# Prediction-Based Strategies for Reducing Data Transmissions in the IoT

Gabriel Martins Dias

DOCTORAL THESIS UPF / 2016

THESIS SUPERVISORS

Dr. Boris Bellalta
Dr. Simon Oechsner

Departament of Information and Communication Technologies

Universitat Pompeu Fabra Barcelona

This thesis is dedicated to my parents and to Carol.

*Dedico esta tese aos meus pais e à Carol.*

# Acknowledgements

First and foremost, I want to thank my parents, Maria Lucia Martins Dias and Orlando Antonio Dias, for supporting my dreams and ambitions over the years. They have never opposed my desire to learn and explore the world, even though they knew that we would end up in hours, days, weeks and months without meeting each other. More than their kind and wise words that helped me to relief when facing the hardest situations, they gave me examples of how to not give up on my goals.

*Em primeiro lugar, quero agradecer a meus pais, Maria Lucia Martins Dias e Orlando Antonio Dias, por apoiar meus sonhos e ambições ao longo dos anos. Eles nunca se opuseram à minha vontade de aprender e explorar o mundo, mesmo sabendo que passaríamos horas, dias, semanas e meses sem nos encontrar. Mais do que suas palavras gentis e sábias que me ajudaram a ficar bem ao enfrentar as situações mais difíceis, eles me deram exemplos de como não desistir dos meus objetivos.*

Next, I would like to express my sincere gratitude to my girlfriend, Carolina Tuzia. She always gave me support and encouragement to follow my dreams and helped me during the difficult moments in the last years. Thanks to her love, patience, encouragement, and support, I could go ahead and finish this work. Also, I thank her parents, José Orlando and Pasqualina, for welcoming me into their family and supporting us all these years.

I would like to thank my supervisors Dr. Boris Bellalta and Dr. Simon Oechsner. Boris has dedicated many hours to discuss ideas, develop

proper critics to the future of this thesis and, more importantly, to propose directions for this work, which I would not be able to do by myself. Simon has questioned my preliminary findings several times, showing the importance of facing with skepticism results that would support my initial (biased) intuitions. Without his questionings, the scientific contribution of this work would be severely weakened.

I would like to express my gratitude to some of my colleagues, with whom I shared great moments in the University all these years: Lorena Recalde, David Nettleton, Sergio Barrachina, Miquel Oliver, Luis Sanabria, Albert Domingo, Manuel Palacin, Anna Sfairopolou and Azadeh Faridi. Some of them were also personally meaningful during my adaptation in Barcelona. Therefore, I would like them not only for sharing of knowledge but also for the great moments that we spent together outside the University: Ruizhi Liao, Trang Minh, Albert Bel, Anna Carreras, Veronica Moreno, Toni Adame, Maddalena Nurchis, and Alex Bikfalvi.

To all my bachelor students during these years, I would like to express my gratitude. Especially those that I supervised in their Bachelor's thesis: José Antonio Pérez, Kevin Enrique Johnson, and Daniel Vivas. They gave me the opportunity to learn many things and look for relevant answers that I might have never asked myself without their help.

In the last months of my PhD, I visited the University of São Paulo. I want to thank Prof. Dr. Cintia Margi for hosting me and for connecting me with her students. In particular, I want to thank Filipe Oliveira, Bruno Oliveira, and Henrique Silva for all their support during my stay there.

I would like to thank the European Union, the Spanish Government, the Catalan Government and Universitat Pompeu Fabra for the financial supports respectively under the Seventh Framework Programme (FP7-SME-2013-605073-ENTOMATIC), the CISNETs project (TEC2012-32354), the SGR project (SGR-2014-1173), and the ETIC PhD program.

I also would like to thank my brother, Regis, my friend Dheinny, and my cousin Caren for their encouragement and support during my PhD.

Thank you all my extended family and anybody else that I forgot to mention but might have helped me directly or indirectly during this work.

# Abstract

Predictions about the evolution of the Internet of Things (IoT) in the next years are optimistic. The number of interconnected devices will continue to grow exponentially, as well as the amount of data that they report.

Part of this data will be generated by wireless sensor nodes organized in Wireless Sensor Networks (WSNs) to transmit their measurements to Gateways (GWs). However, wireless sensor nodes are mainly designed to have low costs, which implies constrained memory and energy supplies, and does not permit the streaming of measured data at high data rates.

Meanwhile, modern uses of WSNs rely on the knowledge acquired by sensor nodes to trigger reactions in other systems, and sensed data has become critical to avoid economic–and living–losses. Therefore, it is important to optimize data transmissions in WSNs to support not only a higher number of wireless sensor nodes but also a higher diversity of sensed parameters.

Solutions for data aggregation and data compression have reduced the number of gross transmissions, but they did not solve the problem of transmitting measurements that do not convey knowledge to the WSNs' managers. These solutions do not exploit the fact that, fortunately, WSNs are asymmetric and, contrary to ordinary wireless sensor nodes, GWs have an Internet connection with no critical computational, power or communication limitations. Hence, GWs can run algorithms and process amounts of data that wireless sensor nodes do not support, which permits them to predict the data that will be measured.

This thesis extends a paradigm that exploits WSNs to the utmost: data that can be predicted does not have to be transmitted.

First, we design a self-managing WSN architecture that adopts a standardized communication to integrate WSNs into data analysis services in the cloud. To evaluate our idea in experiments, we implement the Data Analytics for Sensors Dashboard (DAS-Dashboard) to control and optimize, using specialized cloud services, a WSN via the Internet. Our experimental results show that the interconnection of remote components does not imply a significant overhead and that the architecture is feasible

in practice.

Then, relying on this architecture, we design a mechanism to adjust the sensor nodes' sampling intervals according to the changes observed in the environment. The novelty of this mechanism is in the use of a Reinforcement Learning (RL) technique called Q-Learning. Simulation and experimental results show that this mechanism provides necessary means to make a smart WSN with the capacity of self-optimizing.

As a result of hardware evolution, new wireless sensor nodes have extended memory and computing capabilities; and more sophisticated prediction algorithms were adopted in sensor nodes. In response to that, we analyze the benefits of incorporating the current state-of-the-art prediction algorithms in WSNs. The results are promising: our simulation results show that it is possible to eliminate WSN transmissions without reducing the quality of the measurements provided in several sensor network applications.

For the future generations of WSNs, we design a theoretical model for characterizing the number of transmissions in WSNs, which can provide reliable estimations about the efficiency of prediction-based data reduction methods. The new model will support the WSNs' growth regarding the number of sensor nodes in a single network and the quality of information processed by their GWs.

The prediction-based strategies investigated in this thesis can impact the present and the future of the IoT. Current WSNs can be optimized to avoid unnecessary transmissions with the help of the cloud. Also, coming generations of WSNs will be supported by our WSN transmission model to adopt prediction algorithms and maintain strict control over the quality of the reported data without being harmed by the adoption of a higher number of sensor nodes; hence, collaborating to the IoT's growth.

# Resumen

Las predicciones sobre la evolución del *Internet of Things* (IoT) en los próximos años son optimistas. El número de dispositivos interconectados continuará creciendo exponencialmente, así como la cantidad de datos generados.

Parte de estos datos serán generados por los nodos sensores inalámbricos organizados en Redes de Sensores Inalámbricas (del inglés *Wireless Sensor Networks*, o abreviado WSNs) para transmitir sus mediciones a sus correspondientes *Gateways* (GWs). Sin embargo, los sensores inalámbricos están diseñados principalmente para ser de bajo coste, lo que implica recursos de memoria y de energía finitos, y no permite la transmisión de los datos medidos a altas velocidades.

Mientras tanto, los usos actuales de las WSNs se basan en los conocimientos adquiridos por los nodos sensores para desencadenar reacciones en otros sistemas, de modo que estos datos se han convertido en fundamentales para evitar pérdidas económicas–y de vidas. Por lo tanto, es importante optimizar las transmisiones de datos en WSNs para soportar no sólo un mayor número de nodos sensores inalámbricos, sino también una mayor diversidad de parámetros detectados.

Las soluciones para la agregación y compresión de datos han reducido el número de transmisiones brutas, pero no han resuelto el problema de la transmisión de las mediciones que no llevan información útil a los administradores de las WSNs. Estas soluciones no explotan el hecho de que, afortunadamente, las WSNs son asimétricas y, contrariamente a los nodos sensores inalámbricos habituales, los GWs tienen una conexión a Internet sin limitaciones críticas en términos computacionales, energéticos o de comunicación. Por lo tanto, los GWs pueden ejecutar algoritmos y procesar grandes volúmenes de datos no ejecutables en los sensores inalámbricos, lo que les permite predecir los datos que se van a medir.

Esta tesis extiende un paradigma que explota las WSNs al máximo: los datos que pueden ser predichos no tienen que ser transmitidos.

En primer lugar, diseñamos una arquitectura de autogestión para WSNs que adopta una comunicación estandarizada para integrar las

WSNs con los servicios de análisis de datos en la nube. Para evaluar nuestra idea de forma experimental, se ha implementado el *Data Analytics for Sensors Dashboard* (DAS-Dashboard) para controlar y optimizar, mediante servicios especializados en la nube, una WSN a través de Internet. Nuestros resultados experimentales muestran que la interconexión de componentes remotos no implica una sobrecarga significativa y que la arquitectura resultante es factible en la práctica.

Basándonos en esta arquitectura, diseñamos un mecanismo para ajustar los intervalos de muestreo de los nodos sensores a partir de los cambios observados en el medio. La novedad de este mecanismo está en el uso de una técnica de *Reinforcement Learning* (RL) llamada Q-Learning. Los resultados de la simulación y los experimentos muestran que este mecanismo proporciona los medios necesarios para hacer una WSN inteligente con capacidad de auto-optimización.

Como resultado de la evolución hardware, nuevos nodos sensores inalámbricos han extendido las capacidades de memoria y de cómputo; como consecuencia, se ha adoptado algoritmos más sofisticados de predicción en los nodos sensores. En respuesta a ello, analizamos los beneficios de la incorporación de los algoritmos de predicción actuales del estado del arte en WSNs. Los resultados de nuestras simulaciones son prometedores: estos demuestran que en varias aplicaciones de redes de sensores es posible eliminar algunas transmisiones sin reducir la calidad de las medidas proporcionadas.

Para las futuras generaciones de WSNs, diseñamos un modelo teórico para caracterizar el número de transmisiones en WSNs, que pueden proporcionar estimaciones fiables acerca de la eficiencia de los métodos de reducción de datos basados en predicciones. El nuevo modelo permite el crecimiento de las WSNs en relación al número de nodos sensores en una sola red y la calidad de la información procesada por el GW.

Las estrategias basadas en la predicción de datos investigadas en esta tesis pueden tener un impacto en el presente y el futuro del IoT. Las WSNs actuales pueden ser optimizadas para evitar transmisiones innecesarias con la ayuda del *cloud*. Además, las nuevas generaciones de WSNs estarán respaldadas por nuestro modelo de transmisión para adoptar algorit-

mos de predicción y mantener un estricto control sobre la calidad de los datos notificados sin ser dañadas por la adopción de un mayor número de nodos sensores; por consiguiente, colaborando en el crecimiento del IoT.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**AI** Artificial Intelligence. 7, 16, 17, 71, 75, 100

**AIC** Akaike Information Criterion. 103, 108

**AICc** AIC with a correction for finite sample sizes. 103, 108

**ANN** Artificial Neural Network. 51, 52, 61, 81, 107, 109, 118

**API** Application Programming Interface. 23, 24

**AR** AutoRegressive. 51, 114, 116

**ARIMA** AutoRegressive Integrated Moving Average. 51, 53, 61, 63, 65, 67, 69, 70, 107, 109, 114, 116

**BIC** Bayes Information Criterion. 103, 108

**CH** Cluster Head. 30

**CTP** Collection Tree Protocol. 4

**DAS-Dashboard** Data Analytics for Sensors Dashboard. vii, 11, 13, 14, 16, 19, 23–26, 44, 99, 100

**DB** database. 12–14, 23, 24

**DISON** DIstributed Self-Organizing NEtwork management. 20, 21

**DPS** Dual Prediction Scheme. 8, 17, 19, 24, 46, 47, 49, 51, 59, 61–63, 69, 71, 75, 80–82, 85, 91, 97, 98, 103–105, 127

**ES** Exponential Smoothing. 51, 52, 61, 63, 65, 67, 70, 107, 109, 114, 116

**GPS** Global Positioning System. 53, 59

**GW** Gateway. vii–x, 1, 2, 4–7, 9, 12–14, 17, 19, 24, 25, 27–31, 47, 49–54, 58, 61, 63, 64, 71, 75–77, 79–86, 91, 92, 96, 98, 105, 122, 123, 129

**HTTP** Hypertext Transfer Protocol. 23

**IoT** Internet of Things. vii–x, 1, 3, 9, 10, 12, 26, 31, 70, 71, 99, 100

**LEACH** Low-Energy Adaptive Clustering Hierarchy. 4

**LMS** Least Mean Squares. 50

**MA** Moving Average. 114, 116

**MAC** Medium Access Control. 2, 5, 91, 97, 123

**MAE** Mean Absolute Error. 104

**MAPE** Mean Absolute Percentage Error. 104

**MCU** Microcontroller Unit. 3, 27, 75

**MSE** Mean Square Error. 104

**MVN** Multivariate Normal. 86, 87, 89, 92

**OS** Operating System. 12, 20, 97

**PCA** Principal Component Analysis. 30

**RelMAE** Relative Mean Absolute Error. 104

**RFID** radio-frequency identification. 58

**RL** Reinforcement Learning. viii, x, 8, 28, 31–33, 35, 36, 45

**RMSE** Root Mean Square Error. 104, 109

**SDN** Software Defined Networking. 21

**SM** Simple Mean. 61, 109, 112, 116

**sMAPE** symmetric Mean Absolute Percentage Error. 104

**SPS** Simple Prediction Scheme. 8, 17, 24, 26–29, 31, 45–47

**TCP** Transmission Control Protocol. 14

**WARM** WSN Application development and Resource Management. 21, 22, 25

**WLAN** Wireless Local Area Network. 121

**WSN** Wireless Sensor Network. vii–x, 1–14, 16, 17, 19–31, 33, 34, 36, 38, 42, 45–49, 51–53, 55, 71, 75–77, 80–84, 91, 96–101, 104, 105, 107, 109, 121, 122, 124, 127

# List of Publications

## Referred Journal and Magazine Papers

- G. M. Dias, B. Bellalta, and S. Oechsner, "A Survey About Prediction-Based Data Reduction in Wireless Sensor Networks," *submitted to ACM Computing Surveys*, 2016.
  Available at: `https://arxiv.org/abs/1607.03443`

- G. M. Dias, B. Bellalta, and S. Oechsner, "The Impact of Dual Prediction Schemes on the Reduction of the Number of Transmissions in Sensor Networks," submitted to *Computer Communications Journal*, 2016.
  Available at: `https://arxiv.org/abs/1509.08778`

- G. M. Dias, B. Bellalta, and S. Oechsner, "On the Importance and Feasibility of Forecasting Data in Sensors," submitted to *Transactions on Mobile Computing Journal*, 2016.
  Available at: `https://arxiv.org/abs/1604.01275`

- G. M. Dias, C. B. Margi, F. C. P. Oliveira, and B. Bellalta, "Cloud Empowered Self-Managing WSNs," submitted to *IEEE Communications Magazine*, 2016.
  Available at: `https://arxiv.org/abs/1607.03607`

# Referred Conference, Workshop and Summit Papers

- G. M. Dias, "Performance optimization of WSNs using external information," in *2013 IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Jun. 2013.
  Available at: `https://arxiv.org/abs/1607.03408`

- G. M. Dias, B. Bellalta, and S. Oechsner, "Towards Information-Centric WSN Simulations", in *1st OMNeT++ Community Summit 2014, Sep. 2014*.
  Available at: `https://arxiv.org/abs/1409.1001`

- G. M. Dias, S. Oechsner, and B. Bellalta, "A Centralized Mechanism to Make Predictions Based on Data from Multiple WSNs", in *International Workshop on Multiple Access Communications, 2015. Springer International Publishing, Aug. 2015*.
  Available at: `https://arxiv.org/abs/1407.0981`

- G. M. Dias, B. Bellalta, and S. Oechsner, "Predicting Occupancy Trends in Barcelona's Bicycle Service Stations Using Open Data," in *2015 SAI Intelligent Systems Conference (IntelliSys)*, Nov. 2015.
  Available at: `https://arxiv.org/abs/1505.03662`

- G. M. Dias, T. Adame, B. Bellalta, and S. Oechsner, "A Self-Managed Architecture for Sensor Networks Based on Real Time Data Analysis," to appear in *2016 IEEE Future Technologies Conference. IEEE, Dec. 2016*.
  Available at: `https://arxiv.org/abs/1605.09011`

- G. M. Dias, M. Nurchis, and B. Bellalta, "Adapting sampling interval of sensors using reinforcement learning," submitted to *2016 IEEE World Forum on Internet of Things. IEEE, Dec. 2016*.
  Available at: `https://arxiv.org/abs/1606.02193`

# Chapter 1

# INTRODUCTION

*Where any form of terrestrial life exists it is safe to assume there will be sensors deployed close by*[1].

According to Gartner [2], the number of interconnected devices in the Internet of Things (IoT) will triple in the next few years, reaching $13.5$ billion in $2020$. Cameras, accelerometers, thermometers, barometers, fire detectors and air quality monitors are only some examples of sensors that can be found in wireless sensor nodes used to monitor environments and track objects.

Indeed, wireless sensor nodes are taken into account in Gartner's estimates as they are also considered "Things" that form the IoT. However, in practice, wireless sensor nodes communicate only among themselves in Wireless Sensor Networks (WSNs) to report sensed data to Gateways (GWs). A typical GW consists in a wireless sensor node connected to a workstation (e.g., via USB) with Internet access; hence, usually, GWs can interact with remote systems that compute, analyze and extract valuable information from the collected datasets. On the other hand, ordinary wireless sensor nodes can neither process high complexity algorithms nor

[1] In allusion to "Where any form of terrestrial life exists it is safe to assume there will be spiders living close by." [1]

communicate directly with other Internet devices without the help of the GWs, due to their constrained computing power and hardware limitations.

Thanks to this limited communication between sensor nodes and the external world, the WSNs' growth can be described by the increasing number of wireless sensor nodes measuring and reporting data to GWs, and by the diversity of data types transmitted in WSNs. Modern applications of WSNs do not simply monitor changes in the environment anymore; they also trigger reactions to these changes. For example, in agriculture, sensed data can be used to apply pesticides after detecting that the number of insects exceeded a certain threshold [3]. Such a threshold, in turn, may vary according to the season or get affected by temperature and relative humidity changes during the days. In these applications, sensor nodes may need a high number of transmissions to communicate the number of insects, temperature, and relative humidity. If the WSN cannot handle all transmissions that sensor nodes make, it will collapse and end up in significant economic losses.

In other cases, losses can go beyond the ecological and economic ambits. For example, structural health monitoring for aircraft can include engine control systems that rely on enhanced data analysis to detect accidents, and report unexplained phenomena to people responsible for maintaining their safety and healthy conditions [4]. Extended data analysis require, in comparison to the simplest monitoring tasks, more parameters and a higher amount of informative data. Therefore, in these cases, more sensor nodes transmitting higher amounts of data might collapse the WSN and provoke accidents resulting in losses of lives.

These situations help us to understand that WSNs are data-oriented networks, i.e., the data that sensor nodes can produce is their most valuable asset [5]. Hence, as typical wireless sensor nodes have computing power constraints and are mainly supplied by capacity-constrained batteries, most of the works about WSNs focus on optimizing their energy consumption [6, 7]. In fact, efforts to reduce energy consumption in WSNs provide efficient solutions at different levels, such as Medium Access Control (MAC) protocols [8, 9], routing schemes [10, 11], and data compression [12, 13]. Fortunately, there are promising advances in en-

ergy supply methods for sensor nodes, including energy harvesting [14] and wireless power transfer [15]. These techniques facilitate the design of scalable methods to refill sensor nodes' batteries. Therefore, also considering the recent discoveries regarding batteries that could be charged hundreds of thousands of times without losing capacity [16], wireless sensor nodes' energy constraint may be–sooner or later–overcome.

Meanwhile, although energy limitations tend to disappear, the medium access has been named as one of the key challenges in the next generations of wireless networks, due to the increase in the number of wireless devices and traffic profiles [17]. In other words, the access to the data produced by neighboring sensor nodes might remain as an issue in the next generations of WSNs. Hence, if the number of sensor data transmissions follows the growth in the number of devices, the IoT will not be able to support all wireless communications. De facto, this situation raises a dilemma in WSNs because attempting to deliver more of their most valuable asset can, in fact, reduce their throughput. In conclusion, the spectrum scarcity is a candidate for becoming the greatest obstacle in the WSNs' growth thanks to the number of interconnected devices and data produced.

To workaround such a limitation, it would be necessary a deep improvement in the wireless sensor nodes' architecture, which would include new radio antennas, more powerful Microcontroller Units (MCUs), and extended memory capacities. On the other hand, the resulting costs of the new hardware might not be compatible with the mass adoption of sensor nodes.

Given that, an interesting question to ask, and which shall be addressed in this thesis, is:

**Could we build *smart* WSNs that optimize their transmissions?**

To answer this question, we must investigate other problems beforehand. First, what does a WSN need to become *smart* and *optimize itself*? Second, how can sensor *data transmissions* be *optimized*?

To become smart and optimize itself, a WSN needs enough knowledge about the environment, which can be produced by other systems (in-

cluding other WSNs). To obtain this information directly, sensor nodes might be programmed to access data from other devices external to their WSNs, such as other sensor nodes, cloud servers, and smartphones. This option, however, is not feasible in practice, because it would require a standard data interchange format for measurements, extra computing time in sensor nodes and occasionally new hardware to overhear the medium. Moreover, sensor nodes would need other sources of knowledge to verify the trust of their neighbors' data. In fact, all of these problems have already been addressed in Internet applications and are not compatible with WSNs' purposes.

An alternative solution to make a WSN optimize itself is to rely on GWs, exploiting their Internet connection and the fact that they do have neither critical computational, power nor communication limitations [18, 19]. GWs can access external information from several sources and provide direct access to WSNs' managers, who may intervene in the actions taken in the WSNs.

To optimize the number of transmissions in a WSN, it is important to observe the frequency of data generation and the data length. Using distributed mechanisms, a WSN can update its routing tables and reduce the overall number of transmissions, or adopt self-organization schemes to find clusters and avoid long-distance transmissions [20]. At this moment, there are several mechanisms to improve WSNs' transmissions and, even though there is no *one-size-fits-all* solution, many combinations have been tested and extended to work properly in several scenarios, such as the B-MAC [8], the Collection Tree Protocols (CTPs) [21], and the Low-Energy Adaptive Clustering Hierarchy (LEACH) [22] methods.

It is important to highlight, however, that the effort to find the best routing algorithm that can successfully coordinate the medium access is worthless if the data transmitted by the sensor nodes does not convey knowledge to the WSNs' managers. It is not only important to use sensor nodes to generate relevant knowledge, but it is also critical to maintain such a relevance during the WSNs' operation. The relevance of the knowledge acquired by sensor nodes depends mainly on the frequency that the data is generated and transmitted, and it can be wasted due to

long delays in the delivery process.

Data-driven mechanisms can rely on the foundation provided by combinations of protocols at the lower layers, such as MAC protocols and routing mechanisms. For instance, relying on lower layer solutions, existing data aggregation and data compression mechanisms can respectively reduce up to $40\%$ and $50\%$ the number of transmissions without affecting the data throughput [23, 24].

Nonetheless, in fact, data aggregation and data compression are examples of techniques that do not consider the most typical characteristics of WSNs: (i) differently from traditional (cabled and wireless) networks, WSNs are asymmetric, i.e., the computing power of sensor nodes is extremely smaller than that of workstations and servers used for data processing; and (ii) data collected and transmitted by sensor nodes may be predicted using basic algorithms. For instance, using such algorithms, GWs can rely on several sources of knowledge to estimate the data measured by sensor nodes. Meanwhile, sensor nodes can compute the same estimations, verify locally if the GW's predictions are accurate, and transmit their measurements only in cases when predictions fail. Therefore, adopting data prediction in WSNs can optimize transmissions in a way that neither data compression nor data aggregation can: eliminating the necessity of communicating the data.

Most of the WSNs predict sensed data, even though this is not explicitly addressed in their specification. It happens, for example, when a sensor measurement cannot be sampled on demand and the value informed to a WSN's manager is the last measurement transmitted by a sensor node. This is the same technique used by the *Constant* prediction method: the system simply assumes that there was no change in the environment after the last observation [25]. Sensor nodes, in turn, can exploit this behavior to avoid unnecessary transmissions by transmitting a measurement only if its value differs by more than a fixed tolerance threshold from the last value transmitted.

In this work, we concentrate our efforts to overcome the paradigm of using sensor networks to measure and transmit as much raw data as possible. Our goal is to optimize the data analysis while taking into account

the typical limitations of WSNs. Thus, we focus on using data predictions as a means of reducing the number of transmissions without affecting the quality of the measurements made by sensor nodes. To achieve our goal, we rely on the WSNs' asymmetric capacities: sensor nodes are closer to the data's sources, and GWs can communicate with cloud services that compute complex algorithms for data analysis.

## 1.1 Objectives

The primary objective of this work is to facilitate the optimization of sensor networks at the application layer. To achieve this goal, we will investigate scalable optimization methods that rely upon: (i) tools that facilitate sensor data acquisition, storage, and management; and (ii) statistical methods and models for data analysis.

These optimization methods will provide means for expanding WSNs both in the number of sensor nodes and applications, besides facilitating the inter-WSNs communication. To achieve that, we must provide a sufficiently solid basis on which future applications can rely to create more complex systems that interact with the environment and manage themselves. To achieve our primary goal, we also accomplish other steps during the research process:

1. **Investigate the state of the art and collect information about how data prediction has been incorporated into WSNs.** As a consequence, this investigation will show how the architecture schemes used in WSNs can better exploit the most recent methods used by data scientists for inferring and forecasting values.

2. **Evaluate in practice the benefits that cloud services give to WSNs, extending sensor nodes' computational power and incorporating WSNs into self-managing environments.** As cloud services may perform complex algorithms that are incompatible with ordinary wireless sensor nodes (due to their constrained computing capacity), their benefits may transcend currently known lev-

els of WSN optimization, for example, avoiding a high volume of unnecessary transmissions.

3. **Explore algorithms for predictions under the constraints observed in current wireless sensor nodes, and study their adoption in next generations of WSNs, considering their intrinsic hardware evolution.** This study will provide concrete evidence of predictions' effectiveness and their potential for the coming generations of sensor nodes and WSNs.

4. **Design a model for characterizing the number of transmissions in sensor networks, such that it will provide reliable estimations about the efficiency of prediction-based data reduction methods.** The new model will support the WSNs' growth regarding the number of sensor nodes in a single network and the quality of data analysis in their GWs.

## 1.2 Contributions

The main contribution of this thesis is the integration of mechanisms for fine-tuning WSNs with the most powerful Artificial Intelligence (AI) techniques. This connection can empower WSNs with cloud services, improve the quality of their data, and open space for new business opportunities. For the current WSNs, we focused on a self-managing architecture that supports existing frameworks and technologies used in sensor nodes. For future generations of WSNs, we deliver a theoretical analysis about the use of data prediction, considering sensor nodes' hardware evolution and sensor data quality.

We begin proposing a self-managing WSN architecture that integrates cloud computing, data analysis, and sensor networks. In the real implementation described in Chapter 2, we adopt a standardized communication for reporting sensed data and communicating remote decisions. Results show that the interconnection of remote components does not imply a significant overhead and is feasible in practice.

As existing wireless sensor nodes have strict memory and computing power constraints, our first contributions–for existing technologies–avoid extra computation in sensor nodes. Using an Reinforcement Learning (RL) technique called Q-Learning, we design a mechanism to adjust the sensor nodes' sampling intervals according to the changes observed in the environment. In Chapter 3, we describe our simulations and discuss the experimental results of this mechanism on the self-managing architecture proposed beforehand.

