby observations.

Although a simple, single term covariance function will suffice for some data sets, it is often the case that the data will have numerous, complex features that need to be modelled. For instance, a data set may be modelled to be the sum of two independent functions $f_1(x)$ and $f_2(x)$. In this case, the covariance function can be extended accordingly by adding terms for each function. A parallel can be drawn here to other time series forecasting techniques, such as Holt Winters' exponential smoothing [9], where extra parameters are introduced to encode multiple trends in the data.

To provide accurate predictions, it is important that terms of the covariance function can assert assumptions about patterns in the data gathered.

The generalised Matèrn covariance function is

$$k_{matern} = h^2 \frac{2^{1-v}}{\Gamma(v)} \left(\frac{\sqrt{2vd}}{w} \right)^v K_v \left(\frac{\sqrt{2vd}}{w} \right) \tag{2.7}$$

where $d = |x - x'|$ is the absolute distance between input samples, v is a smoothness parameter (with higher values of v giving smoother functions), K_v is the modified Bessel function and Γ is the Gamma function [2]. The height parameter is denoted by h and w is the width parameter. these are commonly known as *hyperparameters* of the Gaussian Process and are discussed in more detail later in this section.

The mapping between points in time can be modified to model periodic functions (such as diurnal patterns) by replacing the input space function $d = |x - x'|$ with $d = sin\ \pi|x - x'|$.

The width and height Hyperparameters w and h can be used to alter the behaviour of the function. The width parameter w affects the relationship of the distance between samples and the output, tuning to what degree samples further from x should influence the covariance. Larger values for w produce smoother functions, but likely at the expense of short term detail (and hence model fit). The output height h is effectively a scaling factor on the output of the covariance function, limiting the variance of the covariance function output; h is typically set to be the standard deviation of the observed data following the recommendation of Rasmussen and Williams [88].

In monitoring situations it is not possible to know the ground truth measurements, as the sensors can only observe a noise corrupted version. To account for the estimated measurement noise, the covariance function can be extended as follows

$$K(t, t') = k_1(t, t') + k_2(t, t') + \sigma_n \delta(t, t') \tag{2.8}$$

where σ_n represents the estimated measurement noise and δ is the Kronecker delta.

To capture the correlation between multiple inputs, a multiplicative term over the sensor identifier l is applied to the covariance. This term is defined by the Pearson r correlation coefficient [91] between the two sensors under consideration.

When considering time data alone, as the relationship between inputs and outputs decreases, the output of the covariance function approaches zero, and accordingly has less impact in the regression process. A similar approach can be taken to represent the correlation between separate nodes, by weighting the output of the covariance function by a correlation coefficient representing the relationship between sensors. For highly correlated sensors ($r \approx 1$) the output of the covariance remains relatively unchanged. The relationship between uncorrelated sensors ($r \approx 0$) is reduced. During the calculation of the covariance matrices K and more specifically K_*, the weighting the Gaussian Process gives to uncorrelated sensors when producing the conditional probability of test values is reduced accordingly.

The extended covariance function K over x and l then becomes

$$K([x, l], [x', l']) = k_{node}(l, l')(k_1(x, x') + k_2(x, x')) + \sigma_n \delta(x, x') \tag{2.9}$$

This allows the covariance function to capture temporal and spatial changes.

2.6.4 Summary

Missing data values in the data collected by WSN are common, these missing values can impact the understanding of the phenomena. Imputation or prediction of these missing values can aid higher-level analysis. GPR is a machine learning technique that can be use for both imputation and prediction. The use GPR offers some benefits in specifying the model used for prediction over traditional time series prediction techniques. Where sensors are well correlated, the values predicted by GPR can be improved by incorporating these correlated values into the predictive model. GPR uses a covariance matrix to describe the relationship between points in the data stream. Several covariance functions each describing a feature within the data can be combined into a larger function describing the characteristics of the data stream. In this way complex data can be encapsulated within a single model.

Long term prediction of sensor readings can be achieved by either directly, or recursively predicting one step ahead. Direct predictions require a separate model for each predicted time step and are therefore more computationally intensive, recursive predictions while less complex may accumulate prediction error. However, studies with GPR have shown little difference between the two approaches. It is also unclear whether the use of correlation can improve the accuracy of long term recursive predictions made using GPR

2.7 Chapter summary

This chapter has reviewed literature relevant to the work presented in this thesis. The key topics discussed were WSN design choices, increasing node lifetime and handling missing data.

The WSN domain lacks a design pattern based view to application design. A framework for the design of WSN applications is discussed in Chapter 3

With regard to data driven approaches to increasing node lifetime. Current approaches to model based transmission reduction are either simple with poor performance, or have better performance at the expense of complexity. There is a clear gap in the literature for a simple, timely and accurate approach to transmission suppression, this gap is addressed by SIP discussed in Chapter 4.

Node or network failures often mean that the data streams collected by WSN have periods of missing sensor readings. Tools for high level analysis or classification of sensor network data is often ill-equipped to deal with these missing values, GPR can be used for both imputation of missing readings and prediction of future sensor values. Currently GPR has not widely been applied to the WSN domain. Current implementations of GPR for WSNs have addressed computational complexity. However, it is unclear how these approaches will deal with out of order data caused by network latency. Additionally, there has been no on-line method for long term prediction of sensor readings proposed. These factors are addressed in Chapter 5.

This chapter has reviewed the relevant literature, and motivated the work presented in the rest of the thesis. The following section presents a conceptual architecture for the structured design of WSN applications.

Chapter 3

FieldMAP: A Generic Wireless Sensor Network Architecture

3.1 Introduction

This chapter presents *FieldMAP*, a conceptual Wireless Sensor Network (WSN) architecture that aims to capture features of a common data flow found in WSN applications. The use of such an architecture allowed the research questions explored in this thesis to be conceptually examined within the context of a generic WSN, without the need for a specific application, hardware platform, or operating system. The framework serves as a reference design for the work presented in the remainder of this thesis.

The FieldMAP architecture lays claim to a middle ground between the *sense and send* approach and architectures that involve sophisticated interaction between neighbouring wireless nodes. It makes more use of computational ability at the node than many current real-life deployments, while still being more practical than many reported theoretical approaches that have rarely been implemented or deployed.

The architectural components presented in this chapter are based on a set of data processing stages found within a conceptual WSN monitoring application. Each stage represents a distinct phase where data is gathered or processed. While not all real world WSN applications will have exactly the same set of stages (and associated requirements), many will share a subset of the functionality described.

The architecture supports multi-modal sensing, provides opportunities for data processing and information extraction, allows for real-time phenomena visualisation, data and information logging and post-analysis. The architecture caters for applications that use both *sense-and-send* and *on node processing*, through the use of a data modelling component to aid on-node processing of data. However, given the tight coupling needed between communication protocols and inter-node data aggregation, the architecture specifically excludes any node to node communication for data processing or aggregation.

The remainder of this thesis builds upon the *data modelling component* of the architecture at node and sink, to develop a novel transmission reduction strategy aimed at increased WSN longevity in (Chapter 4), and to develop a strategy for estimating missing sensor readings (Chapter 5).

The rest of the chapter is organised as follows: Section 3.2 presents the framework requirements, Section 3.3 presents a formal description of the architecture and its components, and finally the chapter concludes with a summary in Section 3.4.

3.2 Requirements and Data Flow

This section presents the requirements and data flow for a conceptual WSN monitoring application. The range of potential applications for WSN sensing systems is vast, from habitat monitoring [66], sniper localisation [97] and Body Sensor Networks (BSNs) [29] to building monitoring [23].

Although the type of data collected and the mechanics of data transfer are application specific, the basic set of requirements for applications such as the above are the same: collecting data from a set of spatially distributed sensors and transferring this to a sink for storage and analysis. Performing some processing or modelling of data in a decentralised way is also desirable, as it can increase the informational output of the network while reducing the amount of raw data transmitted. The conceptual application considered here thus includes provision for on-node processing of data.

WSN deployments are often experimental, thus the ability to modify functional components will allow the experimental design to be modified based on information gathered during prototype deployments and trials. Therefore the conceptual application takes a compartmentalised design approach which allows for structured and rapid prototyping cycles.

While no requirements at the network level are specified, the architecture is intended to be used in conjunction with modern routing protocols that aim to extend battery life and cope with intermittent communication links in an ad-hoc multi-hop network.

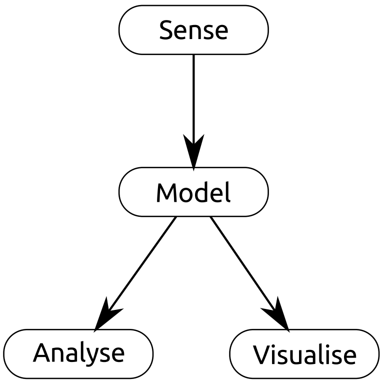


Figure 3.1: Conceptual overview of a WSN application

Given the description above, the basic data flow within this generic WSN application will look somewhat like the one given in Figure 3.1. Data is gathered (*sense*) at a number of locations within the area being monitored, this data is then processed (*model*) into user relevant information before being displayed to the end user (*visualise*) or stored for later analysis (*analyse*).

Requirements for the conceptual monitoring application can be loosely stated as follows:

- Sense environmental parameters at a number of locations.
- Cater for on-node processing.
- Transmit measurements to some base station in an energy efficient way.
- Provide a real time representation of the phenomena being sensed.
- Store the logged readings for later analysis.

Given both the data flow and requirements, the architectural components as defined by Wolf et al. [109] can be derived and they are described in the following sections.

3.3 Architecture Description

As specified above, a WSN application is defined as one involving a distributed set of processing nodes $N = \{n_1, n_2, \ldots\}$. All nodes report to a single base station (or sink node), which supports information visualisation and storage.

Algorithm 3.1 FieldMAP Node Algorithm

```
function SENSE
    $A_t \subseteq S$                                        ▷ Get Active Sensors from attached sensors
    $\mathbf{z}_t \leftarrow sense(A_t)$                       ▷ Vector of readings from Active Sensors
end function

function MODEL
    $\mathbf{x}_t \leftarrow f(\mathbf{z}_t, A_t, \mathbf{x}_{t-1})$          ▷ Filter reading vector $z_t$ using function $f()$
    detected $\leftarrow e(\mathbf{x}_t, \mathbf{x}_{t'}, t')$              ▷ Detect events based on function $e()$
    if detected then
        $enqueue(\mathbf{x}_t)$                               ▷ Add state vector to transmission queue
    end if

end function
function TRANSMIT
    $ordered \leftarrow g(txQueue)$                            ▷ Order Transmission Queue
    while transmission queue is not empty and no sampling scheduled do
        $tx \leftarrow ordered.pop()$                           ▷ Pop First Item from the Queue
        $time \leftarrow t$                                      ▷ Append sensing Time
        $transmit(tx)$
    end while

end function
$schedule()$                                                   ▷ Schedule next sampling period
```

3.3.1 Data Elements

The main element used to transfer data between components is the *state vector.*

The state vector $\mathbf{x}_t$ is partitioned into a data portion (corresponding to the state of the phenomena being sensed) and a management portion (corresponding to the state of the sensors).

In cases where only a subset of the sensors attached to the node are sampled, the same vector based approach can be used with null values for sensors where no sample was requested. To allow processing stages to take account of sensors that have no expected sample, a bitmask of currently *active* sensors can be appended to the management portion of the state vector. In cases where the packet size is critical, where a bitmask is used, the state vector can also be compressed by removing non-present entries from the data or management portions of the state vector.

3.3.2 Node Processing Components

Each node n is assumed to gather readings from a set of attached sensors S. At each sampling instant t each node n must:

1. **Sense:** Given a set of sensors $S = \{s_1, s_2, \ldots\}$ attached to the node n, sense a vector of parameters $\mathbf{z}_t$ from the *active* subset $A_t \subseteq S$. The vector may involve parameters of different modalities, and may consist of a time-series "chunk", such as a one second microphone sample.

2. **Model:** On the node, the modelling process is composed of two distinct phases:

   (a) **Filter and manage faults:** Transform the vector of sensor readings $\mathbf{z}_t$ into an estimate of the current state $\mathbf{x}_t \leftarrow f\left(\mathbf{z}_t, A_t, \mathbf{x}_{t-1}\right)$, as described in section 3.3.1 In general, this transformation function f must take account of which sensors are active on the node (A_t) when the sample was requested. Transformation functions can also make use of the previous state estimate $\mathbf{x}_{t-1}$, to infer how a system is evolving. To allow the model function to take account of when a sample was collected the state estimate should include the time when the sample was gathered.

   Where there are a series of uncorrelated sensors, or several filtering operations to be performed, the filter can be implemented as a series of functions $[f_1, f_2, \ldots]$ that each operate on all or part of the state vector $\mathbf{x}_t$.

   (b) **Detect events:** based on a predicate function $e\left(\mathbf{x}_t, \mathbf{x}_{t'}, t'\right)$, where $\mathbf{x}_{t'}$ is the last transmitted state and t' is the time of the last transmission, decide whether to transmit the transformed vector $\mathbf{x}_t$ to the base station (for example, to allow event detection through threshold rules). By considering the last transmitted state, the event detection predicate can identify the value to the receiver of the new state information. The event detection decision is made locally, without reference to other locations or nodes. The node architecture does not provide for considering global information or even neighbouring nodes to aid event detection but in some situations this may be an appropriate extension.

   (c) **Queue:** If an event is detected, append the vector $\mathbf{x}_t$ to the transmission queue.

3. **Transmit:** while a channel is available and the transmit queue is non-empty, sort the queue according to some priority ordering $g\left(\mathbf{x}_t\right)$, transmit the first sample in the queue, and update the last state time t'. The transmission time $t_{transmit}$ is included in the message.

Algorithm 3.2 FieldMAP Sink Algorithm

```
$\mathbf{x}_{n,k} \leftarrow receive(\mathbf{x}_t, t, n)$                ▷ Store Received Sample

function Model
    $\Delta t_n \leftarrow t - k$                ▷ Calculate time delta for last received sample
    $N_{fresh} \leftarrow (n \mid n \in N \textit{ and } \Delta t_x \leq t_{max})$                ▷ Calculate fresh samples based on threshold
    $N_{stale} \leftarrow N \setminus N_{fresh}$                ▷ Stale sensors are those who are not fresh

    if $k > t^{'}$ then                ▷ If a more recent sample
        $\mathbf{y}_{n,t} \leftarrow m\left(\mathbf{x}_{n,k}, t - k\right)$                ▷ Update model using the most recent sample
    else
        $\mathbf{y}_{n,t} \leftarrow m\left(\mathbf{y}_{n,t^{'}}, t^{'}\right)$                ▷ Estimate state using prior state estimate
    end if
end function

$visualise(\mathbf{y}_{t,n})$
```

4. **Schedule:** Based on the last attempted sampling time for each sensor, and the required sampling rate for each sensor, schedule the next sampling period. In the simplest case, this may involve gathering a sample from each sensor then sleeping for a fixed duration. A discrete event scheduling approach could also be used to allow different per sensor sample rates.

3.3.3 Sink Components

The base station has an on-line, real-time component, which performs the following generic procedure:

1. **Store:** store received vectors $\mathbf{x}_{n,k}$ for replay, post analysis, and deployment management, along with their associated node n and receipt time k. The transmission time does not have to be stored, as it is already encoded in the state vector by the node. With transmission and receipt times, the two clocks, one at the node and one at the base station, need not necessarily be synchronised.

2. **Model:** on the sink the modelling component consists of two phases:

    - **Identify fresh locations:** given a newly received state estimate $\mathbf{x}_{n,t}$, the state estimate "age" is updated to $\Delta t_n \leftarrow t - k$, where t is the current time, and k is the receipt time. *Fresh* nodes are those whose sample age is within some limit Δt_{max}; Conversely, *stale* locations are those that are not fresh. Identifying nodes where data has been received within some given time limit as fresh allows any modelling function to take account of a sample's age when estimating the state of the system, thus allowing samples that have expired to be excluded from the model, or to be predicted with an alternate model. This will mean that the output of any such model is not affected by out of date information, (for example, if a node has stopped transmitting due to sensor or communications failure).

- **Update node state:** predict the current state as,

$$\mathbf{y}_{n,t} \leftarrow \begin{cases} m\left(\mathbf{x}_{n,k}, t-h\right) \text{ if } k \geq t' & (a) \\ m\left(\mathbf{y}_{n,t'}, t-t^{'}\right) \text{ otherwise} & (b) \end{cases}$$

  Where m is a model of the evolution of the state, $t^{'}$ is the time for which the location state was last updated, and $\mathbf{y}_{n,t'}$ is the previous prediction for the node. The top case (a) is where newer information has arrived, and the bottom (b) where received information is older than that used to estimate the previous prediction. This approach allows the evolution function to continue to estimate the current state when no recent sample has been received (for example due to communications failure).

3. **Visualise:** display the information given by the current state estimate $\mathbf{y}_{t,n}$ to the user in an appropriate form.

4. **Recall / replay:** on request, data from a previous period can be recalled and optionally replayed, showing the evolution of the data over a specified period of time.

5. **Feature / anomaly extraction:** identify features or anomalies based on a set of rules, given $g\left(\mathbf{y_{n,t}}\right)$. For example, in a household monitoring application a typical rule might be to provide an alert if a temperature sample exceeds 30 ℃.

3.4 Summary

In this chapter, FieldMAP was presented and defined as an conceptual framework for WSN systems. This framework is intended to serve as a foundation for the algorithms proposed in Chapter 4 and Chapter 5 of this thesis, providing a reference point for their implementation, without relying on a particular application. In the remainder of this thesis, the contributions to knowledge are examined within the context of the framework.

Frameworks such as FieldMAP are a common computer science design tool, providing a common language and reference point for application designers. These frameworks break software into components with specific roles and responsibilities. Rather than a specific implementation, such frameworks provide a general description of functionality. The benefits of a compartmentalised design include increased scope for software reuse, as well as providing software designers with a structure within which to design and evaluate individual components.

A key part of the FieldMAP framework is the *modelling* phase that takes place at both node and sink. In many cases, the informational output of an WSN can be improved by performing some data processing in a decentralised way. Chapter 4 builds upon the *model* stage on the node and proposes a method for reducing network energy consumption through transmission suppression. While the framework does not specifically cater for inter-node communication, the model phase at the sink may make use of its global view of the data to perform aggregation. Chapter 5 focuses on applying a new *model* to the sink, allowing the prediction of missing data samples and providing an increase in network resolution.

Another key component is the use of a *state vector* to transfer data between each node and the sink. While the mechanics of data transfer, such as routing algorithms, remain application specific, the use of a state vector provides a standardised way to move data between both components of the framework and nodes within the network. The state vector encapsulates the state phenomena sensed by the node in two components: data and management. The data portion of the state vector holds information about the sensor readings, while the management portion holds information about the state of the sensors, or the node.

In the following chapter, the use of a state vector and modelling component of the framework is explored. Encapsulating trends in the sensed data within the state vector, allows a predictive model to be used to reduce the number of transmissions required to reproduce a data stream at the sink.

Chapter 4

The Spanish Inquisition Protocol

4.1 Introduction

The previous chapter discussed FieldMAP, a framework for Wireless Sensor Network (WSN) design. This chapter presents the Spanish Inquisition Protocol (SIP), which builds on FieldMAP with the aim of increasing network lifetime by reducing energy consumption. The algorithm reduces energy cost by transmitting only unexpected information and is so-named because "Nobody expects the Spanish Inquisition!" [70].

SIP is an instantiation of the node *model* component within FieldMAP and makes use of the state vector to support a transmission suppression scheme to reduce the amount of data transmitted, thus reducing the energy cost of using the radio. The algorithm aims to reduce transmission cost while maintaining both timeliness and the sink's ability to reconstruct the rich data stream required by many WSN applications. It does so by suppressing those samples that report a predictable evolution of the system being monitored.

The approach is based around encoding the data using a simple, approximate model of the phenomena. This model is shared by both node (transmitter) and sink (receiver). Rather than transmit each sample, the parameters for the model are transmitted from node to sink when they change significantly. Specifically, the node assumes that the sink can apply the model and predict the current state of a sensor. By keeping track of what the sink knows, the node can identify when the error in the sink's prediction will exceed some predefined threshold, triggering an update message to be sent to the sink. Additionally, many applications can tolerate some uncertainty in the measurement (for example, household temperature monitoring may only need an accuracy of ±0.5 ℃). SIP allows a trade off between accuracy and the number of transmission though an error budget which can be tuned to the desired accuracy, with a lower accuracy typically leading to a reduction in the number of transmitted packets.

SIP extends prior work on Dual Prediction System (DPS) algorithms (see Section 2.5.3) by transmitting a state vector rather than individual readings. This means that the state of the phenomena is taken into account when any predictions are made, allowing not only phenomena with a steady state to be predicted (as in Piece-wise Constant Approximation (PCA) Section 2.5.3) but trends in the data stream to be taken into account. The state vector approach also allows multiple sensor types to be encoded in a single predictive model. This is an improvement on prior DPS work that has so far focused on compressing individual streams.

The research questions this chapter attempts to address are:

1. Is transmission reduction beyond the current state of the art possible by combining model based filtering with dual prediction approaches in environmental monitoring applications?

2. When using a dual prediction based approach is there a consistent relationship between the number of transmissions required and the error budget for a given data set?

3. Can combining multiple sensors readings into a single predictive model allow a greater reduction in the number of packets transmitted, than when compressing each stream individually?

The rest of this chapter is organised as follows. Section 4.2 examines the motivation behind the protocol. A description of the SIP algorithm is given in Section 4.3 followed by experimental results over real world data sets in Section 4.4. A summary of the chapter is given in Section 4.5

4.2 Motivation

WSNs are deployed to monitor, control, and improve understanding of environments and phenomena. Domain scientists and industrial technicians need robust and reliable data to allow them to find patterns and confirm hypotheses about how the monitored phenomena changes over time. When monitoring is being used to ensure that the state of a system remains within certain bounds, the timeliness of the information is also important so that corrective action can be taken quickly.

A critical factor in the design of WSN systems is often the energy cost of communicating the data [85]. Although other components, such as sensing and processing, play a part in the energy budget, these are typically much less than the energy cost of communication.

In general, WSN applications collect time series data that are used to examine the evolution of the phenomena's state over time, allowing not only an understanding of what is actually happening now, but also the events leading up to the current state. However, many time series tend to contain little relevant information per bit of data (*High Data — Low Information*). Given the need to conserve energy in WSN deployments and the high cost of using the radio, transmission of redundant data can represent a significant proportion of the energy budget.

A common strategy to reduce the amount of redundant data transmitted is to reduce the sampling frequency. However, this has drawbacks. Firstly a higher than Nyquist sampling frequency provides insight into the evolution of the phenomena, giving a greater understanding of the system under consideration. Furthermore, event detection is desirable, and a low sample rate increases the chance that a significant data stream event will be missed.

The motivation for SIP is to provide a simple method that allows a reduction in the amount of data transmitted while still allowing the original data stream to be reconstructed within a user specified error budget.

4.3 Algorithm Description

SIP builds upon prior work in Dual Prediction Systems (DPSs) (See Section 2.5.3). These algorithms use shared predictive models at both node and sink to estimate the current state of the system. SIPs key advancement is the use of a state vector estimate as the predictive model. In brief, for DPS rather than transmitting each sample to the sink, the node estimates the current sensor reading using the shared model. The node's estimated value is then compared to the actual sensor value. If the values differ by more than a given error threshold, the model parameters are updated and transmitted to the sink for use in future predictions. When responding to a query, the sink uses its copy of the predictive model to generate estimated readings on request.