After observing the real benefits of a Simple Prediction Scheme (SPS), in Chapter 4, we analyze the accuracy that the current state-of-the-art algorithms for predictions have when applied to sensed data. We show that adopting sophisticated algorithms in sensor nodes makes Dual Prediction Schemes (DPSs) promising: using these algorithms in our simulations, we observe that it is possible to reduce the number of transmissions without reducing the quality of the measurements provided by WSNs in different sensor applications.

Foreseeing the sensor nodes' hardware evolution and new algorithms for predictions, we design a WSN transmission model in Chapter 5. It can be used to plan WSNs, considering their sensor nodes' hardware capabilities and configuration parameters, such as transmission ranges and sampling rates. Based on this model, we make a theoretical study to observe how the number of transmissions is affected by the predictions' accuracy and the correlation between data collected by neighbor sensor nodes. Our analysis shows that adopting a prediction-based and a data aggregation scheme can reduce the number of transmissions by up to nearly $92\%$.

# Chapter 2

# CLOUD EMPOWERED SELF-MANAGING WSNS

The IoT is composed of several smart devices with sensors; these devices collect, transmit and process environmental parameters, such as temperature, relative humidity and solar radiation. Finally, all this information is shared via the Internet with other "things".

In comparison with traditional WSNs, smart devices can be more powerful and perform high-complexity algorithms, interact with humans, provide machine-to-machine communication and also connect to cloud services to extend their computing power. Thus, thanks to extended communication and computing capabilities, household appliances, machines, personal devices and living beings can self-configure and self-manage resources in reaction to external phenomena that impact their operation.

Fortunately, rather than just forwarding data periodically reported by sensor nodes, GWs can become the connection point between WSNs and IoT environments; hence, WSNs can exploit cloud services for optimizing their operation [26]. Among several cloud services, data analysis can be ranked as the most relevant for WSNs, given the sensor nodes' dedication to measuring and reporting information about the external world. The ease of access to data analysis services is a twofold facility. First, because it provides information that cannot be computed in WSNs, due to the

constrained sensor nodes' computing power and limited access to external resources. Second, because analyzing data at WSN runtime facilitates the control over the quality of the data provided to WSNs' managers and the WSNs fine-tuning.

Indeed, fine-tuning a WSN requires appropriate management to detect and handle several problems at different levels. Also, each WSN has its requirements and particularities, such as maximum delay to deliver the data and tolerance about node failures and unreliable transmissions. For instance, if a sensor node's radio is experiencing a temporary interference, its transmissions will suffer occasional errors. Then, if the WSN tolerates packet losses, the routing table may be maintained to avoid the overhead communication that would provoke new transmissions and probably worsen the data delivery. Since the final decision affects the data collection, managing multiple WSNs that may co-exist in an IoT environment would require several configurations and parallel tasks.

Besides that, as data cannot be properly manipulated in WSNs, a proper data analysis manager must be able to execute several instances of analysis over different datasets at the same time. However, even simple linear regression algorithms can consume nearly 30 times more memory than the size of the dataset [27]. Therefore, if all the data computation is centralized in a single server, it could overload and collapse a server that must handle dozens (occasionally hundreds) of sensor nodes at the same time. Hence, a scalable solution that integrates WSNs in IoT environments must exploit specialized knowledge about the advantages and disadvantages in WSNs fine-tuning and data handling.

In this Chapter, we describe the design and the deployment of a scalable architecture that integrates solutions at different layers to incorporate WSNs in IoT environments and exploit cloud services to work autonomously. As part of this architecture, we describe the design and implementation of two other components: (i) a dashboard that provides means for collecting, storing and publishing data transmitted by wireless sensor nodes; and (ii) a data analytics server that supports several types of data analysis algorithms and handles data collected by WSNs. Finally, we integrate an existing solution for WSN application and resource man-

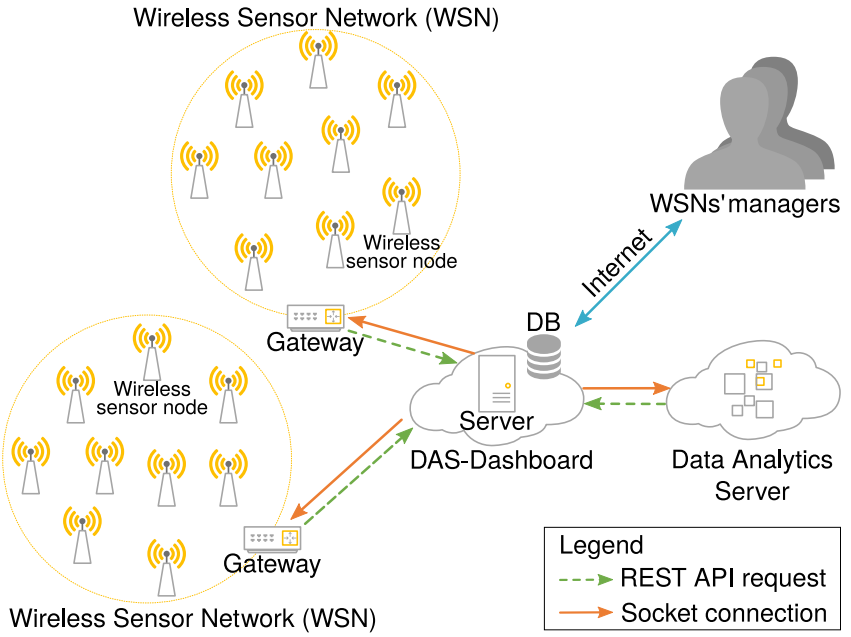



Figure 2.1: The self-managing architecture for WSNs.

agement. The result is a self-managing architecture that controls WSNs based on real-time data analysis, which is shown in Figure 2.1.

## 2.1 A self-managing architecture

The architecture proposed in this work aims for scalability and relies on the power of shifting most of the computation to the cloud. It fits the main principles of data science concerning sensed data: its collection; description; storage; maintenance; discovery; visualization; and analysis. The Data Analytics for Sensors Dashboard (DAS-Dashboard) is used to delegate the WSNs' management and the data analysis to appropriate mechanisms that may perform their operations remotely in the cloud. As the DAS-Dashboard stores and publishes collected sensed data, it is possible to visualize measurements and other collected values, as well as repro-

duce IoT scenarios to optimize data acquisition and its further analysis. Finally, a Data Analytics Server supports different data analytics tools and algorithms that can optimize the data collection regarding the quality of the measurements provided by the WSNs.

IoT environments have an information flow that extends the standard WSNs' data collection, which permits WSNs' operation to be improved based on data analysis executed at runtime. The WISEBED project has already addressed the need of a shared platform for programming WSNs and collecting data from sensor nodes [28]. In that project, several testbeds were deployed, and remote users were able to program sensor nodes and run experiments to prove concepts modeled in simulations. That solution, however, did not integrate tools for remote data analysis and real-time WSN optimization.

More recently, a new architecture was proposed to integrate WSNs in IoT environments [29]. There, a central server communicates with sensor nodes and receives their statistics, which are analyzed by a third-party tool every hour. Based on the data analysis, the central server can react to dynamic changes in network conditions and provide flexibility to the WSN, through remote reconfiguration of the sensor nodes. As a drawback, that architecture may face limitations to support several WSNs simultaneously, because the management in the central server is highly coupled. In short, the server must have installed the same Operating System (OS) used in the sensor nodes, besides hosting the database (DB) to store collected data, and the tool used for data analysis. Hence, that architecture empowers WSNs with Internet services, but it is bounded by the capacity of the central server to scale up and integrate many WSNs, data analytics tools and provide simultaneous access to several remote WSNs' managers.

### 2.1.1 Structure of the proposed architecture

We consider a typical IoT environment composed by several WSNs with ordinary wireless sensor nodes reporting to predetermined sinks, named as *Gateways (GWs)* in Figure 2.1. The proposed architecture is centered

in the *DAS-Dashboard* and consists of interconnected components that can exchange information with trusted entities, eventually from different domains. The components of this architecture are:

**Wireless sensor nodes**

Taking advantage of their proximity to the data origin, they perform default sensing tasks in their deployment area and transmit their measurements via radio to a GW.

**Gateway**

They forward the gathered information to the central server and disseminate occasional instructions and updates to wireless sensor nodes. GWs are the link between ordinary wireless sensor nodes and the DAS-Dashboard, using a point-to-point connection over the Internet or a local network.

**Data Analytics for Sensors Dashboard (DAS-Dashboard)**

The central component of this architecture has three primary responsibilities: collecting, storing and publishing data transmitted by wireless sensor nodes. The ability to collect data requires direct communication with the WSNs and is fundamental for the other two responsibilities. Storing the collected data in a DB allows further access to historical information, besides providing data visualization to WSNs' managers and other users. Finally, communicating data to other systems allows the DAS-Dashboard to outsource data processing, which may involve filtering and analyzing the collected data, besides predicting future measurements.

**Data Analytics Server**

The Data Analytics Server can process computationally expensive real-time analysis over data. To do that, it can rely on external data resources,

such as public services and other DBs on the Internet. Indeed, tasks processed by the Data Analysis Server could not be assigned to sensor nodes, due to their constrained hardware and limited communication with external sources of knowledge.

### 2.1.2 Main data flow

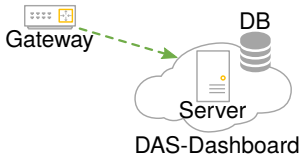
Supported by Transmission Control Protocol (TCP) connections established among different components, each component of this architecture can be moved to the cloud and become remotely reachable. Such an accessibility permits WSNs to interact with other WSNs, communicate with other systems and simultaneously serve several users.

As detailed in Figure 2.2, first, sensor nodes must report data to their respective GW, which is connected to the DAS-Dashboard via the Internet. The data provided by sensor nodes can be, for example, their measurements or statistics about network conditions, such as delays, packet losses, and data throughput. After receiving the reported values, a Data Analytics Server may generate a report and recommend the most proper changes to the WSN's operation. The data analysis can rely on complex computations to extract knowledge from the collected data. Optionally, the resulting environmental analysis can be offered to systems external to the WSN at a certain cost [30].

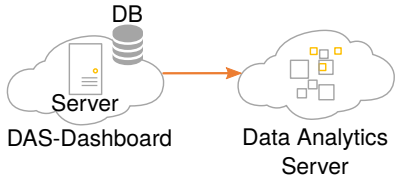
Finally, sensor nodes can be updated to adjust their tasks according to the results of the data analysis. Thanks to this architecture, the data analysis can be delivered as a service to WSNs, such that sensor nodes will be able to apply the knowledge in their favor, e.g., changing their operation to report measurements more often and detail the variations in the environment.

## 2.2 Applications of the architecture

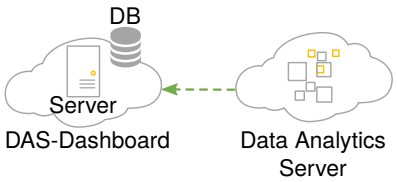
As mentioned before, WSNs are data-oriented, i.e., the sensed data is their most valuable asset. Hence, it is common to rely on analysis of the



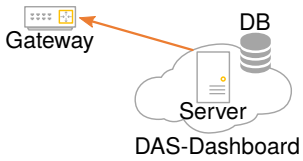

(a) **Step 1**. Sensors perform their default sensing tasks, before transmitting their measurements via radio to GWs. Data is reported to the DAS-Dashboard and stored in the DB for further access.




(b) **Step 2**. Measurements received by the DAS-Dashboard are communicated to the Data Analytics Server. The Data Analytics Server performs the data analysis to infer whether a sensor is gathering informative data or not.




(c) **Step 3**. The Data Analytics Server transmits to the DAS-Dashboard new plans for the WSN, depending on the application type. These plans may be linked with network details (such as sensor nodes' positions) and can rely on external factors (such as the time of the day).




(d) **Step 4**. At this point, the DAS-Dashboard announces the recommendations received from the Data Analytics Server. Then, sensor nodes are reprogrammed according to the instructions communicated to their GW.

Figure 2.2: The self-managing architecture in detail.

collected information to take further actions, and the proposed architecture facilitates this process. There is a vast number of ways to exploit the proposed architecture, from optimizing existing WSNs to starting a new business focused on sensor data. These applications can involve different algorithms for data analysis, new modules that can be attached to the DAS-Dashboard, or algorithms that can be implemented in sensor nodes. In the following, we highlight some examples of ways to exploit this architecture.

### 2.2.1 Multi-scope Integration

Data analysis algorithms can be substituted by AI techniques that evaluate historical trends and trigger actions, activate machines or communicate with other systems according to pre-defined policies. For example, recommendations generated through data analysis can be targeted to WSNs' managers that would take manual actions, such as adding new sensor nodes or replacing existing ones.

### 2.2.2 Sensing as a Service

If a user management layer is attached to the DAS-Dashboard back-end, it will facilitate the identified communication with other systems and allow the information sharing, creating a new business model that offers the information retrieved by sensor nodes as a service. For instance, WSNs can establish data-sharing agreements involving monetary compensations for future cooperation and use shared data from one or more WSNs to optimize another WSN's performance, after analyzing and evaluating how the environment is changing at a precise moment. Alternatively, users may pay for measurements made in a particular region. Meanwhile, data analytics algorithms can improve the resource utilization and offer confidence intervals to estimated values computed locally.

### 2.2.3 Prediction-based data reduction

For a while, prediction algorithms have been underexploited in WSNs, because higher complexity algorithms were thought to be unsuitable for sensor nodes [31]. More recently, real deployments incorporated advanced forecasting algorithms [32] and other AI techniques [33], challenging such an assumption. As a consequence, data mining and other machine learning techniques started to be used to find patterns in the sensed data, improving its collection and delivery [34]. In summary, while machine learning techniques rely on their ability of learning and evolving their predictions in response to changes in the environment, other prediction-based methods may use traditional time series algorithms that depend on the statistics of the studied data to make predictions.

In the proposed architecture, the Data Analytics Server can compute prediction algorithms that will optimize the data transmissions. Thus, predictions can be made only in the Data Analytics Server, as proposed by SPSs, or also in the sensor nodes, as proposed by DPSs.

#### Single Prediction Schemes

In SPSs, predictions are made in a single point of the network, which can be either close to the origin of the data (in sensor nodes) or close to the data collection point (in GWs). For instance, the Data Analytics Server can predict the data measured by sensor nodes and decide when the GWs must pull more measurements, based on the reliability of the predictions. Alternatively, sensor nodes can predict changes in their surroundings to avoid unnecessary measurements and–consequently–their transmissions. The latter alternative is especially beneficial if a sensor node spends more energy to sample the environment than to predict the future measurements, which usually does not happen with wireless sensor nodes that monitor temperature, relative humidity, and other environmental parameters [6].

In the proposed architecture, as GWs are connected to the cloud and have with higher computational power and energy availability, they can have access to predictions and take important decisions about the WSNs'

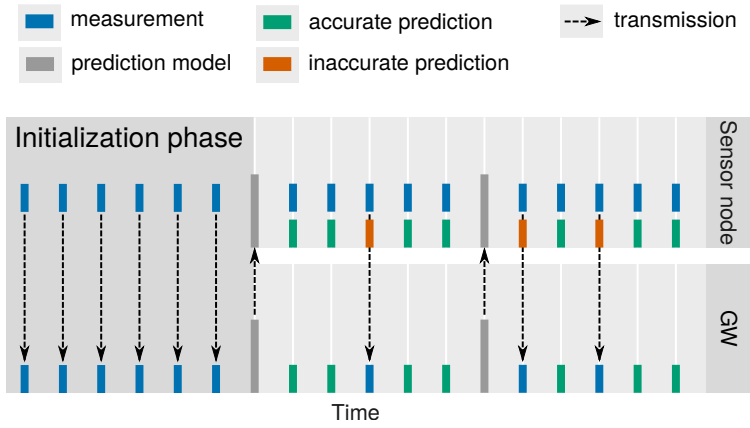



Figure 2.3: In a DPS, a measurement is transmitted only if its forecast is inaccurate. The GW may be responsible for transmitting new prediction models every time interval after the initialization phase.

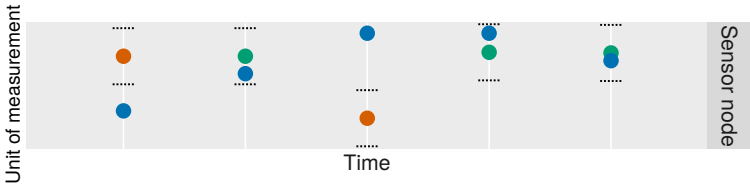



Figure 2.4: Measurements that fall inside the accepted threshold do not trigger any action.

operation without compromising the quality of the information provided by the measurements [35]. Meanwhile, a conservative strategy is adopted in sensor nodes, which become merely responsible for their primary tasks, i.e., measuring environmental parameters and transmitting the raw data collected by their sensors.

#### Dual Prediction Schemes

In DPSs, predictions will be simultaneously made in the GW and sensor nodes. The general idea behind such mechanisms is that sensor nodes can produce the same "a priori" knowledge than the GW is, but sensor nodes can locally check the predictions' accuracy and avoid unnecessary transmissions. As shown in Figure 2.3, the same predictions are made in the sensor node and the GW. Then, every time the sensor node measures a value that falls outside an acceptance threshold defined for the predictions (as represented by the first and the third measurements in Figure 2.4), it must transmit the real value to the GW, which substitutes the values locally predicted. Hence, sensor nodes can consume fewer energy resources and avoid unnecessary transmissions, because measurements will be transmitted to the GW only when the predictions are not sufficiently accurate.

## 2.3 Experimental deployment

To experiment this architecture, we deployed a WSN with TelosB motes [36]. The DAS-Dashboard has been implemented and deployed in a well-dimensioned machine without energy or performance constraints, and with reliable Internet connection and direct access to the WSN's GW [37]. During $4.5$ days, the room-temperature data was collected using four wireless sensor nodes placed in an office, as shown in Figure 2.5. We explain and discuss the experimental deployment in the following.

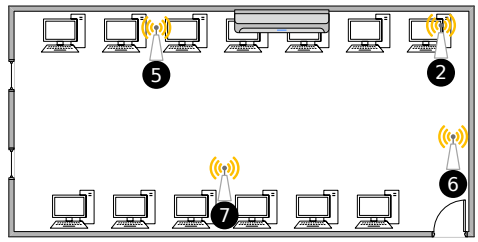



Figure 2.5: Sensor nodes' positions in the office.

### 2.3.1 Adopting a WSN manager

Wireless sensor nodes are typically close to the data origin and have constrained computing capabilities to store and process information. In homogeneous WSNs with similar wireless sensor nodes (regarding their software and hardware technologies), each sensor node may have different configurations, according to its location and measurements' relevance. In heterogeneous WSNs, sensor nodes may differ in more ways, such as computing capabilities, OSs, and clustering roles. Because of these particularities, addressing the architecture's scalability and aiming for simultaneously controlling multiple WSNs require a framework that can avoid resource underutilization and handle eventual changes in WSNs' topology. These aspects facilitate sensor nodes (re)placement and favor the expansion and evolution of WSNs, boosted by advances in software and hardware solutions.

TinyDB [38] is a framework that allows sensor nodes to be queried to measure environmental parameters, such as temperature and relative humidity, periodically. That is, given a set of queries, TinyDB can analyze and optimize the use of the WSN's resources, shortening delivery routes and reducing the overall energy consumption. However, even though TinyDB considers the sensor nodes' energy consumption to decide for the most suitable execution plan, it does not adapt to topology changes that may also impact the routing and the end-to-end delays in the data delivering, which does not favor the scalability of WSNs.

Similar to TinyDB, DIstributed Self-Organizing NEtwork manage-

ment (DISON) [39] is a framework that permits WSNs to self-organize, reacting to changes in the topology and optimizing the overall energy consumption. However, DISON forces each sensor node to reconfigure its operations according to its resources and the network state, which may become an overhead in large WSNs.

**WSN Application development and Resource Management (WARM)**

The architecture proposed in this work is focused on supporting several sensor node types and tasks, while reducing the management control at the most. The scalability of this system can be provided by a framework that abstracts sensor nodes at the application layer and reduces management tasks, such as the WSN Application development and Resource Management (WARM) framework [40]. WARM relies on Software Defined Networking (SDN) features to simplify the development and resource management for WSN applications. An SDN controller implemented in the sensor nodes is responsible for dealing with topology changes and exploring their resources at the best. To achieve that, the SDN controller abstracts the control and data layers [41], which facilitates the adoption of low-level tasks in the format of sensor node applications that can be easily configured. Thus, to meet the scalability requirements described above, WARM was adopted as the WSN manager in our experimental deployment. Figure 2.6 shows the WARM's architecture in detail.

To control the WSN, a sensor node with the WARM controller installed is responsible for receiving, via serial port, any configuration and management commands from the WSN's managers. These commands are further translated to the format used by the SDN controller. The WARM controller also communicates eventual notifications from sensor nodes to the WSN's managers, such as new sensor nodes associations and acknowledgments after updates.

To configure and manage WSN applications, WARM provides a hardware-independent application layer, which makes wireless sensor nodes' particularities transparent to any component external to the WSN.

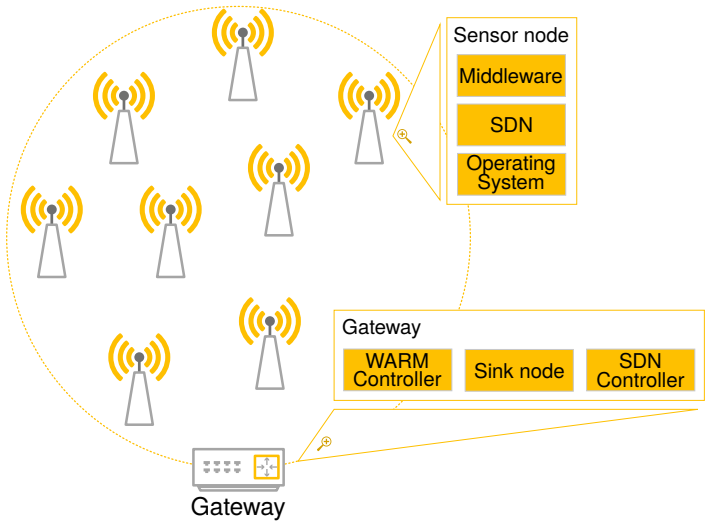



Figure 2.6: The WARMs's architecture in the WSN.

Meanwhile, sensor nodes can still be controlled via an interface that provides means to retrieve their statuses and to schedule tasks, such as sensing environmental parameters or processing collected data periodically. In practice, to program a sensor node to sense the environment, the WSN's managers need only to inform which node will receive the data and the period between consecutive measurements. WARM also specifies interfaces to support new applications that can be further designed by WSNs' managers.

In conclusion, WSNs' managers do not need to spend their time adjusting WSNs with the best set of parameters, configuring protocols that may enhance their power savings or (re-)synchronizing network components after replacing sensor nodes, because all of these tasks are automatically–and properly–guaranteed by WARM.

### 2.3.2 Implementing the DAS-Dashboard

WSNs' managers may adopt different strategies to monitor and analyze all the information retrieved by their WSNs. Storing collected data in

DBs allows further access to historical information, besides providing data visualization to other systems. If the DB is connected to the Internet, collected data can be available to remote users, overcoming physical limitations and providing access to external systems that may independently process such information.

As a drawback, setting up a new DB for each WSN implies overhead, because their schema must be designed, servers must be configured, and the communication with the WSN must be properly established. A standard storage medium can make the data handling transparent to the WSNs' managers, remove the overhead to customize data formats and facilitate the communication with other systems. As a consequence, WSNs' managers can outsource the data processing to systems that filter and analyze the collected data, and, especially, predict future measurements.

The DAS-Dashboard [42] was designed and implemented to communicate with WSNs to collect, store and publish: (i) values reported by sensor nodes; and (ii) recommendations generated by external data analysis servers. To store the data, a PostgreSQL DB server was connected to the system's back-end, allowing further access to historical information. The data input is handled by the back-end, implemented in Sails.js [43], which facilitates the creation of Application Programming Interfaces (APIs) to manage the insertion of new information, and provides further access, on demand, via Hypertext Transfer Protocol (HTTP) Requests. The use of APIs allows the communication among different servers and also guarantees the loose coupling with the front-end. The front-end, implemented in AngularJS [44], provides data visualization to WSNs' managers and remote users via the Internet.

Finally, the DAS-Dashboard guarantees that the data inserted in the DB is communicated via socket connections in the form of events. Thus, external servers can be registered and receive updates about a WSN via the Internet. Establishing standards for data insertion (API requests) and data publishing (socket connections) allows the DAS-Dashboard to integrate with Big Data services and update the operation of the sensor nodes according to the data that they have reported.

### 2.3.3 Implementing the Data Analytics Server

As explained before, typical wireless sensor nodes have constrained memory and computing power, besides limited communication with external networks. Therefore, they do not have enough capacity to store measurements or execute high-complexity algorithms to analyze the environment's evolution and appropriately optimize their data collection. For example, a sensor node could use its data to forecast if it is going to rain and therefore report more often to detail an abrupt change in temperature that would occasionally happen.

Specialized tools, such as the Riverbed Modeler [45], can communicate with the DAS-Dashboard and receive network statistics, such as end-to-end delays, packet arrival times and transmission times. Later, an optimization plan may be generated, based on simulation results obtained in parallel to the WSN's operation. Alternatively, as the DAS-Dashboard provides sufficient tools to store and publish reported data, it could be possible to analyze WSNs' data using public APIs, such as the Google Prediction [46] and Amazon Machine Learning Prediction services [47].

In this work, a customized Data Analytics Server was implemented in R [48]. It can perform different types of data analysis and recommendations, occasionally relying on external data resources, such as public services and other DBs on the Internet. Based on the data analysis results, this server can generate two types of recommendations for sensor nodes: (i) sampling intervals, in which an SPS is used to define the time interval between two measurements; and (ii) predicted values, in which a DPS is used to inform sensor nodes about their future measurements and avoid unnecessary transmissions.

### 2.3.4 Benchmarking

The proposed architecture adds some delay inherent to the communication between the different components (GWs, DAS-Dashboard, and the Data Analytics Server). In our implementation, there are also delays due to data processing in the DAS-Dashboard and Data Analytics Server, besides the time needed to reprogram the sensor nodes and adjust their sam-

| Sensor node | Average clock drift (ms) | $\sigma$ of clock drift (ms) | Average introduced delay (ms) | $\sigma$ of introduced delay (ms) |
|---|---|---|---|---|
| 2 | $-993$ | 824 | 813 | 2514 |
| 5 | $-979$ | 832 | 1173 | 2446 |
| 6 | $-938$ | 862 | 702 | 2326 |
| 7 | $-850$ | 787 | 896 | 2281 |

Table 2.1: Delays observed in the experiments.

pling intervals via WARM. To observe the average run-time overhead introduced by the new data flow, we deployed the GW, the DAS-Dashboard and the Data Analytics Server in the same machine (a PC with 64-bit Intel Core i5 3.33Ghz, 8 GB RAM, and Ubuntu Linux 14.04 LTS), which eliminated any network delays that could impact their operation. Table 2.1 shows the average delays and their respective standard deviations ($\sigma$). These values can be considered lower bounds for future distributed deployments.

Note that when sensor nodes are programmed to sample in fixed time intervals, the period between consecutive measurements may vary, due to their clock drift. In these experiments, sensor nodes sampled, on average, nearly one second earlier than they should have reported. This value is not taken into account when calculating the delays caused by the new data flow. Thus, on average, the process of receiving data in the DAS-Dashboard, analyzing its content in the Data Analytics Server, reporting a recommendation back to the DAS-Dashboard and reprogramming a sensor node did not take longer than $1.2$ seconds.

## 2.4 Summary

A typical WSN is composed of dozens (occasionally hundreds) of ordinary sensor nodes connected to a central workstation that is responsible for providing the communication between WSNs' managers and sensor

nodes. In this Chapter, we described the implementation of an architecture that integrates WSNs in IoT environments. In practice, data gathered by WSNs can be displayed to their managers and other stakeholders, such as data consumers and third-party services that can benefit from the knowledge generated by sensor nodes. The difference from "traditional" WSNs is the online data analysis and the capacity of self-management resulting from the interconnection of several managers at the application layer.