The SIP algorithm aims to transmit a stream of sensor data x at times $x_t = \{x_1, x_2 \ldots, x_n\}$ to the sink within some user specified error threshold ε such that the sink can reconstruct the sensor values

Algorithm 4.1 SIP Node Algorithm

```
 1: $\mathbf{z} \leftarrow$ querySensors()                                  ▷ Obtain vector of sensor readings
 2: $\hat{\mathbf{z}}_t \leftarrow$ filter $(\mathbf{z}_t)$                                           ▷ Filter Readings
 3: $\mathbf{x}_t \leftarrow f(\hat{\mathbf{z}}_t, \mathbf{x}_{t-1}, t_{-1})$                                   ▷ Estimate current state
 4: $\mathbf{x}_s \leftarrow (\mathbf{x}_{\text{sink}}, t_{\text{sink}}, t)$                                    ▷ Predict sink state
 5: $y \leftarrow \mathbf{x}_t - \mathbf{x}_s$                                          ▷ Calculate model error
 6: **if** $\exists k \in A_t : |y_k| > \varepsilon_k$ or $t - t_{\text{sink}} \geq t_{\text{heartbeat}}$ **then**                ▷ Detect Events
 7:     transmit $(\mathbf{x}_t, n, t)$                                      ▷ Transmit current model
 8:     $n \leftarrow n + 1$                                          ▷ Increment sequence number
 9:     **when** (acknowledgement received)
10:         $\mathbf{x}_{\text{sink}} \leftarrow \mathbf{x}_t$                                    ▷ Update copy of sinks state
11:         $t_{\text{sink}} \leftarrow t$
12:
13: **end if**
14: $\mathbf{x}_{t-1} \leftarrow \mathbf{x}_t$                              ▷ update previous state estimate with current
```

$x \pm \varepsilon$.

To achieve this, a simple, approximate model of the phenomena is shared between node and sink. A state vector forms the parameter for this model, allowing a forward prediction of the state of the phenomena to be made. Rather than report the last received sensor reading, the sink predicts the state of the phenomena based on the last received state vector.

Using knowledge of the last state vector transmitted to the sink, the node can identify when the error in such a prediction will exceed some threshold ε. Different applications will have different requirements for the error threshold. For example, household temperature monitoring might only need an accuracy of ±0.5 ℃ ($\varepsilon = 0.5$ ℃).

The algorithm on both node and sink is discussed in the next section.

4.3.1 Node Algorithm

Each time a new sample is acquired, the SIP algorithm on the node, compares the actual sensor reading against the most recent prediction. If these differ by more than a pre-specified error threshold, then the predictive model is updated and transmitted to the sink for use in future predictions, otherwise the node completes the sensing cycle and waits for the next sample.

At each time sampling interval t the node algorithm presented in Algorithm 4.1 performs *sensing* (1), *filtering* (2), *event detection* (3-6) and conditionally *transmitting* (7-11).

1. (*Sense*) Obtain a vector of sensor readings $\mathbf{z}_t$ from the currently *active* sensors. This follows the same

approach as put forward in the FieldMAP framework, for example, readings may not necessarily be gathered from all sensors at each time step.

2. (*Filter*) Filter the vector of sensor readings $\hat{\mathbf{z}}_t \leftarrow \text{filter}(\mathbf{z}_t)$. By filtering the data the algorithm aims to provide an estimate of the ground truth system state, rather than use the noise corrupted sensor value. The choice of filter is dependent on the application. However, simple filters such as the Exponentially Weighted Moving Average (EWMA) have been shown to perform well during experimentation. Filters are discussed in more detail in Section 4.3.4. At this stage it may also be useful to do some form of error detection, for example outlier detection.

3. (*Estimate state*) Derive the current state of the system. This stage estimates the current state of the phenomena under consideration, transforming the vector of filtered sensor readings $\hat{\mathbf{z}}_t$ into an estimate of the state $\mathbf{x}_t$. To incorporate evolution of the phenomena and allow forward prediction, the state estimation function $f(\hat{\mathbf{z}}_t, \mathbf{x}_{t-1}, t_{-1})$ will need to take account of the previous state estimate $\mathbf{x}_{t-1}$ and its associated time t as well as the current state.

   Where multiple sensors are involved, the method used to encapsulate state may differ. For example temperature or humidity data may use Piece-wise Linear Approximation (PLA), while Piece-wise Constant Approximation (PCA) is more appropriate for light levels. However, regardless of the estimation method used, the output of the state estimation function should contain sufficient information to allow prediction of the node state. For example, in the case of PLA model discussed above, the state vector will contain the filtered sensor reading and current rate of change $(\hat{z}_t, \Delta\hat{z}_t)$.

4. (*Estimate sink state*) $\mathbf{x}_s \leftarrow (\mathbf{x}_{\text{sink}}, t_{\text{sink}}, t)$ The node estimates the sensor values that will be reported by the sink $\mathbf{x}_s$ using its copy of the sinks predictive model, the time it was transmitted to the sink t_{sink} and the current time t. If using the PLA in the stage above, then linear prediction will be used at this stage $\hat{s}_t \leftarrow x_{sink} + (\Delta x_{sink} \times (t - t_{sink}))$

5. (*Detect Events*) Compare the current node state and estimated sink states and transmit a model update to the sink if required. $(\mathbf{y} \leftarrow \mathbf{x}_t - \mathbf{x}_s)$. $\mathbf{y}$ represents the difference between current system state and the predicted sink state. If any active element of the state vector exceeds some error threshold ε or the time since the last transmission exceeds some heartbeat time $t_{\text{heartbeat}}$ then the new state $\mathbf{x}_t$ is transmitted to the sink for use in future transmissions. Otherwise, the node

suppresses the state update transmission.

(a) (If error threshold or heartbeat time exceeded) $\exists k \in A_t : |y_k| > \varepsilon_k$ or $t - t_{\text{sink}} \geq t_{\text{heartbeat}}$ The difference between the current system state and the sinks estimate state is calculated. If this difference exceeds some threshold ε, then the sink's prediction will fall outside of the error budget. The use of a heartbeat time can help reduce the impact of sensor failure as the sink can detect when the heartbeat time is exceeded and will not report predictions for this node until a new model update is received.

(b) (Transmit current model) The new state estimate $\mathbf{x}_t$ is transmitted to the sink for use in future predictions, along with the sequence number n and node time t. The payload for the packet that is transmitted will consist of:

Sequence Number n Used to identify the packet number at the sink. In the case of transmission failure the sequence number can be used by the sink to identify missing packets.

Node Time t Used to identify when a sample was gathered.

State Estimate $\mathbf{x}_t$ A copy of the updated state estimate generated by the *model* phase on the node. When more that one datastream is being processed, this may consist of one or more state vectors corresponding to each sensing modality under consideration (see Section 4.4.8).

(c) (Update stored sink state) Upon acknowledgement of the packet being stored at the sink, the node updates its copy of the predictive model to match the one held at the sink $\mathbf{x}_{\text{sink}} \leftarrow \mathbf{x}_t$ and the time of transmission $t_{\text{sink}} \leftarrow t$. If no acknowledgement is received, retransmit the packet (with an appropriate back off mechanism). In the case of network failure it may be appropriate to store the packet locally until end to end transmission is possible. It is important that an end-to-end acknowledgement is used, as the accuracy of the algorithm is based on the node and sink sharing the same predictive model.

(d) (Update previous state estimate) At the end of each sensing cycle the current sink state and time are stored for use in the state estimation stage of the next sensing cycle $\mathbf{x}_{t-1} \leftarrow \mathbf{x}_t$.

Algorithm 4.2 Spanish Inquisition protocol (sink algorithm)

```
procedure RECEIVE($\mathbf{x}_t, n, t$)    ▷ Store received state vector and associated time
    $\mathbf{x}_{\text{sink}}(t) \leftarrow \mathbf{x}_t$
    $t_{\text{last}} \leftarrow t$
    acknowledge($n$)    ▷ Send acknowledgement
end procedure

procedure ESTIMATE($t$)    ▷ Estimate value for time $t$
    if $t \geq t_{\text{last}}$ then
        predict from $\mathbf{x}_{\text{sink}}(t_{\text{last}})$
    else
        interpolate from neighbouring $\mathbf{x}_{\text{sink}}$
    end if
end procedure
```

4.3.2 Sink Algorithm

The sink algorithm is given in Algorithm 4.2. The sink maintains a set of state vector and time stamp pairs for each node. Upon receipt of a new state vector from a node, the sink stores the state vector along with its associated time stamp. The sink then transmits an acknowledgement to the node to confirm that the state vector has been stored.

Rather than use raw sensor values to respond to queries, the sink will estimate the sensor value using the stored state vector. If a current sensor reading is required by the user, the value is predicted based on the last received state vector. Otherwise the sensor value is estimated by interpolating between temporally neighbouring state vectors.

4.3.3 SIP Assumptions

The SIPs algorithm makes several assumptions as described in the following sections.

Collecting sensor data is less expensive than transmitting

SIP assumes that sensing and processing of sensor data is less energy intensive than transmitting that data to the sink. In the general case this is a valid assumption (See Section 2.3). However there are cases when this assumption does not hold. For instance, certain sensor types such as CO_2, have a high sampling energy cost. This means that the most significant consumer of energy will be the sensor itself during the sensing phase. Thus reducing transmissions will only have a minor effect on the network lifetime. For such sensors with a high current draw, reducing the sampling frequency will have a greater effect on energy consumption. Here approaches such as compressive sensing [19] may be of benefit.

Guaranteed delivery scheme

In common to all DPS algorithms, SIP relies on the node and sink sharing the same predictive model. Therefore, the parameters used to estimate the sensor state must be synchronised for accurate predictions to be made. If the model parameters are not synchronised any prediction made at the sink will not accurately represent the true state of the node's readings, meaning that any prediction made could exceed the error budget, until the next state estimate is received. A guaranteed packet delivery scheme may be in place at the network level (through the use of protocols such as TCP), or simple acknowledgement mechanisms for the received state-vector being stored at the sink. This latter approach may be preferable since if a packet is received by the sink but not stored, the prediction mechanism on the sink will not return the same value as the prediction at the node, meaning any accuracy requirements cannot be met.

The data is predictable

The principle behind the algorithm is to reduce the amount of data transmitted by only reporting a change in the state of the sensed phenomena. This does not mean that the phenomena must be unchanging for best performance, but that its rate of change is steady. The PLA approach is valid for non-linear data because there will be sections of that data which exhibit linear trends. For example, a sine wave transmitted using SIP will still show a reduction in the number of packets transmitted when compared to the raw signal. Environmental temperature data shows a diurnal pattern, the rate of increase or decrease is relatively small. This characteristic can be found in many natural phenomena.

SIP may not perform well with data sets that contain many rapid changes of state over short periods of time, such as accelerometer data used to monitor body movement.

4.3.4 Choice of filter

To improve the estimate of the ground truth sensor reading, some form of filtering is generally needed. Selection of a filter depends partly on the model used and partly on the requirements of the application. Basic filtering can be performed using an EWMA filter. This recursive filter returns an estimate of the measurand that combines the current reading with past readings. Apart from removing some of the noise in the signal, the filter also smooths over quantisation introduced by the node's Analog to Digital Converter (ADC). Least Mean Squares Adaptive Filter (LMS) or Normalised Least Mean Squares

Adaptive Filter (NLMS) is a more sophisticated filter that has been evaluated, and has been used in prior dual prediction system work, as discussed in Section 2.5.3 .

The Kalman Filter (KF) or Extended Kalman Filter (EKF) are sophisticated approaches that use linear or nonlinear, respectively, models of the state of the environment. However, these are more computationally costly and may be more difficult to tune without prior knowledge of the phenomena.

To aid filter selection we identify three key characteristics for consideration:

Accuracy: How well the filter represents the underlying data stream

Responsiveness: How well the filter responds to transient events

Complexity: How computationally expensive the filter is.

Filter accuracy affects the representation of the filtered data stream. For example, filters based on a moving average introduce lag, where it takes some time for the value reported to reflect recent changes in the data. This delay is dependent on parameters such as the window size, or whether a simple or exponential moving average is used. Therefore these parameters should be selected based on the application constraints. The LMS adaptive filter [38] has less lag, but suffers from a period of convergence where samples reported during the convergence period are not representative of the underlying data. This can have a significant effect on the performance of the algorithm, as during this convergence period the filtered output differs significantly from the ground truth data. This can lead to the following issues:

1. The diverged filter reading is used in calculating the state vector. This means that the values reported by the forward prediction mechanism will be incorrect, and the error in the predicted and reported samples against the ground truth will increase.

2. The node may drop the diverged filter reading, meaning that information is lost during the start up process.

3. The node may transmit the raw sensor value, leading to an increased number of transmissions during the convergence period.

A filter's responsiveness denotes how quickly changes in the data stream are reported in the output of the filter. This becomes important when the application is used for event detection, or to monitor the state of a phenomenon in real-time.

The *complexity of a filter* is also a consideration. One aspect of a filter's complexity is the amount of computation required. Given that nodes within an WSN tend to have low processing capabilities, a computationally efficient filter will reduce the amount of time and energy required to process the readings. A second aspect of filter complexity is the number of parameters and ease of tuning of the filter. While filters with a greater number of parameters (such as the Kalman filter) can provide a more accurate representation of the data, in most cases these parameters require a significant amount of tuning to yield optimal results. This tuning process can require a significant amount of *a priori* knowledge of the data steam, and therefore makes the filter less useful for ad-hoc deployments.

4.3.5 Prediction mechanism

Algorithms such as SIP rely on a shared predictive model to allow the sink to successfully estimate the node's readings. Prior work with DPS has lead to several models being proposed in the literature. For example, the *adaptive filtering* approach proposed by Olston, Jiang and Widom [73] makes use of a simple piece-wise constant approximation, while Santini and Römer [95] propose the use of the LMS linear filter.

While more complex models may better fit the data stream and provide more accurate predictions, a case can be made for using simple approximations. As well as computational overhead, tuning complex models such as the Kalman filter approach proposed by Jain, Chang and Wang [50] to get the best results requires extensive *a priori* knowledge of the data stream. This in turn makes the algorithm less appropriate for deployment in ad-hoc networks.

Two simple models have been shown to produce good results with SIP, allowing the algorithm to cater for data sets with different characteristics without the need for extensive prior knowledge of the data.

Piece-wise Linear Approximation (PLA) is suitable for data that shows linear trends. Many environmental data sets (such as temperature and humidity) have a relatively slow rate of change between samples. The PLA model requires an estimate of the gradient (or rate of change) Δ, as well as an estimate of the point in time value of the measured. PLA is the model described in Section 4.3.1, and has been shown to perform well over a range of data sets. Figure 4.1 shows the output of the algorithm when PLA is used. Some notable trends in this modelling approach include the spikes in the predicted values during rapid changes in the underlying data. This is due to the rate of change calculation taking the difference between the two most recent samples. If there is a sudden change in trends within the data, then the estimated rate of change will be large, and may not accurately represent the general trend.

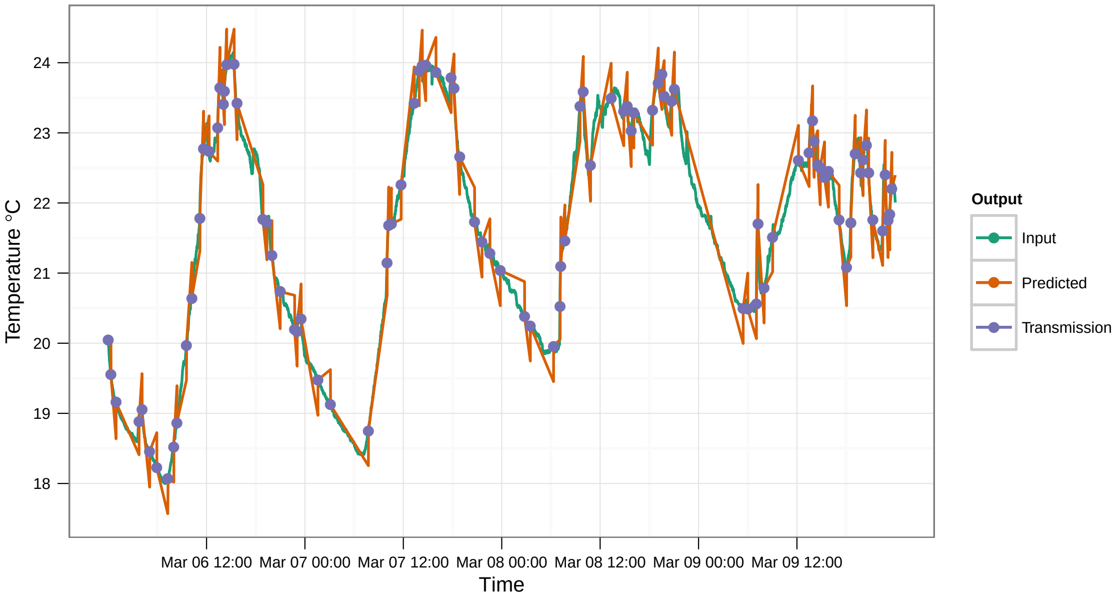


Figure 4.1: SIP example with piecewise-linear approximation

Not all sensing modalities show the linear trend that would make PLA suitable. A model based on PCA has been shown to be appropriate for data sets that exhibit abrupt changes in the reported values (for example data from light sensors), or show binary behaviour. Rather than use a linear estimate for the gradient, the PCA approach expects the value to remain the same, thus the rate of change is set to zero. Therefore, when using PCA a sample is only transmitted to the sink if the difference between the current reading and the last transmitted model is greater than the user specified error threshold. When PCA is used SIP demonstrates similar characteristics to approximate caching approaches (see Section 2.5.3) such as K-RLE [15] or Poor Man's Compression (PMC) [56]. However, these approaches can increase latency by only transmitting a summary when a complete segment has been gathered. SIP addresses this problem by transmitting at the start of each segment, and using this value in future predictions.

Figure 4.2 shows the output of SIP when a Piece-wise Constant Approximation (PCA) model is used. Rather than the spikes present when the PLA model is used, the estimated values are a quantised version of the original data set.

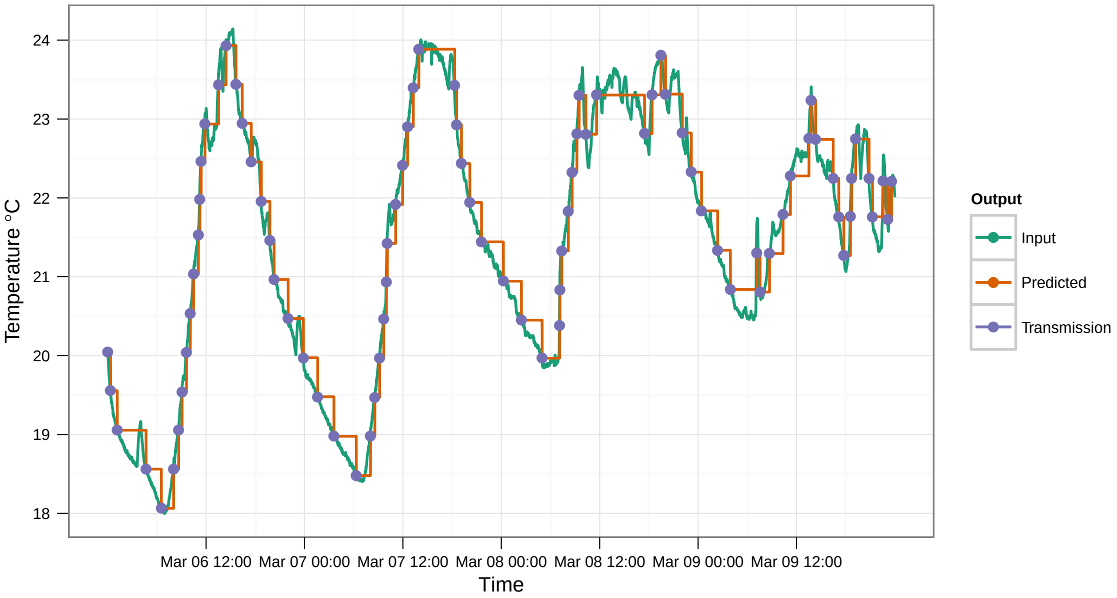


Figure 4.2: Example with piecewise-constant approximation

4.4 Evaluation

This section evaluates the performance of the SIP algorithm. The transmission suppression performance is evaluated against real world data sets.

4.4.1 Metrics And Datasets Used In Evaluation

Key metrics for evaluating the performance of any DPS algorithm are the percentage of data transmitted ($\%tx$) and the Root Mean Squared Error (RMSE) in the reconstructed signal. In principle, reducing the percentage of data transmitted will produce a corresponding reduction in energy use, as fewer transmissions will lead to the radio being used less frequently. However, given that the choice of communication protocols will have an effect on the number of transmissions made to transport data from node to sink, the work described here focuses solely on the reduction in data that needs to be transmitted. A detailed examination of the energy savings when deployed on physical node hardware is left for future work (see Section 6.2). The second metric for evaluation is the RMSE of the values predicted by the model. This provides an indicator of the quality of the reconstructed signal at the sink and thus the lossiness of the protocol. In the general case, this error value is calculated using the values that are estimated by the model (prediction error) rather than the error when interpolating between model points (reconstruction error). Future work will evaluate the use of other interpolation methods, such as a spline based approach, to evaluate the impact of the interpolation method on the reconstruction error.

The data sets used in evaluation are:

The **Intel Lab Data set** [63] , which consists of data collected from 54 sensors deployed in the Intel Berkeley Research between February 28th and April 5th 2004. The data consists of temperature, humidity, light and voltage collected at 31 second intervals, and is commonly used in the evaluation of WSN algorithms [57, 95]. To allow comparison with prior work, unless otherwise specified the data used for evaluation is restricted to values collected between the 6th and 9th of March. This allows the results reported here to directly match those reported in Santini and Römer [95]. Where a longer period of data is required some pre-processing of the data was performed to remove erroneous readings due to low battery levels at the end of the deployment. Each data set was truncated to remove values when the battery level was below 2.35 Volts. No other pre-processing was performed.

HomeREACT data set consists of data collected from a household monitoring application [23]. Temperature, humidity and light levels were collected at 5 minute intervals. This data set is similar to the Intel data described above, however the longer sampling period is similar to the duty cycle found in many real world deployments, and allows evaluation of how the duty cycle can affect the performance of SIP.

National Data Buoy Centre (NDBC) data set [71] consists of samples taken from weather buoys in the Atlantic. This data set has been selected to allow comparison with prior DPS work. However, while the Dual Kalman Filter (DKF) [69] and Racing [57] algorithms were both evaluated against data from the NDBC, no description of the actual data streams used were available, making a direct comparison difficult. The data used for evaluation is hourly averages of data gathered by buoys located at 2N 137E and 8N 137E.

4.4.2 Example SIP algorithm output

To illustrate the terms used in the rest of this section, this section demonstrates and discusses the output of the SIP algorithm over an example data set.

Figure 4.3 shows the output of the SIP algorithm using a piece-wise linear model, for Node 1 of the

Intel Lab Data set[63]. The input and output parameters together with any related performance metrics are described below. The performance metrics are further summarised in Table 4.1.

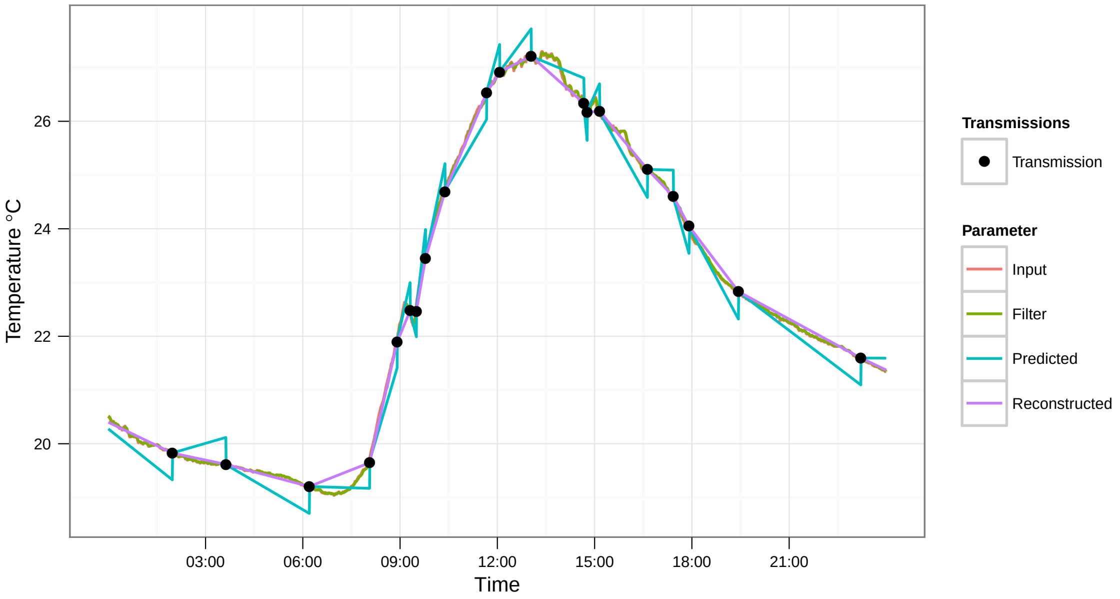


Figure 4.3: Example SIP output

Input represents the values reported by the sensor. In this case 2049 samples were taken during the 24 hour period.

Filter represents the filtered sensor readings, in this case the output of an EWMA filter with window length 5.

Predicted represents the output of the shared model, here a piece-wise linear model is used, with the output based on the rate of change value encoded in the state vector. The RMSE between the input data, and the values predicted by the SIP algorithm was 0.27 ℃.

Reconstructed represents the output of the model when a historical sample is requested. Here, the data is reconstructed using linear interpolation between transmitted values, with a error of 0.11 ℃ against the input data.

Transmission represent points where the node transmits a new model to the sink. Here 20 model updates (0.98% of collected samples) were required to represent the phenomena within the $\varepsilon = 0.5$ ℃ error threshold.

			RMSE	
Samples	Transmissions	% Transmitted	Predicted	Reconstructed
2049	20	0.98%	0.27 ℃	0.11 ℃

Table 4.1: Example SIP performance metrics

4.4.3 Evaluation: Error Budget

To aid comparison to prior work, each experiment in this section makes use of the Intel Lab data set, in a similar approach to that used by Santini and Römer [95]. The SIP algorithm was used to compress temperature data from nodes 1, 11, 13 and 49 between the 6th and 9th of March with a range of error thresholds. In all cases an EWMA filter was used.

Relationship between Error Budget and Number of Transmissions made

The SIP algorithm allows the developer to trade some accuracy in the reconstructed data for a reduced number of transmissions. This is achieved though the use of a user specified error threshold ε.

For this experiment, a hypothesis was proposed that:

H 1. *When SIP is used to suppress (compress) temperature data transmissions, the choice of error budget will affect the suppression (compression) performance of the algorithm. With a small error budget a greater percentage of collected samples will require transmission. As the error threshold ε increases, the number of transmissions required to represent the data will decrease.*

SIP was used to compress data streams from the Intel Lab data set, and the percentage of data that would require transmission was recorded. A PLA model was used to encode the data. The error thresholds were $\varepsilon = \{0.1^{\circ}C, 0.2^{\circ}C, \ldots, 5.0^{\circ}C\}$.

It is clear from Figure 4.4 that the number of transmissions made decreases as the size of the error budget is increased. For all Datasets, error budget ($\varepsilon = 0^{\circ}C$) resulted in all samples being transmitted. Transmissions drop to approximately 0.1% of collected samples when $\varepsilon = 5^{\circ}C$.

As the error threshold ε increases, the number of transmissions decreases dramatically, with an average of approximately 5% of samples requiring transmission when $\varepsilon = 0.1^{\circ}C$, and an average of 1.4% of samples

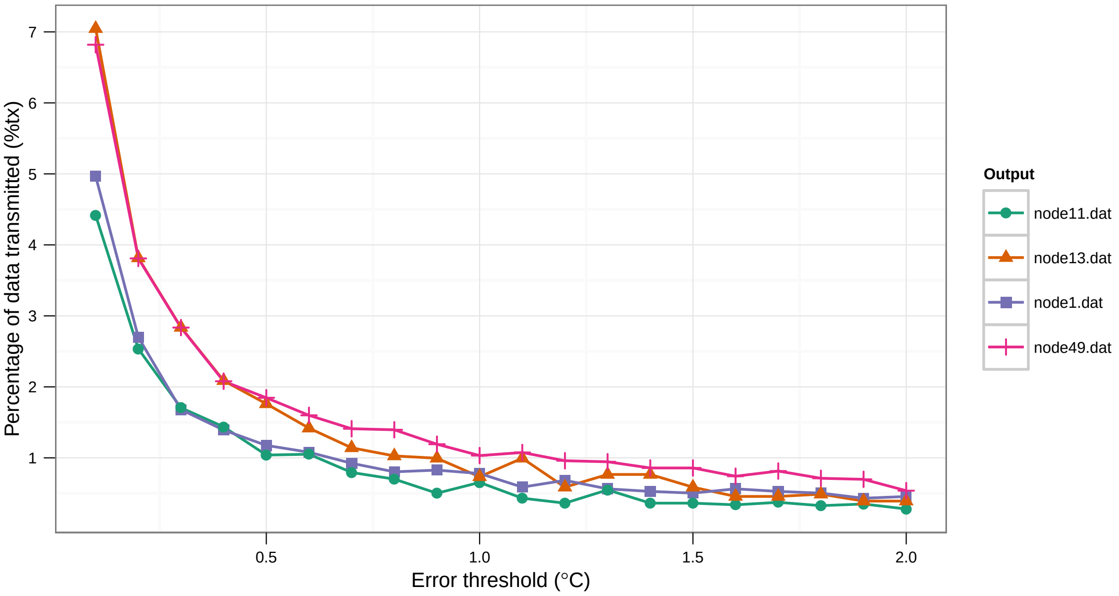


Figure 4.4: Percentage of data transmitted over a range of error thresholds for thresholds 0.1℃–1.0℃

transmitted with $\varepsilon = 0.5^{\circ}C$. These results compare favourably with the transmission reduction of 90% reported by Santini and Römer [95] using the same data sets and an error budget of $\varepsilon = 0.5^{\circ}C$.

The exponential relationship between the error threshold and the number of transmissions means that, in the case of temperature, increasing the error threshold above $\varepsilon > 1.0^{\circ}C$ yields little improvement in the transmission suppression performance, with an increasingly smaller change in data reduction for each 0.1 ℃ increase in error threshold. In the extreme case the error threshold will be greater than the range of the data in which case no transmissions past the first will be made.

To conclude, this experiment has demonstrated that for the nodes selected from the Intel temperature data set, the hypothesis H 1 is true. In all cases, as the error budget increases the number of transmission required to represent this data to this error budget decreases. Higher error thresholds give greater scope for transmission suppression. Small values for the error threshold will provide a more accurate representation of the data, but require more transmissions to be made.

Relationship between Error Budget and Model Error

Having demonstrated that SIP is able to reduce the number of transmissions, this next experiment demonstrates the effect on the model accuracy of increasing the error threshold ε. Given that SIP allows users to trade accuracy in estimated readings for a reduction the number of samples transmitted, it is expected that larger error budgets will result in a greater error in the modelled data.

For this experiment, the following hypothesis was proposed:

H 2. *When used to evaluate temperature data from the Intel lab data set where SIP has been shown to decrease the number of packets transmitted. Increasing the error budget will produce a corresponding increase in the RMSE of the data predicted by the model.*

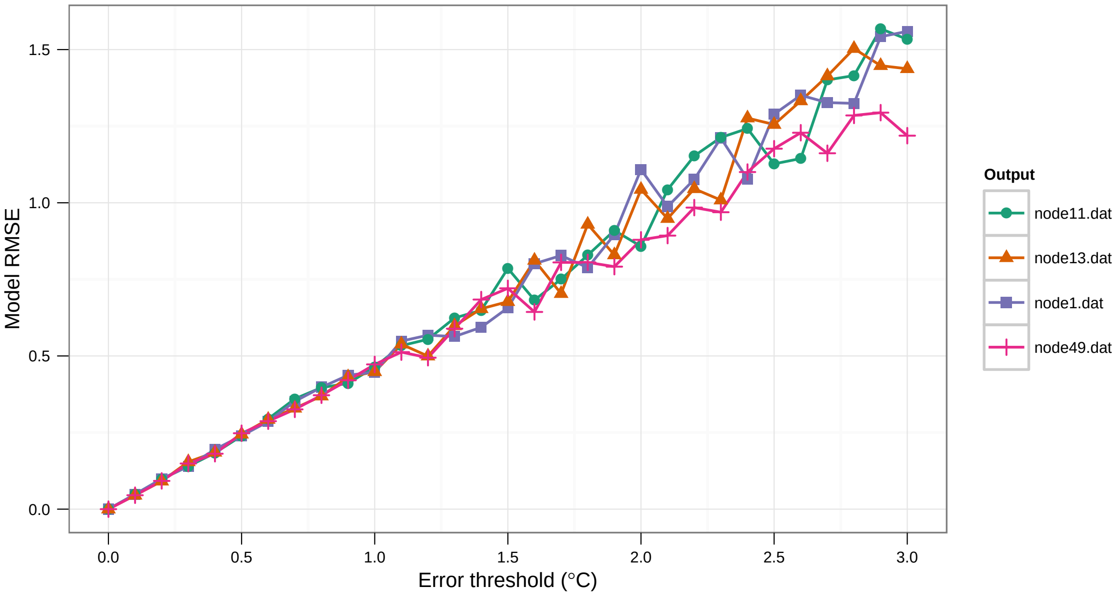


Figure 4.5: RMSE of model predictions for Intel Lab data over a range of thresholds

Data set	RMSE (predicted)	Max Error (Predicted)
Node 1	0.23℃	0.49℃
Node 11	0.24℃	0.49℃
Node 13	0.23℃	0.49℃
Node 49	0.23℃	0.49℃

Table 4.2: Error for predicted and reconstructed sensor readings

Figure 4.5 shows the measured RMSE of the model's predictions, for selected nodes in the Intel Lab data set. The RMSE of the predicted data increases linearly in line with ε. At no point does the RMSE of the estimated data exceed the user specified error threshold.

Increasing the error budget leads to diminishing returns in performance. In particular, the exponential relationship between the number of packets transmitted and the particular error threshold means that while the error increases in line with ε, increasing the error budget beyond a certain point will only induce more error in the model, whilst yielding little increase in transmission suppression performance.

Relationship between Error Budget and Reconstructed Data

The SIP algorithm offers two modes for estimating sensor readings. For estimating the current reading, an *online* mode is used, that estimates the current sensor value based on the most recent model transmitted to the sink. Historical data is generated by interpolating between previously transmitted states. These two approaches produce different error levels in the estimated datastream: the model based approach tends to have spikes, where the estimated reading diverges from the underlying data before the error threshold is reached. The interpolation approach used to estimate historical readings will interpolate between transmitted state estimates, without considering the gradient component of the model.

For this experiment, we evaluate the effect the differing estimation approaches have on the error in the reconstructed data. Table 4.3 lists the RMSE of the predicted and reconstructed data streams when the error threshold $\varepsilon = 0.5$. It can be seen that the RMSE of the data is reduced when using the reconstructed data approach, although the maximum error is up to twice the predicted.

Dataset	RMSE (Predicted)	Max Error (Predicted)	RMSE (Reconstructed)	Max Error (Reconstructed)
Node 1	0.23	0.49	0.21	0.90
Node 11	0.24	0.49	0.18	0.75
Node 13	0.23	0.49	0.21	0.85
Node 49	0.23	0.49	0.20	1.01

Table 4.3: Error for predicted and reconstructed sensor readings, using a 0.5 ℃ error threshold, for the Intel Lab dataset.

When using the reconstruction approach, the gradients are not considered. This interpolation can cause a larger under- or over-estimate then when considering the gradient, meaning the absolute error for a interpolated sample can exceed the specified error threshold. This effect can most easily be observed when there has been a rapid change in the direction of the data. Given that SIP provides both the sensor value, and the rate of change it is possible that the use of spline based interpolation will improve the accuracy of these reconstructed values. A spline based approach will be examined in future work.

4.4.4 Effect of Different Prediction Models

This section evaluates the performance of SIP when different predictive models (see Section 4.3.5) are used. Two simple predictive models that have been shown to work well with SIP (Piece-wise Constant Approximation (PCA). PLA are evaluated using data from the Intel Lab dataset.

Table 4.4 shows the RMSE and percentage of data transmitted for temperature data from Intel node 11 when a PLA and PCA model was used.

For temperature data, it is clear that the linear approximation provides an improvement in both RMSE and transmission percentage over a simple constant approximation. This is due to the temperature data showing linear trends that can be captured by the model (see Section 4.3.5).

Node	Model	Threshold	RMSE	Percent Transmitted
node11	linear	0.10°C	0.05°C	4.42%
node11	constant	0.10°C	0.05°C	6.10%
node11	linear	0.50°C	0.24°C	1.04%
node11	constant	0.50°C	0.26°C	0.86%
node11	linear	1.00°C	0.46°C	0.65%
node11	constant	1.00°C	0.52°C	0.35%

Table 4.4: Performance of SIP for Intel Node 11 temperature data with different predictive models

4.4.5 Effect of Different Filters

The filter used to smooth the data collected by the sensors can have an effect on the performace of SIP.

Table 4.5 shows the performance of SIP for the HomeREACT datasets with different filter types. For both temperature and humidity the data Kalman Filter produced the greatest reduction in packets transmitted. However, the parameters for this filter required a significant amount of tuning to achieve this performance. On the other hand, the less complex EWMA produces a similar reduction in the number of transmissions, without requiring tuning.

One interesting effect of the filter choice can be seen with the NLMS. This filter has a convergence

Data-set	Error threshold (ε)	Filter	RMSE	Transmitted (%)
HomeREACT Temperature	0.5 ℃	EWMA	0.24 ℃	4.1
(sensor 1)	0.5 ℃	NLMS	0.75 ℃	4.0
	0.5 ℃	KF	0.25 ℃	3.9
HomeREACT Humidity	0.5 %RH	EWMA	0.46 % RH	13.3
	0.5 %RH	NLMS	2.2 % RH	12.7
	0.5 %RH	KF	0.58% RH	11.3

Table 4.5: SIP Effect of different filter types

period where the output of the filter does not provide a good indication of the actual sensor reading (see Section 4.3.4). This leads to an increased RMSE in the predicted data, caused by the increased error between the filtered and raw sensor data during the convergence period. Strategies for dealing with filters that have a convergence period are discussed in Section 4.3.4. However, implementing this approach is left for future work.

4.4.6 Compression Performance with Different Data types

Having established that the SIP algorithm is able to reduce the number of transmissions required to represent temperature data within a user specified accuracy for selected data sets, the following section evaluates the performance of the algorithm over the different data types and data sets described in section 4.4.1.

Compression Performance for Air Temperature Data

This section evaluates the performance of the SIP algorithm over air temperature data. Having demonstrated that the algorithm is able to produce a significant reduction in the number of transmissions for given streams of temperature data in the Intel data set, the algorithm is shown to give comparable performance when used to compress temperature data from a range of other data sets.

The following hypothesis is proposed:

H 3. *SIP will produce similar transmission suppression results to those from the Intel Nodes 1, 11, 13 and 49 in data sets that show similar characteristics (such as, sample rate, rate of change and derivative of the rate of change)*

To evaluate this hypothesis SIP was first used to compress data from all 51 nodes in the Intel data set, and the transmission suppression performance recorded. The parameters for the algorithms were a

EWMA filter and PLA model. The error thresholds for this test were set at 0.25 ℃ , 0.5 ℃ 0.75 ℃, and 1 ℃.

Figure 4.6 shows the results of this experiment. For all selected error thresholds, the algorithm provides a greater than 90% reduction in the number of samples that need transmission. This compares favourably with other transmission suppression schemes evaluated using the Intel data set (see Table 2.4). Datasets for nodes nodes 30 and 32 consistently produce higher numbers of transmissions. Both nodes have several periods with missing data points, which require extra transmissions to bring the model back inline, as each group of missing samples can affect the rate of change calculations. The best performing node was node 7, which required just 0.6% of the data to be transmitted with an error threshold of 1℃.

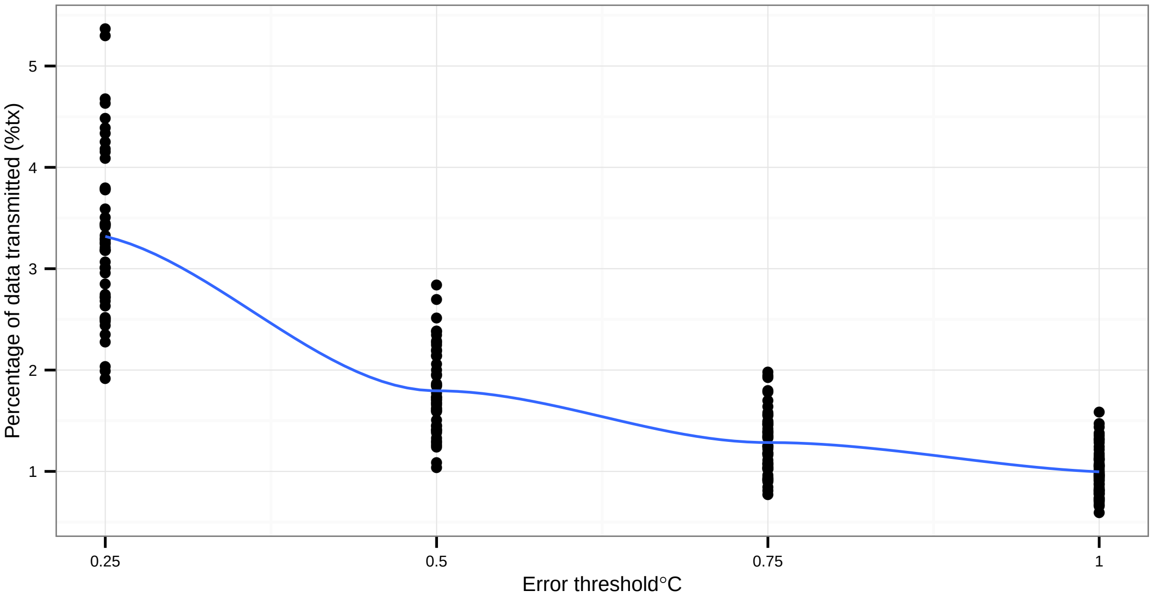


Figure 4.6: Transmission reduction performance over full Intel Lab data set (temperature)

Next, SIP was used to evaluate data streams from additional data sets: two from HomeREACT and two from NDBC. The thresholds chosen for the evaluation are 0.1℃, 0.5℃, 1℃. Additionally, thresholds at 0.5%, 1%, 5% and 10% of the data amplitude range are evaluated. The algorithm output was evaluated with both the EWMA and NLMS filters, using a Piece-wise Linear Approximation (PLA) model.

For these data sets there is a marginal gain in transmission suppression performance using an EWMA filter as opposed to an NLMS. The NLMS filter has a convergence period where the filtered values do not accurately represent the underlying reading. During this convergence period, the algorithm detects this error as a transmission event, and consequently several extra packets are transmitted while the filter converges.

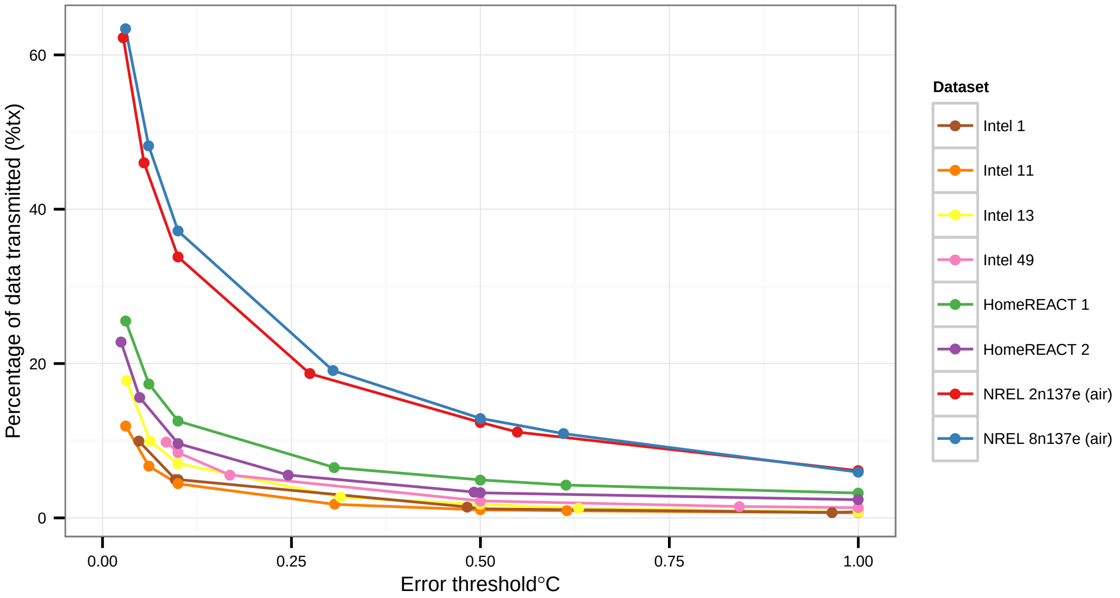


Figure 4.7: SIP performance over temperature data sets using EWMA filter

Figure 4.7 shows the transmission compression performance over the full range of air temperature data sets (Intel Lab, HomeREACT, and NDBC). It can be seen that like the previous experiments with the Intel Lab data set, in most cases SIP manages a significant reduction in transmissions even with a small error budget. In the worst case (NDBC air temperature) a 37% reduction in the number of samples was achieved at 1% of the range of the data (0.02 ℃). To maintain the benchmark error threshold of 0.5 ℃, on average of 8% of the samples required transmission, with a minimum of 1.15% and maximum of 19.79% (Table 4.6). This represents an up to 10 fold reduction in the percentage of data transmitted,when compared to achievements in prior art (Section 2.5.3).

Individual streams within each group of data (Intel Lab, HomeREACT, and NDBC) show similar compression ratios. However, each group has its own particular suppression characteristic.

Two factors affect the percentage of data transmitted:

- The characteristics of the data

- The sampling frequency

While all data sets for air temperature examined show a diurnal pattern, the area where the data was gathered introduces short term variations to this pattern. For example, the Intel data set used sensors placed in an office. Data streams within this set tend to show a variation in the pattern around midday that may be explained by the office lunch break, or other occupancy hours. These variations signal a change in the trends found in the data, and require a transmission to be made to ensure the model represents the underlying data stream. On the other hand the NDBC data sets, are collected from a buoy in the mid Atlantic. These data sets tend to show a stable pattern, as there are unlikely to be any outside influences on the temperature data collected, and thus require fewer transmissions to represent trends in the data.

The second factor is the sampling rate. A higher sampling frequency means that transient events in the data steam are more likely to be captured, increasing the understanding of the system under consideration at the expense of a greater number of transmissions. However, during periods where there is little change to trends within the data, a large amount of redundant data is collected. This means that the SIP algorithm is able to suppress more samples, and thus produce a lower transmission percentage. Conversely, the relative percentage of data transmitted will be higher for sparser (in terms of samples) data sets, as the state of the system is more likely to have changed between sensor readings. Table 4.6 provides examples demonstrating the relationship between sparsity and compression ratios. The Intel data set has the highest sampling frequency, with a sample being gathered approximately every 30 seconds. While the Intel data set transmits the most packets (approximately 32 per day), it also has the lowest transmission percentage, given the high proportion of redundant data in the high frequency data collected. Conversely, the NDBC data set only transmits an hourly summary of data, which means that a large number of these samples represent periods where the trends within the data have changed. Therefore each sample collected is more likely to require a model update. Additionally, given the low number of samples collected even transmitting a single packet would represent over 4% of the data collected that day. Given the diurnal pattern, it can be expected that at least 2 samples would require transmission daily (one for each peak and trough in the data).

In all the temperature data sets evaluated SIP was able to produce a significant reduction in the number of samples that required transmitting to the sink. In the case of the Intel and HomeREACT data

Data set	Samples per day	Range of data	Percentage transmitted
HomeREACT 1	275.71	6.13	4.92
HomeREACT 2	272.95	4.92	3.25
Intel node 1	1549.95	11.77	1.40
Intel node 2	1400.18	10.04	1.26
Intel node 7	1819.15	8.69	1.04
Intel node 13	969.87	10.44	2.52
Intel node 30	967.73	17.58	2.84
Intel node 32	1201.15	18.85	2.69
Intel node 42	1618.26	17.34	2.06
Intel node 49	1343.89	16.86	2.20
NREL 2n 137e	24.00	5.49	12.38
NREL 8n 137e	22.81	6.10	12.89

Table 4.6: Performance over temperature data with threshold 0.5 ℃

sets, less than 5% of the data collected required transmission at a 0.5 ℃ error threshold. This transmission ratio increased to approximately 13% for the NDBC. However, while the Intel and HomeREACT data had relatively high sampling rates (30 seconds and 5 minutes respectively), the NDBC data consisted of hourly averages and therefore a low number of transmissions would represent a high proportion of the data gathered. In all instances these results outperform prior DPS based transmission suppression work.