To demonstrate the feasibility of this architecture, we implemented and observed the delays of processing sensor data, and communicating with sensor nodes. In our experiments, we observed a small delay (less than $1.2$ seconds) to collect and process the WSN's data, which illustrates the architecture's real-timeliness that can be exploited in several use cases.

Thanks to the scalability of the proposed architecture, future works may integrate several WSNs that report data simultaneously. Meanwhile, the DAS-Dashboard can provide access to cloud services that exploit the sensor data to a high degree. Additionally, new business models can explore the remote access to sensor data provided by the DAS-Dashboard. The success of our deployment shows that WSNs can be incorporated into self-managing IoT environments that do not depend on human intervention for fine-tuning their operation. In the following Chapter, we validate the architecture's applicability running an SPS application to optimize the WSN's operation.

# Chapter 3

# REINFORCEMENT LEARNING FOR CONTROLLING SENSOR NODES

In WSNs, GWs are the most powerful devices regarding computing power and energy supply. Since these devices have access to information retrieved from several locations, they may assemble a broad picture of the environment's evolution and infer predictive models at a large time-space scale. Existing SPSs show that GWs can predict measurements of a sensor node without communicating directly with it. These mechanisms reduce the number of transmissions and predict missing measurements using past and recent measurements of sensor nodes in the same neighborhood [49, 50, 51, 52]. To do that, GWs may account for possibly existing cyclic behavior, global trends or other aspects not discernible from the sensors' limited (time and space-wise) perspective. Besides reducing the number of WSN transmissions, an SPS can completely turn off sensor nodes' MCUs for a while, which reduces their energy consumption, and may also be an important step in the direction of improving the overall WSN's lifetime.

In the architecture presented in Chapter 2, cloud services empower GWs, giving them access to more information about the environment in the form of knowledge built outside the WSNs. In this Chapter, we propose an SPS that provides on-line sampling interval adaptation using the architecture described in Chapter 2. First, we formally represent the scenario of a monitoring WSN through the RL model and apply a Q-Learning algorithm that learns the most suitable sampling intervals under different conditions, without an a-priori model of the environment's evolution.

Sampling intervals affect the wireless medium access, end-to-end delays, and sensor nodes' energy consumption. Their values are particularly useful in monitoring WSNs, where data quality is reduced if the time interval between two measurements is not sufficiently short to report significant changes in the monitored parameters, or if it is too brief and several similar (and, consequently, unimportant) values are reported. Also, a sampling interval set under some conditions may occasionally become too short (or too long) within time, due to the environment's evolution. Thus, our primary goal is to guarantee the minimum number of transmissions needed to avoid losing valuable environmental data. To achieve that, we aim to keep the maximum difference between two consecutive measurements below an application-defined threshold. In the end, we simulate the proposed mechanism to observe the factors that impact the scheme's performance and experiment the algorithm in a real deployment.

## 3.1 Single Prediction Schemes with model generation in Gateways

Especially in environmental monitoring WSNs, measurements made by closely positioned sensor nodes have a spatio-temporal correlation, which can be used to generate probabilistic models, approximate the data to well-known distributions and associate confidence levels to predictions. Hence, the number of transmissions can be reduced if GWs predict measurements and locally check whether the user-imposed quality constraints are matched or not. Because of the autonomy is given to GWs, SPSs have

been used in several application types, such as topology controlling, clustering and adaptive sampling.

### 3.1.1 Topology control

Some approaches use SPSs to exploit the spatio-temporal correlation between sensor nodes' measurements and build sets of nodes that can provide "trustful" measurements and should, therefore, be regularly sampled [50, 52]. In these approaches, only a subset of sensor nodes is activated during a time interval and all the others have their radios and sensors turned off to reduce the number of transmissions, save energy and extend the WSN's lifetime. Every subset of sensor nodes provides the values used to predict the measurements of the whole WSN. The predicted values, on average, should differ by less than a user-defined threshold from the real measurements.

The Binocular framework requires, before the WSN deployment, the knowledge about which subsets of sensor nodes must be active at a time [50]. During the so-called *data processing* phase, the GW receives measurements from sensor nodes and calculates linear transformations that will be used to make predictions using data from the sensor nodes that will remain active. Simulations using real data showed that this framework could be used to extend the WSNs' lifetime when the requirements about the accuracy were not very strict, namely, when the temperature could be wrong by $\pm 0.5^{o}$C with a confidence level of $95\%$. Furthermore, the WSN must be dense enough so that some sensor nodes could be switched off and their measurements inferred using their neighbors' measurements.

### 3.1.2 Clustering

Alternatively, SPSs can be used to build clusters of sensor nodes based on the similarity of their measurements. This criterion reduces the divergences between measurements and their deviation from the average in a cluster, which facilitates the data compression [53]. Simulations using

data collected by real sensor nodes showed that it was possible to reduce the number of transmissions in a WSN without injecting significant errors to the data reported to the WSN's managers. Alternatively, some authors used Principal Component Analysis (PCA) to reduce the number of dimensions of the data and make fewer transmissions from Cluster Heads (CHs) to GWs [13].

The main drawback of this kind of clustering is that sensor nodes' roles in the clusters rely on data analysis and not in the computational power of the sensor nodes. Consequently, CHs may not have the computational power needed to analyze sensed data, because the analysis algorithm may require some advanced instructions that cannot be computed in the simplest wireless sensor nodes, such as the multiplication of large matrices in the PCA method.

### 3.1.3 Adaptive sampling

User queries contain, besides the data that should be returned, the error tolerated by the user. Therefore, GWs can answer that the current measurements are inside a range of values if their confidence is high enough to satisfy a user-tolerated error. To do that, GWs must be able to predict future measurements based on the statistics of the historical data–considering the uncertainty about the current values–and autonomously decide whether to pull more measurements or not [49]. As an alternative, GWs can use inferential statistics to determine which sensor nodes have to be sampled, based on their odds of providing valuable information to the WSNs' managers.

For example, the mechanism called BBQ adopts linear regressions to exploit the correlation between different types of data that the sensor nodes may be able to measure, such as their voltage and local temperature [51]. Results of simulations using real data suggest that this mechanism can reduce the number of transmissions, save energy and keep a high confidence level ($95\%$) about the information retrieved, besides keeping a low number of mistakes in stable scenarios (where changes are easier to predict). Alternatively, the PCA method was used to select only the

sensor nodes that measured most of the variance observed in the environment [54]. This technique reduced the workload of the sensor nodes and prolonged twice the WSNs' lifetime, according to the results obtained from experiments in real testbeds.

## 3.2 Adapting sampling intervals using a Reinforcement Learning algorithm

Inspired on existing SPSs, we designed a mechanism to adapt sensor nodes' sampling intervals using an RL algorithm [55]. Differently from existing works, our proposal does not necessarily rely on the computational capacity of sensor nodes or GWs, because the incorporation of WSNs into the IoT allows the use of cloud services that can perform powerful machine learning techniques over sensed data. To the best of our knowledge, this is the first approach that dynamically adapts the sampling interval of the sensor nodes based on an RL technique.

### 3.2.1 Background - Reinforcement Learning

RL is a machine learning technique that allows an autonomic *agent* to determine a system's optimal behavior to achieve its goal [56]. Such an optimal behavior is based on the positive and negative feedbacks received from the environment after taking certain *actions*. Assuming that interactions between the *agent* and the environment occur at a sequence of discrete time instant $t$, an RL model is defined by:

- A set of possible *observations* $O$ that the *agent* may make, such that $o_t \in O$ is the observation made at time $t$;
- A set of *states* $S$, such that the *state* $s_t \in S$ is observed at time $t$;
- A set of *actions* $A$, such that the *action* $a_t \in A$ is taken at time $t$;

- A state transition function $T(s_t, a_t, s_{t+1})$ that calculates the probability of making a transition from $s_t$ to $s_{t+1}$ after performing $a_t$; and

- A set of rules that determine the scalar immediate *reward* $r_{t+1} = R(s_t, a_t)$, which scales the goodness of taking $a_t$ in $s_t$.

Each *state* should satisfy the Markov property[1], that is, to be independent of any *state* or *action* previous to time $t$. An RL *agent* aims to obtain the maximum long-term *reward* for a Markov Decision Process environment, even when the model of the environment is unknown or difficult to learn. The strategy adopted to maximize the long-term *reward* defines the *agent*'s way of behaving at a particular time and is called a *policy*.

### Q-Learning

Q-Learning is an RL algorithm that does not depend on a state transition function to work. More precisely, the algorithm relies on an optimal action-value function $Q(s, a)$, which value is the estimated *reward* of executing $a$ in $s$, assuming that the *agent* will always follow the *policy* that provides the maximum long-term *reward*.

At any state $s_t$, a selected action $a_t$ determines the transition to the state $s_{t+1}$ and the value associated to the pair ($s_t$, $a_t$) is updated:

$$Q_{t+1}(s_t, a_t) = \alpha \left( r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right) + Q_t(s_t, a_t), \tag{3.1}$$

where $s_{t+1}$ and $r_{t+1}$ are the *state* and *reward*, respectively, obtained after performing $a_t$ in $s_t$, the *learning rate* $\alpha \in [0, 1]$ is a positive step-size parameter, and the *discount factor* $\gamma \in [0, 1]$ is used to determine the weight of future *rewards*. If $\gamma = 0$, the *agent* will behave so as to maximize its immediate *reward*, even if this would imply a lower long-term return.

[1] RL can also be applied to cases that do not satisfy the Markov property [56]

By visiting several times each ($s$, $a$) pair, the *agent* learns which is the *action* that gives the best long-term *reward* in each *state*. Hence, if the number of *states* is high, the algorithm takes longer and requires more data to find the best *action* for each *state*, i.e., to *converge*. Therefore, it is very critical to have a concise representation of the environment, thus to define the set of *states* according to the goals of the algorithm and do not include unnecessary information. In short, the set of *states* should illustrate only and all the characteristics that are relevant for the problem under consideration.

### 3.2.2 Adaptive sampling interval problem as a Reinforcement Learning problem

To formulate the adaptive sampling interval problem as an RL problem, we consider the number of transmissions as a general measure of resource optimization (which is also valuable in scenarios where sensor nodes have energy constraints). Thus, to illustrate further concepts, we took as a reference the real scenario of a WSN with several nodes measuring temperature values in an office [57]. From this real dataset, we used a subset of measurements in the preliminary analysis presented in this Section, and a different subset in the performance evaluation in Section 3.3.3 (missing values were interpolated and added to a small white noise). Moreover, as the sensors of this scenario were set to sample temperature nearly every 30 seconds, we set this as the shortest sampling interval and let the range of possible sampling intervals be (i) **30 seconds**; (ii) **60 seconds**; (iii) **120 seconds**; or (iv) **240 seconds**.

#### Goal

A valuable adaptive sampling algorithm should systematically set up the most proper sampling interval so as to guarantee the best quality-resource trade-off under the current environmental conditions. As for the quality, we define the goal of the *agent* regarding an *accepted threshold* $\tau$, such that the algorithm should avoid that the absolute difference between

| Factor | Description |
|---|---|
| **Quality** ($q$) | $q \triangleq \lvert o_t - o_{t-1} \rvert \leq \tau \therefore q \in \{\text{true}, \text{false}\}$ |
| **Hour of the day** ($h$) | $h \in [0, 1, \ldots, 23]$ |
| **Is it working hour?** ($\omega$) | $\omega \triangleq h \geq 7$ and $h \leq 18 \therefore \omega \in \{\text{true}, \text{false}\}$ |
| **Day of the week** ($d$) | $d \in \{\text{Monday}, \text{ Tuesday}, \ldots, \text{ Sunday}\}$ |
| **Is it weekend?** ($e$) | $e \triangleq d \in \{\text{Sat.}, \text{ Sun.}\} \therefore e \in \{\text{true}, \text{false}\}$ |
| **Sampling interval** ($k$) | $k \in \{30, 60, 120, 240\}$ seconds |
| **Node ID** ($i$) | Individual sensor node identification. |

Table 3.1: Factors that can impact the quality of measurements.

consecutive measurements exceeds an application-specific predefined $\tau$. Meanwhile, higher sampling intervals are preferred to reduce the number of transmissions and, consequently, the energy consumption in sensor nodes.

### Observations

We define wireless sensor nodes as the source of the observations made by an *agent*. Observations may vary, among other parameters, between temperature, relative humidity and solar radiation. In our scenario example, an *observation* $o_t$ is a temperature measured at time $t$.

### States

To accurately define the set of possible *states*, we make a preliminary evaluation of part of the data collected by the WSN and identify characteristics that have a high correlation with our goal. Such characteristics are transformed into predictors, and a Random Forest [58] is built to classify in which periods of time each sampling interval would make consecutive measurements that differ by less than $\tau$.

To illustrate this process, we used data from three different sensor nodes sampled every $30$ seconds for three days. We removed intermediate measurements to simulate different sampling intervals. Then, we

observed that if the value of $\tau$ were set to $0.02^o$C, a sampling interval of $120$ seconds would be sufficient to observe a difference of less than $\tau$ in nearly one-half of the time. Furthermore, sampling intervals of $30$, $60$ and $240$ seconds would be sufficient to observe an absolute difference of less than $\tau$ in, respectively, approximately $73.2\%$, $59.2\%$ and $26.1\%$ of the time. Note that we do not expect to measure fast enough to make all measurements differ by less than $\tau$, but we know that there are many cases in which sensors do not have to sample every $30$ seconds to make it.

Finally, we annotated each measurement according to the characteristics shown in Table 3.1 and built a Random Forest to predict if a measurement would differ by less than $\tau$ from the previous one (i.e., $q =$ true). In fact, the Random Forest method permits us to observe which factors have the most positive impact on the predictions' accuracy.

Based on the results obtained and considering the importance of keeping a small number of *states*, we defined the set of *states* for our RL model. Each *state* is a combination of quality, sampling interval and a verification of whether it is a working hour or not: $\{q, k, \omega\}$. In short, as we were looking for *states* that could be used by different sensor nodes, we ignored any factor that was less important than the node ID to predict the quality of the measurements, plus the hour of the day ($h$), because it would represent a significant increase in the number of states.

### Actions and Transitions

In the adaptive sampling interval problem, *actions* are used to control the sensor nodes' sampling interval. An *action* can be specific, such as "set the sampling interval to $30$ seconds", or more abstract, like "increase the sampling interval" requiring the new sampling interval to be calculated based on the current one. To avoid abrupt changes provoked by occasional outliers and noise in the data, we adopted "smooth" *actions* that only move to neighboring *states*. Therefore, in our illustration scenario an *action* $a$ could take one of the following values: (i) **increase the sampling interval**; (ii) **keep the sampling interval**; or (iii) **reduce the sampling interval**.

### Reward

A *reward* is a mathematical representation of the gains obtained after reacting to the environment with a particular *action*. In our case, it is calculated after changing the sampling interval to a new value, while in $s$. As the reward defines the target of the algorithm, in our problem, it should ensure that the difference between consecutive measurements is less than $\tau$, while not oversampling.

The algorithm adopted for the *reward* is based on the rate of transmissions avoided. For instance, if the sampling interval is $120$ seconds, the sensor node is transmitting four times less than if it was $30$ seconds. In this case, therefore, the original *reward* would be set to $\mathbf{4}$. Then, if the absolute difference between two consecutive measurements ($\delta$) is less than $\tau$, we assume that the sampling interval is small enough to avoid losing significant changes in the environment and take the original *reward*. If $\delta$ is greater than one-half of $\tau$, the sampling interval is the best one possible, so we multiply the original *reward* by $\mathbf{1.5}$. Otherwise, if $\delta$ is greater than $\tau$, the sampling interval is too long, and the reported data may be missing important changes in the environment. In this case, we multiply the original *reward* by $\mathbf{-1}$.

## 3.3 Simulations

To check the feasibility of using RL as a means of intelligently adapting sampling intervals, we simulated its use in artificial and realistic scenarios. Using OMNeT++ [59] and MiXiM [60], we reproduced a WSN with a single sensor node controlled by our RL algorithm [55, 61].

To represent the simplest situations, we generated synthetic data with evident attributes, such as large and significant (versus small and negligible) variations in a short period. Having control over the data characteristics allows us first to verify if the RL algorithm decides for the most proper *actions* in different scenarios. Later, we will analyze the impact of the values of *learning rate* $\alpha$ and *discount factor* $\gamma$ in the decisions taken by the *agent*.

| Learning rate $\alpha$ | Discount factor $\gamma$ | Convergence time (s) | % of wrong decisions | % of measurements over $\tau^2$ |
|---|---|---|---|---|
| 0.9 | 0.1 | 1013.00 | 4.79 | 2.31 |
| 0.8 | 0.2 | 1050.32 | 6.11 | 4.38 |
| 0.8 | 0.1 | 1088.01 | 8.36 | 1.80 |
| 0.8 | 0.4 | 1163.05 | 13.46 | 8.80 |
| 0.8 | 0.5 | 1640.30 | 12.82 | 10.79 |
| 0.9 | 0.2 | 1920.57 | 11.57 | 3.65 |
| 0.5 | 0.2 | 18330.53 | 25.39 | 10.89 |
| 0.9 | 0.4 | 18457.82 | 19.90 | 4.63 |
| 0.6 | 0.1 | 21922.80 | 18.47 | 5.47 |
| 0.9 | 0.5 | 23512.98 | 20.68 | 4.34 |

Table 3.2: Simulations over the *Controlled* datasets: average convergence times and percentage of wrongly taken decisions by each combination of $\alpha$ and $\gamma$. These values are sorted by the lowest convergence times.

Finally, we observed how long the Q-Learning algorithm takes to decide for the correct sampling interval. We call this period *convergence time*, and we assumed that the *agent* has converged to a final value if the sampling interval did not change in, at least, $75\%$ of the future decisions. The reported values are the average of all considered scenarios.

### 3.3.1 Synthetic scenarios with fixed expectations

We generated six synthetic scenarios in which we had control about the sampling intervals that the algorithm should set. In these datasets, the difference between consecutive measurements is proportional to $\tau$. For instance, if the difference between two consecutive measurements made at periods of $60$ seconds is always smaller than $\tau$, setting the sampling interval to $30$ or $60$ seconds is sufficient to satisfy the requirements of quality. However, setting it to $60$ seconds is preferred, because it reduces the number of transmissions, in comparison with the $30$-second interval.

In the *Controlled 30* dataset, the difference between consecutive measurements made at intervals of $30$ seconds is always $110\%$ of $\tau$. In practice, even the smallest sampling interval ($30$ seconds) is not sufficient to provide measurements in which consecutive values differ by less than $\tau$. Therefore, the *agent* must define the ideal sampling interval as $30$ seconds to reduce as much as possible the quality loss. Note that in this particular scenario, a difference between consecutive values higher than the maximum threshold is unavoidable. Thus, we did not consider this dataset when reporting the percentage of measurements over $\tau$.

In the *Controlled 60*, *Controlled 120* and *Controlled 240* datasets, the difference between consecutive measurements made at intervals of $30$ seconds is respectively $47.5\%$ $23.75\%$ and $10\%$ of $\tau$. Hence, $60$, $120$ and $240$ seconds are respectively the largest possible sampling intervals such that the sensor node will never report a difference greater than $\tau$. Hence, the *agent* must define the ideal sampling interval respectively to $60$, $120$ and $240$ seconds in each scenario. Note that, antagonistically to the *Controlled 30* dataset, in *Controlled 240*, successive measurements never have an absolute difference higher than the maximum threshold. Thus, we also did not consider this dataset when reporting the percentage of measurements over $\tau$.

Table 3.2 shows the combinations with the ten lowest average convergence times, the percentage of times that the *agent* took a wrong decision (using the expected sampling interval as a reference) and the percentage of consecutive measurements that differed by more than $\tau$. In our simulations, the average convergence time was less than $2000$ seconds (around $33$ minutes) only in six cases and nearly ten times longer in the remaining. We highlight, as WSNs are usually long-term deployments that last for months (or years), the period of one day (or less) spent to find the most proper sampling intervals represents less than $1\%$ of their average lifetime.

Half of the combinations shown in Table 3.2 had high $\alpha$ (i.e., $\alpha \in \{0.8, 0.9\}$) and low $\gamma$ (i.e., $\gamma \in \{0.1, 0.2\}$), which means that the *agent* performs better when its decisions are mostly based on the current status of the environment, and future estimated rewards have little importance.

In practice, it shows that if a certain *action* resulted in high *rewards* in the day before, it would not necessarily result in high *rewards* in the future, due to the environment's evolution.

### 3.3.2 Synthetic scenarios with moving expectations

In real world applications, the environment may be continuously changing and evolving, requiring that *agents* never stop to learn, because there might not exist an answer that stands forever as the most proper one. To synthesize these situations, we generated three datasets that are combinations of the *Controlled* datasets presented before. Finally, we simulate four days in which the *agent* should converge to a new value each day, updating its previous belief. In practice, these scenarios will show how good is the algorithm to update its decisions in response to the environment's evolution.

The sequence of expected sampling intervals varies in each dataset. In *Evolving I*, the sampling interval that satisfies $\tau$ evolves in the sequence: $30$ seconds in the first day, $60$ seconds in the second day, $120$ seconds in the third day, and $240$ seconds in the last day. In *Evolving II*, the most proper sequence of sampling intervals is $240$, $120$, $60$, and $30$ seconds. In *Evolving III*, the most proper sequence of sampling intervals is $60$, $120$, $240$, and $30$ seconds.

Table 3.3 shows the three parameter combinations that took less than $16$ hours to converge on every simulated day and the respective percentage of wrong decisions. Recall that, in these–more realistic–scenarios, the conditions change every $24$ hours. Therefore, every day, the *agent* revisits *states* and updates its knowledge to set the most proper sampling intervals, which increases the time necessary to converge: on average, at least $4.5$ hours more than in the previous simulations.

Once again, a high $\alpha$ (namely, $\alpha = 0.9$) combined with low $\gamma$ (i.e., $\gamma \in \{0.1, 0.2, 0.5\}$) would be the best option to reduce the average time that the *agent* took to converge to the most proper sampling interval value, considering that the environment is continuously evolving.

| Learning rate | Discount factor | Convergence time (s) | % of wrong decisions | % of measurements over $\tau$ |
|---|---|---|---|---|
| 0.9 | 0.2 | 18417.80 | 22.78 | 7.56 |
| 0.9 | 0.1 | 31420.31 | 37.45 | 3.24 |
| 0.9 | 0.5 | 31782.82 | 48.30 | 13.64 |

Table 3.3: Simulations over the *Evolving* datasets: average convergence times and percentage of wrongly taken decisions by each combination of $\alpha$ and $\gamma$. The values are sorted by the lowest convergence times.

### 3.3.3 Real world scenarios

In real world scenarios, the environment is always changing, and there are external (uncontrolled) factors that impact the measurements. To simulate that, we adopted real measurements collected during five days by five wireless sensor nodes and set $0.02^{o}C$ as the value of $\tau$, using the strategy explained in Section 3.2.2. These measurements were collected in the same experiment we considered to set up the *states* in Section 3.2.2, but in these simulations, we used data from different sensor nodes.

To illustrate the results, we assume that during the first $12$ hours, the Q-Learning algorithm "calibrated" the action-value function, i.e., it tried to visit all state-action pairs to estimate the long-term *rewards* that each *action* would provide in each *state*. Therefore, we considered only the results observed in the last $4.5$ simulated days.

As before, we observed how the values of $\alpha$ and $\gamma$ impacted the quality of the measurements and the number of wireless transmissions in the sensor node. However, differently from the synthetic scenarios, it is not possible to define the expected values in real world situations. The main reason is that the environment is constantly changing and evolving, besides external factors that produce noise and change the environment itself. Indeed, this is the core motivation of this work and what requires the design of a solution that can adaptively adjust sensor nodes' sampling intervals.

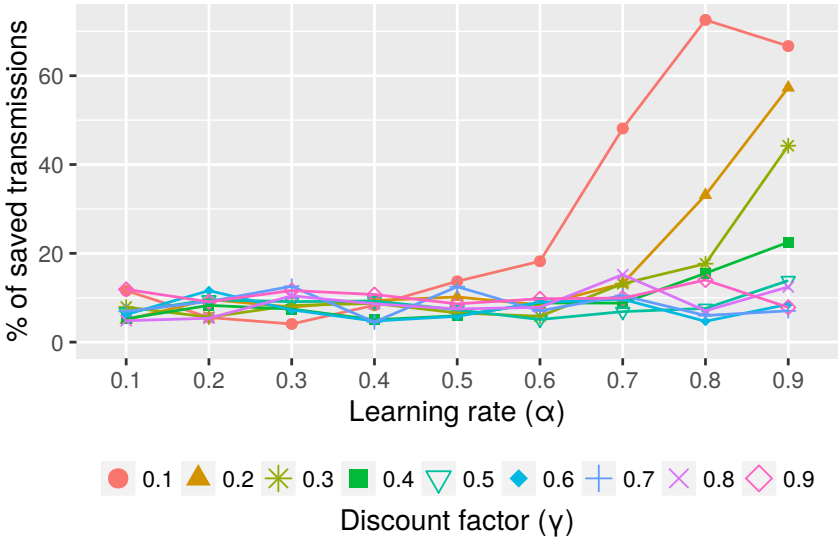



Figure 3.1: Impact of the Q-Learning *agent* in the reduction of the number of transmissions.

### Number of transmissions

In our experiments, the number of transmissions in a sensor node achieves its maximum when the sampling interval is $30$ seconds and its minimum when the sampling interval is $240$ seconds. Intuitively, setting the sampling interval to $60$, $120$ and $240$ seconds represents a reduction of respectively $50\%$, $75\%$ and $87.5\%$ in the maximum number of transmissions.

Figure 3.1 illustrates how many transmissions could be saved when the Q-Learning was adopted to adjust the sensor node's sampling interval. To make this plot, we considered that the Q-Learning *agent* triggered one new transmission every time a new sampling interval was set. In the best case, the number of transmissions could be reduced to $72.57\%$ of its maximum, when $\alpha = 0.8$ and $\gamma = 0.1$. As observed in our preliminary results, the highest savings happened when $\alpha$ was high (i.e., $\alpha \in \{0.7, 0.8, 0.9\}$) and $\gamma$ was low (i.e., $\gamma \in \{0.1, 0.2, 0.3\}$). That is, when the *agent* learned mostly from recent environment feedback and minimally from the ex-

pected *reward*. We highlight the importance of setting proper values to $\alpha$ and $\gamma$, given that most of the cases did not reduce by more than $15\%$ the maximum number of transmissions.

**Efficiency**

Figure 3.2 shows the percentage of consecutive measurements that differed by more than $\tau$. To help the understanding of the magnitude of the errors, we added four baselines that represent the rates that would be observed if the sampling intervals were fixed, based on the same data used in the simulations. Again, the best results happened in scenarios using higher $\alpha$ (i.e., $\alpha \in \{0.7, 0.8, 0.9\}$) and lower $\gamma$ (i.e., $\gamma \in \{0.1, 0.2, 0.3\}$).

In the best result ($\alpha = 0.9$ and $\gamma = 0.1$), the rate of measurements over $\tau$ was similar to the scenario with a fixed sampling interval of $30$ seconds. With the sampling interval fixed to $30$ seconds, we observed $16509$ pairs of consecutive measurements that differed by more than $\tau$ with an average of $0.063$ºC. Using Q-Learning, we observed $13679$ pairs of consecutive measurements that differed by more than $\tau$, which differed by $0.078$ºC on average. These small values strengthen the relevance of the reduction in the number of transmissions shown above, because they indicate that the avoided transmissions are, in fact, worthless in this scenario. In conclusion, a real application that adopted Q-Learning with $\alpha = 0.9$ and $\gamma = 0.1$ would have saved around $65\%$ of its transmissions and observed an average of $0.024$ºC in the absolute difference between two consecutive measurements.

## 3.4 Experimental results

To test the proposed scheme, we utilized the architecture implemented in Chapter 2, which has a WSN to monitor room-temperature in an office [37]. In this office, the temperature is constantly changing, and several factors impact the measurements, such as the presence of people and the air conditioning system. Considering measurements from all sensor nodes during the first two days, we set $\tau = 0.5^{\text{o}}C$. We highlight that

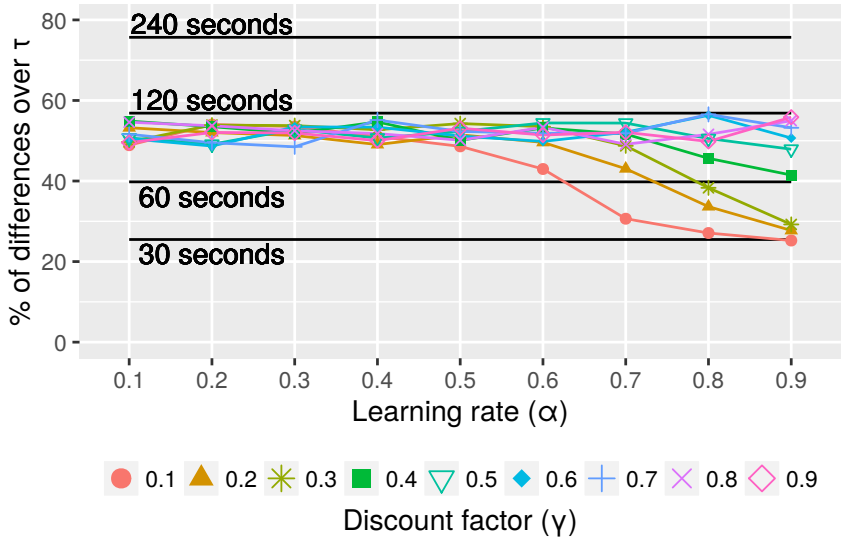



Figure 3.2: Percentage of consecutive measurements that differ by more than $\tau$. The black lines show the percentage observed the sampling intervals fixed to the values written in the plot.

around $6.4\%$ of consecutive measurements made at intervals of $480$ seconds would differ by more than $0.5^{o}C$ and that around $0.15\%$ of consecutive measurements made at intervals of $30$ seconds would still differ by more than $0.5^{o}C$. Therefore, we did not expect to measure fast sufficiently to have all measurements differ by less than this value, but we knew that there might exist several cases in which sensors did not have to sample every $30$ seconds because the environment was not (significantly) changing every time.

From the third day, the Data Analytics Server systematically set up the most proper sampling interval so as to guarantee the best quality-resource trade-off under the current environmental conditions. The parameters of the Q-Learning algorithm were chosen based on the best results obtained in previous simulations, i.e., $\alpha = 0.9$ and $\gamma = 0.1$. Again, to illustrate the results, we considered the first $12$ hours as the time necessary for the Q-

| Sensor node | Transmissions saved | $\delta > \tau$ | Average $\delta$ |
|---|---|---|---|
| 2 | 82.71% | 7.68% | 0.29 |
| 5 | 84.62% | 4.52% | 0.25 |
| 6 | 75.82% | 11.32% | 0.29 |
| 7 | 43.35% | 2.78% | 0.09 |

Table 3.4: Transmission savings observed in the experiments

Learning algorithm to "calibrate" the action-value function, i.e., when it visits all possible action-state pairs to find the best actions it should take in the future. Thus, after the initial 12 hours needed to "calibrate", the experiment ran for two days more.

### Number of transmissions

In our experiments, the number of transmissions in a sensor node achieved its maximum when the sampling interval was 30 seconds and its minimum when the sampling interval was 480 seconds. Intuitively, setting the sampling interval to 60, 120, 240 and 480 seconds would represent a reduction of respectively 50%, 75%, 87.5% and 93.75% in the maximum number of transmissions.

Every time that the DAS-Dashboard reprogrammed a sensor node, at least one extra transmission was made (due to packet losses, commands were retransmitted after a delay to guarantee their delivery). Considering also these transmissions, Table 3.4 illustrates how many transmissions could be saved in each case when the Q-Learning algorithm was adopted to adjust the sensor nodes' sampling interval. The baseline is the scenario with the sensor nodes always sampling every 30 seconds without extra transmissions from the DAS-Dashboard. In the best case, the number of transmissions was reduced to 84.62% of its maximum.

**Efficiency**

In Table 3.4, it is possible to see that the average absolute difference between consecutive measurements was always smaller than $\tau$. In the worst case, sensor nodes $2$ and $6$ reported an average of $0.29$°C in the absolute difference between consecutive measurements.

Concerning the algorithm's efficiency, the number of consecutive measurements that differed by more than $\tau$ could be extremely low. For example, sensor node $7$ had the best results: only $2.78\%$ of the consecutive measurements differing by more than $\tau$. We could observe, in the reported measurements, that sensor node $7$ was directly affected by the air conditioning system, which was positioned in front of it. As a response to the great variations, the *agent* took the most precautionary *actions* to avoid negative *rewards*. As a result, this sensor sampled most of the time in periods of $30$ seconds, which increased the average number of transmissions, but generated satisfactory results regarding the quality of the measurements.

## 3.5 Summary

We conclude that an SPS that uses an RL algorithm to control sensor nodes' sampling intervals can be very profitable. Furthermore, we highlight that the proper choice of its parameters ($\alpha$ and $\gamma$) can significantly impact the results. For instance, in our simulations, higher values of $\alpha$ (i.e., $\alpha \in \{0.8, 0.9\}$) and lower values of $\gamma$ (i.e., $\gamma \in \{0.1, 0.2\}$) provided the best cost-benefit in the "needed transmissions"-"high quality measurements" relationship. In our experiments, a real WSN could be automatically configured using an RL algorithm that learned from the historical data and generated instructions to adapt sensor nodes in reaction to the environmental changes. In practice, the algorithm could reduce up to $84.62\%$ the number of transmissions in the sensor nodes. Most importantly, results show that no human intervention is necessary to adjust the sampling intervals according to the environment's evolution

The most significant contribution of this Chapter is to show that the ar-

chitecture proposed in Chapter 2 provides necessary means to make smart WSNs with the capacity of self-optimizing. Then, having clarified that SPSs can benefit sensor networks, we point out an unanswered question: *Is it feasible to forecast the sensors' measurements in the sensors?*

Discarding the direct access to other sources of data that could improve the predictions' accuracy, the list of prediction algorithms is reduced, and the results are limited by the sensor nodes' constrained hardware. For the next generation of sensor applications, a detailed study is necessary to show how accurate can be the predictions made by the sensors and their respective cost-benefit analysis. In the following Chapter, we will investigate the feasibility of using complex prediction algorithms in DPSs to reduce the number of transmissions *without reducing the quality of their measurements*.

# Chapter 4

# EFFICIENCY OF DUAL PREDICTION SCHEMES

SPSs exploit the computational power of the GWs to guarantee that sensor nodes will only measure and transmit data. Such a conservative strategy fits wireless sensor nodes with memory and energy constraints, but inevitably reduces the quality of the measurements provided by WSNs, when compared with scenarios in which sensor nodes are programmed to measure and transmit raw data regularly at the highest frequency possible.

Differently from SPSs, in DPSs, sensor nodes do not abstain from sensing the environment, and, therefore, the reduction in the quality of the measurements provided by the WSNs can be controlled by the sensor nodes. Fortunately, sensor nodes have evolved in the last years from devices with constrained energy and memory resources [36] to the point where some modern hardware can harvest energy and work autonomously for longer periods [62]. Consequently, several approaches adopted complex algorithms for predictions in DPSs to avoid only the unnecessary transmissions and reduce the energy consumption in sensor nodes without compromising the quality of their measurements.

However, the results published so far are not sufficient to support a broad adoption of DPSs. In some cases, state-of-the-art algorithms for predictions have been compared and shown to be accurate in several uses

that do not involve WSNs [63]. In other works, researchers also considered the limited data access and processing power inherent to sensor nodes, but few scenarios were experimented [32, 64]. In this Chapter, we investigate if the current state-of-the-art forecasting methods (i.e., algorithms to predict future values) can keep the quality of the measurements provided by different WSNs, given sensor nodes' memory limitations and their constrained access to external knowledge.

To do that, we conduct a study over datasets encompassing several data types, originated in different scenarios [65]. These datasets represent some of the sensor applications and illustrate the data heterogeneity inherent to sensor networks. As for the evaluation, we focus on the number of transmissions avoided and the quality of the reported data.

## 4.1 Background - Prediction methods

The term *prediction* can either refer to the process of inferring missing values in a dataset based on statistics or empirical probability, or to the estimation of future values based on the historical data. The latter mechanism is also called *forecast*, and it is the class of predictions we will refer to in this Chapter. To predict the future, we must consider a high number of possible outcomes, given the uncertainty about the factors that may impact the scenario under consideration. Thus, forecasts differ from other predictions because this range of possibilities tends to be wider than in cases when missing values are inferred.

A *prediction method* $P$ is a deterministic algorithm that produces predictions based on two input variables: a set of *observed values* $X$ and a *set of parameters* $\theta$. A *prediction model* $p$ is an instance of a prediction method $P$, such that $p_\theta(X) = P(X, \theta)$. Thus, a prediction model is determined by $P$ and $\theta$.

The values of $\theta$ can be provided by a utility function that measures the predictions' accuracy, models' complexity or information loss. Thus, *choosing a prediction model* means finding the values of $\theta$ that best summarizes the current measurements under the criteria adopted by the utility

function (e.g., the minimum information loss estimated). In Appendix A, we summarize methods that have been used to choose prediction models in WSNs.

## 4.2 Background - Dual Prediction Schemes

In DPSs, there is an *initialization phase*, i.e., a period during which sensor nodes report all the data that they have measured to the GWs [66]. After this phase, prediction models can be chosen. To do that, there are three possible strategies (some mechanisms allow more than one approach [67]):

1. GW and sensor nodes choose the same prediction models independently, based on the data collected during the *initialization phase*.

2. Each sensor node chooses its prediction model and transmits the parameters to the GW.

3. The GW chooses prediction models and transmits them to sensor nodes.

After choosing the prediction models, sensor nodes have the advantage of locally verifying if predictions are inaccurate and transmit the actual measurements if needed. Thus, sensor nodes may either regularly report the data to the GW, due to the lack of accuracy in predictions, or not report any sensor reading at all, in case that the predictions are sufficiently accurate. Finally, sensor nodes and GW may periodically restart the whole cycle with a new *initialization phase* to choose new prediction models.

### 4.2.1 Independent model choice

The *initialization phase* ensures that the GW will have complete information about the environment before any prediction model is chosen. Therefore, after this phase, the GW can choose the same prediction models as

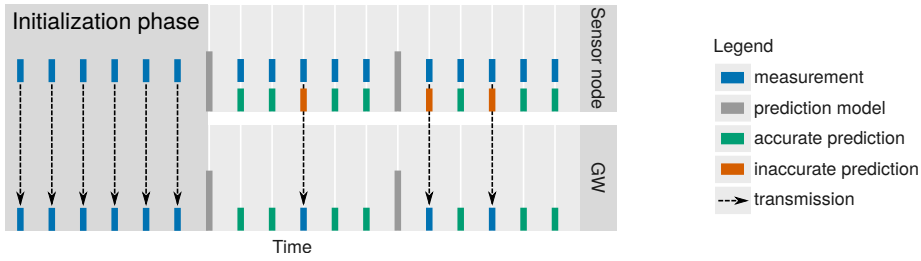



Figure 4.1: A variant of the DPS with independent model generation. The sensor node and the GW can compute the same prediction model because they are programmed to use the same data.

sensor nodes, without making any new transmission. Figure 4.1 illustrates the sensor nodes' and GW's behaviors. New prediction models can be regularly chosen based on the knowledge simultaneously available to sensor nodes and GW. As a drawback, the variety of the prediction models is restricted by the memory and computing power limitations of sensor nodes.

The Least Mean Squares (LMS) method provided accurate predictions in simulations where sensor nodes and GW generated their prediction models independently [66, 68, 69]. For instance, in a particular scenario, only $10\%$ of the measurements would be necessary to monitor room temperature accurately [66].

### 4.2.2 Model choice in sensor nodes

Alternatively, prediction models can be chosen in sensor nodes, as illustrated in Figure 4.2. As before, sensor nodes start transmitting all the measurements to the GW. However, a new responsibility is assigned to sensor nodes: after collecting local measurements and choosing a prediction model that fits the current environment, they must communicate the prediction model to the GW. The main advantage of this mechanism is that sensor nodes can decide for new prediction models using all the measured data, instead of using only the information that they share with the GW. On the other hand, sensor nodes need extra transmissions to inform

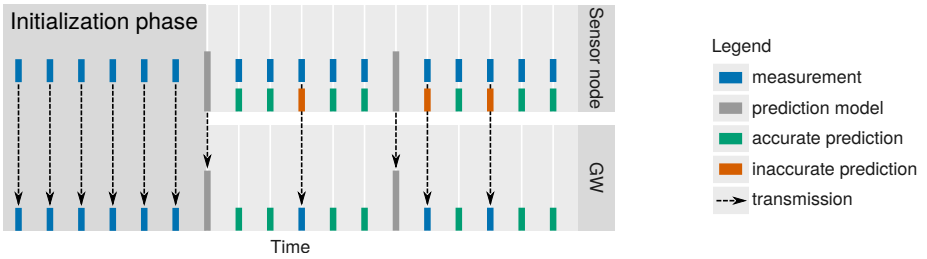



Figure 4.2: As sensor nodes can overhear their neighbors' data without overloading the network or congesting the medium, they may locally decide the best prediction method and later inform their decision to the GWs.

the GW about their decisions.

Simulation results using real data from WSNs showed that this approach could reduce the number of data transmissions using AutoRegressive (AR), AutoRegressive Integrated Moving Average (ARIMA), and Exponential Smoothing (ES) models[1] with neither exceeding the constrained memory nor the computational resources of typical wireless sensor nodes [70, 71, 72]. Alternatively, a hybrid mechanism may improve the quality of the predictions if sensor nodes have the autonomy to adopt more complex prediction methods (e.g., Artificial Neural Networks (ANNs)) when the simplest predictions (e.g., ARIMA) are inaccurate [33]. Only in the worst case, if the predictions using the most complex method also fall outside the accepted threshold, sensor nodes will have to transmit the real measurements to the GW.

### 4.2.3 Model choice in the Gateway

In this type of DPS, the GW is responsible for periodically choosing and transmitting new prediction models' parameters and error acceptance levels to sensor nodes, as shown in Figure 4.3. Generating the prediction models in the GW exploits the asymmetric computational power

[1] In Appendix B, we describe all of these methods in detail.

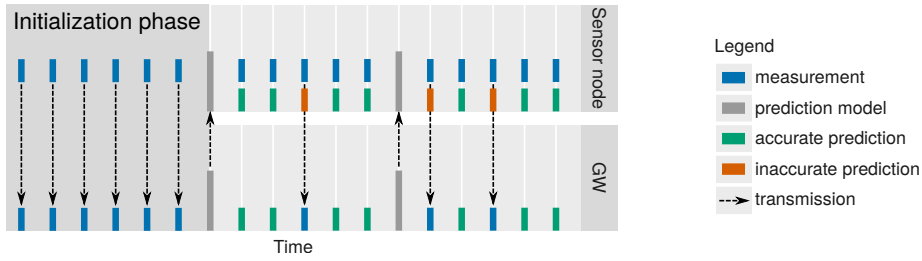



Figure 4.3: In a DPS, a measurement is only transmitted if its forecast is inaccurate. The GWs may be responsible for transmitting new prediction models every time interval after the initialization phase.

availability in WSNs: GWs usually have cheaper energy sources and more resources (such as memory and processing power) than ordinary sensor nodes that are mainly used to measure and report environmental data [73, 74]. The self-managing architecture described in Chapter 2 boosts GWs' computational power with cloud services that can analyze the collected data and have the potential to choose more accurate prediction models. For example, ANNs can provide higher accuracy than other methods, but they may not fit sensor nodes' constraints because this method requires a computation intensive training phase over a large amount of data to make accurate predictions.

Additionally, the GW can assume the responsibility of adapting sensor nodes' operations according to the potential savings that predictions may introduce. In such cases, the GW can estimate if it is worth to make predictions in sensor nodes, based on the relation between the predictions' accuracy, the correlation between measurements, and the error tolerated by the WSN's managers [75]. According to the expected gains, sensor nodes can be set to: (i) go to sleep mode without making any measurement; (ii) make measurements and transmit every measurement done; or (iii) make measurements and transmit them to the GW whenever the predictions are inaccurate.

Notably, some works implemented real sensor nodes to compare the savings using several prediction methods, such as the *Constant*, ES, and

ARIMA [32, 64]. In their particular use case, the *Constant* prediction method was the best trade-off between accuracy and energy consumption in sensor nodes.

## 4.3 Sensor network applications

We use the term sensor network to denote any set of sensor nodes that can measure environmental parameters and report them (through their neighbors if necessary) to a GW that can collect and store all the data. Such networks, however, may have different applications, measure different data types and be influenced by contrasting environmental circumstances. To categorize their characteristics and requirements, we classify sensor network applications into two broad classes according to their nature: monitoring and tracking.

Monitoring and tracking applications share the data heterogeneity. For example, numeric values may be stored as nominals, ordinals, intervals or ratios [76]. Moreover, different datasets may use different units of measurement, for example, distances can be represented in kilometers or miles, temperatures can be represented in Kelvin, Fahrenheit or Celsius, and time intervals can be represented in seconds, minutes, hours or days. In conclusion, the difference between sensor measurements does not stem only from the origin of the data, but also from its representation.

To study the efficacy provided by successfully computing accurate forecasts in several types of sensor networks, we will adopt sensed data collected in different experiments. To represent monitoring applications, we will use two different datasets obtained from real world deployments with WSNs: the *Intel* and the *Sensorscope* datasets. To represent the tracking applications, we will use the *Ball* dataset, which was synthetically created based on a model of projectile launching, and the *Running* dataset, which was collected using a Global Positioning System (GPS) monitor carried by a person while running through a city.

We highlight a particular use of sensor networks that may be incorporated in both (monitoring and tracking) application types: the event

detection. GWs and sensor nodes can detect events after analyzing the available data, but detection algorithms are tightly tied to the scenarios where they are applied and require deep domain knowledge to be designed. That is, missing or wrongly reporting an event (i.e., false negatives and false positives) have inherent costs that impact the operation of the system. Given that our goal is to find a solution that would satisfy as many applications as possible, we did not focus on the act of detecting an event, because it would imply assessing different (higher) costs generated by false positives and false negatives. Instead, we aim our attention at the data collection for general purposes, presuming that it can be adopted in a sensor network that detects events if needed.

### 4.3.1 Monitoring applications

As defined Yick et al., monitoring applications comprise mainly "indoor and outdoor environmental monitoring, health and wellness monitoring, power monitoring, inventory location monitoring, factory and process automation, and seismic and structural monitoring" [77]. Hence, in these applications, it is more common to encounter temperature, relative humidity, light, solar radiation, wind speed and soil moisture sensors, among others, that can measure environmental parameters. Moreover, data types are usually periodic, i.e., each one follows a similar pattern through the days (or weeks). As shown in Chapter 3, sensor nodes' sampling interval could be updated at a particular time of the day, based on a data analysis. However, sensor nodes usually do not have the computational power to store enough data and compute such a decision without being assisted by GWs.

#### Intel data

The *Intel* data was extracted from the same dataset used in the simulations explained in Chapter 3 [57]. This dataset has been broadly used in several works in the field to represent data that can be collected by indoor WSNs [33, 51, 52, 75, 78].

In this study, only the temperature values will be utilized. We selected five consecutive days and observed the data collected by three sensor nodes to illustrate the performance of the predictions. Two nodes were selected according to the variance in their temperature measurements, i.e., those with the lowest and the greatest variance, and the third one was randomly picked (respectively, sensor nodes $35$, $21$ and $40$). The missing values were linearly interpolated and summed to a small white noise.

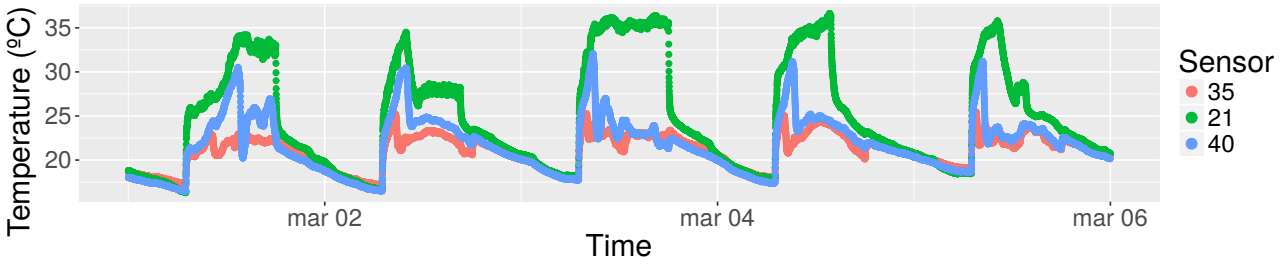
In Figure 4.4a, we can observe that the data collected by the sensors vary between $15^o$C and $37^o$C. Even though the measurements do not look very similar, it is possible to see that there is a daily pattern: their values and variances increase during the days and are more stable and similar at the beginning and the end of the days.

### Sensorscope data

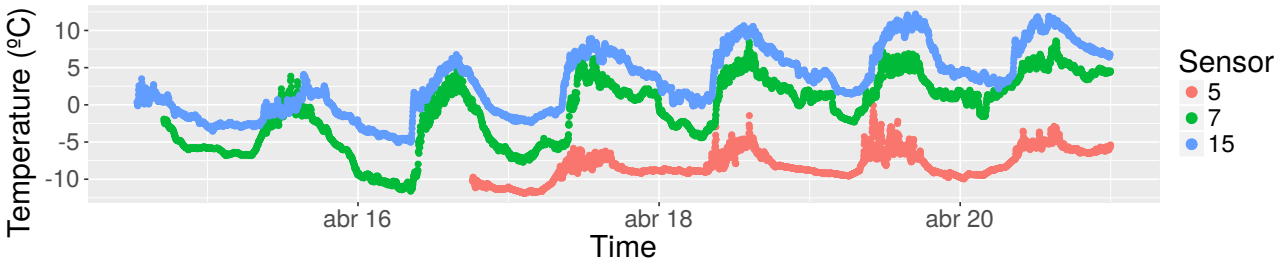
The Sensorscope data was collected by wireless sensor nodes in a deployment made on a rock glacier in Switzerland [79]. The WSN was composed of ten sensor nodes specially designed for environment monitoring. The experiment lasted five days, and each sensor node reported its measurements every two minutes, which resulted in over $3000$ reports per node, each of them including eight different measurement types: temperature, solar radiation, relative humidity, soil moisture, watermark, rain level, wind speed, and wind direction.

We will use the temperature values of three sensor nodes to illustrate the performance of the predictions. The sensor nodes selected were the ones that presented the lowest and the greatest variance in their measurements, and the last one was randomly chosen (respectively, sensor nodes $5$, $7$ and $15$), similar to how we selected the sensor nodes in the *Intel* dataset. Again, the missing values were linearly interpolated and summed to a small white noise.

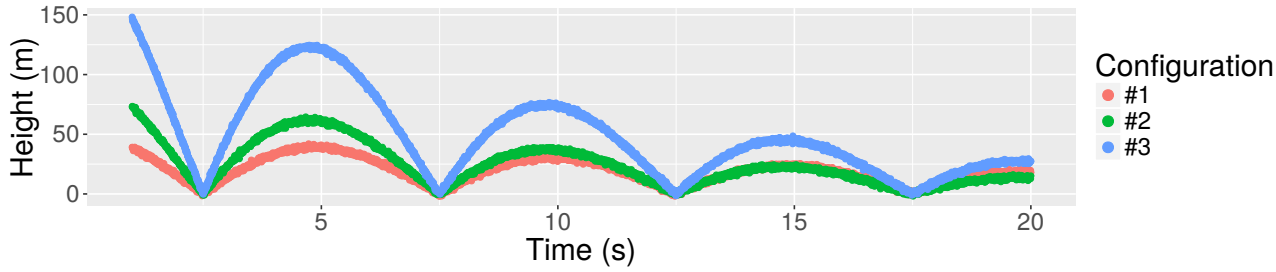
Figure 4.4b shows that, compared with the temperatures observed in the *Intel* dataset, the values are much lower (between $-12^o$C and $12^o$C), which is explained by the sensors' localization and the nature of the experiments. Moreover, there are less abrupt changes, although the presence



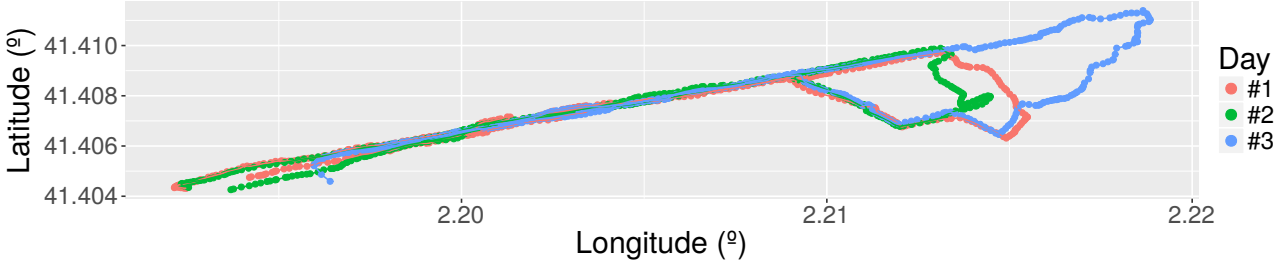

(a) *Intel*: Temperature measured by three sensors in an office [57].




(b) *Sensorscope*: Temperature measured by three sensors in a mountain [79].




(a) *Ball*: Synthetic data created to imitate bouncing objects.




(b) *Running*: GPS coordinates of a person during street runs.

Figure 4.5: Datasets used to illustrate monitoring and object tracking applications.

of the sun clearly changes the values and increases their variance during the days.

### 4.3.2 Tracking applications

Tracking applications include primarily human tracking, battlefield observation (e.g., enemy tracking), animal tracking and car tracking in smart cities. This kind of application usually requires more powerful sensors than those used in monitoring applications, such as cameras, microphones, and radio-frequency identifications (RFIDs). In many target tracking applications, the data is sensed by only one sensor node at a time, differently from monitoring applications that use several sensors to measure the environmental parameters. As a result, tracking applications are less tolerant to delays and single point of failures, and the computation in the sensor nodes is heavier because they tend to oversample the data and avoid missing variations that will eventually happen. Moreover, the data is usually processed, and only the relevant information is transmitted to the GWs [80].

#### Ball movement

The first dataset used to represent tracking applications was synthetically created to simulate an object bouncing on the floor a few times. The data is intended to simulate an object being tracked and can be thought as the vertical position of a ball that hits the floor after being dropped from a certain height. The data points were calculated using the formula of a pendulum with exponential decay [81]:

$$\theta(t) = \theta_0 \, \frac{|cos(2\pi\lambda \, t)|}{e^{\gamma \, t}} + \varepsilon_t, \tag{4.1}$$

where $\theta(t)$ is the height at time $t$, $\theta_0$ is the initial amplitude, $\lambda$ is the frequency, $\gamma$ represents the decay, and $\varepsilon_t$ is an additive zero-mean, unit variance Gaussian white noise. To reproduce different types of movements, we generated three sequences of data, each one with $3800$ data points.

Figure 4.5a shows the values of the three ball movements that compose the *Ball* dataset. Each movement has frequency $\lambda = 0.1$ hertz and values were sampled once every five milliseconds. The first set of points is based on a movement with initial amplitude $\theta_0 = 50$ meters and decay $\gamma = 0.05$. The second set of points has greater initial amplitude and decay ($\theta_0 = 100$ meters, $\gamma = 0.1$), which means that the object moves faster and results on sparser data that may be less predictable. Finally, the third set of points has a decay $\gamma = 0.1$, and the initial amplitude is $200$ meters, which means that the object is faster than before, and, therefore, the changes are more abrupt and less predictable than in the others. Besides these differences, their periodicity can be clearly noticed in the plot.