Compression performance with Light Data

Compared to temperature and humidity data, data collected by light level sensors shows very different characteristics. Firstly, rather than showing linear trends, values reported by light level sensors tend to be quantised, with long periods where the same value is reported, separated by large jumps in value. This means that rather than have a smooth transition between samples, large step changes in the reported value are observed. Further, the size of these changes are much larger than those found in other data sets. For example, the average difference between two neighbouring light readings in the Intel Node 11 data set is 7.27 Lux, compared to 0.047 %RH and 0.012 ℃ for humidity and temperature data respectively. This means that large error thresholds are required for the algorithm to achieve any compression performance.

The step change in reported values means that a different model to the PLA is a more appropriate to represent light levels, in this case assuming a constant rate of change with a PCA representation. With the PCA, the model clamps the rate of change in the state vector to 0, meaning that the estimated values remain static. When data streams with low levels of sensor noise and transient step changes are observed, a *pass-through* filter (that simply returns the raw sensor value) is more appropriate to allow the model to respond to events in the data stream. This is because the EWMA is smoothing the signal.

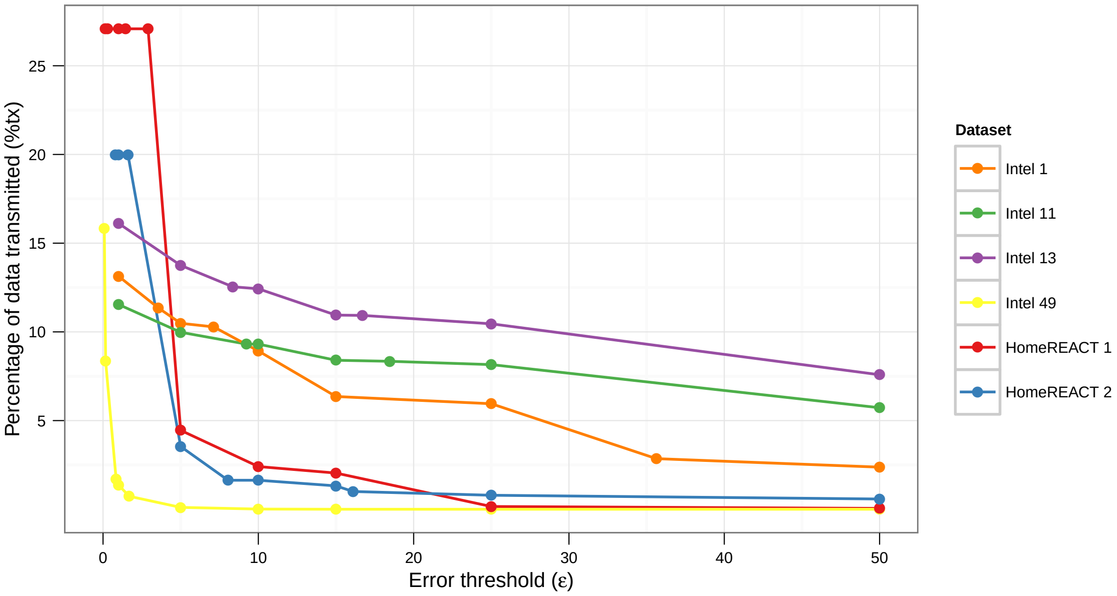


Figure 4.8: Transmission suppression performance over light data

Figure 4.8 shows the transmission suppression performance for selected light data sets. Compared to the performance when compressing temperature or humidity data, the relationship between the error threshold and the number of transmissions is less pronounced. While there is a large decrease in packets transmitted with small error thresholds, there is less benefit to using larger thresholds. This is due to the step changes between values observed in the light data sets. Given the size of the step between values, in many cases any change in the light levels reported will be greater than the specified error threshold, and thus any changes in the data will require transmission.

Evaluation on a physical deployment

The experiments described previously have simulated the performance of SIP over historical data sets. Further evaluation to gain insight into the performance of the SIP algorithm on node hardware was undertaken.

SIP was implemented in TinyOS [40] and deployed on a network of TelosB [84] motes.

For each sensing modality (temperature, humidity and light), a sample was taken every 30 seconds

Phenomenon	Threshold	Samples	Transmissions	% Tx	RMSE (Predicted)
Temperature	0.25°C	844	38	4.5	0.14°C
	0.5°C	844	30	2.4	0.26°C
	0.75°C	844	12	1.4	0.37°C
Humidity	0.25 %RH	844	78	9.2	0.36%RH
	0.5 %RH	844	60	7.1	0.34%RH
	0.75 %RH	844	37	4.4	0.39%RH
Light (PAR)	5 Lux	844	506	59.953	7.1 Lux
	10 Lux	844	313	37.085	3.092 Lux
	20 Lux	844	152	18.09	7.47 Lux

Table 4.7: Results of SIP deployment on TelosB nodes

(to allow comparison with the Intel Data) over a 7 hour period, for a range of error thresholds. Filters and state evolution models were chosen based on the best performing parameters from simulated results discussed above. Humidity and temperature data were filtered using an EWMA, with a PLA model. Light data was filtered using pass-through filter and PCA model.

To allow the error between the output of SIP and the values gathered by the sensor to be calculated, sensed values were logged to the TelosB external flash, while model updates were transmitted to the sink using single hop communications.

Table 4.7 shows the results of these deployments. The transmission suppression performance for each sensor type is similar to that observed during simulations, with larger error thresholds resulting in fewer model updates.

This experiment has demonstrated that SIP is appropriate for use on node hardware, and the transmission suppression performance when deployed on node is comparable to that found during simulation.

Summary of Results over various data sets

A summary of results for SIP is given in Table 4.8. The table shows results for 6 data sets, the last being results from a deployed system rather than a simulation. The filters used in evaluation were chosen to represent a range of computationally simple (i.e. EWMA) and complex filters (i.e. KF) that are commonly used for signal processing, or have been used in prior work (i.e. NLMS and KF). These results consistently improve over prior art results shown in Table 2.4. In particular, the performance for Intel temperature data shows an improvement by a factor of 10. The SIP algorithm is scale invariant and so if the scale of the data range is increased along with the error threshold, performance will be maintained. The rate of fluctuation of the data also plays a part in the performance of the algorithm but whether this

Data-set	Error threshold (ε)	Filter	RMSE	Transmitted (%)
HomeREACT Temperature (sensor 1)	0.5 ℃	EWMA	0.24 ℃	4.1
	0.5 ℃	NLMS	0.75 ℃	4.0
	0.5 ℃	KF	0.25 ℃	3.9
HomeREACT Humidity	0.5 %RH	EWMA	0.46 % RH	13.3
	0.5 %RH	NLMS	2.2 % RH	12.7
	0.5 %RH	KF	0.58 % RH	11.3
HomeREACT Light	5 lux	Pass-through	2.2 lux	4.4
	9 lux	Pass-through	2.5 lux	2.4
	5 lux	EWMA	2.7 lux	1.4
	9 lux	EWMA	5.8 lux	0.37
Intel (Node 13)	0.5 ℃	EWMA	0.24 ℃	1.0
	0.5 ℃	NLMS	0.41 ℃	1.1
	0.5 ℃	KF	0.26 ℃	1.4
	0.05 ℃	EWMA	0.06 ℃	5.3
Telos Deployment	0.5 ℃	EWMA	0.22 ℃	1.7

Table 4.8: Summary of SIP performance for various data sets.

has a significant effect depends on the filter and its parameters.

Figure 4.9 shows the quoted transmission reduction over the Intel Lab dataset for prior DPS algorithms. In each case an error threshold of 0.5℃ is used. Early approaches such as the probabilistic models used in Ken and PCA (section 2.5.3) were able to represent the sensed data within this error threshold by transmitting around 65% of the samples collected. In the case of Ken this improves to 45% of the collected samples when a cluster based approach is used. Filter based approaches such as DPS and DKF improve on this and were able to represent the data stream using approximately 10% of the collected samples. However as discussed in section 2.5.3 the DKF approach requires extensive tuning of the Kalman filter to achieve this performance, while the LMS used by Santini and Römer [95] can suffer from a convergence period where the filter may not accurately represent the data stream.

The Similarity-based Adaptive Framework (SAF) approach (section 2.5.3) offers a similar transmission reduction performance to SIP. However, the Intel dataset was pre-processed to adjust the sampling rate [100], and thus the results may not truly represent the performance compared to SIP.

4.4.7 Tuning the Error Budget

Given that there is a clear relationship between the error budget and the number of transmissions made, it should be possible to estimate the transmissions that would be required to compress a data stream with

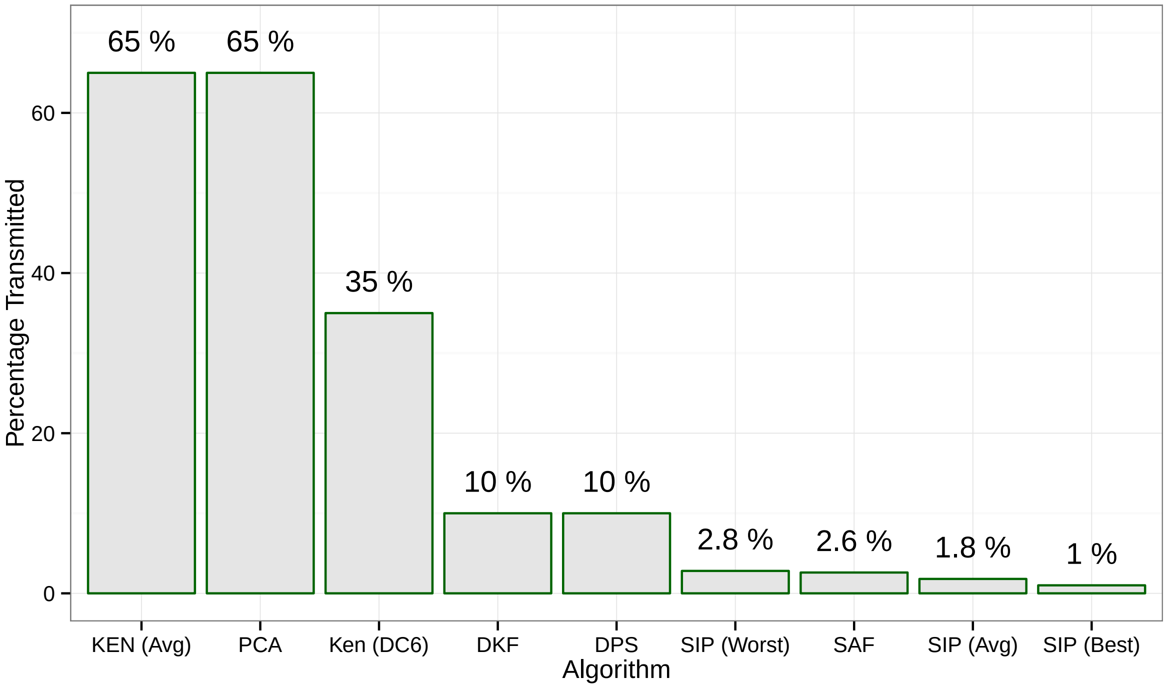


Figure 4.9: SIP Comparison against prior work

a given error budget. Rather than use the algorithm to compress a wide range of error thresholds, and find the required error threshold by brute force, a curve fit of this relationship can be used to estimate the number of transmissions resulting from a given error threshold.

To evaluate this the following hypothesis was proposed:

H 4. *The number of transmission required for a particular data stream is related to the error threshold according to, $y = ax^c$ where y is the error threshold, x is the number of packets transmitted, and a, c are constants.*

A power law curve fit of the error threshold (ε) against the percentage of data transmitted ($\%tx$) was used. The curve fit itself used a linear model such that $log(\varepsilon) = a \times log(\%tx) + c$. Thresholds chosen as inputs for the curve fit were at 0.1 ℃ 0.2 ℃ and 0.5 ℃, corresponding to approximately 1%, 2% and 5% of the of the range of the averaged Intel Node 1 temperature data.

Table 4.9 presents the curve fit parameters for the Intel data sets. The standard error and coefficient of determination (R^2) values demonstrate that the curve is a good fit for the supplied data.

Phenomena	Data set	Intercept (a)	Std. err. intercept	Slope (c)	Std. err. slope	R^2
Temperature	Node 1	-0.32	0.010	-0.95	0.006	0.999
	Node 11	-0.22	0.07	-0.88	0.044	0.995
	Node 13	0.33	0.06	-0.87	0.04	0.995
	Node 49	0.21	0.046	-0.83	0.027	0.997
Humidity	Node 1	0.614	0.04	-0.99	0.02	0.998
	Node 11	0.715	0.04	-0.92	0.02	0.998
	Node 13	1.2	0.03	-0.89	0.02	0.998
	Node 49	0.614	0.04	-0.99	0.02	0.998

Table 4.9: Transmission estimation curve fit parameters based on a linear fit of log(%tx)-log(ε) (Intel temperature data)

The curve fit was calculated using a limited number of error thresholds. To demonstrate how the parameters match a wider set of thresholds, the number of transmissions for a threshold range $\{0.1°\text{C}, 0.2°\text{C}, \ldots, 0.5°\text{C}\}$ was then calculated for each data set using the parameters of the curve fit. Figure 4.10 shows the estimated transmission percentage calculated using the curve fit parameters against the actual transmission percentage of the SIP. The RMSE of the calculated percentage values was under 0.2 ℃ for each of the data streams. This clearly demonstrates that the curve fit approach is able to estimate the number of transmissions that will be made with a given error threshold within a reasonable accuracy.

An alternate application of the curve fit approach is to estimate the error threshold required to achieve a particular number of transmissions. This can be useful if there is a target number of transmissions, (for example when calculating the lifetime of a network). In the case where more than one variable is encoded in the state vector, it can be advantageous that the expected transmission rate is similar for each phenomena.

To evaluate how the curve fit parameters can be used to achieve this, the error thresholds to meet a range of desired transmission percentages were calculated using the parameters for each curve fit.

SIP was then used to compress the data using the calculated threshold, and the actual transmission suppression performance recorded. Table 4.10 shows a summary of these results. It can be seen that the curve fit parameters allow the threshold needed to achieve a desired transmission suppression ratio to be calculated. In this case the maximum error between the desired and actual percentage of data transmitted was 0.52% with an RMSE of 0.24% over the cases shown here.

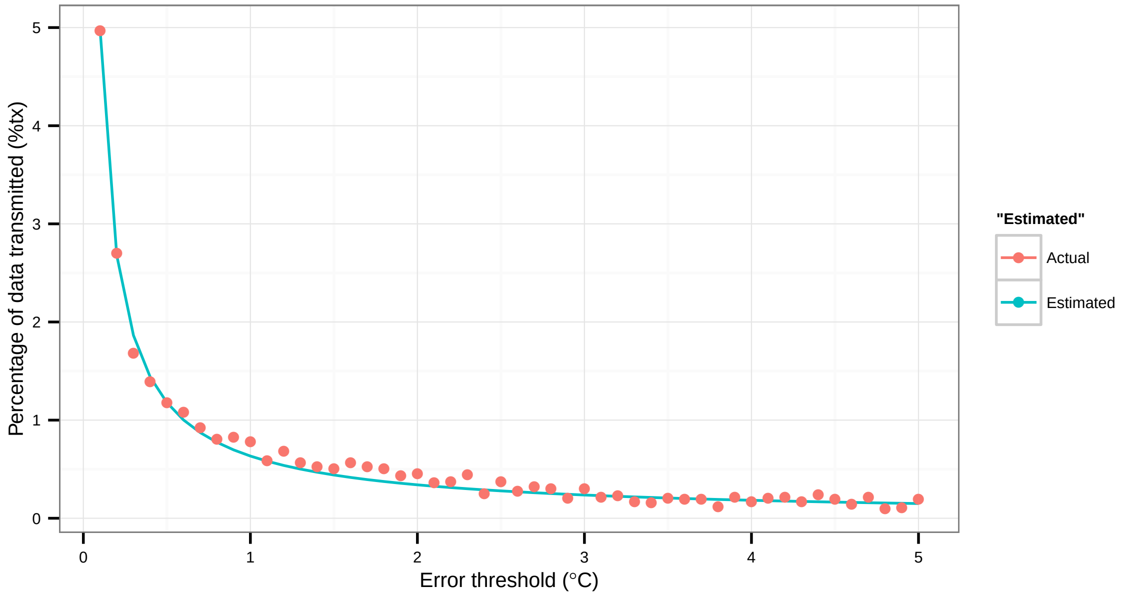


Figure 4.10: Estimated transmissions derived from linear fit plotted against actual transmissions required for the Intel Lab temperature data

Node	Desired Tx %	Estimated threshold	Actual Tx %	Error
node1	0.10	7.85	0.08	0.02
	0.50	1.30	0.59	0.09
	1.00	0.60	1.07	0.07
	10.00	0.05	9.63	0.37
node11	0.10	6.84	0.01	0.09
	0.50	1.15	0.56	0.06
	1.00	0.53	1.18	0.18
	10.0	0.04	9.48	0.52

Table 4.10: Estimated against actual transmission ratios for error thresholds calculated using power law curve fit approach (Intel Nodes 1 and 11)

4.4.8 Compression of more than one sensor reading

So far, the evaluation of SIP has concentrated on performance over single sensor data sets. However, in practice it is rare that a node will collect only one data type. It is common to combine several sensor readings from a node taken at a particular sampling interval, into a single packet.

Prior work in dual prediction schemes has focused on encoding a single data stream in each model. Thus, where multiple sensors are present on a node multiple models will be used. SIP allows multiple readings to be combined into a single state vector which can reduce the overhead of processing the data.

This section evaluates the performance of SIP when multiple inputs are used to the state vector. The following hypothesis is proposed:

H 5. *Combining multiple sensor readings into a single state vector, will produce a greater reduction in the number of transmissions made, compared to compressing individual data streams.*

To illustrate this, consider a situation where SIP compresses data from the Intel Lab node 1. For each of the 8559 samples in the data set, temperature, humidity and light level readings are taken. With sense and send, combining these readings into one packet would mean that a total of 8559 transmissions are made. Making use of SIP to compress each data stream would require 98 transmissions for temperature, 269 transmissions for humidity (assuming a threshold of 0.5℃/%RH for each) and 952 transmissions for light (assuming a 5Lux threshold). This results in a total of 1319 transmissions (15%) overall.

Table 4.11 summarises the performance of SIP over individual temperature, humidity and light data streams for nodes 1 and 13 of the Intel lab data set. Temperature and humidity data were processed using an EWMA filter and PLA model. Light data was processed using a pass-through filter and PCA. The total percentage of data that would be transmitted using individual models is calculated by summing the total number of samples transmitted for each parameter and dividing this by the number of samples in the data set. Table 4.12 shows the number of packets required to transmit the same data sets when a combined state vector is used. A paired t-test over these data sets (using temperature, humidity and light) confirms that there is a significant ($p < 2 \times 10^{-16}$) improvement when moving from separate state vectors for each sensor type to a combined state vector.

When combining all sensor types together into a single packet, the transmission suppression performance of this approach is limited by the worst performing individual data stream. For example, in both the cases below light data required the greatest number of transmissions to reconstruct the data stream.

data set	phenomena	filter	model	threshold	samples	transmitted	Percent Tx
Node 1	Temperature	ewma	linear	0.50	8336	98	1.18
	Humidity	ewma	linear	0.50	8336	269	3.23
	Light	pass	PCA	10.00	8336	744	8.93
Total						1111	13.32
Node 13	Temperature	ewma	linear	0.50	6131	108	1.76
	Humidity	ewma	linear	0.50	6131	344	5.61
	Light	pass	PCA	10.00	6131	762	12.43
Total						1214	19.80

Table 4.11: Summary of transmission suppression performance for individual streams in the Intel data set

Node	Thresholds	Samples	Transmissions	Percent Transmitted

Node 1	Temperature = 0.5
	Humidity = 0.5
	Light = 10.0
		8336	922	11.06%

Node 13	Temperature = 0.5
	Humidity = 0.5
	Light = 10.0
		6131	936	15.26%

Table 4.12: Summary of transmission suppression performance with combined state vector

This transmission percentage is close to that of when combined readings are transmitted. However, given the packet overhead (approximately 32B per packet for the TelosB mote) this approach should still yield a large reduction in the number of bits transmitted.

4.5 Summary

In the general case, the radio unit is the most energy intensive component on a sensor node. Thus, transmitting data through the network will have a significant cost compared to processing this data. Reducing the amount of data transmitted will save energy and increase a node's lifetime.

SIP lessens sensor node energy consumption by reducing the number of transmissions required to obtain a data stream at the sink within a user specified accuracy. SIP is an outgrowth of the structured view of WSNs in FieldMAP (Chapter 3) and makes use of the same basic elements to achieve its goal. As a data-driven approach to transmission reduction, SIP can be used to complement other energy saving techniques such as low energy routing.

The performance of SIP is evaluated over several sensing modalities (temperature, humidity, light) from three different sources (two building monitoring applications and the NDBC). Evaluation also took place on mote hardware, with comparable reductions in the amount of data transmitted to that experienced during simulation.

SIP is shown to outperform similar algorithms, producing greater reduction in the number of packets transmitted while maintaining a similar level of accuracy. In the case of temperature data from the Intel Lab data set, the amount of data requiring transmission was 10 times less than when using a comparable algorithm with the same error threshold parameters.

SIP allows the developer to trade accuracy in the reconstructed data stream for a reduction in the number of transmissions required. There is an exponential relationship between the error budget and the number of transmissions with low error thresholds allowing significant reductions in the percentage of samples transmitted. In the case of temperature data, a 0.1 ℃ threshold with SIP causes only 5% of collected samples to be transmitted, while the percentage transmitted drops to 1.4% for an error threshold of 0.5 ℃. This pattern is observed for other environmental monitoring data (humidity and light).

The relationship between the error budget and percentage of data transmitted has been evaluated. Given a small amount of training data, a power law curve fit can be found that relates the transmission reduction to the error threshold for a given signal. This method allows the user to tune the error threshold to achieve a desired number of transmissions. Tuning SIP can be useful if the application has a desired transmission ratio (for example, to meet a target lifetime). The tuning process is shown to allow the user to estimate the percentage of data that a given error threshold will transmit to within 0.5%.

Past work in this domain has not evaluated transmission reduction with multiple modalities. Nonetheless, collecting multiple modalities from a single node is commonplace. With SIP, it has been found that combining multiple modalities into a single state vector yields better performance than processing the different modalities separately.

In addition to an increase in transmission suppression performance, the simple linear models used by SIP have several other advantages compared to prior work:

- Approaches based on probabilistic models (such as, BBQ and Ken) or Autoregressive Integrated Moving Average (ARIMA)-based approaches (for example PAQ and SAF) require a period of training before they can be used for compression. The LMS filter used by Santini and Römer [95] also requires a period of convergence before the filter can accurately represent the data. If the user requires data during the training period then the raw data must be transmitted to the sink. The

EWMA filter and linear model used by SIP do not require training.

- Both the Kalman filter and ARIMA-based approaches require tuning of the model parameters to achieve a high level of performance. Both require *a priori* knowledge of the phenomena. While some difficulties with selection of ARIMA-based models was addressed by Le Borgne, Santini and Bontempi [57], their approach requires a number of models to be run in parallel and the same training period as other ARIMA-based approaches. SIP is simpler than either a Kalman filter or ARIMA, in both the number of parameters and the need for tuning of those parameters.

This chapter has presented a node-based approach to modelling that yields a significant reduction in the number of transmissions. In the next chapter, an approach to modelling the phenomena on the sink is used to produce a virtual sensor.

Chapter 5

Virtual Sensors

5.1 Introduction

The previous chapter presented the Spanish Inquisition Protocol (SIP) transmission suppression algorithm, that aims to reduce energy consumption in Wireless Sensor Networks (WSNs) by modelling the phenomena at the sensor node to reduce the amount of packets that are transmitted. While SIP is one method of extending the modelling phase of the architecture presented in Chapter 3, modelling can also be effective at the sink, to estimate missing sensor values or predict future sensor readings.

This chapter explores imputation of missing data and prediction of future sensor values in an online pressure monitoring application for Water Distribution System (WDS). A non-parametric, machine learning technique called Gaussian Process Regression (GPR) is used to impute and predict data streams of individual sensor nodes based on historical measurements and correlations with other sensor nodes' data streams. This approach is extended to develop the concept of *virtual sensors*, where data is predicted for a location where no physical sensor is deployed. The prediction is based on a short window of training data and correlated readings from other sensors in the network. The target WDS application includes a component to model demand and consumption in the water network that can benefit from an increased number of inputs. Using virtual sensors the number of inputs can be increased without the need to deploy additional sensors.