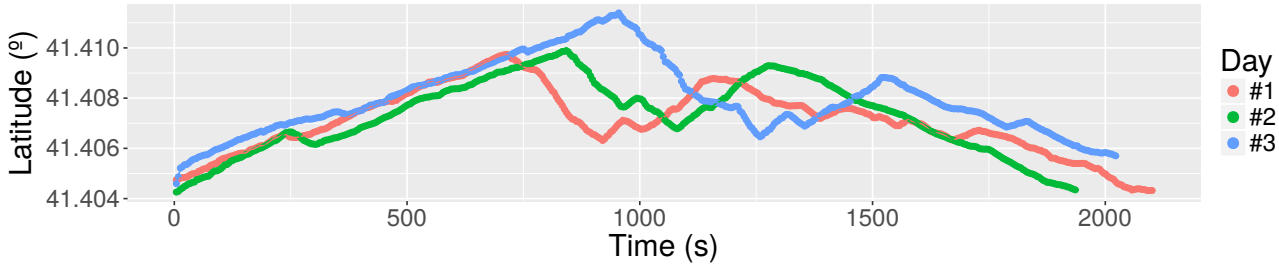
### Street runner

This dataset consists of the GPS position of a person while running across the city of Barcelona. Each position is represented by the respective latitude and longitude coordinates. The data was collected by a GPS-enabled watch taken by a person in three different days, and each observation was registered in an interval between one and five seconds after the last one, summing up to $480$ data points. Even though the trajectories are similar, the measurements contain noise and variations that are expected to be encountered in other applications for object tracking.

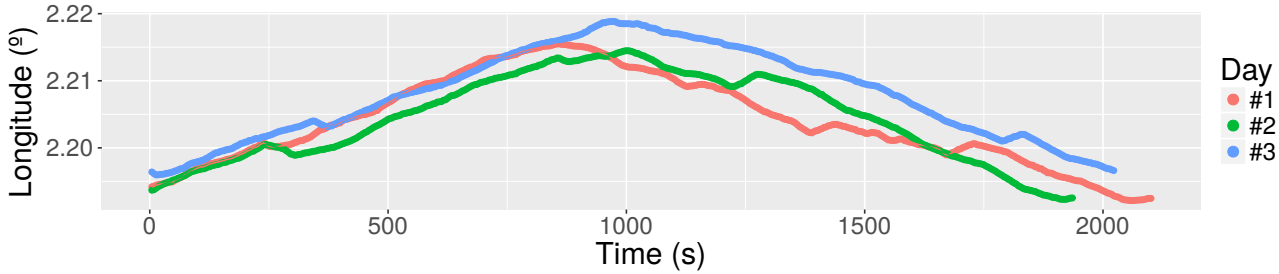
In Figure 4.5b, the *Running* dataset is shown. The different trajectories among the days are illustrated, and it is possible to observe their similarity and the absence of periodicity. As we can see in Figure 4.6a, changes in the latitude are abrupt and do not follow any pattern. On the other hand, Figure 4.6b shows that the longitude varies almost linearly in time and is more intuitive than the changes in the latitude.

## 4.4 Experimental results

The first step to validate the efficiency of forecasting methods in DPSs is to apply the different methods over splits of the data collected by real sensors in various experiments. Each split is defined by a *history* plus a




(a) *Running* (latitude): Geographic latitude of a person.




(b) *Running* (longitude): Geographic longitude of a person.

Figure 4.6: Changes in GPS coordinates over time.

*window*: the former represents the measurements made in the past and the latter the measurements to be forecast. The idea of observing different *history* and *window* lengths is to synthesize the evolution of the sensor nodes' memory and computational capacities. Hence, the experimental results illustrate the comparison between different configurations and how the sensor nodes' evolution impact the forecasts' accuracy.

### 4.4.1 Parameter study

From now, we define as a *scenario* each combination of *history* and *window* lengths observed in a *dataset*. In the experiments, for example, a scenario with *history* length $100$ and *window* length $10$ in the *Ball* dataset has been represented by $600$ splits of data randomly picked from that dataset.

Before testing each scenario and the different forecasting methods, we detail their characteristics in the following.

#### Datasets

The tests were made using all the datasets presented before: *Intel*, *Sensorscope*, *Ball* and *Running*. Each dataset is composed of three *groups* of data, i.e., *Intel* and *Sensorscope* have three different sensors each, *Ball* was constructed using three different parameter configurations and *Running* contains data from three different days. Particularly, the *Running* dataset was separated along two dimensions (latitude and longitude), and considered as two distinct sets of data.

Finally, the forecasts were made based on $200$ splits of data randomly picked from each *group*. This setup represents the data heterogeneity expected in different sensor networks.

#### Forecasting methods

In our experiments, we tested the forecasting methods that are broadly used in data applications, thanks to their scalability and reliability: the

| Method | Preprocessing time complexity | Runtime complexity | Space complexity |
|---|---|---|---|
| **Constant** | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ |
| **Linear** | $\mathcal{O}(1)$ | $\mathcal{O}(w)$ | $\mathcal{O}(1)$ |
| **SM** | $\mathcal{O}(h)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ |
| **ES** | $\mathcal{O}(k^3\ h)$ | $\mathcal{O}(w)$ | $\mathcal{O}(1)$ |
| **ARIMA**$(p,d,q)$ | $\mathcal{O}(k^3\ h^2)$ | $\mathcal{O}((p+q)w)$ | $\mathcal{O}(max(p,q+1))$ |

Table 4.1: List of forecasting methods and their complexities[3].

*Constant*, *Linear*, Simple Mean (SM), ES, ARIMA and ANN methods[2]. The computing complexities of the tested methods are summarized in Table 4.1. Usually, the complexities are given in function of the number of values used to choose a forecasting model ($h$) and the number of values that will be forecast ($w$). In some cases, other specific method parameters ($p$, $q$, and $k$) also impact the algorithms' complexity. Usually, $0 \leq p \leq 2$, $0 \leq q \leq 2$ and $0 \leq k \leq 10$.

We highlight that ES and ARIMA models use sets of parameters to forecast new values, which does not happen if the *Constant* method is adopted. As explained before, in some DPSs, the prediction models' parameters must be transmitted (from sensor nodes to the GWs, or vice-versa), and this can either be done by triggering a new transmission or simply by attaching them to a measurement transmission. Therefore, if we use ES or ARIMA to predict only one value, we cannot reduce the number of transmissions, given that the model will have to be updated after each measurement. In fact, updating the model after each measurement will also increase the energy consumption in the sensor nodes, given that their parameters require an extra computational time inexistent in the *Constant* method.

[2]In Appendix B, we describe each forecasting method in detail.

[3]ANNs are not included in Table 4.1 because they are soft computing solutions that cannot be bounded by a computational time limit [82, 83].

### History

The *history* is the set of data points used during the *initialization phase*, and its length impacts the preprocessing time complexity, i.e., the computing time required to set up the best prediction model's parameters. Thus, the simplest methods, such as the *Constant* and the *Linear*, are not affected by the *history* length, but the time spent to set up an ARIMA model increases quadratically according to the *history* length and must be considered before its adoption in sensor nodes with critical memory limitations. We examined cases in which the *history* length varied among $5$, $10$, $20$, $50$, $100$, $200$, $500$ and $1000$.

### Window

The *window* is the set of values that must be forecast. These values represent future measurements and, therefore, they are only considered to measure the predictions' accuracy. The *window* length might affect the runtime, which can be either constant or increase linearly, as shown in Table 4.1. We experimented scenarios where $1$, $5$, $10$, $20$, $50$, $100$, $200$, $500$ and $1000$ values were predicted at a time.

## 4.4.2 Effectiveness of the forecasts

To define if a transmission could be avoided using a DPS, it is necessary to annotate, firstly, which values are relevant in a scenario. Measures of predictions' accuracy are useful to compare numerical differences, but they do not provide the practical benefit that the predictions would represent in a real scenario. For example, a sensor network that is measuring the temperature in a data center might require higher precision than one placed in the mountains, because the indoor temperature may be used to control an air conditioning system that avoids damages from excessive heat.

To evaluate the effective impact of each forecasting method, we assume that if the absolute difference between the actual measurement and

the predicted value is smaller than an *acceptance threshold* $\tau_{\text{min}}$, the current measurement does not provide valuable information to the network and its transmission might be avoided. To estimate the number of transmissions that could be avoided using a DPS, we first calculate how many transmissions could be avoided if the *Constant* method were used. Then, we compare the results with the ES and ARIMA methods, which were the only methods that could forecast as accurate as the former one in our preliminary observations[4].

We highlight that inaccurate forecasts trigger the transmission of the real measurements to the GW. Then, if the *Constant* method has been adopted, each new measurement can be used to forecast new values, resetting the forecasting *window*. On the other hand, the ES and ARIMA methods may trigger (at least) one new transmission to establish the same forecasting model in the sensor node and the GW. Therefore, we made a fair comparison by focusing only on the scenarios in which the ES and ARIMA methods could reduce at least two transmissions more than the *Constant* method. To estimate the percentage of avoided transmissions, we considered that the *window* length was the maximum number of measurements that could be transmitted, which would happen if all forecasts were inaccurate.

### Results without quality loss

To observe if it is possible to reduce the number of transmissions using forecasting methods without reducing the quality of the measurements, we adopted the smallest values possible for $\tau_{\text{min}}$: the resolution of the sensors used in each experiment. A sensor's resolution is the smallest measurement that can be indicated reliably, e.g., if a temperature sensor's resolution is $0.01^o$C, it cannot precisely measure the difference between $20.001^o$C and $20.007^o$C. In this case, we could assume that any change smaller than $0.01^o$C in the temperature is never relevant; therefore, if a forecast differs by less than $0.01^o$C from the real observation, it will be considered *accurate* and will not trigger a transmission from the sensor

[4]In Appendix B, we detail our preliminary tests over the data used in this Chapter.

node to the GW.

To estimate the number of transmissions that could be avoided without affecting the data quality, we considered the following values of $\tau_{\text{min}}$ in each dataset:

- ***Intel* dataset**: $0.01^o$C. Since the specification of Mica2Dot[5] does not include the temperature sensor's resolution [84], we defined $\tau_{\text{min}}$ as the minimum difference between any pair of measurements observed in the original data. On a side note, we highlight that it is the same resolution of Sensirion SHT11 [85], a temperature sensor broadly used in several wireless sensor nodes similar to Mica2Dot [86].

- ***Sensorscope* dataset**: $0.045^o$C. TinyNode[6] uses an analog temperature sensor (LM20) that can measure from $-55^o$C to $130^o$C and the microcontroller has a 12-bit analog-to-digital converter [87]. Therefore, we calculated the sensor's resolution as $(130 - (-55)) \,/\, 2^{12}$ $^o$C.

- ***Ball* dataset**: $0.001$ meters. As a reference, we adopted the specification of the AR3000 sensor, a long range laser sensor that can accurately measure distances of up to 300 meters [88].

- ***Running* datasets**: $8.38 \cdot 10^{-8}$ degree. We defined the $\tau_{\text{min}}$ as the minimum difference between any pair of measurements observed in the data, because the sensor's specification is not publicly available. This value represents nearly $1$ centimeter of distance along the latitude direction and around $0.7$ centimeters along the longitude orientation [89].

Table 4.2 shows the percentage of absolute differences between consecutive measurements that are below or equal the $\tau_{\text{min}}$. Note that only the *Intel* and the *Sensorscope* datasets have consecutive values such that their

[5]Mica2Dot motes were used to collect the measurements in the *Intel* dataset.

[6]TinyNode motes were used to collect the measurements in the *Sensorscope* dataset.

| Dataset | Group 1 | Group 2 | Group 3 |
|---|---|---|---|
| ***Intel*** | 58.13% | 46.36% | 60.14% |
| ***Sensorscope*** | 61.36% | 33.02% | 33.78% |

Table 4.2: Percentage of absolute differences between consecutive measurements that are not greater than $\tau_{\text{min}}$ in the dataset groups used in the experiments.

absolute difference is smaller or equal than $\tau_{\text{min}}$, due to the consecutive similar measurements and the missing values filled beforehand.

Figure 4.7 shows the percentage of transmissions that could be avoided using the *Constant*, ES, and ARIMA methods. For this representation, we did not consider any extra transmission that could occasionally be required to transmit the forecasting models' parameters. However, as explained above, we included only the cases in which the ES and ARIMA methods accurately predicted, on average, at least two values more than the *Constant* method.

The results show that the *Constant* method is the best option when the *window* length is smaller than 20. Curiously, in our preliminary results, the *Constant* method was regularly less accurate than the ES and ARIMA methods when the *window* length was 1, 5 or 10. This difference illustrates the importance of analyzing the real effectiveness of the forecasts when applied to real use cases.

Considering all scenarios, the greatest improvement of a forecasting method in comparison with the *Constant* method was observed in *Sensorscope* when the *window* length was set to 20. In such cases, the number of transmissions avoided using ARIMA was 18.1% greater than using the *Constant* method, i.e., the ARIMA models could accurately forecast nearly 3.9 measurements more than the *Constant* method. In this scenario, the ARIMA method could be used to avoid an average of 6.33 transmissions every 20 measurements, which represents a reduction of 31.65% in the number of transmissions.

Regarding the *history* length, we observed that increasing the number

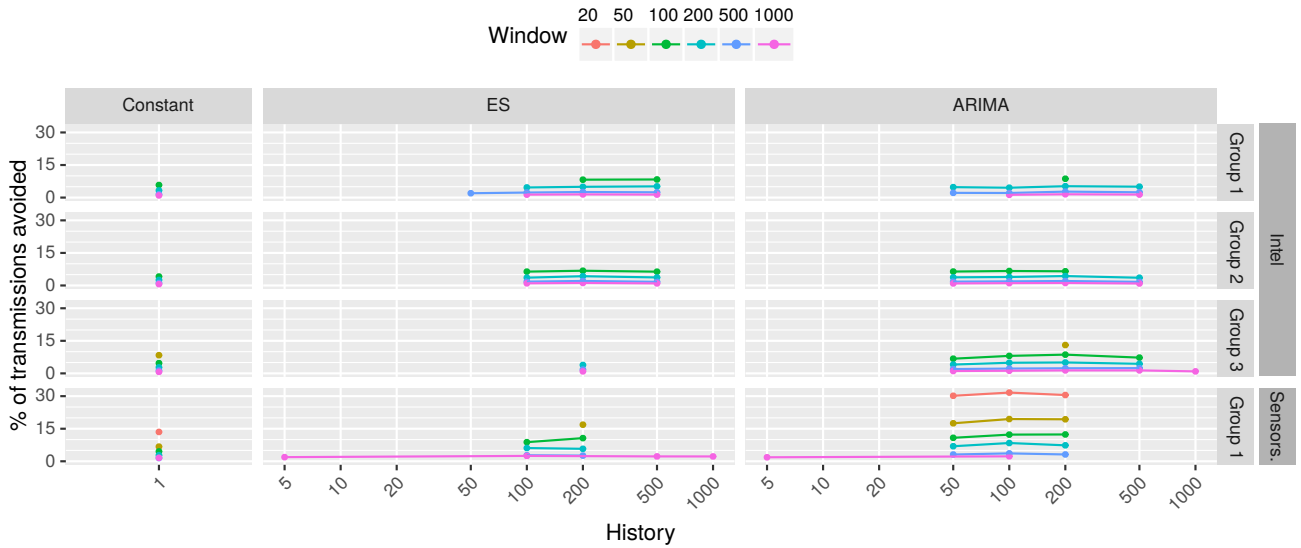



Figure 4.7: The percentage of transmissions that could be avoided using each forecasting method without reducing the quality of the measurements generated by the sensor nodes.

of values in the *history* from $100$ to $200$ increased the number of avoided transmissions in $70\%$ of the cases illustrated in the plot. In general, the ES models could reduce more transmissions when the *history* length was between $100$ and $200$, while the ARIMA models also performed well using only $50$ values. These results suggest that the ES and ARIMA methods can be more accurate and bring improvements when the *history* length is increased up to $200$ (and not to $500$ or $1000$).

As a baseline for comparison, we programmed a TelosB [36] mote running Contiki [90] without any optimization. Besides a simple application that reported temperature periodically, there was enough space to include one floating point array with length up to $1350$. Therefore, it would not be possible to configure this sensor node to forecast a scenario with *window* length $1000$ and *history* length $500$ (or vice-versa), but it would be possible to adopt *history* and *window* lengths of $200$, for example. In conclusion, based on the results observed before, a TelosB mote has enough memory and the computing power necessary to reduce its transmissions using the ES or ARIMA methods.

**Results with different sensors' resolution**

Considering the four cases illustrated in Figure 4.7, more than $45\%$ of the absolute differences between consecutive measurements were below the *acceptance threshold* (see Table 4.2). Such a coincidence suggests that if sensor nodes sample frequent enough to measure similar values, the ES and ARIMA models will be sufficiently accurate to reduce their number of transmissions.

Let us consider the hypothetical sequence of temperature values measured at periods of one minute between each other by a sensor with resolution equal to $0.1^o$C:

$$\{20.1^o\text{C}, 20.1^o\text{C}, 20.4^o\text{C}, 20.5^o\text{C}, 21.7^o\text{C}, 21.8^o\text{C}, 21.9^o\text{C}\}.$$

In this sequence, we can observe only one pair of similar consecutive values (out of six), i.e., $16.7\%$ of similar measurements.

There are two scenarios that could make this sensor measure a higher rate of similar values:

| Dataset | Group 1 | Group 2 | Group 3 |
|---|---|---|---|
| ***Intel*** | $0.0098^o$C | $0.0101953^o$C | $0.009653105^o$C |
| ***Sensorscope*** | $0.02137582^o$C | $0.0840074^o$C | $0.07867205^o$C |
| ***Ball*** | 0.9410768 meter | 0.9699227 meter | 0.9555685 meter |
| ***Running* (latitude)** | $3.35276 \cdot 10^{-5}$ degree ($\sim$ 3.72 meters) | $3.704801 \cdot 10^{-5}$ degree ($\sim$ 4.11 meters) | $3.43658 \cdot 10^{-5}$ degree ($\sim$ 3.81 meters) |
| ***Running* (longitude)** | 0.0001015048 degree ($\sim$ 8.49 meters) | 0.0001128204 degree ($\sim$ 9.43 meters) | 0.0001072884 degree ($\sim$ 8.97 meters) |

Table 4.3: New sensors' resolutions and *acceptance thresholds* set in order to have similar consecutive values in $50\%$ of the time.



| Dataset | *window* length | *history* length | Saved transmissions using the *Constant* method | Saved transmissions using another forecasting method |
|---|---|---|---|---|
| ***Intel*** | 20 | 200 | 8.59% | 20.91% using ARIMA |
| ***Sensorscope*** | 20 | 100 | 6.87% | 21.08% using ARIMA |
| ***Ball*** | 50 | 1000 | 14.95% | 43.03% using ARIMA |
| ***Running* (latitude)** | 50 | 100 | 0.58% | 5.21% using ES |
| ***Running* (longitude)** | 5 | 200 | 13.91% | 51.74% using ARIMA |

Table 4.4: Highest improvements in the percentage of saved transmissions.

1. **Reducing the sampling interval.** Assuming a linear change in the temperature, if the same sensor measured at periods of $30$ seconds, it would have been registered the sequence:

   $$\{20.1^o\text{C}, \mathbf{20.1^o\text{C}}, 20.1^o\text{C}, \mathbf{20.2^o\text{C}}, 20.4^o\text{C}, \mathbf{20.4^o\text{C}},$$
   $$20.5^o\text{C}, \mathbf{21.1^o\text{C}}, 21.7^o\text{C}, \mathbf{21.7^o\text{C}}, 21.8^o\text{C}, \mathbf{21.8^o\text{C}}, 21.9^o\text{C}\},$$

   where the highlighted values are the new measurements. In this sequence, we can observe five pairs of similar consecutive values (out of $12$), i.e., $41.6\%$ of similar measurements.

2. **Changing the sensor resolution.** If the sensor's resolution was $0.5^o$C, the original sequence (with one measurement per minute) would have been registered as:

   $$\{20.0^o\text{C}, 20.0^o\text{C}, 20.5^o\text{C}, 20.5^o\text{C}, 21.5^o\text{C}, 21.5^o\text{C}, 22.0^o\text{C}\}.$$

   In this sequence, we can observe three pairs of similar consecutive values (out of six), i.e., $50\%$ of similar measurements.

To observe the impact of the similarity between consecutive values in the number of transmissions avoided using the same forecasting methods, we changed the *acceptance thresholds* and simulated new sensors' resolutions in each *Group*. The new values (shown in Table 4.3) were chosen such that the probability of observing two similar consecutive values was $0.5$. This change is equivalent to changing the sensors' sampling interval to a value in which the measurements are the same as the last one measured in $50\%$ of the time.

After obtaining the desired scenarios, we observed the number of transmissions that would have been saved using a DPS. Table 4.4 shows the greatest reductions in the number of transmissions after setting new resolutions for the sensors. We can observe that it is possible to reduce the number of transmissions regardless of the application type, even though each dataset has a different average percentage of avoided transmissions per *window* length. In the best case, the ARIMA models could reduce more than $50\%$ of the transmissions in the *Running* (longitude) dataset,

which represents a significant improvement especially taking into consideration its potential to improve the end-to-end throughput and reduce the delays in other tracking applications.

Figures 4.8 and 4.9 include all the cases in which the ES and ARIMA models accurately predicted, on average, at least two values more than the *Constant* method. In comparison with the previous results, now it is possible to observe cases in which there is a reduction in the number of transmissions with a smaller *window* (lengths $5$ and $10$). However, similar to the previous tests, the vast majority of the improvements using ES and ARIMA needed between $50$ and $200$ values in *history*. Moreover, when the *window* length was set to $1000$, the ES and ARIMA models could reduce the number of transmissions only in the monitoring applications. This difference may have happened due to the nature of the measurements and an eventual seasonality in the observations.

## 4.5 Summary

The first generation of wireless sensor nodes (e.g., Mica2Dot, TinyNode, and TelosB motes) has constrained energy resources and computational power, which discourages applications to process any task other than measuring and transmitting towards a central server. However, nowadays, sensor networks are part of the IoT, and the software and hardware evolution may change the old strategy of avoiding data computation in the sensor nodes.

In this Chapter, we showed that it is possible to reduce the number of transmissions using forecasting methods without reducing the quality of the measurements provided by the sensors. The effectiveness of such methods, however, may vary from sensor to sensor even if the environment and the phenomena observed are the same. The decision about whether to adopt forecasting methods or not can be made based on the expected reduction in the number of transmissions. This value can be estimated after observing the sensor nodes' computing capacities and estimating the forecasts' accuracy that a method will have in the environment

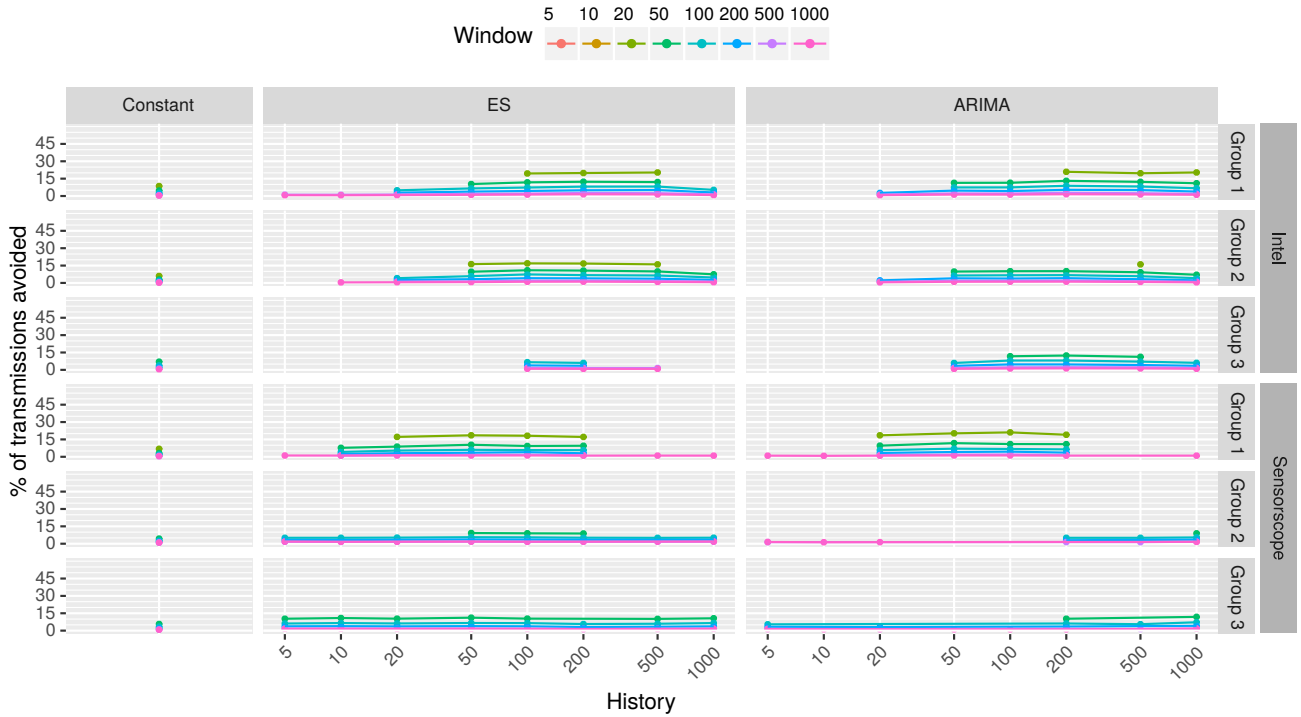



Figure 4.8: If the chances of measuring two consecutive values is $50\%$, it is possible to have a high reduction in the number of transmissions without reducing the quality of the measurements provided by the sensor network.

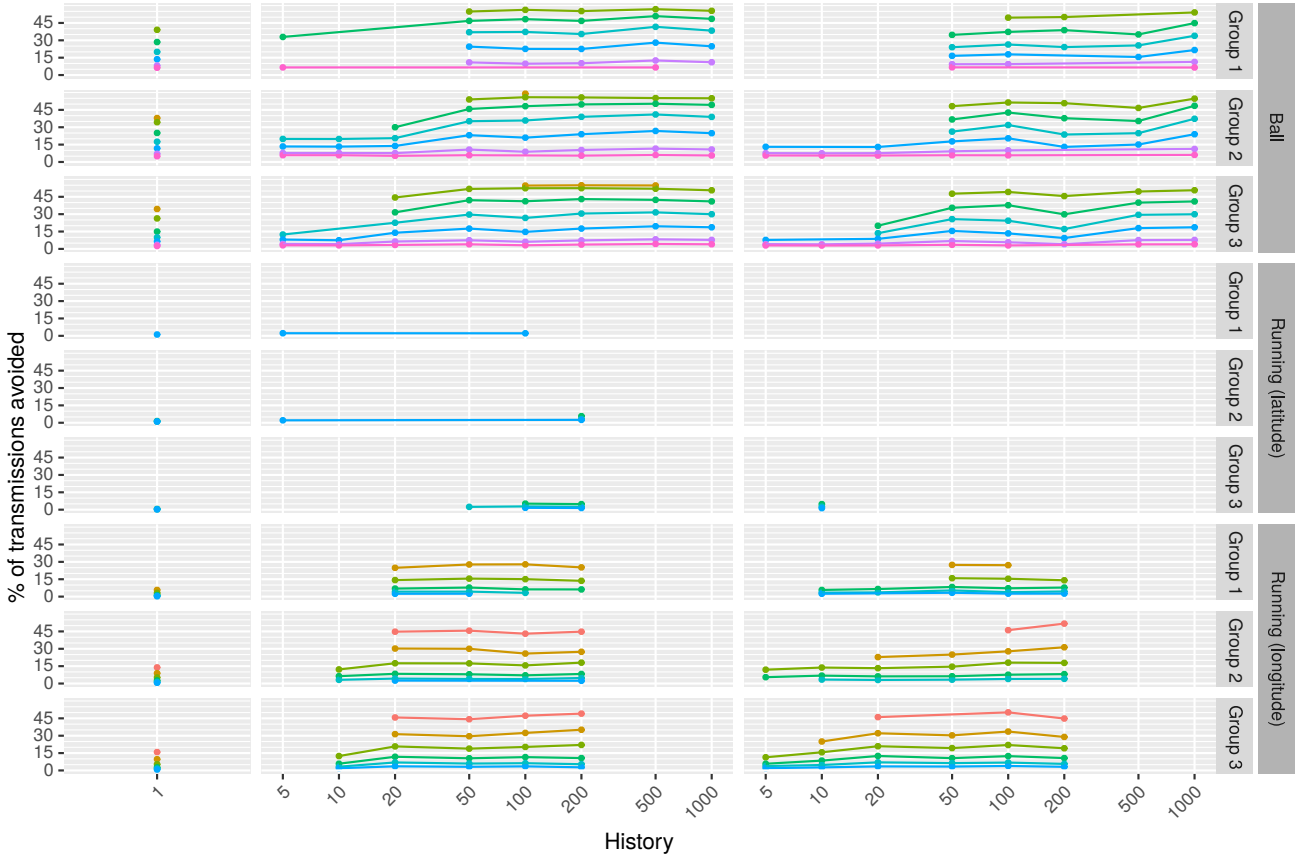



Figure 4.9: In object tracking applications, we also observed a high reduction in the number of transmissions when the chances of measuring two consecutive values is $50\%$.

under observation.

We conclude that there is an old paradigm that is no longer the most beneficial, which is the strategy of always transmitting a measurement when it differs by more than a threshold from the last one transmitted. Adopting more complex forecasting methods in DPSs is an alternative to significantly reduce the number of transmissions without compromising the quality of their measurements, and therefore support the exponential growth of the IoT. In our experiments, we could simulate the sensor nodes' hardware evolution by increasing *window* and *history* lengths. However, empowered by cloud services, GWs will be ready to produce more accurate predictions using advanced AI techniques. In the next Chapter, we make a theoretical analysis about the impact of predictions' accuracy in WSNs with dozens of sensor nodes.

# Chapter 5

# A MODEL FOR DUAL PREDICTION SCHEMES

Prediction algorithms and other AI techniques tend to become more accurate within time. To achieve higher accuracy levels, new methods compute higher amounts of data from more sources in less time. Thus, we cannot expect that designing sensor nodes with better MCUs will be sufficient to perform more accurate predictions because of their limited access to external information about the environment. Fortunately, considering the architecture proposed in Chapter 2, we can expect that GWs will attend the requirements of the most accurate prediction methods. Hence, sensor nodes can exploit such a higher accuracy if they work in collaboration with GWs in DPSs.

Based on statistical theory, we propose, in this Chapter, a WSN transmission model that will consider the predictions' (in)accuracy and the correlation between measurements made by different sensor nodes in a typical monitoring application [91]. This model will provide a reliable foundation for future (scalable) optimizations using prediction-based strategies. In the end, we will use the proposed model to evaluate the impact of DPSs in the number of transmissions in a WSN and compare our estimations with results obtained through simulations.

## 5.1 A WSN transmission model

At the application layer, sensor nodes are usually organized as sensor networks with two fundamental roles: GWs and ordinary sensor nodes. Ordinary sensor nodes are typically close to the data origin and may just perform default sensing tasks and transmit their measurements via radio to a GW. GWs are responsible for forwarding the gathered data to WSNs' managers and for disseminating occasional instructions and updates to sensor nodes.

### 5.1.1 Original model

Langendoen and Meier [92] presented a ring model for WSN topologies to describe a multi-hop network based on the average number of neighbors ($C$) of a sensor node and the number of hops from the GW to the furthest nodes ($D$). Assuming a uniform node density on the plane and defining it as $C + 1$ nodes per unit disk, the first ring ($d = 1$) is expected to have $C$ nodes. Figure 5.1 shows an example of a sensor network based on this model with $D = 3$ and $C = 5$.

In this model, the GW is always in ring zero, and transmissions made by a component (either the GW or a sensor node) can reach neighbors that are up to one unit of length from it. The set of neighbors of a sensor node $i$ is defined by all sensor nodes in the unit disk centered in $i$. The unit disk represents the sensor node's transmission range and does not necessarily imply that neighbor sensor nodes will establish a direct communication at the routing layer.

In fact, communication links are defined by underlying routing protocols. Langendoen and Meier assumed that these protocols aim to keep the smallest number of hops in a WSN and that sensor nodes only transmit to sensor nodes in the previous ring, i.e., the next ring closer to the GW. For example, to reach the GW from ring $d$, we can expect a $d$-hop transmission. Therefore, the distance from the GW also defines in which ring a sensor node is placed.

The expected number of sensor nodes $N_d$ in ring $d$ can be calculated

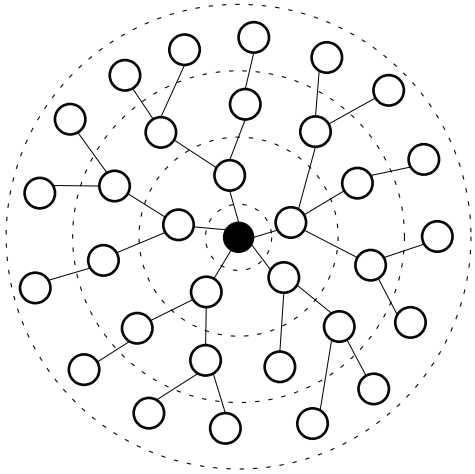


Figure 5.1: Sensor network model based on the density of the sensor nodes and their coverage. Each node has an average of five ($C = 5$) neighbors at the physical layer, and the vertices represent communication links established in an average (optimistic) scenario. The dark circle represents the GW.

based on the surface area of the annulus[1]:

$$N_d = \begin{cases} 0, & \text{if } d = 0 \\ Cd^2 - C(d-1)^2 = (2d-1)C, & \text{otherwise} \end{cases} \tag{5.1}$$

The number of nodes in the WSN is equal to $CD^2$. Given that the first ring has $C$ sensor nodes, it is expected $C$ branches starting in the GW with $D^2$ sensor nodes each. In this work, each of these branches will be referenced as a *sub-tree*.

Assuming a sensor node in ring $d$, the expected number of direct children ($I_d$) can be calculated by the relation $N_{d+1}/N_d$. This value does not

[1]The region bounded by two concentric circles.

depend on the value of $C$:

$$I_d = \begin{cases} 0, & \text{if } d = D \\ \frac{2d+1}{2d-1}, & \text{otherwise} \end{cases} \tag{5.2}$$

### 5.1.2 Model extension

We assume that the number of transmissions and receptions made by sensor nodes is the primary concern in monitoring applications, not only due to the challenges to access the medium but also due to the energy required for the external communication. Although these challenges are commonly observed in irregular real-world topologies, they are often neglected by other models, due to their complexity [51]. Having said that, we highlight that the main advantage of this model is its simplicity to identify and describe the operation of the bottlenecks in a sensor network, i.e., the sensor nodes in the first ring.

In this work, we will extend the original model and derive the number of transmissions based on the sensor nodes' positions. First, we define the set of children nodes of a sensor node $i$ in ring $d$ as $H_{i,d}$. We define the expected cardinality of $H_{i,d}$ as $K_d$. The value of $K_d$ is the number of direct children times the expected number of their children plus one (representing themselves):

$$K_d \triangleq |H_{i,d}| = \begin{cases} 0, & \text{if } d = D \\ I_d(K_{d+1} + 1), & \text{otherwise.} \end{cases} \tag{5.3}$$

Recall that the expected number of sensor nodes is $CD^2$, and the first ring is expected to have $C$ nodes. Thus, the expected number of children of the sensor nodes in the first ring (i.e., $K_1$) is $D^2 - 1$ if $D > 1$.

#### Node-to-GW transmissions

In monitoring applications, sensor nodes usually measure and transmit their data at a pre-defined time interval $(1/f)$ that can vary from few seconds to hours. The number of measurements per second $(f)$, the period

| Parameter | Description |
|---|---|
| $f$ | Number of measurements per time slot |
| $T$ | Period between the choice of two new prediction models |
| $C$ | Expected number of neighbors of each sensor node |
| $D$ | Expected number of rings/hops in the sensor network |
| $\rho$ | Correlation between measurements in a *sub-tree* |
| $\alpha$ | Expected prediction's accuracy |

Table 5.1: Parameters taken into account to calculate the number of transmissions and receptions using the model.

between predictions ($T$) and the other parameters shown in Table 5.1 may vary from case to case.

In the simplest approach, measurements are transmitted right after their creation. We will assume this behavior as the baseline for further comparisons. Alternatively, these transmissions, which we call *node-to-GW*, may not happen right after measurements' creation if sensor nodes aggregate the data received from other sensor nodes or past measurements. The impact of aggregating transmissions will also be modeled in the following.

Given that sensor nodes must forward data from their children towards the GW, the expected number of transmissions ($S_d$) during a period of $1/f$ seconds in a sensor node $i$ in ring $d$ is

$$S_{i,d} \;=\; (K_d + 1), \tag{5.4}$$

and the number of receptions is

$$R_{i,d} \;=\; K_d. \tag{5.5}$$

Finally, the total number of transmissions during a period $T$ in a sensor node $i$ is the sum of transmissions and receptions:

$$\begin{aligned} X_{i,d} &= S_d + R_d \\ &= ((K_d + 1) + K_d)\; fT \\ &= (2K_d + 1)\; fT. \end{aligned} \tag{5.6}$$

Based on (5.3), we can affirm that $K_1 > K_d$, if $d > 1$. Applying this inequality to (5.4), (5.5), and (5.6), we mathematically show that, if $d > 1$, then $S_{i,1} > S_{i,d}$, $R_{i,1} > R_{i,d}$, and $X_{i,1} > X_{i,d}$. Thus, in a homogeneous sensor network, sensor nodes in the first ring make more transmissions and are the bottlenecks that limit the number of transmissions in their *sub-trees*, according to their capacity. It shows the importance of focusing on optimizing the number of transmissions in the first ring because it can prevent new sensor nodes of being appended to the network[2].

### GW-to-node transmissions

*GW-to-node* transmissions are those initiated by the GW, for example, to change a configuration or update the software in the sensor nodes. Assuming one unicast transmission per packet, if the GW transmits a packet to every sensor node in the WSN, a sensor node $i$ in ring $d$ will receive one *GW-to-node* transmission to itself, plus $K_d$ transmissions (one per child), which must be forwarded towards their receivers. Therefore, the number of transmissions and receptions at a sensor node $i$ in ring $d$ is

$$S^*_{i,d} = K_d, \tag{5.7}$$

and

$$R^*_{i,d} = K_d + 1. \tag{5.8}$$

In this case, the number of transmissions made by the GW to a *sub-tree* is $D^2$, i.e., the number of nodes in each *sub-tree*. Therefore, a sensor node in the first ring will make $K_1 + 1 = D^2$ receptions and $K_1 = D^2 - 1$ transmissions.

## 5.2 Modeling Dual Prediction Schemes

As explained in Chapter 4, DPSs exploit the proximity of the sensor nodes to the sources of the data, avoiding unnecessary transmissions and han-

[2]In Appendix C, we use this model to detail the importance of reducing the number of transmissions in a WSN.

dling occasional sensor nodes' hardware limitations that might reduce WSNs' lifetime. A DPS has two tasks that may be executed either by GWs or by sensor nodes, namely the *prediction model choice* and the *prediction model dissemination*. The *dissemination* is the process of transmitting the prediction model either from sensor nodes to the GW or from the GW to sensor nodes.

In the following, we describe the impact of these tasks in the network load, concerning the number of transmissions. Before that, we discuss the assumptions and limitations of this model.

### 5.2.1 Assumptions and limitations

In this model, we assume that the quality of the measurements provided by a WSN can be scaled as "acceptable" if the values at the GW do not differ by more than a certain threshold. Since sensor nodes can compare their predictions with real measurements locally (without making any transmission), no transmission will be required if a prediction is accurate, i.e., it does not differ by more than an accepted threshold from the measured value.

In some cases, WSNs' managers have no information about the statistics of the data that is going to be retrieved by the sensor nodes. Thus, it may be necessary a long *initialization phase* before starting to make predictions. For example, schemes that use advanced prediction methods, like ANNs, require larger amounts of data to find stable models, due to their high complexity and the vast number of parameters to estimate [93]. We do not include this phase in this model because we assume that the GW has no energy constraints.

Finally, in this work, we do not expect distributed algorithms, i.e., sensor nodes do not have to synchronize with their neighbors. However, this can be easily extended from our model, given the number of expected neighbors of each sensor node.

### 5.2.2 Prediction model choice and dissemination

In a DPS, the same prediction model is shared between a sensor node and the GW. Each sensor node (or group of sensor nodes) has its prediction model, and the prediction models in a WSN can be independently chosen by both (sensor nodes and GW) without making any new transmission. Alternatively, prediction models can be chosen in the GW or in sensor nodes. In case that prediction models are chosen in sensor nodes, the GW must receive the parameter values and, in some cases, also the prediction method selected. On the other hand, if the GW is responsible for choosing prediction models, sensor nodes must be informed about the decisions taken.

Assuming that the *dissemination* of a prediction model is made through a unicast transmission, sensor nodes in the first ring will receive and forward every transmission to their children towards the proper destinations. Thus, if the GW is responsible for generating the prediction models, the sensor nodes in the first ring will have to forward the transmissions to their children. In such cases, a sensor node in the first ring will receive $D^2$ packets. From these packets, $D^2 - 1$ will be forwarded to its children. Therefore, to disseminate the prediction models generated in the GW, there will be

$$\begin{aligned} X_{\text{DIS-GW}} &= R^*_{i,1} + S^*_{i,1} \\ &= D^2 + (D^2 - 1) \\ &= 2D^2 - 1 \end{aligned} \tag{5.9}$$

transmissions (including receptions) in each sensor node in the first ring.

Analogously, in case that prediction models are chosen in the sensor nodes, every sensor node in the first ring will make $D^2$ transmissions to the GW after receiving $D^2 - 1$ prediction models. Thus, the number of transmissions at the first ring will be, once again, equal to $2D^2 - 1$.

If packets to the same *sub-tree* are aggregated, sensor nodes in the first ring will receive only one packet that will be split before being retransmitted to their direct children in the second ring. In such cases, a sensor node in the first ring will need

$$X_{\text{DIS-GW-AGG}} = X_{\text{DIS-SN-AGG}} = I_1 + 1 \tag{5.10}$$

transmissions to *disseminate* the prediction models, where, from (5.2), $I_1 = 3$, if $D > 1$.

Finally, if the GW uses broadcast (or multicast) transmissions, sensor nodes will receive and forward only one packet, i.e.,

$$X_{\text{DIS-GW-BC}} = 1. \tag{5.11}$$

### 5.2.3 Impact of predictions in the number of transmissions

As described before, adopting a data prediction scheme can benefit the WSN reducing the number of transmissions and optimizing the medium access control, which may eventually reduce energy consumption and extend the WSN's lifetime. To estimate the number of transmissions in homogeneous networks, we develop a formula based on the predictions' accuracy and the correlation of the monitored data.

Let us assume that $\alpha_i$ is the accuracy of the predictions in sensor node $i$, i.e., $\alpha_i$ is the probability that a measurement of $i$ matches to the prediction and does not have to be transmitted to the GW, and $\alpha_i^c = 1 - \alpha_i$. Therefore, the expected number of transmissions and receptions in a sensor node $i$ during a time interval of $1/f$ seconds (i.e., between two measurements) is respectively represented by $S'_{i,d}$ and $R'_{i,d}$ as

$$S'_{i,d} = \alpha_i^c + \sum_{j \in H_{i,d}} \alpha_j^c, \tag{5.12}$$

and

$$R'_{i,d} = \sum_{j \in H_{i,d}} \alpha_j^c. \tag{5.13}$$

Considering an eventual *dissemination* of the prediction models, the expected number of transmissions and receptions during a period of $T$ seconds is

$$
\begin{aligned}
X'_{i,d} &= (S'_{i,d} + R'_{i,d})fT + X_{\text{DIS}} \\
&= (\alpha_i^c + \sum_{j \in H_{i,d}} \alpha_j^c + \sum_{j \in H_{i,d}} \alpha_i^c)fT + X_{\text{DIS}} \\
&= (\alpha_i^c + 2 \sum_{j \in H_{i,d}} \alpha_j^c)fT + X_{\text{DIS}}
\end{aligned}
\tag{5.14}
$$

Note that a low accuracy in predictions used in sensor nodes that are far from the GW has a higher impact on the total number of WSN transmissions than a low accuracy in predictions used in sensor nodes in the first rings. However, concerning the number of transmissions at a single sensor node, the bottleneck of the WSN is still represented by the sensor nodes in the first ring.

Let us define the minimum average accuracy ($\alpha_{\text{min}}$) necessary to reduce the number of transmissions, according to the size of the network and its number of rings. This value can be used to define the maximum number of transmissions ($S'_{i,d,\text{max}}$) and receptions ($R'_{i,d,\text{max}}$) in a sensor node $i$ in ring $d$:

$$
\begin{aligned}
S'_{i,d,\text{max}} &= (1 - \alpha_{\text{min}}) + \sum_{j \in H_{i,d}} (1 - \alpha_{\text{min}}) \\
&= (1 + K_d)\,(1 - \alpha_{\text{min}})
\end{aligned}
\tag{5.15}
$$

and

$$
\begin{aligned}
R'_{i,d,\text{max}} &= \sum_{j \in H_{i,d}} (1 - \alpha_{\text{min}}) \\
&= K_d\,(1 - \alpha_{\text{min}})
\end{aligned}
\tag{5.16}
$$

Recall that $K_d \triangleq |H_{i,d}|$, for a sensor node $i$ in ring $d$. Therefore,

$$
X'_{i,d} \leq ((1 + K_d)\,(1 - \alpha_{\text{min}}) + K_d\,(1 - \alpha_{\text{min}}))fT + X_{\text{DIS}} \tag{5.17}
$$

Finally, the use of predictions will reduce the number of transmissions if $X'_{i,d} < X_{i,d}$. After some mathematical development shown in Appendix D, we arrive at the following expression for the minimum average accuracy of the predictions:

$$\alpha_{\text{min}} > \frac{X_{\text{DIS}}}{(2D^2 - 1)fT} \tag{5.18}$$

In conclusion, if prediction models are not independently generated in GWs and sensor nodes, the DPS requires a minimum accuracy to ensure the reduction in the number of transmissions. Thus, extra transmissions used to disseminate new prediction models may turn the prediction scheme into an inefficient option. Hence, the efficiency of a DPS also depends on how many transmissions are required for disseminating the prediction models because the number of transmissions will be proportional to the number of hops between sensor nodes and the GW.

Moreover, the minimum accuracy is a lower bound that depends only on the network layout (i.e., the number of rings $D$), the frequency of the measurements ($f$) and the time between two predictions ($T$). Therefore, if the predictions' accuracy does not reach this limit, there will exist three actions to improve the network operation, either to set new values for $f$ and $T$, to adopt a DPS with independent model generation, or to turn the DPS off.

### 5.2.4 Impact of predictions and aggregations

Additionally to DPSs, it may be possible to adopt aggregation schemes in sensor nodes, such that a sensor node aggregates data received from its children and transmits only after making its measurement. In the following, we model an aggregation scheme and compare its efficiency with the use of DPSs. Finally, we estimate the reduction in the number of transmissions if both techniques are simultaneously adopted.

To make it clear for the reader, we introduce a scenario with only two sensor nodes to clarify the normalization of the data and its application. Later, we will extend the model to a more complex scenario with $D$ rings.

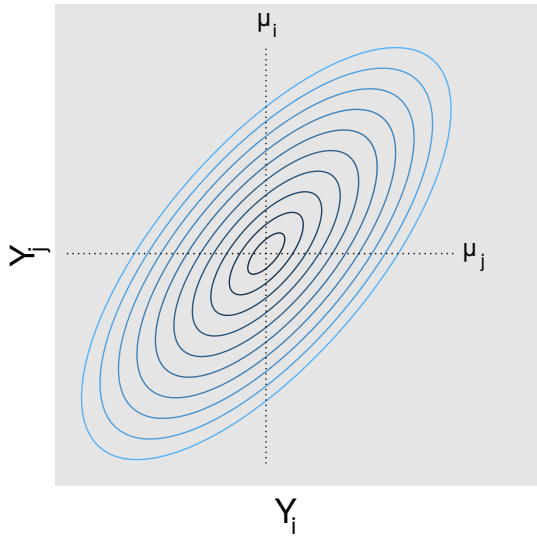



Figure 5.2: Values of $Y_i$ and $Y_j$ are correlated ($\rho_{i,j} = 0.7$), and each line represents a different density of points.

### Network with two sensor nodes

Let us consider a section of the sensor network with the GW and a sensor node $i$ with a single child $j$. Due to the sensor network's topology, transmissions from sensor node $j$ can reach the GW only through $i$. Thus, every $1/f$ seconds, $i$ may transmit to the GW if its prediction has failed or if it had happened to $j$.

We assume that the measurements of $i$ and $j$ follow the Normal distributions respectively represented by $Y_i = N(\mu_i, \sigma_i^2)$ and $Y_j = N(\mu_j, \sigma_j^2)$. Such a Multivariate Normal (MVN) distribution can be defined based on the correlation between their values, i.e., the relationship between each pair of measurements made by $i$ and $j$. An illustration of the MVN density containing $Y_i$ and $Y_j$ is shown in Figure 5.2.

Assuming that the predictions ($\bar{y}$) are not biased (i.e., $\bar{y} = \mu$), we may also approximate them to Normal distributions[3] and label an outcome as

[3]In Appendix E, we detail how to estimate the predictions' accuracy for normally

incorrect whenever a measurement falls outside the interval defined by the accepted threshold $\varepsilon$. In such cases, the probability that the sensor node $j$ will transmit (including its measurement) after $1/f$ seconds is $1 - \alpha_j$. Hence, the probability of $i$ receiving a packet is also $1 - \alpha_j$.

Similarly, $i$ will transmit if the prediction about its measurement fails (i.e., it falls out of the accepted threshold $\varepsilon_i$) or if the prediction in sensor node $j$ had failed and a measurement has been received. In other words, there will be a transmission if at least one of the two predictions fail.

If $i$ can aggregate transmissions, its total number of transmissions is not a simple sum as in the case without aggregation because it depends on the correlation of the measurements of $i$ and $j$. Let us assume that the correlation between $Y_i$ and $Y_j$ is defined by the Pearson correlation coefficient and represented by $\rho_{i,j}$. Therefore, to model the probability of having at least one wrong prediction, we must calculate the correlation matrix ($\Sigma$), which is defined as

$$\Sigma = \begin{bmatrix} \sigma_i^2 & \rho_{i,j}\ \sigma_i\sigma_j \\ \rho_{i,j}\ \sigma_i\sigma_j & \sigma_j^2 \end{bmatrix} \tag{5.19}$$

Finally, given the lower limits

$$l_i = \bar{y}_i - \varepsilon_i \text{ and } l_j = \bar{y}_j - \varepsilon_j, \tag{5.20}$$

the upper limits

$$u_i = \bar{y}_i + \varepsilon_i \text{ and } u_j = \bar{y}_j + \varepsilon_j, \tag{5.21}$$

and the correlation matrix ($\Sigma$), it is possible to calculate the following MVN probability:

$$F(y_i, y_j) = \frac{1}{\sqrt{|\Sigma|(2\pi)^2}} \int_{l_i}^{u_i} \int_{l_j}^{u_j} e^{\left(-\frac{1}{2}\theta^t \Sigma^{-1}\theta\right)} d\theta \tag{5.22}$$

The value of $F(y_i, y_j)$ represents the probability that both predictions (in $i$ and $j$) are correct and can be illustrated by the density inside the crosshatched rectangle in Figure 5.3. Thus, the probability of at least one

distributed measurements, based on the user's accepted threshold for errors.

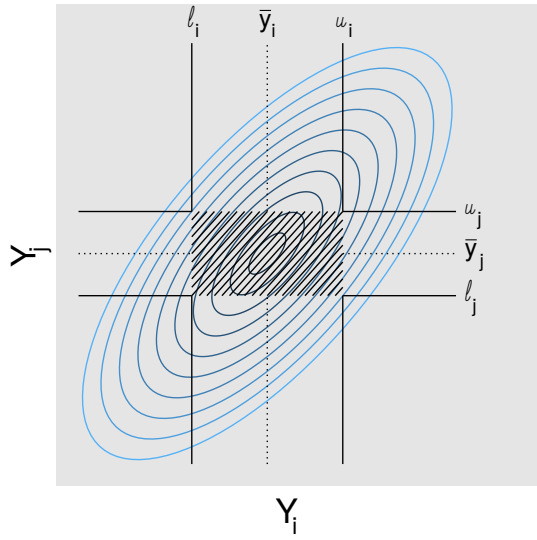



Figure 5.3: The hashed rectangle in the center illustrates the points in which both predictions ($\bar{y}_i$ and $\bar{y}_j$) are correct.

prediction failing can be calculated as $(1 - F(y_i, y_j))$, which, in fact, is the probability of sensor node $i$ making a transmission after $1/f$ seconds.

Finally, considering occasional extra transmissions to disseminate the prediction model, the number of transmissions at $j$ during a period of $T$ seconds can be calculated as $(1 - \alpha_j)fT + X_{\text{DIS}}$, and the expected sum of transmissions and receptions at sensor node $i$ during a period of $T$ seconds can be modeled as $((1 - F(y_i, y_j)) + (1 - \alpha_j))fT + X_{\text{DIS}}$.

### Larger networks

Now, we will extend the previous example to larger sensor networks. The correlation matrix ($\Sigma$) of several data distributions can be calculated as

$$\Sigma = \begin{bmatrix} \sigma_a^2 & \rho_{a,b}\,\sigma_a\sigma_b & \cdots & \rho_{a,z}\,\sigma_a\sigma_z \\ \rho_{b,a}\,\sigma_b\sigma_a & \sigma_b^2 & \cdots & \rho_{b,z}\,\sigma_b\sigma_z \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{z,a}\,\sigma_z\sigma_a & \rho_{z,b}\,\sigma_z\sigma_b & \cdots & \sigma_z^2 \end{bmatrix}, \tag{5.23}$$

and, similarly to the two-dimensional model, the expected number of transmissions made by sensor node $i$ in ring $d$ (represented by $S''_{i,d}$) depends not only on its predictions but also on the predictions used in all of its children. The value of $S''_{i,d}$ can be calculated as

$$S''_{i,d} = 1 - F(i, a, b, \ldots, z), \tag{5.24}$$

where $\{a, b, \ldots, z \in H_{i,d}\}$, and the function $F$ is the MVN probability function integrated from the lower accepted limits to the upper accepted limits over the $k = 1 + K_d$ distributions:

$$F(i, a, b, \ldots, z) = \frac{1}{\sqrt{|\Sigma|(2\pi)^k}} \int_{l_i}^{u_i} \int_{l_a}^{u_a} \int_{l_b}^{u_b} \cdots \int_{l_z}^{u_z} e^{\left(-\frac{1}{2}\theta^t \Sigma^{-1}\theta\right)} d\theta, \tag{5.25}$$

which can be efficiently calculated with the use of Monte Carlo methods for higher dimensions [94].