The author proposes a methodology for identifying suitable locations for virtual sensors, gathering training data and maintaining the predicted data streams.

The chapter is organised as follows: Section 5.2 provides a background to the motivating application. Following this Section 5.3 presents a description of the on-line missing data imputation and prediction algorithm. In Section 5.4 the algorithm is evaluated in the context of a WDS case study. Section 5.3.5 extends the on-line imputation and prediction algorithm to enable the creation of virtual sensors. Section 5.5 evaluates the virtual sensors approach through an in-situ deployment within the WDS in Singapore. Finally the chapter is summarised in Section 5.6.

5.2 Motivating application

The work presented in this chapter has been motivated by and developed as part of the Wireless Water Sentinel (WaterWiSe@SG) project [106], a collaboration between Singapore MIT Alliance for Research and Technology (SMART), the Singapore Singapore Public Utilities Board (PUB), and Nanyang Technological University (NTU). The author was part of the project team for a period of three months, from July to September 2010.

The goal of the WaterWiSe project is to apply wireless sensor network technologies and capabilities to the continuous monitoring of drinking water distribution systems. WaterWiSe has a deployment of wireless sensor nodes in the city area of Singapore that measure water pressure and flow rate, transmitting data in real-time back to a central server. At the server, data is analysed for events and assimilated into an on-line hydraulic model that is used to provide rolling predictions of demand and consumption in the water network. This processing and modelling is used by the water utility to aid decision making in operating the WDS. Since the modelling is intended to run in an on-line manner, these inputs must be provided at consistent time steps, in this case hourly intervals [96, 86].

The need to provide data to the on-line model at regular intervals presents several challenges. First, sensor or network failures may mean that some nodes may not have reported sensor readings by the time that the model is updated. This missing data can affect the output of the on-line model as the stream must be ignored. Since there are only limited inputs to the on-line model, losing a stream may mean that the demand prediction algorithm may not converge. Estimating these missing sensor values through imputation will ensure that a complete data set is available to the on-line model. Secondly, the on-line model attempts to solve a complex estimation and calibration optimisation problem using a limited number of data streams. Therefore, it is possible that the output of the model would be improved thorough the addition of more inputs.

5.3 An on-line GPR based imputation and prediction algorithm

The previous section motivated the need for an on-line imputation and prediction mechanism for a WDS application. While there are several well studied independent algorithms for both imputation and forecasting, both techniques will be required to generate values for the hydraulic model during the analysis. Therefore combining both imputation and prediction into one model is advantageous. This section describes an on-line algorithm based around GPR (see Section 2.6.3) to address this need.

Using GPR for data imputation and prediction allows flexible modelling within a non-parametric Bayesian framework. This is powerful because it allows regression (and hence prediction) without overfitting (where a complex model describes noise or random errors rather than the datastream) though the optimisation of the marginal likelihood[88] and offers advantages over traditional prediction techniques (for example, see Section 2.6.3) as it can reduce the complexity of discovering an appropriate regression function for the data.

GPR can also incorporate multiple inputs into a single model, this allows inputs from one data stream to influence the output of other streams prediction. It has been shown that the use of correlated inputs to GPR will improve the accuracy of the predicted values (see Section 2.6.2). Given that data collected by sensors in the WDS tends to be highly correlated, the use of an prediction algorithm that exploits correlation can be expected to improve the quality of estimated sensor readings.

In order to apply GPR the following steps must be taken:

1. Understand the characteristics of the data

2. Identify an appropriate Covariance Function

3. Add correlation between sensors nodes to the covariance function

Sections 5.3.1 to 5.3.3 discusses these steps.

In order to incorporate GPR into an on-line algorithm it is important to identify an appropriate windowing function (to update predictions) before performing the imputation and prediction of sensor values. The prediction process is discussed in Section 5.3.4 and the window function in Section 5.3.6.

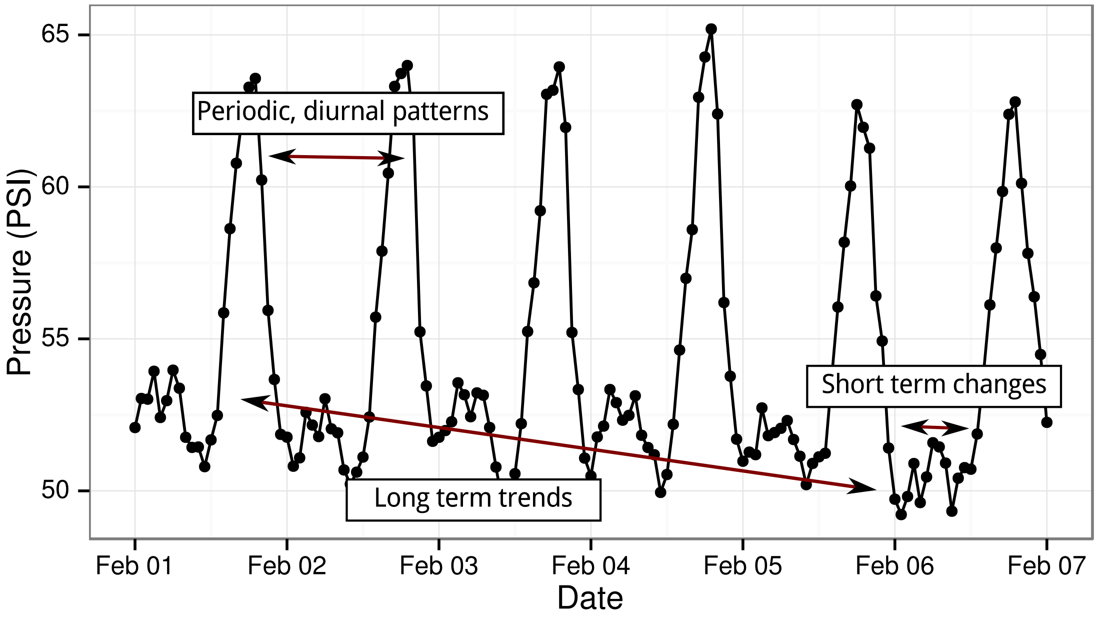


Figure 5.1: Annotated data showing data stream trends between 1st February 2010 and 7th February 2010

5.3.1 Identifying data characteristics

The accuracy of the output of the regression function (that is, imputation and forecasting of sensor values), is dependent on the GPR being able to take account of the characteristics of the data stream. In the water monitoring application, the parameter being predicted is water pressure in Pounds per Square Inch (PSI). Figure 5.1 shows hourly averages of water pressure data taken from a sensor node between 15th February and 15th March 2010. There are several characteristics that are present in this data stream:

- Short term trends, representing minor fluctuations in the water pressure data (as water is consumed during the day).

- A diurnal trend, representing the normal water consumption pattern of the network (pressure is higher at night because of lower consumption).

- A longer term baseline trend (related to operational of seasonal changes in demand)

An appropriate window size for training is at least two weeks; this accounts for one week of training data and a second week to incorporate recent trends.

Having identified the trend components of the data stream, a suitable covariance function K must be defined for the GPR that takes into account these characteristics.

5.3.2 Choice of covariance function

The covariance function of a Gaussian Process (GP) encodes the assumptions about the function that represents the data and specifies the relationship between pairs of inputs (see Section 2.6.3). To provide accurate predictions, it is important that terms of the covariance function can assert assumptions about patterns in the data gathered. Rather than use some arbitrary function to represent the correlation between inputs, there are a wide range of covariance functions (for examples see Rasmussen and Williams [88]) that can be used to represent components of the data stream. Individual covariance functions can be combined to represent more complex models.

Having previously identified the patterns in the water pressure data stream, the covariance function shown in Equation (5.1) is constructed to be the sum of a periodic function k_1 to represent the daily pattern (the diurnal trend identified above), a non periodic component k_2 that maps the trend over time (the baseline trend). This fits the expectation that sensed pressure readings can be modelled by the combination of a periodic signal, and a second term to take account of longer term trends.

$$K(t,t') = k_1(t,t') + k_2(t,t') \tag{5.1}$$

Both covariance terms k_1 and k_2 are represented by the Matèrn function (see Section 2.6.3). Through experimentation, a smoothness parameter $v = 5/2$ for both k_1 and k_2 (Equation (5.2)) has been shown to be a good match to pressure patterns seen in the water distribution system.

$$k_{matern\frac{5}{2}} = h^2 \frac{2^{1-v}}{\Gamma(v)} \left(\frac{\sqrt{2vd}}{w} \right)^v K_v \left(\frac{\sqrt{2vd}}{w} \right) \tag{5.2}$$

To represent periodic signals, the mapping of values between points in time is modified by replacing the input space function $d = |x - x'|$ of the periodic term k_1 with $d = sin\ \pi \, |x - x'|$. This modification of the distance function is commonly used to encode periodicity in covariance functions.

Finally, to represent measurement noise the covariance function is extended with a third term $\sigma_n \delta(t, t')$ where σ_n is the estimated measurement noise, and δ the Kronecker delta to become

$$K(t, t') = k_1(t, t') + k_2(t, t') + \sigma_n \delta(t, t') \tag{5.3}$$

When selecting the hyper parameters for the covariance function, since regular, hourly sampling intervals are expected for the pressure data, the scale is set to the unit length, that is $w = 1$. The output height h is effectively a scaling factor on the output of the covariance function, limiting the variance of the covariance function output; h is typically set to be the standard deviation of the observed data following the recommendation of Rasmussen and Williams [88]. The amount of estimated measurement noise becomes another hyper-parameter of the Gaussian Process and is set to $\sigma_n = 0.2$ based on prior experimentation.

The covariance function described in this section allows the GP to encode the relationship between samples in water pressure data within a WDS. Each term in the covariance function corresponds to a trend observed in these data streams.

5.3.3 Adding correlation to the covariance function

The output of the GPR prediction can be improved by exploiting correlations between nodes. Neighbouring sensors are likely to have well correlated readings (although this may not always be the case), and thus the values reported by these correlated sensors can be used to inform the prediction algorithm and help improve the predicted readings.

Common to all forecasting algorithms the GPR process relies on future data points following similar trends to the historical data used to inform the model and produce accurate predictions (see Section 2.6). However, changes in trends within the underlying data stream will affect the accuracy of the prediction because these are not present in the training set. For example, consider the following hourly averaged data taken over one week from two WaterWiSe sensor nodes. On the 6 and 7th day the average pressure rises (See Figure 5.2). If only the previous five days were used for training, the model would have no way of predicting the change in trend. If there were a node failure on these dates, it is not unreasonable for the prediction mechanism to assume the baseline pressure will static rather than increase to the new value, which will effect the accuracy of the prediction.

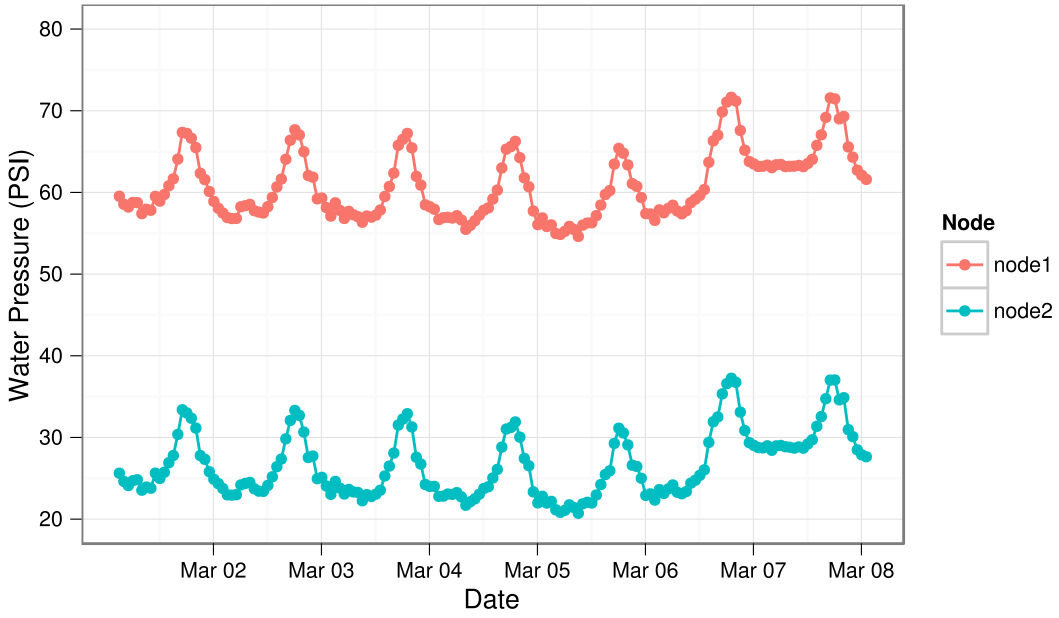


Figure 5.2: Sensor data from a well correlated sites (r = 0.99)

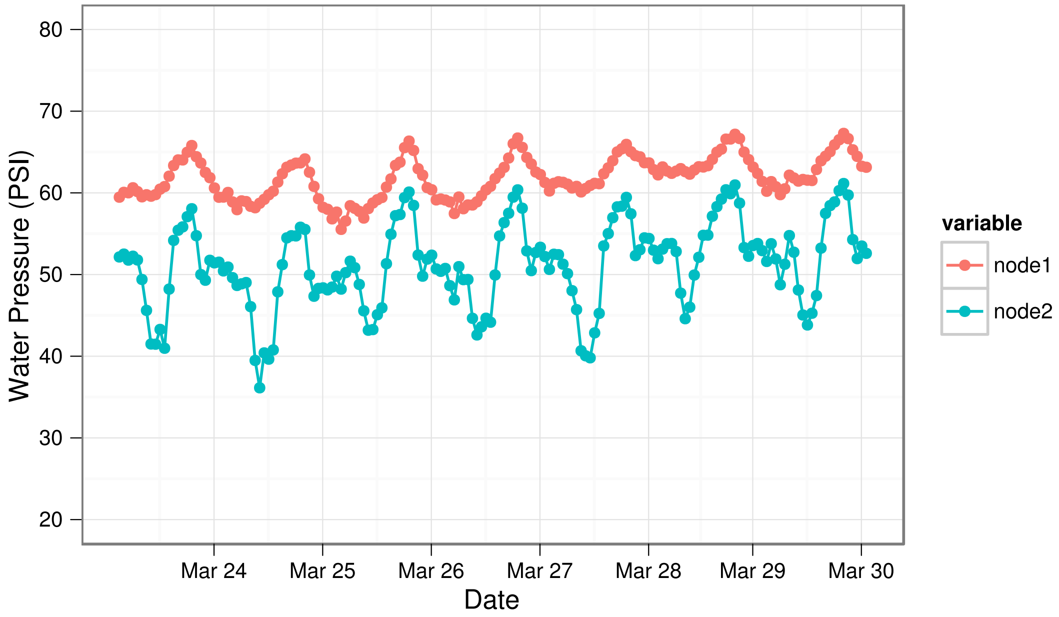


Figure 5.3: Sensor data from a less well correlated sites (r = 0.738)

To determine which sensors are well correlated, the Pearson r correlation coefficient is calculated for each pair of sensors in the network. Well correlated sensor data streams such as those in Figure 5.2 will be used as inputs to a single GPR process. Where data streams are less well correlated (such as those in Figure 5.3) a new GPR prediction process will be used.

To capture the correlation between multiple data streams, a multiplicative term over the sensor identifier ℓ is applied to the covariance. This term is defined by the Pearson r correlation coefficient between the two sensors under consideration [75]. The extended covariance function K over x and ℓ then becomes.

$$K([x,\ell],[x',\ell']) = k_{node}(\ell,\ell')(k_1(x,x') + k_2(x,x')) + \sigma_n\delta(x,x') \tag{5.4}$$

The extended covariance function described in this section allows the GPR to take account of correlated sensors when making predictions. As inputs from correlated nodes can influence the predicted values, the use of a correlation component improves the accuracy of the predictions made and allow unexpected changes to the baseline data to be predicted.

5.3.4 Imputation and Prediction of Sensor Values

Having determined the appropriate covariance function for the characteristics of the data, and included the correlation between data streams in the network in the covariance function. The imputation and prediction algorithm is run hourly and is used to ensure a complete set of inputs is available for the on-line model.

The GPR based prediction algorithm is given in Algorithm 5.1 and has two stages:

1. initialisation (initialise procedure)

2. on-line prediction (run procedure)

Initialisation

During the initialisation phase (Algorithm 5.1 line 1) the GPR prediction process will :

Populate training data For each data stream that is to be included in the GPR, retrieve the previous two weeks of hourly averaged water pressure data from the database and store in the training window.

Algorithm 5.1 GPR prediction and imputation algorithm

```
 1: procedure Initialise                                          ▷ Initialise GPR
 2:     Populate training data
 3:     Initialise hyper-parameters
 4:     Impute()
 5: end procedure

 6: procedure Run                                                 ▷ On-line Operation
 7:     Load training data
 8:     Update training data
 9:     if There are samples missing in the training data then
10:         Impute()
11:     end if
12:     Update hydraulic model
13:     Store training data
14:     sleep until next prediction required
15: end procedure

16: procedure Impute                      ▷ Use GPR to estimate missing sensor values
17:     Predict missing sensor values
18:     Update training data with predicted samples
19: end procedure
```

Initialise hyper-parameters Optimise the hyper-parameters Φ based on the characteristics of the data, and set the correlation term for each pair of sensors to the Pearson r correlation coefficient.

Impute missing data Impute any missing values in the training window using the imputation function described below (Algorithm(5.1 line 16).

Model update (run)

Each time the on-line hydraulic model requires updating (Algorithm 5.1 line 6) the algorithm will:

Load training data The training data for this GPR prediction process is loaded from memory.

Update training data The training data window is updated to take account of any sensor values that have been received since the last time the model was run. Details of the windowing mechanism are given in Section 5.3.6.

Predict missing data Predict any missing sensor values using the imputation function described below. The imputation stage is skipped if there is a complete data set to avoid unnecessary computation.

Update hydraulic model The most recent set of sensor data including predicted values is passed to the hydraulic model

Store training data The training data is saved for use at the next model update phase.

Imputation

When imputation (Algorithm 5.1 line 16) is required the algorithm will:

Impute missing values GPR is used to impute these values using the data in the training window as input. A standard GPR based approach to imputation is used as per Rasmussen and Williams [88].

Update training data Any predicted sensor readings are appended to the window of training data. The update of the window is described in detail in Section 5.3.6.

This section has described the approach used to allow GPR based prediction and imputation of sensor values within an on-line WDS monitoring application. The following section describes the Virtual Sensors algorithm that builds upon the algorithm described in this section to allow long term prediction of values where no sensor node is currently deployed.

5.3.5 Virtual Sensors algorithm

In Section 5.2 it was noted that the on-line demand and consumption prediction component of WaterWiSe could benefit from an increased number of data streams to be used as inputs. One solution would be to use more physical sensors to gather data. However, deployment of these sensors is costly, and requires time and effort in deployment and maintenance of sensor units (for example, see Figure 5.4). Forecasting can be used to estimate future sensor readings based on historical data. Given that a robust forecasting mechanism can estimate sensor values for an extended period of time, it is possible that this approach can be used to create *virtual sensors* to estimate values at sites where no permanent sensor exists.

Figure 5.5 presents an overview of the virtual sensing concept. A set of pressure transducers are deployed within the WDS, and their real-time data is used as an input to the on-line hydraulic model. The solid circles represent real sensors that are permanently deployed on the WDS. The dashed circles represent virtual sensors, where data is predicted based on both a small window of training data from a temporary deployment and the data gathered by a correlated static sensor.

Figure 5.4: Debugging a WaterWiSe sensor node

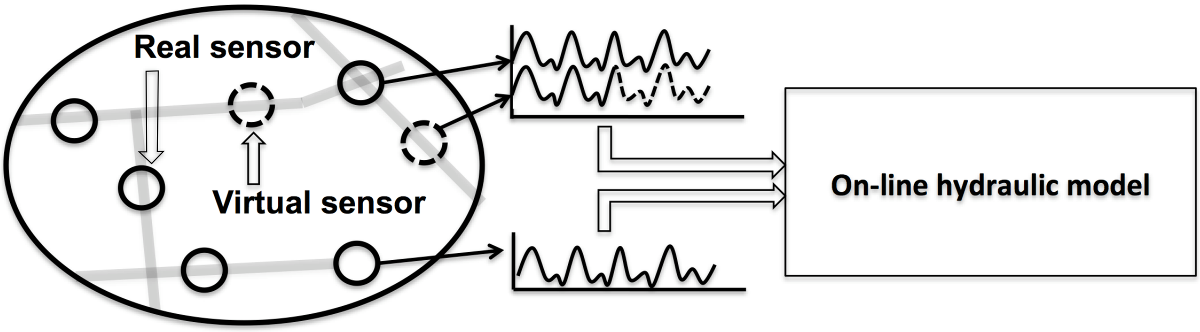


Figure 5.5: Virtual Sensors concept

The process of integrating a virtual sensor into a system has two steps:

1. Initialisation

2. Data Prediction (on-line operation)

Virtual Sensors: Initialisation

Initialisation of the GPR for the Virtual Sensors algorithm uses a modified approach of the algorithm used to initialise the on-line imputation. This modified approach contains an additional stage to identify appropriate locations for virtual sensors. The modified initialisation stage is described below.

Potential candidate sites for virtual sensors are determined by a WDS domain expert, in this case, the team member responsible for developing and maintaining the on-line model.

To determine whether a candidate site is suitable for use as a virtual sensor, a period of *training* data from the candidate site needs to be collected. The training data allows the relationship between data collected at this site and other static sites in the network to be examined. While gathering this training data will require a node to be deployed at that location, the temporary deployment has several advantages over permanently deploying a node. The cost of the hardware can be prohibitive and gathering a period of training data can allow the hardware to be reused at other locations. Additionally, there is a significant cost associated with the infrastructure required to make a permanent deployment. For example, while a temporary connection to the WDS can be made through a hydrant, a permanent node makes use of a more robust connection which may require a new manhole and associated valve to be installed in the network.

The Pearson r correlation coefficient for the candidate site and other sites in the network is calculated. If the collected data values are well correlated with any static sensor the site is suitable to be a Virtual Sensor. Otherwise, sites where the data is poorly correlated with other sites in the network represent ideal sites for the placement of new, static sensors.

Once a site has been validated as suitable, a new virtual sensor is added to the GPR prediction process, using the methodology described in Section 5.3. The correlation term added to the covariance function (Section 5.3.3) is the r value calculated during site selection.

Virtual Sensors: prediction

The process used to update the on-line model when using Virtual Sensors is the same as the one presented in Section 5.3.4. However, as a physical node is not used to collect data at the virtual sensor location a prediction will have to be made at each time step. Additionally, there is the added complexity of maintaining the training data used to inform the Virtual Sensors prediction process.

To retain a suitable set of training data the windowing mechanism makes use of *feedback* to recursively update the input to the GPR. The windowing mechanism is discussed in the following section.

5.3.6 Windowing mechanism

A key part of the GPR based approach is maintaining an appropriate window of training data to allow accurate predictions to be made. Additionally, the Virtual Sensors approach requires the data collected during the training window to be maintained. The following section discusses the creation and maintenance of the training data.

Using GPR for data prediction has a computational complexity $O(N^3)$ where N is the size of the input data set. Since the Virtual Sensors system is intended to provide timely input to the on-line hydraulic model, care must be taken to restrict the input data to provide realistic computation times. Whilst iterative schemes for GPR [75] have been shown to be efficient, there are two factors that in this case mean that such an iterative scheme may not be appropriate for the virtual sensors approach.

1. One of the motivating factors for this work is that a significant amount of latency has been observed in the network. This means that sensor data can arrive out of order, or with a large delay between the time when the sample was gathered and it arriving at the sink. The prediction approach is intended to alleviate these problems by estimating readings when this occurs. However, these samples can add to a greater understanding of the phenomena when they arrive at the sink. Thus the ability to incorporate samples that arrive out of order into the predictive model is desirable.

2. The iterative approach assumes that a stream of data from each node will periodically update the model. However, the virtual sensors extension to the GPR process relies on learning the relationship between sensors over a short period of training data then continuing to use this relationship to inform future predictions in the long term. Therefore, there will be input to the model aside from the initial period of training data. As the *downdate* [77] mechanism removes samples based on their age (and

thus relevance to the current prediction) it is likely that training period will be removed from the input to the model, or that this training data will lose relevance to the current model given the age of the samples.

It is computationally infeasible to use all previously gathered data to train against. To address this, a windowing scheme is applied, using a sliding window of two weeks worth of input data. The first week is treated as training data, and the second is used to learn the current (that is, most recent) trends.

An overview of the feedback mechanism is given in Figure 5.6 when updating the model one of three scenarios will occur:

Case 1: Fresh data The latest sample received by the sensors is used to update the input data for the model. To maintain a constant window size, the oldest sample is removed.

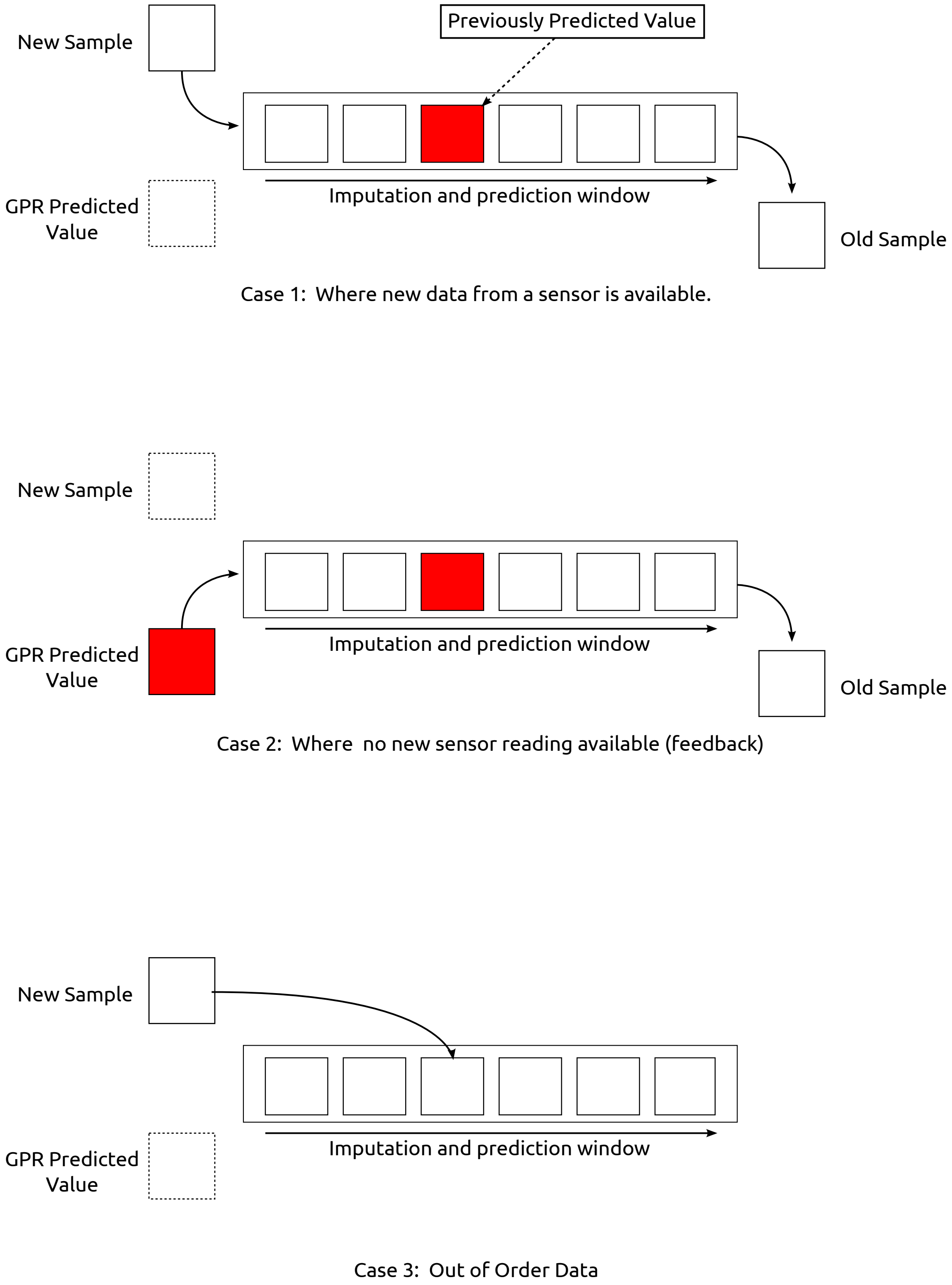
Case 2: No data is available for a node GPR is used to estimate any missing sensor readings. The output of the GPR is then used to update the input data. This process is referred to as the *feedback mechanism*.

Case 3: Samples arrive out of sequence If the samples received would affect the training data contained within the model, then the time the new sample corresponds to is updated based on the received data. Otherwise, if the sample arrives outside of the training data window (either too early or too late), it is dropped.

The feedback mechanism is necessary because during extended periods where no sensor reading is available, the covariance function will give less weighting to older data points in the output of the prediction. Given that the on-line imputation algorithm relies on a short period of training data, then it is inevitable that this training data will lose relevance to the predictions made, thus making the prediction of data for virtual sensors impossible. By recursively using the predicted values as input, the relationship between the training data and any other correlated nodes is retained in the model, allowing predictions to be made indefinitely.

5.4 Evaluation of imputation and prediction process

To evaluate the use of the imputation and prediction algorithm to estimate missing values, several controlled experiments were performed using hourly averages of water pressure data collected from the


New Sample
Previously Predicted Value
GPR Predicted Value
Imputation and prediction window
Old Sample
New Sample
GPR Predicted Value
Imputation and prediction window
Old Sample
New Sample
GPR Predicted Value
Imputation and prediction window

Case 1: Where new data from a sensor is available.

Case 2: Where no new sensor reading available (feedback)

Case 3: Out of Order Data

Figure 5.6: The three cases that can occur when updating the Virtual Sensors window. In case 1 a new sample arrives and the oldest sample is removed; in case 2 a sample has not arrived in time so GPR predicts this value; in case 3 a value arrives out of sequence and is used to replace a previously predicted value.

WaterWiSe@SG network.

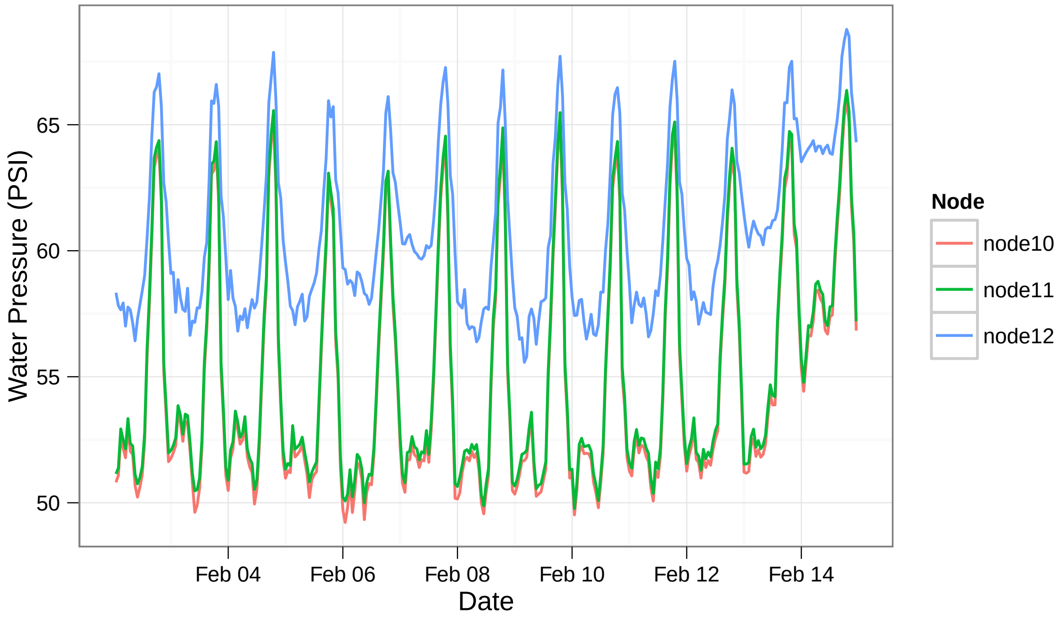


Figure 5.7: Data streams used in experimentation

For each experiment in this section, hourly averaged pressure data traces between the 1st to the 14th of February 2010 were taken from sensors deployed in the WDS. The data traces for these sensors are shown in Figure 5.7.

These particular sensors and time period were chosen as the data shows a significant upward change in trend toward the end of the trace. This trend is not something that could be predicted purely using the time history of the data, and thus indicates the performance of the correlation term in the GPR's covariance function that is fundamental to the approach. Two sensors (herein referred to as node 10 and node 11) are permanently deployed 600 meters apart on the same 800 mm pipe main, these sensors were chosen as the data streams are very well-matched ($r = 0.999$). A third sensor (node 12) is chosen as it is not as correlated ($r = 0.929$ with node 10, and $r = 0.93$ with node 11) and has a different baseline pressure level; however, it shows the same general trends as the other two streams.

In each experiment the input from the first half (168 data points) of all data streams is used as training data, this provides the algorithm with sufficient information to learn the characteristics of the data and any relationships between nodes. Following this the data stream to be evaluated against is removed from

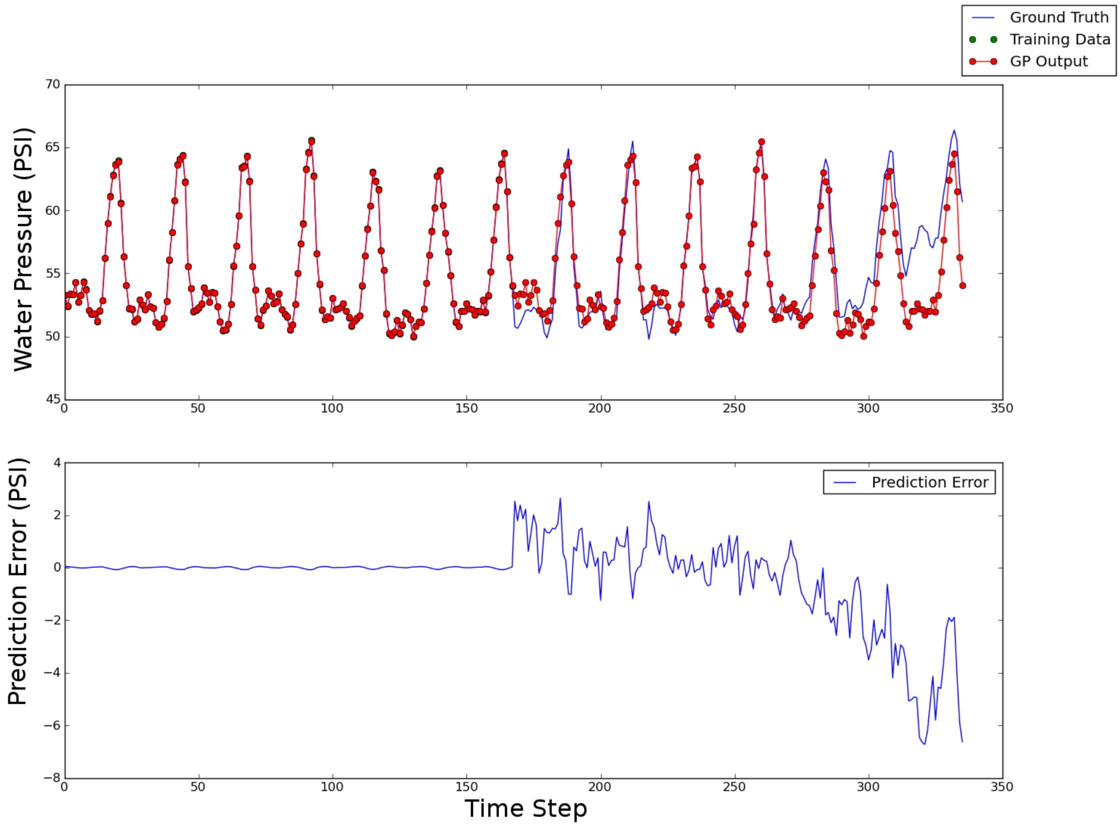


Figure 5.8: GPR algorithm, prediction for node 10 without using a correlated input

the input and its remaining points predicted by the GPR. The Root Mean Squared Error (RMSE) of these predicted values is then calculated to provide an indication of the accuracy of the prediction.

Prediction using a single data stream

Examining the output of the prediction algorithm when no correlation is used provides a baseline performance target against which the experiments in the remainder of this section can be evaluated. Excluding the correlation component of the algorithm means that when the data stream is removed from the input, the GPR process makes use of just the historical data to estimate the missing readings. In this case the prediction mechanism cannot be expected to predict the rise at the end of the data trace (Figure 5.7), since this trend has not been seen in the training data and consequently the estimated values can be expected to continue to follow the trends observed during the first week of training data.

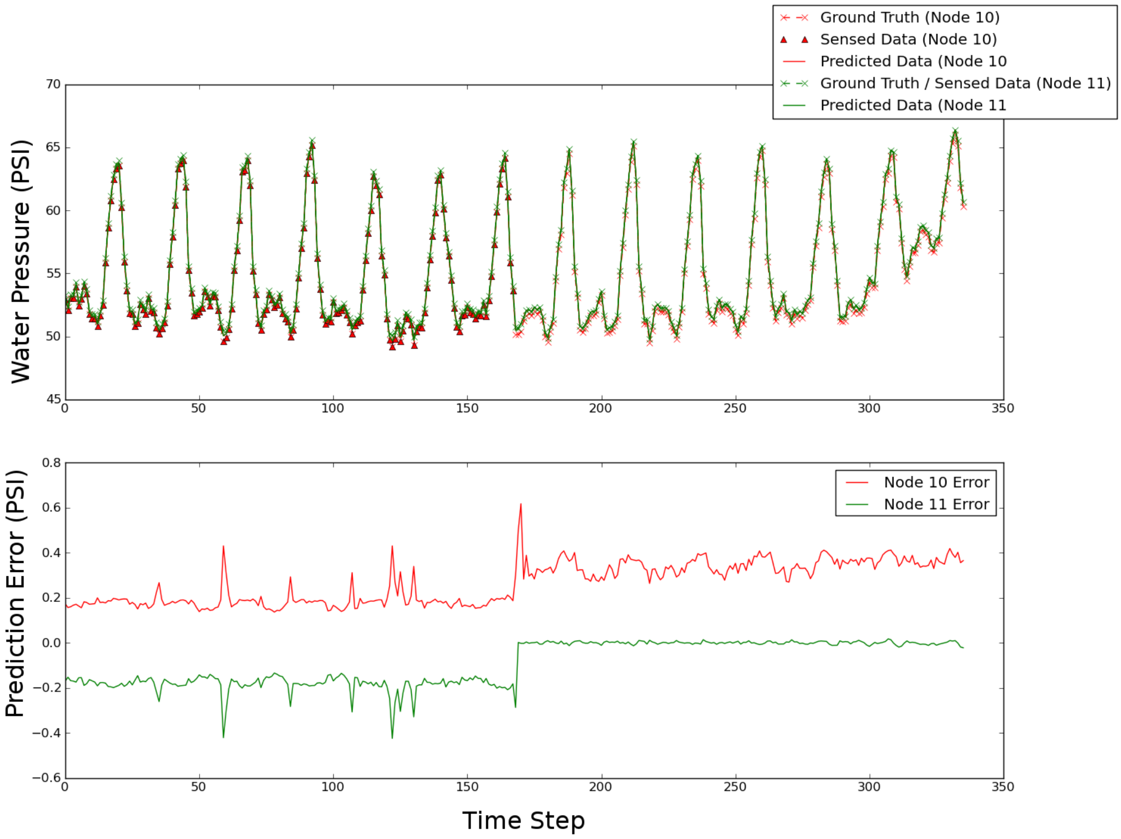


Figure 5.9: GPR output, prediction with correlated sensors

Figure 5.8 shows the output of the prediction algorithm when no correlation component is used. The RMSE value observed was 1.55 PSI. This confirms the expectation that without the correlation component of the algorithm, the output of the GPR is unable to predict the unexpected event at the end of the data trace.

Prediction with correlation

It is expected that adding a correlated node to the input of the virtual sensors algorithm will increase the accuracy of the predicted values and allow the prediction of features that would not have been possible before (such as the rise in water pressure at the end of the data trace).

To illustrate the effect of adding correlated nodes to the input of the prediction algorithm, the previously described experiment was repeated with the addition of a correlation term describing the relationship between the data stream added to the covariance function. After seven days (168 data points), the data stream from node 10 was removed, leaving GPR to predict the remaining data points based on the previous sensor readings and the input from the correlated sensor.

Figure 5.9 shows the predicted points overlaid on the observed data stream, starting from when node 10's observed data stream was removed. It is clear that when correlation is used GPR takes into account the reference data stream from node 11, and adapts to the upward trend change in the data set. The estimated sensor readings more accurately represent the measured values with an RMSE of 0.01 PSI when correlated readings are included in the input. This is a significant improvement in comparison to the RMSE of 1.55 PSI when no correlation component was used.

One drawback to the GPR prediction process presented here is the small error induced in the output when correlated sensors are used. This error takes the form of a small deviation towards a central point of all sensors included in the input. This can be seen most clearly in the training data period, where *mirrored* error readings can be seen in the traces. This prediction error comes from the correlation term in the covariance function being applied to each sensor stream. This means each stream will have a minor effect on the output of any other streams included in the same process, with the covariance function *pulling* the predicted values slightly towards a central point.

Using less correlated sensors

Not all data streams will be as well correlated as the one in the previous examples. However, as long as the underlying trends in the data stream are similar, the correlation component of the covariance function can still be used to influence the prediction of these trends. For this experiment, the same approach was used, but instead of using the node 11 as input, the less correlated node 12 is used (see Figure 5.7). While both data streams follow the same general trends, the data from node 12 has a higher baseline pressure, a lower range between high and low points, and less extreme spikes between each day. However, the rise at the end of week two is still present, which should enable the algorithm to take this trend into account when estimating the missing data stream.

Figure 5.10 shows the output of the GPR prediction algorithm using this data set. While the RMSE of 0.79 PSI in the estimated data is greater than when using a well correlated sensor, it is an improvement over not using a correlated component. In particular, the rise at the end of week two is detected and incorporated into the estimated sensor readings. Additionally, a greater weighting of the historical data

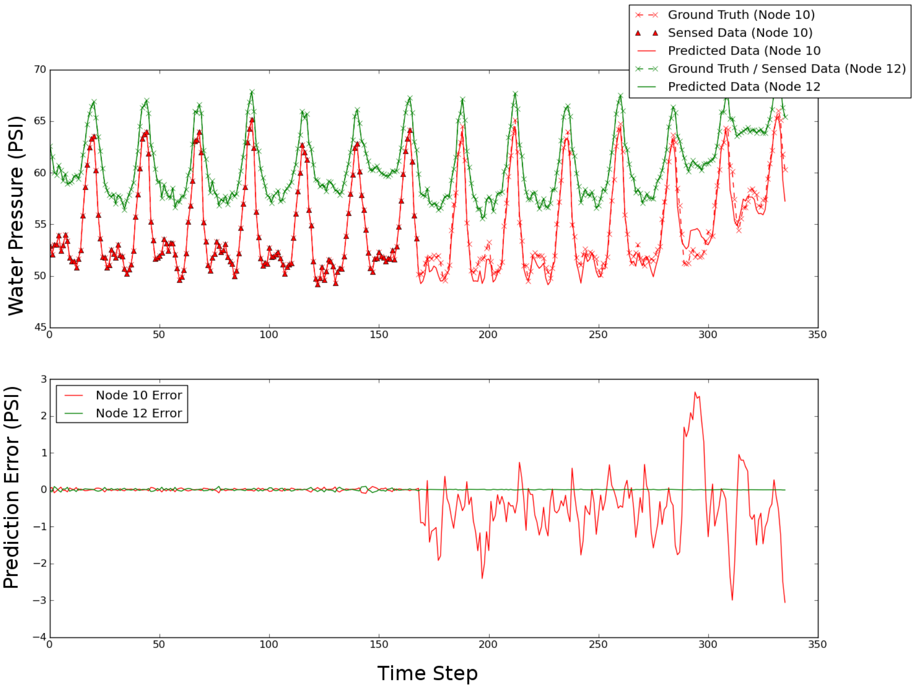


Figure 5.10: Prediction with less correlated node

from node 10, means that the daily trend pattern found in the data stream (specifically the rising pressure in the middle of each diurnal cycle) continues to be observed in the output of the algorithm.

Supporting the loss of all sensors in the input

The above examples have demonstrated that the prediction component of the GPR algorithm is improved through the use of correlated sensors. However, sometimes communication failure can remove the data stream from all correlated nodes. In this case, the algorithm should fall back to using the historical data for prediction.

In this example, the input data for both node 10 and 11 was removed after the first week of training data, leaving the algorithm with no input for both data streams. Figure 5.11 shows the output of the algorithm. As expected, the estimated values are based on historical data values, and the rise at the end of the week is not included in the output.

The RMSE of the predicted sensor readings for this experiment was 1.68 PSI, which is slightly higher than when estimating the same data stream without a correlation term.

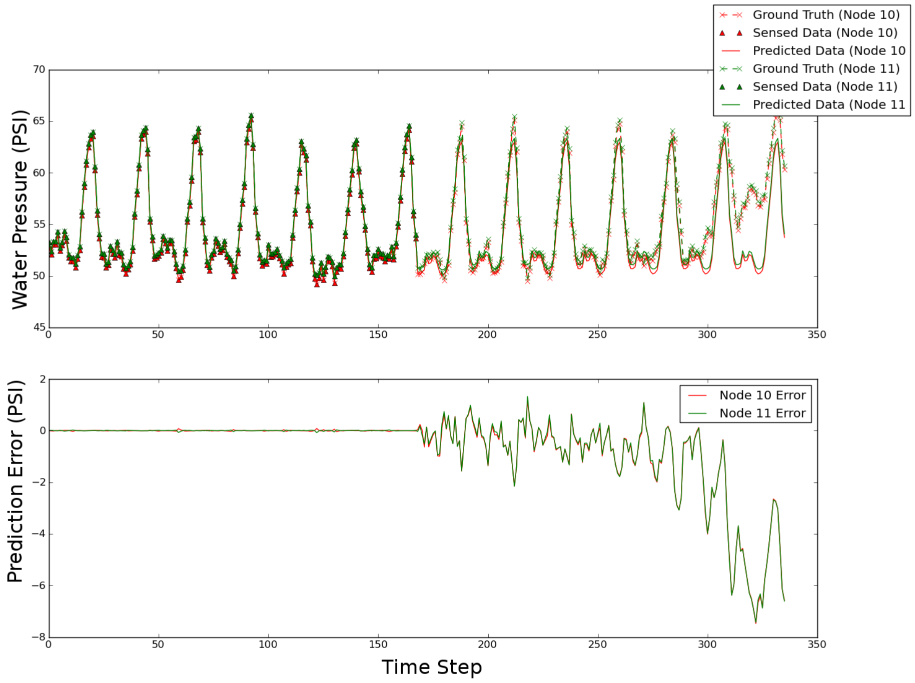


Figure 5.11: Both nodes drop out of the system, fall back onto historical prediction

Given that the algorithm will revert to using historical readings when no sensor data is available, is should be expected that the predicted values would be the same as in the case when no correlation is used. However, the correlation term in the covariance function will still exert some influence on the predicted values for each sensor. This has the effect of adjusting the predicted readings for each node towards a central value.

Using more than one correlated sensor

The algorithm is able to take several correlated sensors as input. This has the advantage of adding some redundancy to the prediction mechanism, as it may be less likely that several nodes will fail at the same time. This means that, as long as there is at least one data stream available, all estimated values will still be influenced by its input and the algorithm will not have to fall back onto historical prediction.