The number of receptions at $i$ ($R''_{i,d}$) is slightly different from the previous example, since now the sensor node may have several children in the next ring, and their transmissions happen independently. Let us define $H'_{i,d}$ as the set of direct children of $i$. Thus, $|H'_{i,d}| \triangleq I_d$. The expected number of receptions can be calculated as

$$R''_{i,d} = \sum_{j \in H'_{i,d}} S''_{j,d+1}, \tag{5.26}$$

and the total number of transmissions and receptions is defined by

$$X''_{i,d} \;=\; (S''_{i,d} + R''_{i,d}) f\, T + X_{\text{DIS}}. \tag{5.27}$$

Even though the function $F$ has no closed formula, it is possible to set a lower bound based on a case when there is absolutely no correlation between the values measured by $i$ and its children. When the correlation is equal to zero, the expected number of transmissions and receptions at sensor node $i$ are the maximum possible. Considering that there will exist a transmission if at least one prediction fails, the probability of having no transmissions at $i$ is $\alpha^{1+K_d}$. Thus,

$$S''_{i,d,\max} = 1 - \alpha^{1+K_d}. \tag{5.28}$$

Recall that $i$ is expected to have $I_d$ direct children and each child be part of a *sub-tree* with $K_d/I_d$ sensor nodes. There may exist $I_d$ independent receptions at $i$, and each reception may not occur with probability $\alpha^{K_d/I_d}$. Thus,

$$R''_{i,d,\max} = I_d\,(1 - \alpha^{K_d/I_d}). \tag{5.29}$$

Therefore,

$$X''_{i,d} \leq\ [\,(1 - \alpha^{1+K_d}) + I_d\,(1 - \alpha^{K_d/I_d})\,]f\,T\ \ + X_{\text{DIS}}. \tag{5.30}$$

We claim that $X''_{i,d} \leq X'_{i,d}$, which means that a mechanism that aggregates the data will not make more transmissions than the one that only makes predictions. Comparing (5.30) with (5.17), we have that for any $\alpha \in [0,1]$ and $K_d \geq 0$, it can be shown[4] that $(1 - \alpha^{1+K_d}) \leq ((1 + K_d)(1 - \alpha))$ and, hence, $S''_{i,d,\max} \leq S'_{i,d,\max}$. Moreover, $R''_{i,d,\max} \leq R'_{i,d,\max}$ and $I_d\,(1 - \alpha^{K_d/I_d}) \leq K_d\,(1 - \alpha)$, which can be similarly proved to be true, since $(K_d/I_d) \geq 1$ when $K_d > 0$ and $\alpha \in [0,1]$. In case of being in the last ring, since there are no children ($K_d = I_d = 0$), no reception is made.

## 5.3 Model experimentation

Using the model presented before, we can estimate the effects of adopting a prediction or an aggregation scheme in a sensor network, concerning the

[4]Based on the proof detailed in Appendix F.

number of transmissions and, eventually, the energy consumption levels. In this Section, we make a parameter study over the model parameters $C$, $D$, $\rho$, and $\alpha$. Our goal is to observe how the number of transmissions varies in different scenarios and validate the model using simulations with normally distributed data.

### 5.3.1 Simulation setup

In OMNET++ [59], we simulated TelosB motes [36] using a TDMA-based MAC protocol. In the MAC protocol adopted, each sensor node has a reserved slot to transmit. Therefore, we did not experience collisions during the transmissions, and there was no overhearing. We highlight that other MAC protocols may obtain different results, due to concurrent transmissions, although we can expect a similar reduction in their number of transmissions.

Regarding the mechanisms adopted to reduce the number of transmissions, we simulated three combinations: (i) with no prediction and no aggregation; (ii) with prediction, but no aggregation; and (iii) with aggregation, but no prediction. When data prediction was adopted, the prediction models were chosen in the GW, and *GW-to-node* transmissions were always aggregated.

As we showed before, in monitoring applications with DPSs, the number of transmissions is highly affected by the correlation between measurements made by the sensor nodes in a *sub-tree*, and by the predictions' accuracy. Therefore, regarding the model parameters, we observed the impact of different values of $\rho$, $\alpha$. Values of $\rho$ varied among $0.1, 0.2, \ldots, 0.9,$ and $0.95$, and values $\alpha$ varied among $0.5, 0.7, 0.9,$ and $0.95$.

Recall that, according to (5.30), the number of transmissions does not depend on the density of sensor nodes ($C$), but on the number of rings ($D$). Thus, to observe the impact of the growth in the number of sensor nodes in WSNs, we observed the number of transmissions with values of $D$ varying among $1, 2, \ldots,$ and $10$. Note that, when new rings are added, the number of sensor nodes increases quadratically if no aggregation is

**Data**: $n =$ number of nodes, $\alpha =$ accuracy, $\rho =$ correlation
**Result**: $P(n, \alpha, \rho) =$ probability that no transmission happens

**if** $n = 0$ **then**
  **return** $P \leftarrow 1$
**else**
  $q \leftarrow \mid \Phi^{-1}\left(\frac{1-\alpha}{2}\right) \mid$
  $Q \leftarrow \{q, q, \ldots, q\}_{1\times n}$
  $Y \leftarrow \{Y_1, Y_2, \ldots, Y_n\}$
  $$\Sigma \leftarrow \begin{bmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{bmatrix}_{n\times n}$$
  **return** $P \leftarrow \Phi\left(Y, \Sigma, Q\right)$
**end**

**Algorithm 1:** Algorithm to calculate the probability that no transmission will be made.

adopted. However, the number of transmissions does not change if sensor nodes aggregate them[5].

Finally, in our simulations, sensor nodes made one measurement per minute ($f = 1/60$), and the GW predicted their measurements once a day during three days ($T = 3 \times 86400$ seconds). Therefore, each sensor node made $4320$ measurements, from which $1440$ happened between each prediction model choice (in the cases when predictions were adopted).

### 5.3.2 Simulated algorithm

Assuming normally distributed values, the expected number of transmissions and receptions can be estimated using the cumulative density functions of MVN distributions. Based on (5.25), we designed the algorithm

[5]In Appendix C, we show that, if the aggregation is not adopted, the number of transmissions increases cubically.

described in Algorithm 1. It calculates the probability $P$ of making no transmissions in a group of $n$ sensor nodes measuring data with correlation $\rho$ if the average predictions' accuracy is $\alpha$.

We highlight that, in our model, the number of children is used to define how many distributions will be used, which means that decimal values cannot be considered. Hence, we rounded all of them up to the next integer, which resulted on an upper bound for the number of transmissions in the simulations.

### 5.3.3 Number of transmissions

Given that the bottlenecks of a sensor network are the sensor nodes in the first ring, we calculate the number of transmissions at a sensor node $i$ in ring $d = 1$ as

$$S''_{i,1} = ( \, ( \, (1 - P(K_1, \alpha, \rho) \, ) \, f \, ) + I_1 \, ) \, T, \tag{5.31}$$

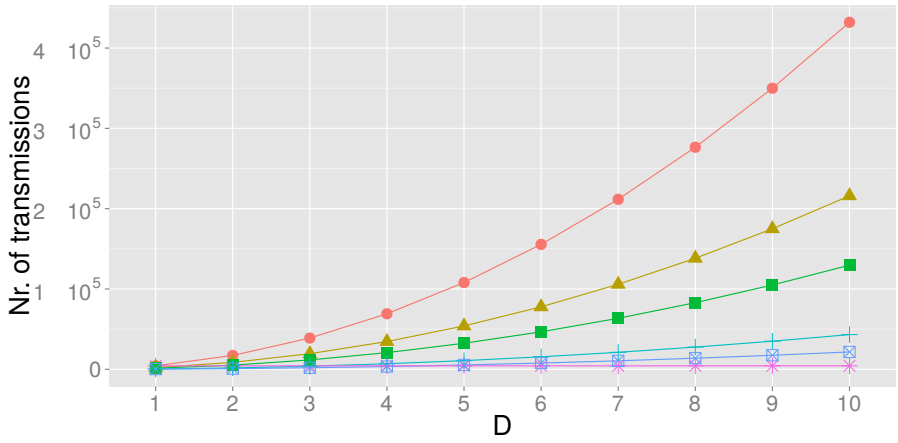
and the number of receptions as

$$R''_{i,1} = (1 - P(K_1, \alpha, \rho) \, ) \, I_1 \, f \, T. \tag{5.32}$$

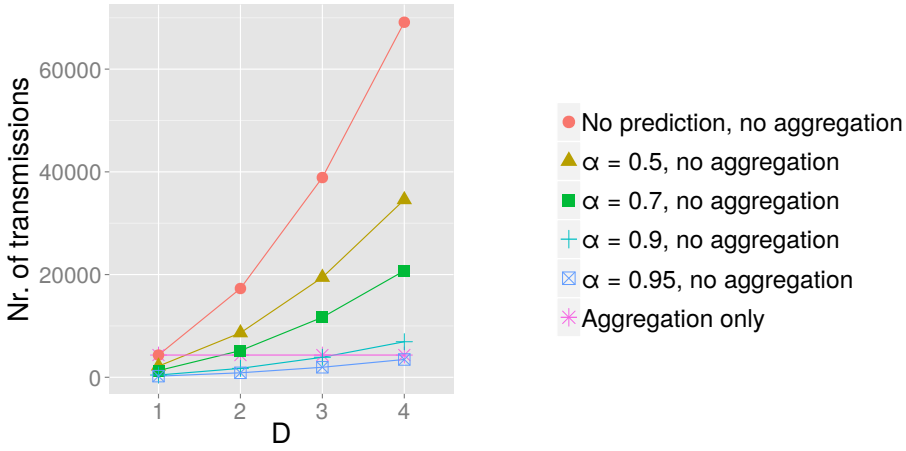
Figure 5.4 shows the results for all the tested configurations. In larger sensor networks ($D > 4$), data aggregation has a higher impact than data prediction in the number of transmissions, as shown in Figure 5.4a. Similar results were observed in another study [66], but the authors did not realize that the predictions had less impact in the final savings and concluded that such optimal achievements happened due to the high accuracy of the predictions.

When the predictions are highly accurate, and the number of rings is small ($D \leq 4$), the data prediction has a higher impact on the number of transmissions if compared with the scenarios where the data is only aggregated. Figure 5.4b highlights scenarios with less than five rings.

To detail the power of the prediction and aggregation schemes, we considered a sensor network with five rings in which the aggregation scheme produces nearly the same number of transmissions observed in




(a) The aggregation reduces the number of transmissions from quadratic to linear order.




(b) When number of rings is small ($D \leq 4$), the use of predictions can lead to fewer transmissions than the aggregation scheme.

Figure 5.4: The impact of the network size in the number of transmissions in the first ring.

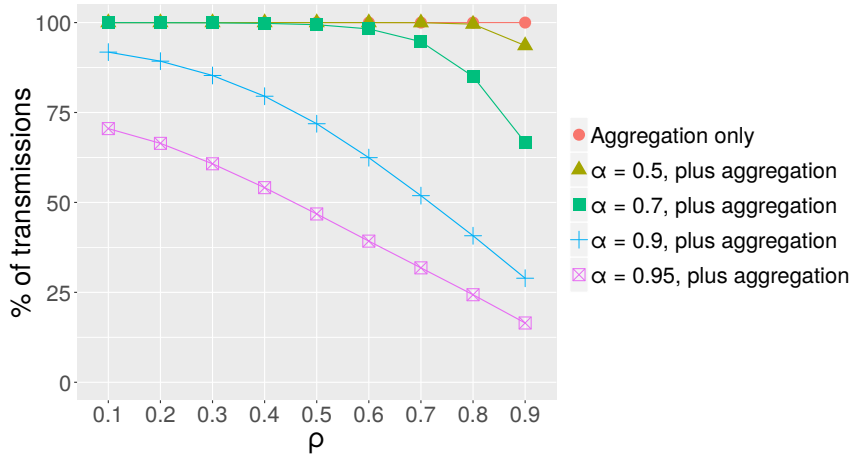



Figure 5.5: The effectiveness of the aggregations depend on the correlation between the measurements in a *sub-tree*.

the scenario with the most accurate predictions. Figure 5.5 highlights the gains obtained by adopting both schemes. First, we can observe that the number of transmissions can be reduced to $15\%$ of its maximum in the best scenario, where the predictions are highly accurate, and the measurements in the *sub-tree* are highly correlated. Additionally, we did not observe any significant gains when the predictions were less accurate (around $0.5$) nor when the predictions were more accurate (around $0.7$), and the correlations were less than $0.7$. Finally, with an average correlation ($0.5$), increasing the accuracy from $0.5$ to $0.9$ reduced by $30\%$ the number of transmissions. Meanwhile, with an accuracy of $0.5$, increasing the correlation from $0.5$ to $0.9$ reduced only by $6.5\%$ the number of transmissions, which illustrates that the impact of making accurate predictions is much higher than having a high correlation between the measurements.

### 5.3.4 Energy consumption

Based on the number of transmissions and receptions, we can use the model presented before to estimate the total energy consumption of a sen-

sor node $i$ in ring $d$ as

$$E''_{i,d} = S''_{i,d}E_{\text{TX}} + R''_{i,d}E_{\text{RX}} + E_{\text{DIS}} + E_{\text{MIN}}, \tag{5.33}$$

where $E_{\text{TX}}$ and $E_{\text{RX}}$ are the extra energy consumption to respectively transmit and receive one packet, $E_{\text{MIN}}$ is the minimum energy necessary to keep sensor nodes working without transmitting and receiving anything, and $E_{\text{DIS}}$ depends on where the prediction models are chosen:

$$E_{\text{DIS}} = \begin{cases} 0, & \text{if they are independently chosen} \\ E_{\text{RX}} + I_d E_{\text{TX}}, & \text{if chosen in the GW and} \\ I_d E_{\text{RX}} + E_{\text{TX}}, & \text{if chosen in sensor nodes.} \end{cases} \tag{5.34}$$

To illustrate the applicability of this model, we estimated the energy consumption in a WSN after three days of operation and compared with the results obtained in our simulations. For this estimation, we considered a homogeneous sensor network with $D = 5$ and $C = 3$ (i.e., 75 sensor nodes plus the GW). To obtain the values of $E_{\text{TX}}$, $E_{\text{RX}}$, and $E_{\text{MIN}}$, we simulated three TelosB motes in OMNET++ transmitting and receiving data without making any predictions. After one simulated day, we calculated the average values for each parameter.

So far, we did neither distinguish delays nor packet lengths used in aggregated transmissions and receptions from the case without aggregation. In fact, in a real implementation, these transmissions could be done in the same packet types if we adopted simple aggregation functions, such as the maximum, minimum and the average of the measurements. However, larger packets would mean higher energy consumption to transmit and receive, in comparison with the non-aggregated transmissions. Therefore, to show the extensibility of our model, we used packets with eight times the payload of the normal packets in the aggregated transmissions.

To illustrate the results, we focused on the energy consumption of a sensor node in the first ring. As sensor nodes in the first ring must handle the highest number of transmissions, they consume more energy than the others. As a consequence of such an energy consumption, these

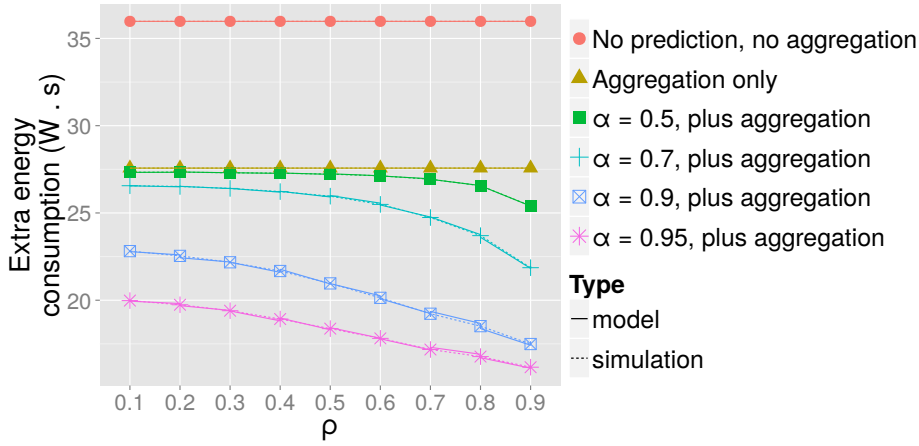



Figure 5.6: The model provides reliable results when compared with the simulations.

sensor nodes can run out of battery earlier than those in the other rings, which has a substantial impact on the WSNs' lifetime. In Figure 5.6, we can see that just adopting the aggregation scheme (without making predictions) reduces the extra energy consumption to $60\%$ of the total, yet larger packets are used. The greatest gains, nonetheless, are obtained after adopting the DPS and the aggregation scheme: they can save up to nearly $92\%$ of the energy consumed by the transmissions. As explained before, the predictions' accuracy are more significant and have a higher impact than the correlation between the measurements in a *sub-tree*. Hence, a very low correlation $(0.1)$ with highly accurate predictions $(0.95)$ give better results than a high correlation $(0.9)$ with an average accuracy $(0.5)$.

In fact, regardless of the values shown in the plot, the exact amount of saved energy depends on the hardware of the sensor nodes, their OS, and the MAC protocol in use, besides other configurations. Nonetheless, the consumption is mainly driven by the relation between the minimum energy necessary to keep a sensor node making measurements and the amount of battery required for transmitting and receiving a packet. In conclusion, the results presented here can facilitate the decision about

adopting a DPS in a WSN with a similar arrangement, even if the sensor nodes' configurations differ from those considered in our investigation.

## 5.4 Summary

In this Chapter, we presented a mathematical framework to calculate the gains and benefits of adopting a DPS to reduce the number of transmissions in a WSN. Using the proposed model, we showed that the benefits of adopting an aggregation scheme are greater than using only predictions in larger WSNs, and that combining both strategies leads to the highest savings. Moreover, we observed the most significant savings when we made accurate predictions in the GW and aggregated intermediate transmissions in the sensor nodes. Finally, our simulations also showed that the predictions' accuracy has a higher impact than the measurements' correlation in the total number of transmissions if a DPS is adopted.

The main contribution for the future generations of WSNs is a model that relies on the statistical theory to show the impact of sensor nodes' hardware evolution and the predictions' accuracy in DPSs. This model can be mainly used to exploit the characteristics of the WSNs to adopt predictions and improve the utilization of the channel resources. Additionally, the proposed model can be extended to calculate the sensor nodes' energy consumption and estimate WSNs' lifetime.

# Chapter 6

# CONCLUSION

The IoT is experiencing an exponential growth in the number of interconnected devices, sensors, and amount of produced data. Two factors can disrupt such an expansion: sensor nodes' energy availability and the wireless medium access. There is enough evidence to believe that sensor nodes' energy consumption can be–at least partially–overcome. Meanwhile, the wireless medium access is pointed out as a problem for the next generations of wireless networks.

To overcome the latter problem (and possibly minimize the former one), we focused on the potential benefits of adopting prediction-based strategies to reduce the number of data transmissions in WSNs. Finally, we conclude that data that can be predicted does not have to be transmitted and that prediction-based strategies can reduce the number of data transmissions in WSNs using the state-of-the-art technologies of wireless sensor nodes and data prediction algorithms.

To reach this conclusion, we first aimed at maximizing the impact of the latest advances in Data Science on the WSNs. Since there is no consensus about how to compute, analyze or publish data collected by WSNs, we designed a platform (DAS-Dashboard) that fits the main Data Science principles concerning sensed data: its collection; description; storage; maintenance; discovery; visualization; and analysis. This affinity with the Data Science principles serves as a starting point for a future set of

standards and solutions that aim to incorporate Data Science techniques in WSN and IoT scenarios.

Furthermore, centered on the DAS-Dashboard, we designed a self-managing architecture that incorporates WSNs into IoT environments. The resulting *smart* WSNs can exploit the power of AI techniques and sensor nodes' proximity to data's sources to self-manage their transmissions efficiently. From now, similarly to the procedure taken in our experiments, any person can deploy a WSN and host their data server, allowing remote users to visualize collected data and to outsource the optimization of their WSNs to external systems. The proposed self-managing architecture will permit new business models focused on sensor data, once it provides WSNs' managers the means to optimize their WSNs simply by choosing among their preferred prediction scheme (single or dual) and a cloud service that makes reliable predictions and recommendations for their applications.

For future generations of wireless sensor nodes and AI techniques, we provided a theoretical contribution. We experimented state-of-the-art prediction methods and designed a WSN transmission model that takes into account the hardware evolution and the consequent use of high complexity algorithms for predictions. The proposed model provides the necessary tools for designing new WSNs, considering the number of sensor nodes, their coverage, the periodicity of their transmissions and the predictions' accuracy. Most important, the mathematical analysis of this model provides us a reliable foundation to adopt prediction-based strategies in future WSNs, besides showing that predictions can extend the improvements provided by data aggregation schemes.

Overall, the work presented in this thesis can be extended to incorporate other prediction methods and data reduction mechanisms in the self-managing architecture proposed. Future research in the field of WSNs and IoT can focus on prediction algorithms and computational power to improve their accuracy, which will permit the improvement of the *smart* WSNs' operations. Moreover, the model used to estimate the number of transmissions can be validated, in practice, with alternative scenarios and applications. For instance, focusing on finding a network configuration

that minimizes the energy consumption in a WSN.

In the long-term, servers used to control WSNs will be able to forecast weather, predict human behavior [95] and infer objects' positions. Hence, most of the data about the environment will be generated without the need of transmitting measurements from sensors; and WSNs will be only necessary to fix few inevitable mistakes and inform a small number of unexpected variations. Therefore, it will be critical to rely on the statistical theory and self-managing architectures that can control other machines, react to the environments' evolution and interact with living beings; and this thesis gives small steps in these two directions.

# Appendix A

# HOW TO CHOOSE PREDICTION MODELS IN DUAL PREDICTION SCHEMES?

In general, to choose a prediction model, it may be necessary experimentation, expert opinion or experience [67]. Considering DPSs, prediction models that can provide more accurate results require more communication among sensor nodes, larger memory buffers, or longer computing times. Indeed, the choice of the prediction model may lead to a successful deployment or promptly make it extremely inefficient [70].

In the literature about statistical methods [96], a common way to choose a prediction model among a list of options is to adopt an information criterion that rewards their accuracy over a training dataset and, in change, penalize their selection according to the number of parameters used to compute the predictions. Methods such as Akaike Information Criterion (AIC), the Bayes Information Criterion (BIC), and AIC with a correction for finite sample sizes (AICc) are some of the existing options of information criteria used to estimate the amount of information loss, given a particular prediction model and a training dataset [97, 98].

If an information criterion is adopted, the first step is to assess the predictions' accuracy, which can be done using measures that attest the quality of a prediction model in a particular use case. Examples of such measures are the well-known Mean Square Error (MSE), the Root Mean Square Error (RMSE), the Mean Absolute Error (MAE), Relative Mean Absolute Error (RelMAE), the Mean Absolute Percentage Error (MAPE), the symmetric Mean Absolute Percentage Error (sMAPE), among others [99]. Once the accuracy has been measured, the information criterion will observe the number of parameters used in each model to indicate the most proper for that situation.

In the literature about DPSs, some mechanisms adopted information criteria to choose the best prediction models [71], while others adopted only accuracy measures [32, 64]. However, WSNs have computational limitations that most networks do not have. Therefore, in DPSs, the decision should be made based on a few aspects that impact the number of transmissions and the energy consumption in the sensor nodes. For example, the number of messages generated by the scheme when the prediction fails and all the engineering concerns, such as the energy consumed to choose new prediction models and, especially, to transmit their parameters.

To overcome the limitations of the traditional methods, Liu et al. created a formula to estimate the Prediction Cost (PC). They considered the percentage of transmitted measurements ($r$) and the user desired level of accuracy ($\alpha$) [74]. Later, an extended model was designed [32]. The new model is more generic and also considers the prediction models' memory footprint ($Ec$) as a significant computational cost for sensor nodes:

$$\text{PC} = [\ \alpha f(e)\ +\ (1-\alpha)r\ ]\ \text{Ec}, \tag{A.1}$$

where $e$ is the measure of the predictions' accuracy (e.g., MSE, RMSE, sMAPE) and $f(e)$ is the accuracy according to the chosen measure. According to this formula, given a list of prediction models, the one with the lowest PC is the most appropriate to that situation.

Periodically, new prediction models may be chosen if the current ones are not predicting as accurate as expected. To decide for a new predic-

tion model, it is necessary to estimate what is the most proper prediction method, given the current environmental conditions, and if making predictions (instead of transmitting all measurements) will reduce the number of transmissions in the WSN. This analysis may be done either in the GW or in sensor nodes and are fundamental to keep the predictions' accuracy in a DPS.

# Appendix B

# BACKGROUND ABOUT FORECASTING METHODS

A time series ($X$) is a sequence of data points, typically consisting of observations made over a time interval and ordered in time [100]. Each observation is usually represented as $x_t$, where the observed value $x$ is indexed by the time $t$ at which it was made. In the literature about time series forecasting [25, 100, 101], it is possible to find several examples of prediction methods that use time series as input (a.k.a., forecasting methods).

In this Appendix, we clarify a set of terms related to forecasting methods. Later, we explain and experiment the forecasting methods used in WSN environments: naive approaches, the ES, the ARIMA and the ANN methods.

## Definitions

**Forecasting method** A forecasting method ($F$) is a prediction method for forecasts. It uses two input variables: a time series ($X$) and a set of parameters ($\theta$).

**Forecasting model** A forecasting model ($f$) is an instance of a forecasting method $F$, such that $f_\theta(X) = F(X, \theta)$. Forecasting models use time series as input to predict future values, which are represented as a function of the past observations and their respective time. That is, $(\bar{x}_{t+1} \dots \bar{x}_{t+i}) = f_\theta(x_t, \dots, x_{t-k})$, where $\bar{x}_{t+1} \dots \bar{x}_{t+i}$ are the forecasts for the period between $t+1$ and $t+i$.

**Accuracy** Accuracy measures are used to evaluate the quality of the predicted values based on the difference between the prediction and the observed value, i.e., the error $e_t = x_t - \bar{x}_t$ [25].

**Information criteria** Information criteria are measures used to estimate the information loss if the time series is modeled by a set of parameters $\theta$. These criteria are useful to choose a prediction model: the estimated information loss is applied to infer the relative quality of the parameters and to choose the best option given a set of candidates. That is, assuming that the future data will have the same characteristics as the observations already made, the goal is to choose the set of parameters that will minimize the information loss, which tends to improve the forecasts' accuracy. In some cases, the number of parameters is also taken into account, i.e., using fewer parameters may be considered an advantage, because it avoids overfitting the training data.

Examples of information criterion measures are the AIC; the BIC; and the AICc [102]. In our experiments, we adopted the AICc as the information criterion to choose the most proper forecasting models.

# Overview of Forecasting Methods

The methods used in this thesis are explained in detail in the literature about forecasting methods [25], and their computing complexities are summarized in Table B.1.

To illustrate each method, we ran a set of preliminary tests over the *Intel* and the *Ball* datasets presented in Chapter 4. In our tests, we used

| Method | Preprocessing time complexity | Runtime complexity | Space complexity |
|---|---|---|---|
| **Constant** | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ |
| **Linear** | $\mathcal{O}(1)$ | $\mathcal{O}(w)$ | $\mathcal{O}(1)$ |
| **SM** | $\mathcal{O}(h)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ |
| **ES** | $\mathcal{O}(k^3\ h)$ | $\mathcal{O}(w)$ | $\mathcal{O}(1)$ |
| **ARIMA**$(p,d,q)$ | $\mathcal{O}(k^3\ h^2)$ | $\mathcal{O}((p+q)w)$ | $\mathcal{O}(max(p,q+1))$ |

Table B.1: List of forecasting methods and their complexities[1].

100 values to forecast 5, 10, 20, and 30 values in the future, i.e., the *history* length was 100, and the *window* length varied among 5, 10, 20, and 30. We call as *period* the set of values observed before a forecast (*history*) plus the predicted values (*window*).

To illustrate the accuracy in our tests, we adopted the RMSE as it is a good indicator of accumulated errors, i.e., it provides a perspective of the cumulative error when the number of predictions is increased. As the RMSE is calculated as

$$\text{RMSE}(e) = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}, \tag{B.1}$$

the smaller values represent more accurate forecasts. The preliminary tests' RMSEs are shown in Tables B.2 and B.3, where the smallest errors in each period are highlighted.