In this example, an additional sensor (the less correlated node 12 used in the above example) was added as input to the algorithm. As in previous experiments node 10 is removed from the input at the end of the training period and its values predicted. Figure 5.12 shows results for this experiment. The

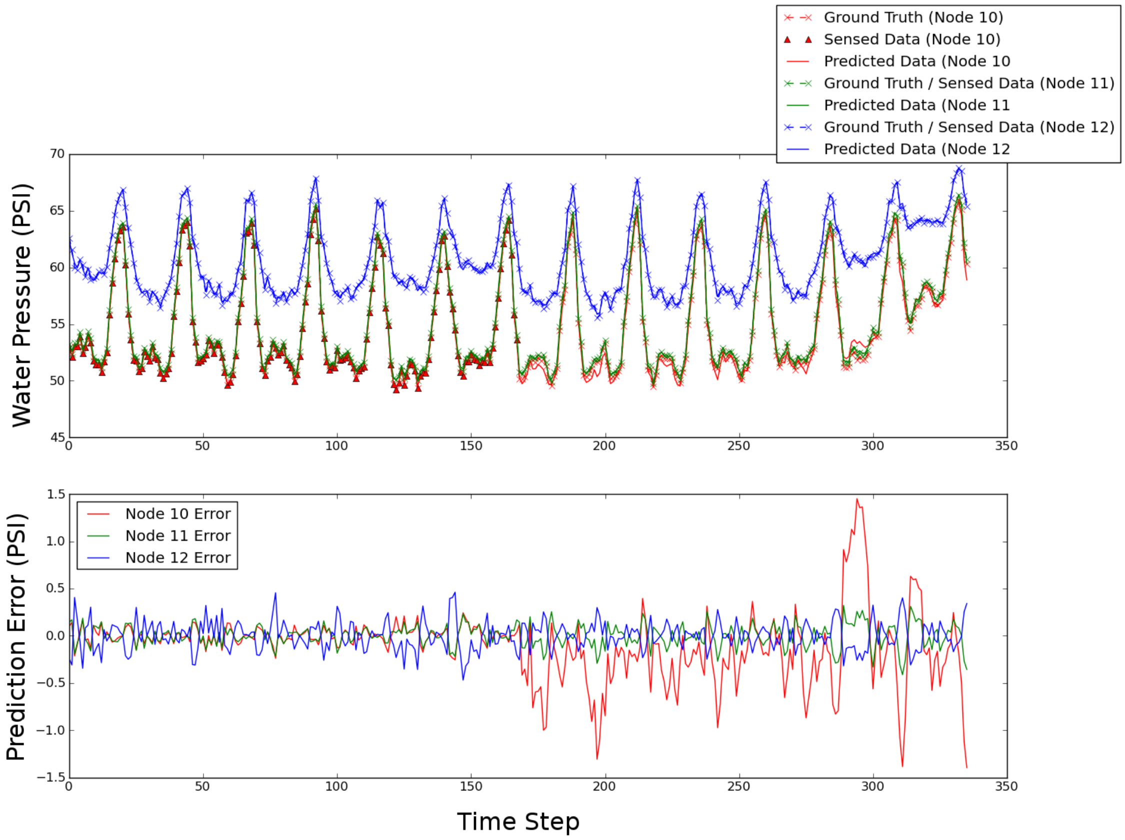


Figure 5.12: Output of prediction algorithm with more than one correlated sensor

Experiment	RMSE (PSI)
Without Correlation	1.55
With Correlation	0.01
Using less well correlated sensors	0.73
Using more than one input sensor	0.38
All nodes drop out of the data stream	1.68

Table 5.1: Summary of performance for missing data prediction

predicted readings for Node 10 had an RMSE of 0.38 PSI, this error is greater than the RMSE of 0.01 PSI when a single correlated sensor is used. The increase in prediction error can be attributed to the use of the less well correlated node 12 as input. While a greater weighting is applied to the data from the well correlated node 11, the sensor readings from node 12 will still have an influence on the predicted readings.

5.4.1 Summary of results for imputation and prediction

Table 5.1 summarises the performance of the imputation and prediction algorithm.

It is clear that the performance of the algorithm is improved by including a well correlated sensor in the input data. In all instances where a correlated sensor was used, the algorithm was successfully able to predict the unexpected rise at the end of the testing data set. In the best case, the RMSE of the estimated data stream was reduced from 1.55 PSI without using correlation to 0.01 PSI when correlation was used, indicating a lower error in each prediction. This demonstrates that the use of a correlation term allows the values predicted by the algorithm to more closely matched the input data.

It is possible to adjust the weight given to the correlated stream by altering the correlation correlation component of the algorithm, with lower r values placing a greater emphasis on the historical segment of the input data rather than the correlated data. This means that it is possible that less correlated sensor traces can still improve the accuracy of the algorithm and aid in the detection of unexpected events.

Finally, if all data traces drop out of the input, the algorithm will fall back to using historical data to make its predictions, which may mean that unexpected events will not be predicted. However, given that the algorithm can take multiple data streams as input, if any one of these streams remains active, the algorithm is able to take account of these trends.

This section has described on-line imputation and prediction for water pressure data when readings are missing but a complete data stream must be provided. The next section explores increasing the number of data streams without increasing the number of permanently deployed sensors.

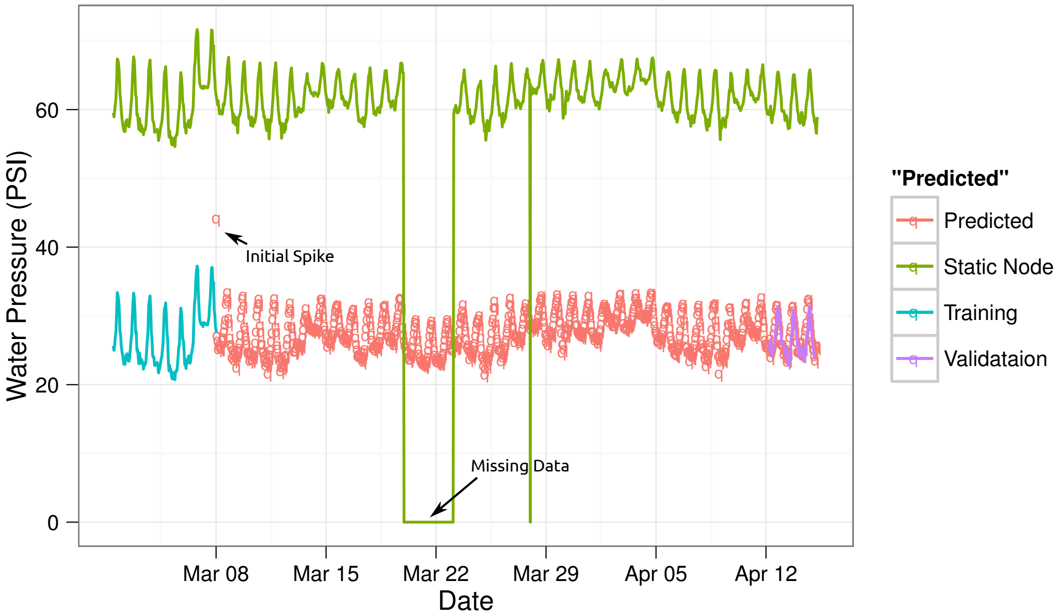


Figure 5.13: Training and prediction data for the virtual sensor, plus the data trace from the real sensor it was matched with. Note the initial spike when prediction starts and the period of missing data from the static sensor.

5.5 In-situ evaluation of Virtual Sensors

In order to validate the complete Virtual Sensors approach a field test was carried out. During March 1st–8th 2010, a mobile sensor node was temporarily deployed to gather pressure data. This data was averaged hourly and observed to be well-correlated both visually and statistically with an existing real sensor ($r = 0.999$). A Virtual Sensor was added to the system for this site, providing predictions based on the historical data and the data being gathered by the correlated sensor node. On April 12th–15th (thirty-six days from the end of the initial deployment), the mobile sensor was re-deployed to provide a data set to validate the predictions.

Figure 5.13 shows an overview of the GPR's predictions of the Virtual Sensor's data trace using the training data taken from March 1st–8th 2010 and the data from the well-matched real sensor. Both the real sensor's data stream and the Virtual Sensor's training data are shown in solid lines and the GPR predictions are shown as a dashed line. The initial spike in prediction after the training data stopped is a side-effect of the correlation setting in the covariance function, and only lasts for the first predicted data point. This effect is not as pronounced when the data streams are almost exactly the same, which

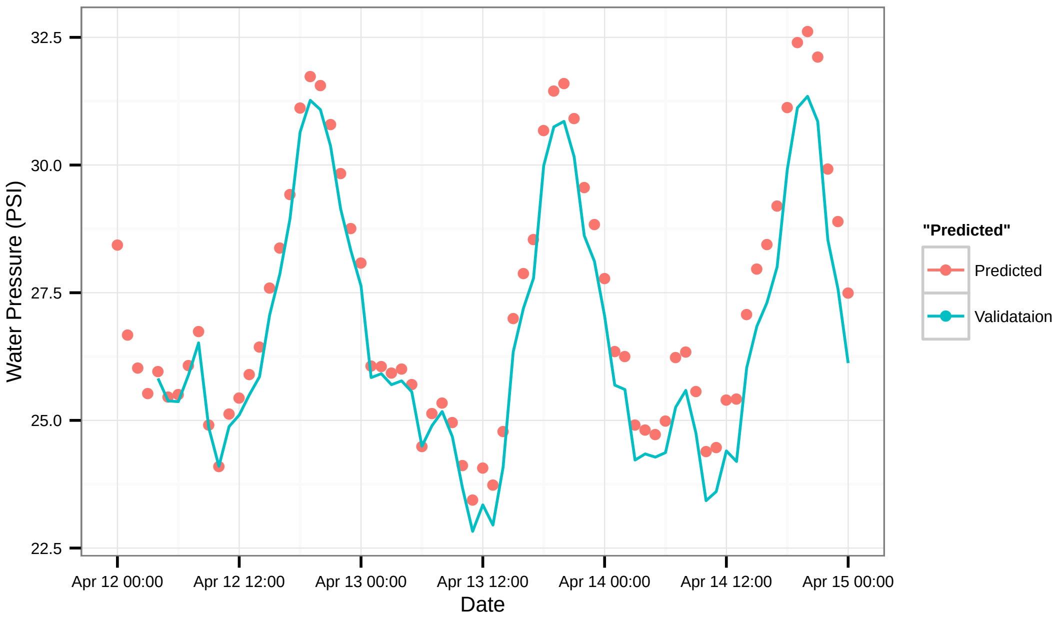


Figure 5.14: Comparison of the data taken for cross validation and predicted data. The RMSE is 0.8 PSI.

explains why this phenomena was not observed in the controlled experiment. The prediction period includes several periods with missing values from the static sensor, in which case the virtual sensor values were estimated using historical data.

Figure 5.14 shows the observed and predicted pressure data traces during the cross-validation period. The RMSE between the predicted data and the observed values was 0.756 PSI, with a maximum error of 1.394 PSI. These values are still within the acceptable boundaries of measurement uncertainty allowed by the on-line model. This result shows that the predicted data is still well-matched with the observed pressure data after an extended time period (thirty-six days).

5.5.1 Relationship to the FieldMAP framework

This section describes how the virtual sensors approach with regard to the FieldMAP Framework (Section 3).

The virtual sensors algorithm is an instantiation of the model component on the sink. The algorithm is able to exploit the sinks global view of the network to improve the informational output of the data received.

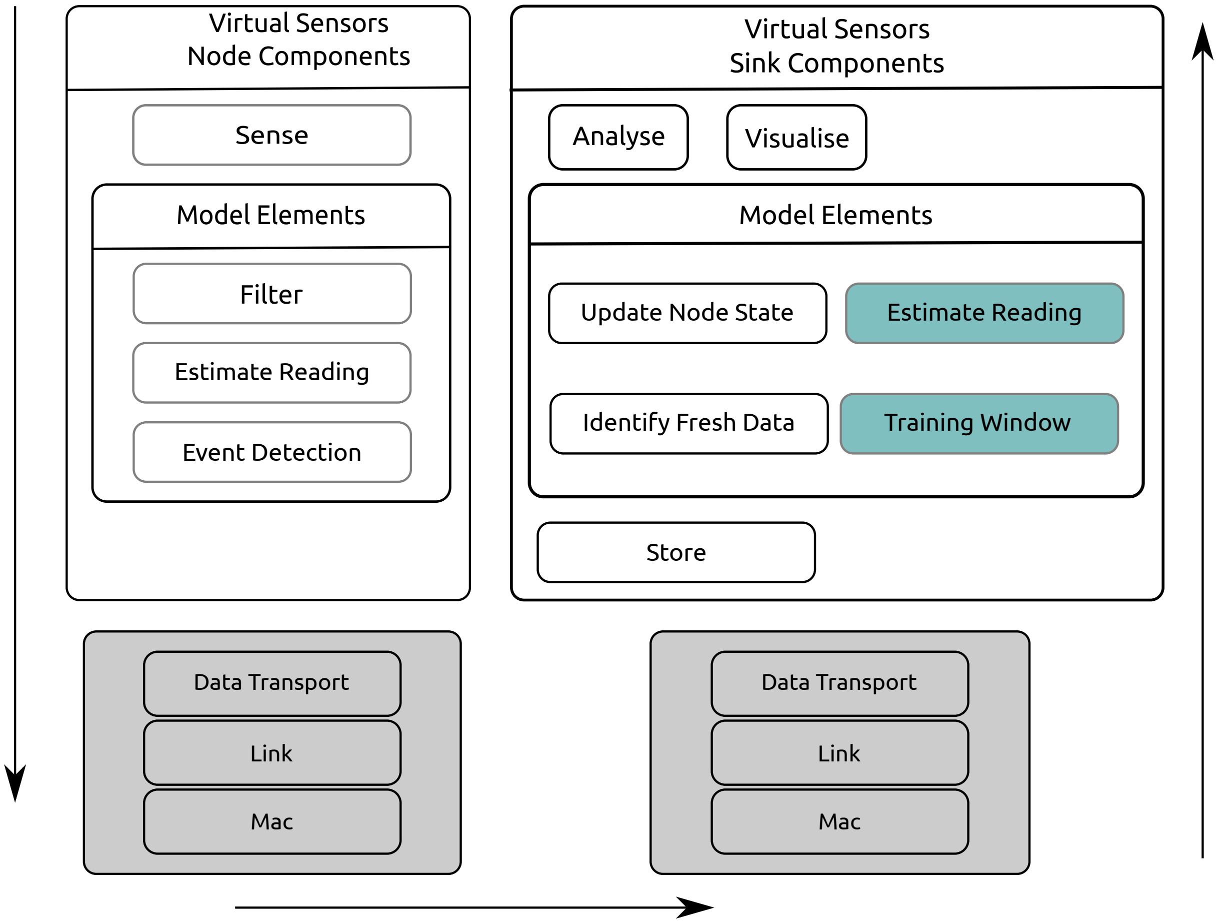


Figure 5.15: Overview of Virtual Sensors as part of FieldMAP

Figure 5.15 shows the extended FieldMAP architecture with the virtual sensors components highlighted in blue. The virtual sensors functionally takes place only at the sink. The node algorithm functions in the same way as in Chapter 3.3.2, transmitting a full stream of data without making use of the transmission suppression functionality of SIP. Future work will evaluate the performance of the where both SIP and virtual sensors are used together.

The *identify fresh data* is responsible for maintaining the training data. If *fresh* samples have been received from a node then the relevant part of the training window is updated using these values. If no fresh data is available, then the values for this sensor are estimated using the prediction algorithm and used to populate the training window. This process does not necessary take place each time a data sample is received, but may make use of sensor readings retrieved from the store if the model is updated periodically.

The GPR prediction process maps to the *update node state* component of FieldMAP. Where there are missing values or virtual sensors, the GPR algorithm is used to predict the current node state. To avoid unnecessary computation, if there are no missing samples, estimation using GPR can be avoided and the most recent sensor readings used.

5.6 Summary

This chapter used a WDS monitoring application to motivate the need for imputation of missing data, prediction of future values and estimation of a data stream where no sensor is permanently deployed.

Gaussian Process Regression (GPR) is a machine learning technique that can be used for the imputation and prediction of time series. A GPR based approach to estimate missing sensors readings in the data reported by WDS is proposed. The use of GPR is motivated in this situation by its ability to exploit correlated sensor readings to both predict and impute data values. The stages required to incorporate GPR based prediction into the on-line prediction algorithm are defined, along with process of incorporating correlations between sensor streams into the GPR to improve the predicted sensor readings.

The use of correlation has been shown to reduce the prediction error in well-matched data sets. The use of a correlation component allows unexpected trends to be accurately predicted as other data streams that observe these events can influence the output of the GPR.

A windowing scheme is proposed that aims to maintain a realistically sized data set and allow samples that are received out of order to be included in the predictions made by the GPR. To enable long term predictions to be made, this windowing scheme makes use of a feedback mechanism to maintain a complete set of training data by replacing missing values with the prediction for that time step.

Given a well correlated real sensor and a small period of in-situ training data for the Virtual Sensor, it is possible to predict hourly averaged pressure data with an RMSE of less than 1 PSI. When compared with validation data, this accuracy is maintained over a six week period. The approach was originally developed to support a water supply network. Nevertheless it clearly has applicability to many wireless sensing problems where either (a) missing sensor data needs to be estimated and / or (b) there is a need to estimate the phenomena at a location that is not physically sensed. Future work will further investigate the application of the virtual sensors algorithm over a longer period and a greater number of data types.

Chapter 6

Conclusions and Future Work

This thesis aims to remove roadblocks to the widespread use of Wireless Sensor Networks (WSNs), firstly by making them longer lived (through energy savings) and secondly by ensuring that the resulting data is suitable for higher-level analysis. A core element of the work is to define clearly the FieldMAP framework for WSN design that abstracts away considerations of operating system or programming language while defining the main components and identifying some of the design decisions that must be made. The Spanish Inquisition Protocol (SIP) algorithm, that arose from this framework, is both computationally simple and provides high-levels of compression for typical WSN data streams. The Virtual Sensors approach builds on prior work for Gaussian Process Regression (GPR) in the context of WSNs. Furthermore, the work is the first implementation of this approach to be used for an on-line, production Water Distribution System (WDS) monitoring system.

6.1 Research Questions

This thesis has explored the advantages of using model-based approaches to improve the efficiency and informational output of sensor networks. The research questions examined by the thesis can be summarised as follows.

1. Is transmission reduction beyond the current state of the art possible by combining model-based filtering with dual prediction approaches in environmental monitoring applications?

2. When using a dual prediction based approach is there a consistent relationship between the number of transmissions required and the error budget for a given data set?
3. Can combining multiple sensors readings into a single predictive model allow a greater reduction in the number of packets transmitted, than when compressing each stream individually?

Concerning virtual sensing, the following research questions are explored:

1. Can a prediction mechanism that exploits previously learnt temporal correlations between nodes be used to continue to estimate sensor readings where no value is currently available?
2. Can this prediction mechanism be used to estimate sensor readings where there is no sensor deployed, assuming a short period of training data?

6.1.1 Is transmission reduction beyond the current state of the art possible by combining model-based filtering with dual prediction approaches in environmental monitoring applications?

Yes.

SIP has been evaluated over several real world environmental monitoring data sets, and has been shown to outperform the transmission reduction performance of similar Dual Prediction System (DPS) algorithms. In the case of the Intel Lab data set, SIP comparable algorithms required ten times as many model updates to reconstruct the data within the same error thresholds. Additionally, the performance of SIP when deployed on node hardware has been demonstrated to be similar to simulated performance over these data sets.

The key differences between SIP and prior DPS work include:

- Reduced computational cost
- No training or convergence period.
- Decoupling the filter and estimation stages means that predictions do not have to be performed simultaneously.

SIP also offers other improvements over prior DPS work. The simple linear model has low computational cost compared to approaches that use Kalman Filters, or Autoregressive Integrated Moving

Average (ARIMA) models. The Piece-wise Linear Approximation (PLA) approach used to estimate sensor values does not require the training period required for ARIMA based approaches and does not have a convergence period where the reported sensor readings do not match the data as in the Least Mean Squares Adaptive Filter (LMS) based approach of Santini and Römer [95]. Additionally, a rate of change based approach to estimating sensor readings means that the node and sink can make predictions independently of each other, reducing the need for predictions to be performed in simultaneously on the node and sink where a recursive model is used (such as in the case of the DPS algorithm presented by Santini and Römer [95])

6.1.2 When using a dual prediction based approach is there a consistent relationship between the number of transmissions required and the error budget for a given data set?

Yes.

The relationship between the user specified error threshold and number of transmissions required to reconstruct a data set has been explored, and a method to estimate the number of transmissions required to reconstruct the data stream at a given error threshold proposed.

6.1.3 Can combining multiple sensors readings into a single predictive model allow a greater reduction in the number of packets transmitted, than when compressing each stream individually?

Yes.

Rather than have a separate predictive model for each sensor, SIP allows multiple sensors readings to be combined into a single state vector. This has been demonstrated to further reduce the number of model updates required compared to processing each sensor stream individually.

6.1.4 Can a prediction mechanism that exploits previously learnt temporal correlations between nodes be used to continue to estimate sensor readings where no value is currently available?

Yes.

As shown in Chapter 5, GPR can be used to impute missing values and this has been demonstrated for a production WDS monitoring application. The key contributions in this thesis that enable this are the development of a windowing mechanism to deal with out of order data streams and a feedback approach that ensures the training window is continuously supplied with data.

6.1.5 Can this prediction mechanism be used to used to estimate sensor readings where there is no sensor deployed, assuming a short period of training data?

Yes.

The use of the feedback in the windowing mechanism has been shown to allow long term prediction of sensor values where there is no node permanently deployed. This approach has been validated during a 36 day in-situ deployment.

6.2 Future Work

In the case of the SIP several strands of future investigation will be pursued:

1. Further investigation is needed into how signal characteristics and sensing frequency affect the amount of data transmitted.

2. The energy performance gain from SIP when deployed on mote hardware will be characterised.

3. It may be possible improve the accuracy of the *reconstructed* signal by using a more sophisticated approach than piece-wise linear approximation. Preliminary results suggest that using a spline-based method to reconstruct the readings improves the accuracy of the reconstructed signal.

4. The use of summary statistics rather than sensor values to derive the state [28] may further improve the transmission suppression performance of SIP. While this approach can result in a better

compression performance, it is unclear if the abstraction of sensor readings into summary statistics may impact the user's understanding of the evolution of the data, and thus the appropriateness of this method as a research tool.

5. While prior DPS algorithms (for example, Jain, Chang and Wang [50], Santini and Römer [95] or Olston, Jiang and Widom [73]) have focused on the use of predictive models to represent time series data, it may be possible that a similar concept could be applied to other domains (such as the frequency domain, or summary statistics).

6. A more comprehensive evaluation of SIP on real world hardware would help evaluate the impact of lower level concerns (such as the MAC, low-power listening, and multi-hop networks) on the performance of the algorithm. For multi-hop networks it is expected that some of the benefit from SIP will be lost in the overhead of low power listening, since the radio is being used to listen for forwarding traffic. Routing approaches that use a *backbone* approach [94] may be more appropriate in this situation.

7. Much of the prior work on DPS algorithms has been performed in simulation on the desktop. Where hardware is used it is difficult to make a direct comparison, due to differing hardware platforms being used. To fully evaluate the energy reduction that SIP offers, it would be beneficial to compare the performance on node hardware against implementations of similar algorithms.

With regard to the Virtual Sensors concept:

1. While the virtual sensors algorithm has been shown to work well in a WDS environment, the approach should be applicable to other environments where there is a reasonable correlation between sensor readings (for example, the environment within buildings). Evaluating the algorithm in such environments would validate this assumption.

2. The windowing mechanism used to provide training data to the GPR was designed in order to allow long term prediction and incorporate out of sequence sensor data into the model. However, out of sequence data may not be present in every situation. An investigation into the use of the feedback mechanism with the efficient update implementation put forward by Osborne et al. [77], may yield significant performance improvements and still allowing long term predictions to be made.

3. One drawback of the virtual sensors approach is the need to maintain a *window* of data that fully describes the underlying data stream. Since SIP is able to provide a compressed version of this data, it may be possible that using the output of the SIP as input to virtual sensors may also improve the performance of the algorithm.

6.3 Summary

In conclusion, this thesis has demonstrated that model based transmission reduction approaches can significantly reduce the number of packets required to communicate a data stream to the sink. For temperature data with a 0.5 ℃ error threshold, approximately 1% of the samples gathered at 30 second intervals are required to be transmitted. The approach generalises well and has been evaluated with a variety of other sensing modalities such as humidity and light data.

A GPR based algorithm has been developed that allows on-line imputation and prediction of missing sensor values. The approach has been implemented and deployed on a WDS monitoring system. The virtual sensors concept extends this approach to allow prediction of a signal where no sensor is permanently deployed based on a short window of training data and the input from correlated sensors.

References

[1] T. F. Abdelzaher, M. Martonosi and A. Wolisz, eds. *Proceedings of the 6th International Conference on Embedded Networked Sensor Systems, SenSys 2008, Raleigh, NC, USA, November 5-7, 2008.* ACM, 2008.

[2] M. Abramowitz and I. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover publications, 1964.

[3] J. Al-Karaki and A. Kamal. 'Routing techniques in wireless sensor networks: a survey'. In: *Wireless Communications, IEEE* 11.6 (2004), pp. 6–28.

[4] C. Alippi, G. Anastasi, C. Galperti, F. Mancini and M. Roveri. 'Adaptive Sampling for Energy Conservation in Wireless Sensor Networks for Snow Monitoring Applications'. In: *MASS*. IEEE, 2007, pp. 1–6.

[5] M. Allen, L. Girod, R. Newton, S. Madden, D. T. Blumstein and D. Estrin. 'VoxNet: An Interactive, Rapidly-Deployable Acoustic Monitoring Platform'. In: *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on.* 2008, pp. 371–382.

[6] Anonymous. *Basic generation services data room.* Available online [accessed 19/04/2013]. URL: `http://www.bgs-auction.com/bgs.dataroom.home.asp`.

[7] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora and M. Miyashita. 'A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking'. In: *Computer Networks* 46 (2004), pp. 605–634.

[8] W. U. Z. Bajwa, J. Haupt, A. M. Sayeed and R. D. Nowak. 'Compressive wireless sensing'. In: *IPSN*. Ed. by J. A. Stankovic, P. B. Gibbons, S. B. Wicker and J. A. Paradiso. ACM, 2006, pp. 134–142.

[9] B. L. Bowerman, R. O'Connell and A. Koehler. *Forecasting, Time Series, and Regression.* South-Western College, 2004.

[10] B. L. Bowerman, T. Richard and A. Koehler. *Forecasting, time series, and regression: an applied approach.* Thomson Brooks/Cole, Belmont, Calif., 2005.

[11] J. Brusey, E. Gaura, D. Goldsmith and J. Shuttleworth. 'FieldMAP: A Spatiotemporal Field Monitoring Application Prototyping Framework'. In: *Sensors Journal, IEEE* 9.11 (Sept. 2009), pp. 1378 –1390.

[12] J. Brusey, E. Gaura and R. Hazelden. 'A Pattern-Based Framework for Developing Wireless Monitoring Applications'. In: *New Developments in Sensing Technology for Structural Health Monitoring*. Ed. by S. C. Mukhopadhyay. Vol. 96. Lecture Notes in Electrical Engineering. Springer Berlin Heidelberg, 2011, pp. 75–91.

[13] J. Brusey, E. I. Gaura and R. Hazelden. 'WSN Deployments: Designing with Patterns'. In: *Proc. 10th IEEE Sensors Conf.* IEEE Press, Oct. 2011, pp. 71–76.

[14] E. Candes and M. Wakin. 'An Introduction to Compressive Sampling'. In: *IEEE Signal Processing Magazine* 25.2 (2008), pp. 21–30.

[15] E. Capo-Chichi, H. Guyennet and J.-M. Friedt. 'K-RLE: A New Data Compression Algorithm for Wireless Sensor Network'. In: *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on.* 2009, pp. 502–507.

[16] N. Chapados and Y. Bengio. 'Augmented Functional Time Series Representation and Forecasting with Gaussian Processes'. In: *NIPS*. Ed. by J. C. Platt, D. Koller, Y. Singer and S. T. Roweis. Curran Associates, Inc., 2007.

[17] K. Chebrolu, B. Raman, N. Mishra, P. K. Valiveti and R. Kumar. 'Brimon: a sensor network system for railway bridge monitoring'. In: *MobiSys*. Ed. by D. Grunwald, R. Han, E. de Lara and C. S. Ellis. ACM, 2008, pp. 2–14.

[18] W. Chen, M. R. D. Rodrigues and I. J. Wassell. 'Distributed Compressive Sensing Reconstruction via Common Support Discovery'. In: *ICC*. IEEE, 2011, pp. 1–5.

[19] W. Chen and I. J. Wassell. 'Energy-efficient signal acquisition in wireless sensor networks: a compressive sensing framework'. In: *IET Wireless Sensor Systems* 2.1 (2012), pp. 1–8.

[20] D. Chu, A. Deshpande, J. M. Hellerstein and W. Hong. 'Approximate Data Collection in Sensor Networks using Probabilistic Models'. In: *ICDE*. Ed. by L. Liu, A. Reuter, K.-Y. Whang and J. Zhang. IEEE Computer Society, 2006, p. 48.

[21] A. Corberàn-Vallet, J. D. Bermùdez and E. Vercher. 'Forecasting correlated time series with exponential smoothing models'. In: *International Journal of Forecasting* 27.2 (2011), pp. 252 –265. URL: http://www.sciencedirect.com/science/article/pii/S0169207010001172.

[22] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia and D. Moore. 'Environmental Wireless Sensor Networks'. In: *Proceedings of the IEEE* 98.11 (2010), pp. 1903–1917.

[23] T. Daniel, E. I. Gaura and J. Brusey. 'Wireless Sensor Networks to Enable the Passive House - Deployment Experiences'. In: *EuroSSC*. Ed. by P. M. Barnaghi, K. Moessner, M. Presser and S. Meissner. Vol. 5741. Lecture Notes in Computer Science. Springer, 2009, pp. 177–192.

[24] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein and W. Hong. 'Model-based approximate querying in sensor networks'. In: *VLDB J.* 14.4 (2005), pp. 417–443.

[25] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein and W. Hong. 'Model-Driven Data Acquisition in Sensor Networks'. In: *VLDB*. Ed. by M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley and K. B. Schiefer. Morgan Kaufmann, 2004, pp. 588–599.

[26] E. Fasolo, M. Rossi, J. Widmer and M. Zorzi. 'In-network aggregation techniques for wireless sensor networks: a survey'. In: *Wireless Communications, IEEE* 14.2 (April), pp. 70–87.

[27] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design patterns: Elements of reusable object-oriented design.* 1995.

[28] E. Gaura, J. Brusey and R. Wilkins. 'Bare necessities — Knowledge-driven WSN design'. In: *Sensors, 2011 IEEE.* 2011, pp. 66–70.

[29] E. Gaura, J. Brusey, J. Kemp and C. Thake. 'Increasing safety of bomb disposal missions: A body sensor network approach'. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 39.6 (2009), pp. 621–636.

[30] E. Gaura, L. Girod, J. Brusey, M. Allen and G. Challen. *Wireless Sensor Networks: Deployments and Design Frameworks.* 1st. Springer Publishing Company, Incorporated, 2010.

[31] E. I. Gaura, J. Brusey, M. Allen, R. Wilkins, D. Goldsmith and R. Rednic. 'Edge mining the Internet of Things'. In: *IEEE Sensors* (2013). (submitted).

[32] H. Gilgen. *Univariate time series in geosciences: theory and examples.* Springer Verlag, 2006.

[33] A. Girard, C. E. Rasmussen, J. Q. Candela and R. Murray-Smith. 'Gaussian Process Priors with Uncertain Inputs - Application to Multiple-Step Ahead Time Series Forecasting'. In: *NIPS.* Ed. by S. Becker, S. Thrun and K. Obermayer. MIT Press, 2002, pp. 529–536.

[34] D. Goldsmith and J. Brusey. 'The Spanish Inquisition Protocol — Model based transmission reduction for wireless sensor networks'. In: *Sensors, 2010 IEEE.* Nov. Pp. 2043–2048.

[35] D. Goldsmith, E. Gaura, J. Brusey, J. Shuttleworth, R. Hazelden and M Langley. 'Wireless Sensor Networks for Aerospace Application - Thermal Monitoring for a Gas Turbine Engine'. In: *Proceedings of 2009 NSTI Nanotechnology Conference.* Vol. 1. NSTI Nanotech. May 2009, pp. 507–512.

[36] D. Goldsmith, A. Preis, M. Allen and A. J. Whittle. 'Virtual Sensors to Improve On-line Hydraulic Model Calibration'. In: *Proceedings of the 12th annual Water Distribution Systems Analysis (WSDA) conference.* Aug. 2010.

[37] S. Haykin. *Adaptive filter theory.* Prentice hall, 2002.

[38] S. Haykin. *Introduction to adaptive filters.* English. Macmillan ; Collier Macmillan, New York : London : 1984, xii, 217 p. :

[39] W. Heinzelman, A. Chandrakasan and H. Balakrishnan. 'An application-specific protocol architecture for wireless microsensor networks'. In: *IEEE Transactions on Wireless Communications* 1.4 (2002).

[40] J. Hill. 'System Architecure for Wireless Sensor Networks'. PhD thesis. Univerity of California, Berkely, 2003.

[41] D. Hughes, P. Greenwood, G. Blair, G. Coulson, F. Pappenberger, P. Smith and K. Beven. 'An Intelligent and Adaptable Grid-Based Flood Monitoring and Warning System'. In: *Proceedings of the UK eScience All Hands Meeting* (2006).

[42] D. Hughes, P. Greenwood, G. Coulson and G. S. Blair. 'GridStix: Supporting Flood Prediction using Embedded Hardware and Next Generation Grid Middleware'. In: *WOWMOM.* IEEE Computer Society, 2006, pp. 621–626.

[43] E. Inc. *B-530 Datasheet.*

[44] T. Instruments. *MSP430F1161 Datasheet.* URL: `http://www.ti.com/lit/gpn/msp430f1610` (visited on 01/12/2012).

[45] C. Intanagonwiwat, D. Estrin, R. Govindan and J. Heidemann. 'Impact of network density on data aggregation in wireless sensor networks'. In: *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on,* pp. 457–458.

[46] C. Intanagonwiwat, R. Govindan and D. Estrin. 'Directed diffusion: A scalable and robust communication paradigm for sensor networks'. In: *Mobile Computing and Networking.* 2000, pp. 56–67.

[47] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva. 'Directed diffusion for wireless sensor networking'. In: *IEEE/ACM Trans. Netw.* 11.1 (Feb. 2003), pp. 2–16.

[48] T. Intruments. *CC2420 Datasheet.* Mar. 2013. URL: `http://www.ti.com/lit/gpn/cc2420` (visited on 03/09/2013).

[49] A. Jain and E. Y. Chang. 'Adaptive sampling for sensor networks'. In: *DMSN.* Ed. by A. Labrinidis and S. Madden. Vol. 72. ACM International Conference Proceeding Series. ACM, 2004, pp. 10–16.

[50] A. Jain, E. Y. Chang and Y.-F. Wang. 'Adaptive Stream Resource Management Using Kalman Filters'. In: *SIGMOD Conference*. Ed. by G. Weikum, A. C. König and S. Deßloch. ACM, 2004, pp. 11–22.

[51] A. Journel and C. J. Huijbregts. *Mining Geostatistics*. Academic Press, Feb. 1981.

[52] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh and D. Rubenstein. 'Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet'. In: *SIGOPS Oper. Syst. Rev.* 36.5 (2002), pp. 96–107.

[53] K. Kapitanova, S. H. Son and K.-D. Kang. 'Event Detection in Wireless Sensor Networks - Can Fuzzy Values Be Accurate?' In: *ADHOCNETS*. Ed. by J. Zheng, D. Simplot-Ryl and V. C. M. Leung. Vol. 49. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2010, pp. 168–184.

[54] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman and M. D. Yarvis. 'Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea'. In: *SenSys*. Ed. by J. Redi, H. Balakrishnan and F. Zhao. ACM, 2005, pp. 64–75.

[55] Lawrence Berkeley National Laboratory. *The Internet Traffic Archive*. http://www.ita.ee.lbl.gov/. Mar. 2000. URL: `http://www.ita.ee.lbl.gov/`.

[56] I. Lazaridis and S. Mehrotra. 'Capturing Sensor-Generated Time Series with Quality Guarantees'. In: *ICDE*. Ed. by U. Dayal, K. Ramamritham and T. M. Vijayaraman. IEEE Computer Society, 2003, pp. 429–.

[57] Y.-A. Le Borgne, S. Santini and G. Bontempi. 'Adaptive model selection for time series prediction in wireless sensor networks'. In: *Signal Process.* 87 (12 Dec. 2007), pp. 3010–3020.

[58] Y. Li and L. Parker. 'A spatial-temporal imputation technique for classification with missing data in a wireless sensor network'. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. Sept. Pp. 3272–3279.

[59] Y. Li and L. E. Parker. 'Nearest neighbor imputation using spatial-temporal correlations in wireless sensor networks'. In: *Information Fusion* 0 (2012), pp. –. URL: `http://www.sciencedirect.com/science/article/pii/S1566253512000711`.

[60] C. Liu, K. Wu and M. Tsao. 'Energy efficient information collection with the ARIMA model in wireless sensor networks'. In: *GLOBECOM*. IEEE, 2005, p. 5.

[61] K. Lorincz, B. rong Chen, J. Waterman, G. Werner-Allen and M. Welsh. 'Resource aware programming in the Pixie OS'. In: *SenSys*. Ed. by T. F. Abdelzaher, M. Martonosi and A. Wolisz. ACM, 2008, pp. 211–224.

[62] A. LTD. *Atmega 328 datasheet*. URL: `http://www.atmel.com/Images/doc8271.pdf` (visited on 01/12/2012).

[63] S. Madden. *Intel Lab Data*. http://db.lcs.mit.edu/labdata/labdata.html. July 2010.

[64] S. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong. 'TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks'. In: *OSDI*. Ed. by D. E. Culler and P. Druschel. USENIX Association, 2002.

[65] S. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong. 'TinyDB: an acquisitional query processing system for sensor networks'. In: *ACM Trans. Database Syst.* 30.1 (2005), pp. 122–173.

[66] A. M. Mainwaring, D. E. Culler, J. Polastre, R. Szewczyk and J. Anderson. 'Wireless sensor networks for habitat monitoring'. In: *WSNA*. Ed. by C. S. Raghavendra and K. M. Sivalingam. ACM, 2002, pp. 88–97.

[67] D. Malan, T. Fulford-Jones, M. Welsh and S. Moulton. 'CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care'. In: *International Workshop on Wearable and Implantable Body Sensor Networks*. Apr. 2004.

[68] K. Martinez, R. Ong and J. Hart. 'Glacsweb: a sensor network for hostile environments'. In: *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*. IEEE. 2004, pp. 81–87.

[69] M. McPhaden. *Tropical atmosphere ocean project, pacific marine environmental laboratory*. July 2010. URL: `http://www.pmel.noaa.gov/tao/`.

[70] Monty Python. *The Spanish Inquisition Sketch*. Sept. 1970. URL: `transcriptavailableathttp://people.csail.mit.edu/paulfitz/spanish/script.html`.

[71] National Data Bouy Center. *TAO Data Delivery portal*. July 2012. URL: `http://tao.noaa.gov/tao/data_deliv/deliv_ndbc.shtml`.

[72] H. A. Nguyen, A. Forster, D. Puccinelli and S. Giordano. 'Sensor node lifetime: An experimental study'. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. March, pp. 202–207.

[73] C. Olston, J. Jiang and J. Widom. 'Adaptive Filters for Continuous Queries over Distributed Data Streams'. In: *SIGMOD Conference*. Ed. by A. Y. Halevy, Z. G. Ives and A. Doan. ACM, 2003, pp. 563–574.

[74] C. Olston, B. T. Loo and J. Widom. 'Adaptive Precision Setting for Cached Approximate Values'. In: *SIGMOD Conference*. Ed. by S. Mehrotra and T. K. Sellis. ACM, 2001, pp. 355–366.

[75] M. Osborne, S. Roberts, A. Rogers, S. Ramchurn and N. Jennings. 'Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes'. In: *Proceedings of the 7th international conference on Information processing in sensor networks*. IEEE Computer Society. 2008, pp. 109–120.

[76] M. Osborne and S. J. Roberts. *Gaussian processes for prediction*. Tech. rep. PARG-07-01. URL http://www.robots.ox.ac.uk/ parg/pubs/PARG-07-01.pdf. Pattern Analysis and Machine Learning Group, Department of Engineering Science, University of Oxford, 2007.

[77] M. A. Osborne, S. J. Roberts, A. Rogers and N. R. Jennings. 'Real-time information processing of environmental sensor network data using bayesian gaussian processes'. In: *TOSN* 9.1 (2012), p. 1.

[78] P. Padhy, R. K. Dash, K. Martinez and N. R. Jennings. 'A utility-based sensing and communication model for a glacial sensor network'. In: *AAMAS*. Ed. by H. Nakashima, M. P. Wellman, G. Weiss and P. Stone. ACM, 2006, pp. 1353–1360.

[79] T. Palpanas, M. Vlachos, E. J. Keogh, D. Gunopulos and W. Truppel. 'Online Amnesic Approximation of Streaming Time Series'. In: *ICDE*. IEEE Computer Society, 2004, pp. 338–349.

[80] L. Pan and J. Li. 'K-Nearest Neighbor Based Missing Data Estimation Algorithm in Wireless Sensor Networks'. In: (2010).

[81] V. Paxson and S. Floyd. 'Wide area traffic: the failure of Poisson modeling'. In: *IEEE/ACM Trans. Netw.* 3.3 (1995), pp. 226–244.

[82] C. Pereira, S. Gupta, K. Niyogi, I. Lazaridis, S. Mehrotra and R. Gupta. *Energy Efficent Communication for Reliablity and Quality Aware Sensor Networks*. Tech. rep. School of Information & Computer Science University of California Irvine, 2003.

[83] J. Polastre, J. L. Hill and D. E. Culler. 'Versatile low power media access for wireless sensor networks'. In: *SenSys*. Ed. by J. A. Stankovic, A. Arora and R. Govindan. ACM, 2004, pp. 95–107.

[84] J. Polastre, R. Szewczyk and D. Culler. ‘Telos: enabling ultra-low power wireless research’. In: *Proceedings of the 4th international symposium on Information processing in sensor networks.* (Los Angeles, California). IPSN ’05. Piscataway, NJ, USA: IEEE Press, 2005.

[85] G. J. Pottie and W. J. Kaiser. ‘Wireless Integrated Network Sensors’. In: *Commun. ACM* 43.5 (2000), pp. 51–58.

[86] A. Preis, A. J. Whittle and A. Ostfield. ‘On-line hydraulic state prediction for water distribution systems’. In: *WDSA ’09: Proceedings of the 11th Water Distribution Systems Analysis Symposium.* Kansas City, MO, USA, May 2009.

[87] V. Raghunathan, C. Schurgers, S. Park and M. Srivastava. ‘Energy-aware wireless microsensor networks’. In: *Signal Processing Magazine, IEEE* 19.2 (2002), pp. 40–50.

[88] C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning.* Cambridge MA 02142-1493 USA: The MIT Press, 2007.

[89] J. Redi, H. Balakrishnan and F. Zhao, eds. *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys 2005, San Diego, California, USA, November 2-4, 2005.* ACM, 2005.

[90] S. Research. *Frequently asked questions (what is the power consumption of the Shimmer?)”* URL: `http://www.shimmer-research.com/links/faqsH1`.

[91] J. L. Rodgers and W. A. Nicewander. ‘Thirteen ways to look at the correlation coefficient’. In: *The American Statistician* 42.1 (1988), pp. 59–66.

[92] K. Römer. ‘Programming paradigms and middleware for sensor networks’. In: *GI/ITG Workshop on Sensor Networks.* 2004, pp. 49–54.

[93] K. Ròmer, O. Kasten and F. Mattern. ‘Middleware challenges for wireless sensor networks’. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 6.4 (Oct. 2002), pp. 59–61.

[94] P. Santi and J. Simon. ‘Silence Is Golden with High Probability: Maintaining a Connected Backbone in Wireless Sensor Networks’. In: *EWSN.* Ed. by H. Karl, A. Willig and A. Wolisz. Vol. 2920. Lecture Notes in Computer Science. Springer, 2004, pp. 106–121.

[95] S. Santini and K. Römer. ‘An adaptive strategy for quality-based data reduction in wireless sensor networks’. In: *Proceedings of the 3rd International InProceedings on Networked Sensing Systems (INSS 2006).* Citeseer. 2006, pp. 29–36.

[96] F. Shang, J. G. Uber, B. G. van Bloemen Waanders, D. Boccelli and R. Janke. ‘Real Time Water Demand Estimation in Water Distribution System’. In: *WDSA ’06: Proceedings of the 8th Water Distribution Systems Analysis Symposium.* Cincinnati, OH, USA, Aug. 2006.

[97] G. Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai and K. Frampton. ‘Sensor network-based countersniper system’. In: *Proceedings of the 2nd international conference on Embedded networked sensor systems.* SenSys ’04. Baltimore, MD, USA: ACM, 2004, pp. 1–12.

[98] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji and A. Lendasse. ‘Methodology for long-term prediction of time series’. In: *Neurocomputing* 70.16-18 (2007), pp. 2861–2869.

[99] G. Tolle, J. Polastre, R. Szewczyk, D. E. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay and W. Hong. ‘A macroscope in the redwoods’. In: *SenSys.* Ed. by J. Redi, H. Balakrishnan and F. Zhao. ACM, 2005, pp. 51–63.

[100] D. Tulone and S. Madden. ‘An energy-efficient querying framework in sensor networks for detecting node similarities’. In: *MSWiM.* Ed. by E. Alba, C.-F. Chiasserini, N. B. Abu-Ghazaleh and R. L. Cigno. ACM, 2006, pp. 191–300.

[101] D. Tulone and S. Madden. 'PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks'. In: *EWSN*. Ed. by K. Römer, H. Karl and F. Mattern. Vol. 3868. Lecture Notes in Computer Science. Springer, 2006, pp. 21–37.

[102] V. Tuulos, J. Scheible and H. Nyholm. 'Combining Web, Mobile Phones and Public Displays in Large-Scale: Manhattan Story Mashup'. In: *Pervasive Computing* (2007), pp. 37–54.

[103] L. Wang and A. Deshpande. 'Predictive Modeling-Based Data Collection in Wireless Sensor Networks'. In: *EWSN*. Ed. by R. Verdone. Vol. 4913. Lecture Notes in Computer Science. Springer, 2008, pp. 34–51.

[104] G. Welch and G. Bishop. *An introduction to the Kalman filter*. Tech. rep. University of North Carolina at Chapel Hill, 1995.

[105] G. Werner-Allen, S. Dawson-Haggerty and M. Welsh. 'Lance: optimizing high-resolution signal collection in wireless sensor networks'. In: *SenSys*. Ed. by T. F. Abdelzaher, M. Martonosi and A. Wolisz. ACM, 2008, pp. 169–182.

[106] A. Whittle, L. Girod, A. Preis, M. Allen, H. Lim, M. Iqbal, C. Srirangarajan S. Fu, K. J Wong and D. Goldsmith. 'WaterWiSe@SG: a Testbed for Continuous monitoring of the Water Distribution System in Singapore'. In: *Proceedings of the 12th annual Water Distribution Systems Analysis (WSDA) conference*. Aug. 2010.

[107] B. Widrow, J. M. McCool, M. G. Larimore and C. Johnson Jr. 'Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter.' In: *Proceedings of the IEEE* 64.8 (1976), pp. 1151–1162.

[108] C. K. Williams. 'Prediction with Gaussian processes: From linear regression to linear prediction and beyond'. In: *Learning and inference in graphical models*. Ed. by M. I. Jordan. Cambridge, MA, 02142-1493, USA: The MIT Press, 1998, pp. 599–621.

[109] A. L. Wolf, C Fl, D. Perry, D. E. Perry and E. L. Wolf. 'Foundations for the Study of Software Architecture'. In: *ACM SIGSOFT Software Engineering Notes* 17 (1992), pp. 40–52.

[110] W. Yan, H. Qiu and Y. Xue. 'Gaussian process for long-term time-series forecasting'. In: *IJCNN*. IEEE, 2009, pp. 3420–3427.

[111] Y. Yao and J. Gehrke. 'The Cougar Approach to In-Network Query Processing in Sensor Networks'. In: *SIGMOD Record* 31.3 (2002), pp. 9–18.