## Naive approaches

Examples of naive approaches for forecasts may vary between calculating the *average* of the past observations, the *maximum* observed value or assuming the *same* as the last observation made in time. Thanks to their simplicity, they may become an attractive option for WSNs due to the

[1] ANNs are not included in Table B.1 because they are soft computing solutions that cannot be bounded by a computational time limit [82, 83].

| Period | Values observed | Values predicted | Accuracy (RMSE) | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Constant | Linear | SM | ES | ARIMA(3,0,3) | ANN |
| #1 | 100 | 5 | **0.03** | 0.12 | 0.09 | 0.04 | 0.04 | 0.08 |
| #2 | 100 | 10 | 0.23 | **0.1** | 0.82 | 0.28 | 0.27 | 0.88 |
| #3 | 100 | 20 | **0.48** | 56.06 | 0.5 | 0.61 | 0.57 | 0.59 |
| #4 | 100 | 30 | 0.3 | 44.85 | 0.52 | **0.09** | 0.14 | 0.46 |
| #5 | 100 | 40 | **0.13** | 91.22 | 0.47 | 0.17 | 0.33 | 0.44 |
| #6 | 100 | 50 | 0.27 | 7.84 | 0.42 | 0.35 | **0.2** | 0.35 |

Table B.2: Accuracy observed during the primary tests over the *Intel* dataset.



| Period | Values observed | Values predicted | Accuracy (RMSE) | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Constant | Linear | SM | ES | ARIMA(3,0,3) | ANN |
| #1 | 100 | 5 | **0.45** | 2.58 | 14.23 | 0.64 | 1.49 | 4.26 |
| #2 | 100 | 10 | 2.13 | 10.6 | 12.01 | 2.97 | **1.84** | 4.89 |
| #3 | 100 | 20 | 7.81 | 4.14 | 5.63 | 5 | **2.78** | 9.75 |
| #4 | 100 | 30 | 12.23 | 40.78 | 14.73 | **5.95** | 9.11 | 11.54 |

Table B.3: Accuracy observed during the primary tests over the *Ball* dataset.

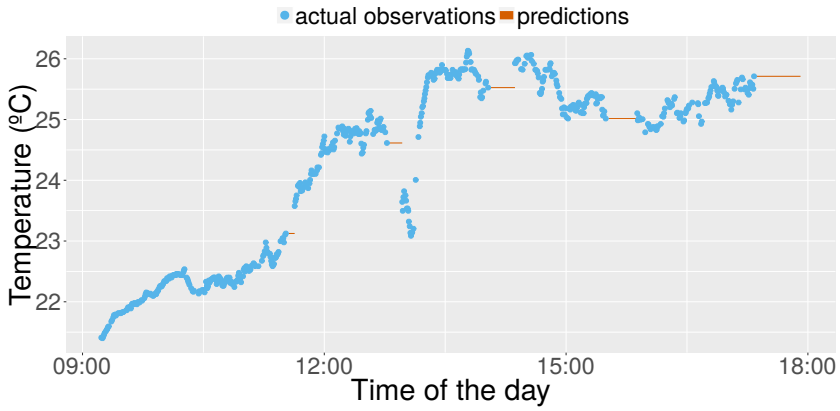



Figure B.1: Temperature forecasts using the *Constant* method has acceptable results when the values do not undergo large variations.

sensor nodes' computing power limitations, even though the predictions are not as accurate as possible

### Constant predictions

As the name suggests, a *Constant* prediction model assumes that no changes will happen in the environment, and it always predicts that the measurements in the future will be the same as it was in the most recent observations, i.e., $\bar{x}_{t+i} = x_t$ [103].

Figure B.1 shows predictions over the *Intel* dataset using a *Constant* model, where the predicted values are the same as the last value measured by the sensor nodes. It is possible to observe that the *Constant* method approximates better to the real values when the variance of the measurements is lower, but it performs poorly when the values change abruptly.

As shown in Table B.1, its computation complexity is constant (and low), which explains why they are often adopted in real sensor networks when a transmission has been missed, or a measurement is not received for any other reason.

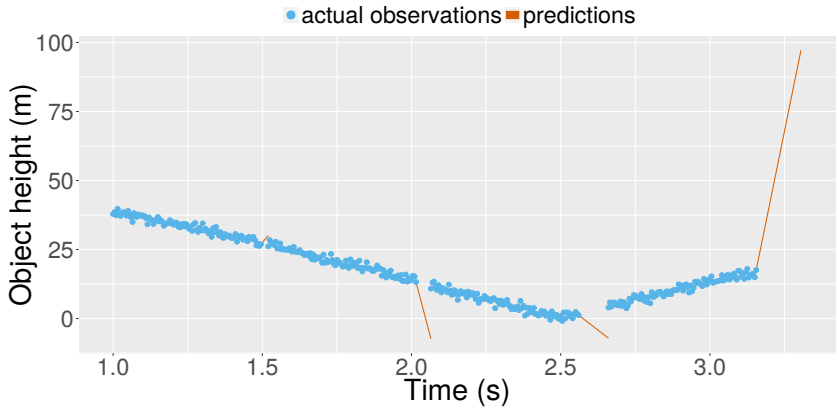


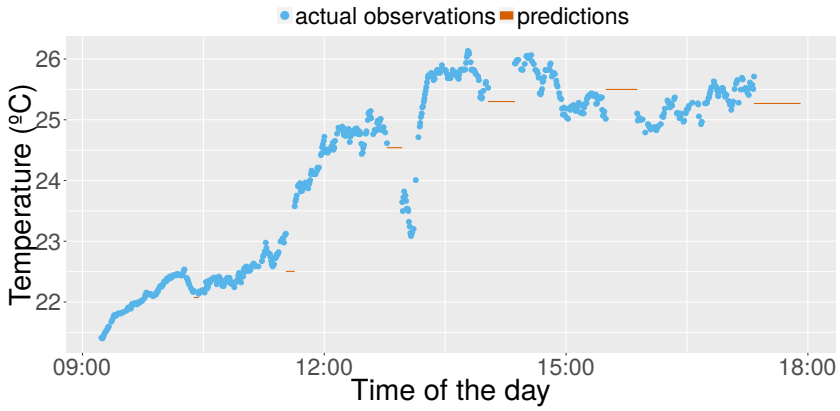

Figure B.2: The *Linear* method assumes the object has a constant speed.

### Linear method

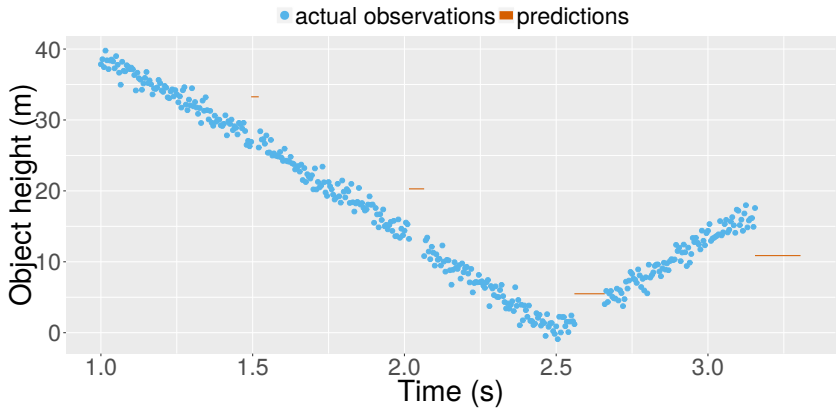
Differently from the *Constant*, the *Linear* method assumes that the measured values may change in the future. Such changes, however, have a linear component that does not vary in time. As an example, the height of the ball is forecast as if its speed were constant in Figure B.2. That is, the predictions were computed as $\bar{x}_{t+i} = (x_t - x_{t-1})i + x_t$. From the plot, it is possible to observe that, even though the movements are very intuitive from a human point of view, the noise in the measurements leads to highly inaccurate predictions that ignore trends and physical laws.

### Simple Mean method

As well as the *Linear*, the SM method predicts in constant time. The method consists in calculating a simple average using the values observed before. Thus, the SM requires longer preprocessing time than the *Linear* method to calculate the average between the last $n$ observations. In practice, at time $t$, it computes $\bar{x}_{t+i} = \frac{1}{n}\sum_{j=t-n}^{t} x_j$. Figure B.3a shows the predictions over the *Intel* and *Ball* datasets computed using the SM method and shows that the predictions can be extremely inaccurate if the data is varying or if there is a significant trend in the latest measurements.




(a) Temperature forecasts using the SM method assumes that the temperatures may vary, but the average will be the same as the one observed more recently.




(b) Forecasting the object position using SM usually breaks the movements' continuity, which is unlikely to be observed in this case.

Figure B.3: SM models may often forecast unrealistic values.

## Exponential Smoothing method

In this method, the value predicted for the time $t + i$ can be calculated using only the most recent observation and the most recent forecast, i.e., in constant time, as shown in Table B.1. For instance, the value of $\bar{x}_t$ is the weighted average:

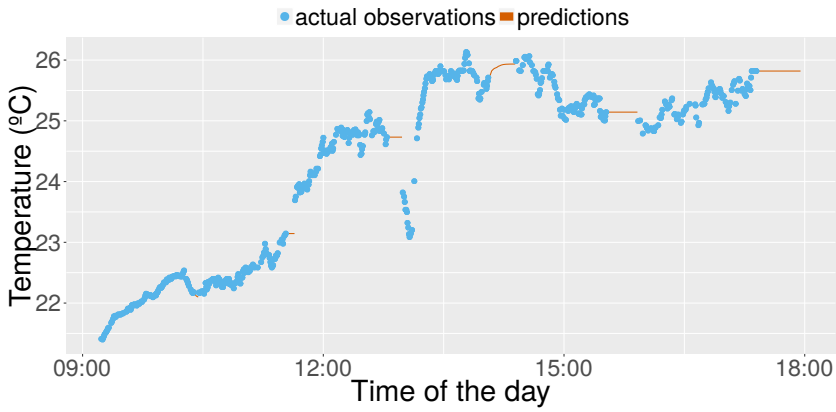
$$\bar{x}_t = \alpha x_{t-1} + (1 - \alpha)\bar{x}_{t-1} \tag{B.2}$$

Guided by the value of $\alpha \in [0, 1]$ (also called *smoothing constant*), the relevance of the old measurements undergo an exponential decay, which justifies the method's name. As shown in Figure B.4, the forecasts converge slowly to average values and tend to follow the trends observed in the measurements during a long period. In practice, they tend to have better accuracy than the naive methods when forecasting longer periods, i.e., more than ten values.

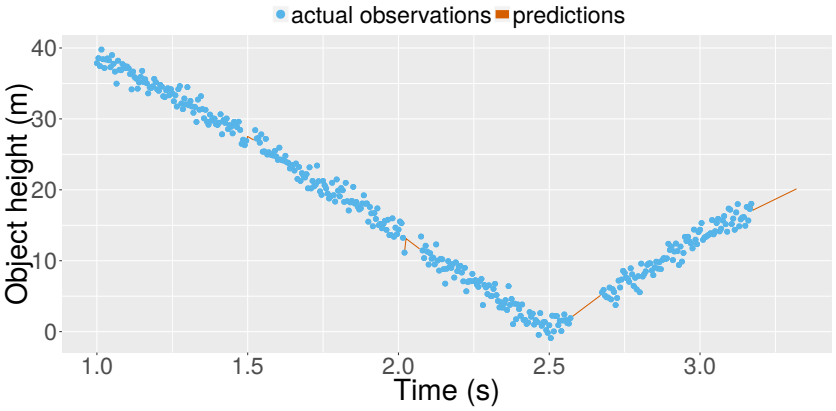
Other formats of the ES are also widely used; they add up to two new parameters ($\beta$ and $\gamma$) to better detect non-linear trends. A common way to set up good values for $\alpha$, $\beta$ and $\gamma$ is by trying among $k$ possible values each (usually, $k = 10$). The choice can be made based on an information criterion. It is important to highlight that adopting an information criterion to choose the best set of parameters can noticeably increase the complexity of the predictions because it requires more calculations and comparisons in the preprocessing stage.

## Autoregressive Integrated Moving Average method

An ARIMA model is a combination of an AR and a Moving Average (MA) model. The ARIMA method considers, to calculate the forecasts, the magnitude of the last observations and their trends (which is provided by the AR model), and the impact of unobserved shocks that influenced their current state (which is provided by the MA model). As the accuracy of the AR and MA models is conditioned to the data stationarity, in the case of observing a non-stationary distribution, an initial differentiating step (corresponding to the "integrated" part of the model) can be applied.




(a) ES models can capture recent trends and provide more accurate predictions even if the values are varying.




(b) ES provides even more accurate predictions when the trends can be easily observed.

Figure B.4: Forecasts using the ES method.

The integration step is represented by the equation

$$y_t = (1 - L)^d x_t, \tag{B.3}$$

where $d$ is the order of the integrated model and $L$ is the Lag operator, such that $L^k x_t = x_{t-k}$ for all $t > k$.

Finally, an ARIMA$(p, d, q)$ model contains an AR model with order $p$ and an MA model with order $q$; and a value observed at time $t$ can be represented as

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}, \tag{B.4}$$

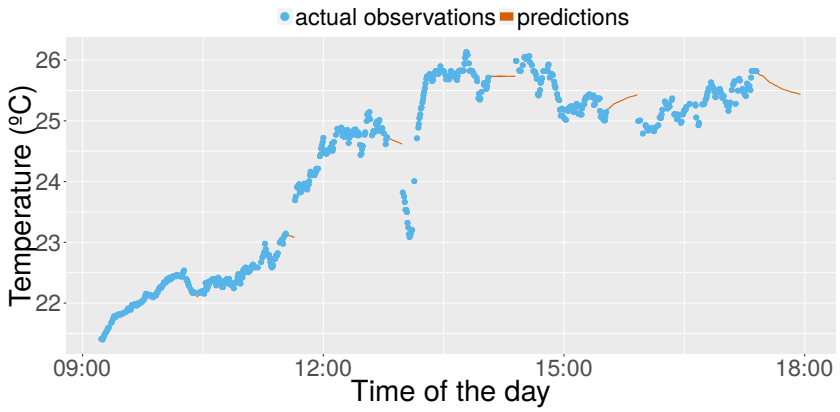
where $c$ is a constant. Note that it is used to predict the value of $y_t$, which is derived from $x_t$, if $d > 0$, or simply equal to $x_t$, if $d = 0$.

As well as the ES method, the parameters $p$, $q$ and $d$ are usually set among $k$ (usually, $0 \leq k \leq 6$) possible values each, using an information criterion measure. In comparison with the simplest methods (such as *Linear* and SM), ARIMA models can predict more accurately the new trends observed in the latest observations, as shown in Figure B.5.

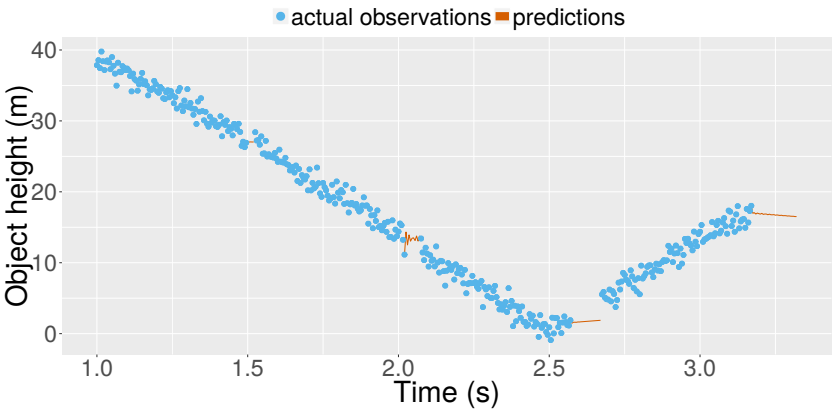
**Special cases of ARIMA** The ARIMA method has a particular characteristic which is the overlapping with other methods. That is, some ARIMA models are equivalent to models of other methods, for example:

- ARIMA$(0, 1, 0)$ is equivalent to the *Constant* method;
- ARIMA$(0, 2, 0)$ is equivalent to the *Linear* method;
- ARIMA$(0, 0, 0)$ is the SM method;
- ARIMA$(0, 1, 1)$ is the simplest model of the ES (with only one parameter); and
- ARIMA$(0, 2, 2)$, ARIMA$(0, 1, 2)$ and ARIMA$(1, 1, 2)$ are equivalent to more complex ES models.

Hence, when choosing the most proper ARIMA model using an information criterion, some of the other methods are also implicitly considered.




(a) Temperature forecasts using ARIMA models are more realistic and accurate than the SM models (see Table B.2).



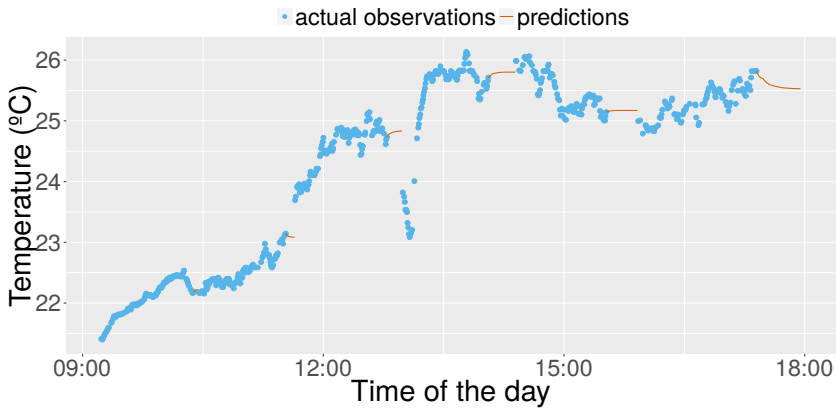

(b) Object positions may not follow a stationary distribution, which may reduce the accuracy of the predictions using ARIMA.

Figure B.5: Forecasts using ARIMA$(3, 0, 3)$ models.

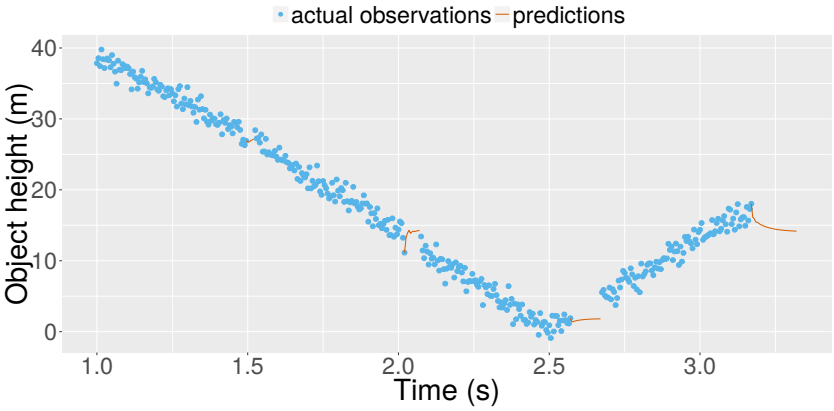
## Artificial Neural Networks

ANNs are targeted to create an artificial version of biological neurons [82, 83]. That is, they simulate a network of components (so-called *neurons*) to predict the output of a system, given a set of inputs. To achieve that, an ANN must go through a *learning phase*, i.e., it needs to adapt its internal parameters and *learn* from available historical data until they converge to the final values. However, differently from the other methods, the time needed to set up the parameters of an ANN can drastically vary from case to case, due to the high number of possible states. Moreover, the convergence time (i.e., the time to find a set of proper parameters) may tend to infinite, which explains why ANNs are soft computing solutions that cannot be bounded by a computational time limit.

ANNs often perform better for long-term predictions than for smaller intervals [104] and their accuracy is better than the traditional methods in time series with discontinuities [105], which may happen in the case of absence of parts of the data. However, their accuracy tends to be worse when the number of values used to forecast is small, which was observed in our primary tests, where the other methods always outperformed the ANNs. Tables B.2 and B.3 shows the differences in their accuracies and Figure B.6 illustrates their limitations to follow some trends.




(a) ANNs tend to perform poorly when few values are available during the *learning* phase.




(b) Predictions about the object movement are as accurate as those using the simpler methods.

Figure B.6: Forecasts using ANNs.

# Appendix C

# ON THE IMPORTANCE OF REDUCING TRANSMISSIONS IN WSNS

The number of wireless transmissions impacts directly the efficient use of spectrum resources, which is one of the key challenges for the next generation of wireless networks, such as Wireless Local Area Networks (WLANs), 4G, 5G, and WSNs [17].

Predicting sensed data is a potential candidate to shorten the increase in the number of WSN transmissions, which is reinforced by its presence in real world sensor network applications. Assuming that sensor nodes can communicate with neighbors within a maximum distance, the model presented in Chapter 5 can be used to evaluate how the number of transmissions is affected when new sensor nodes are added to a WSN.

In this Appendix, we will model the impact of linear growth in WSNs' radius and density, and estimate the maximum number of sensor nodes that a WSN can accommodate, given hardware constraints of real wireless sensor nodes. In the following, we assume a monitoring WSN with homogeneous sensor nodes periodically transmitting measurements every $1/f$ seconds in a payload of $n$ bits.

# Radius growth

Let us suppose a WSN with $D-1$ rings and density $C+1$, i.e., each sensor node has an average of $C$ neighbors, and there is a total of $C(D-1)^2$ sensor nodes. Based on (5.1), adding a new ring to this network represents $C(2D-1)$ new sensor nodes in a total of $CD^2$ nodes. Therefore, the number of sensor nodes grows quadratically in terms of the number of rings.

Recall that a transmission from a sensor node in ring $d$ needs to be forwarded $d-1$ times to reach the GW (summing $d$ transmissions); moreover, according to (5.1), a ring $d$ has $(2d-1)C$ sensor nodes. Therefore, all sensor nodes in ring $d$ will trigger a total of $(2d-1)Cd$ node-to-GW transmissions every $1/f$ seconds. Thus, summing this expression from $d=0$ to $d=D$ gives us that the total number of transmissions in the sensor network during a unit of time (i.e., $1/f$ seconds) is

$$\sum_{d=0}^{D} Cd(2d-1) = \frac{4CD^3+3CD^2-CD}{6}. \tag{C.1}$$

This equation shows that considering a linear growth in the number of rings, the number of transmissions grows cubically.

# Density growth

Similarly, considering the network with $D$ rings and density $C+1$ (i.e., $CD^2$ sensor nodes), a linear increase in the sensor density represents $D^2$ new nodes uniformly distributed and $(C+1)d(2d-1) - Cd(2d-1)$ new transmissions per ring. Summing this expression from $d=0$ to $d=D$ shows us that increasing the density of a sensor network by one will trigger

$$\sum_{d=0}^{D}(C+1)d(2d-1) - Cd(2d-1) = \frac{4CD^3+3CD^2-CD}{6} \tag{C.2}$$

new node-to-GW transmissions per unit of time (i.e., $1/f$ seconds).

In fact, the number of new transmissions calculated in (C.2) is the same as the number of total transmissions in a network with radius $D$ and density $C + 1$ calculated in (C.1). Therefore, increasing the density of a network by one will duplicate its number of transmissions.

# Throughput vs. number of sensor nodes

Let us call the throughput of a sensor node as $\lambda$. If we consider that each sensor node reports its measurements using unicast transmissions, every $1/f$ seconds, a sensor node in the first ring will have to forward the transmissions from all its children $K_1 = D^2 - 1$ besides transmitting its measurement. Hence, it will require $D^2$ transmissions every $1/f$ seconds.

## Minimum throughput

To avoid collisions, a sensor node $i$ cannot transmit simultaneously to any other sensor node in the range of 2 hops of distance, even if they are not direct children of $i$. Considering $i$, there will exist $4C + 1$ sensor nodes in the range of 2 hops from it (i.e., $CD^2 = 4C$, because $D = 2$, plus $i$). Thus, assuming a fair MAC protocol in which each sensor node transmits once per cycle, $i$ cannot transmit more than $\lambda/(n(4C + 1))$ bits per $1/f$ seconds. Such a maximum throughput would happen in a perfect scenario where each sensor node can transmit coordinately at a time with no control overhead.

Therefore, the necessary sensor nodes' throughput ($\lambda'$) can be calculated in function of the disposition of the sensor nodes as

$$\frac{\lambda'}{n(4C + 1)} \geq D^2 n f, \tag{C.3}$$

and the minimum throughput ($\lambda_{\text{min}}$) must be

$$\lambda_{\text{min}} = D^2 n^2 f(4C + 1). \tag{C.4}$$

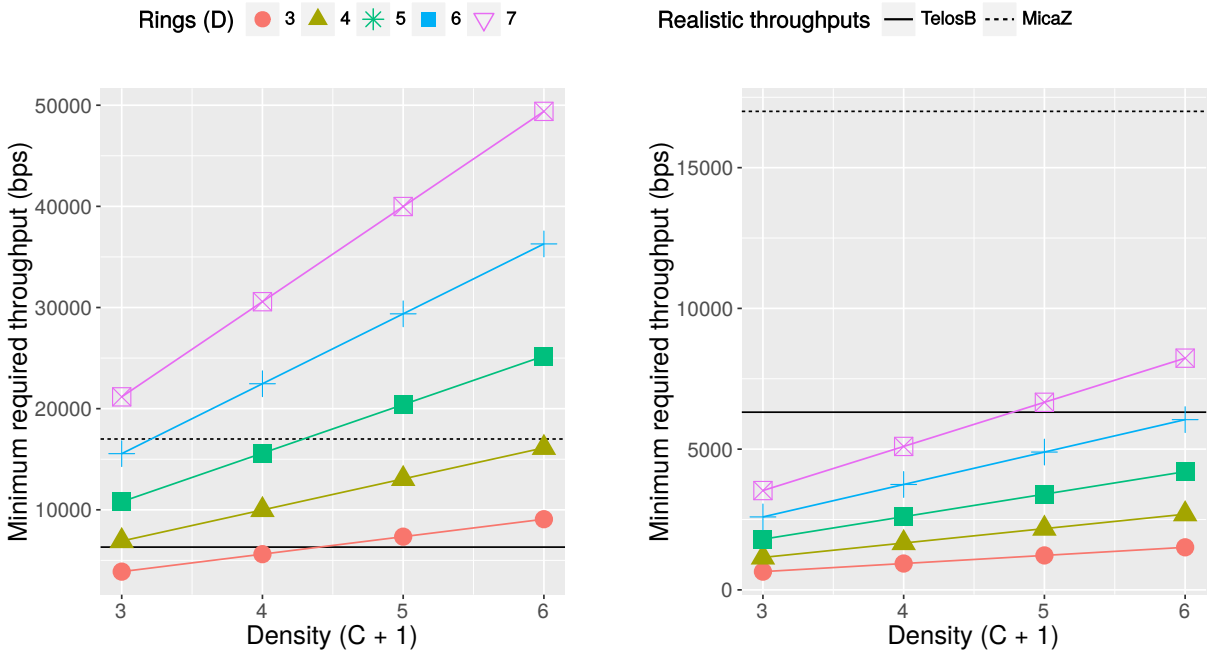
## Realistic throughput

Unfortunately, in practice, wireless sensor nodes have throughputs that differ considerably from their nominal transmission rates. Let us define the *realistic throughput* of a wireless sensor node as the number of application bits that can be transmitted using real hardware. The difference between this value and the nominal transmission rate is due to hardware delays to process and transmit data, and extra bits used to control the communication in lower layers. For example, assuming a payload of $15$ bytes and considering the internal delays to prepare and transmit a packet, the *realistic throughput* of TelosB and MicaZ motes are respectively $6313$ and $17000$ bps [106], even though both mote types have nominal transmission rates of $250000$ bps [36, 107].

Figure C.1a shows the minimum throughput for scenarios with different numbers of rings, densities and a sampling interval of five minutes between consecutive measurements. Based on our model, we can affirm that a WSN composed by TelosB motes can have more than four hops only if it is a linear topology (i.e., $C + 1 \sim 3$). Because of the low *realistic throughput* of TelosB motes, a WSN cannot have more than $35$ nodes, if they are all transmitting once every five minutes with $15$ bytes of payload. If we assume the minimum nominal range of their radio transmitter ($20$ meters), a homogeneous WSN with TelosB motes can have the maximum number of sensor nodes if it is arranged in four rings with a density of four sensor nodes per $1300$ meters$^2$. In practice, such a WSN could cover an area of two hectares.

On the other hand, MicaZ motes have a higher *realistic throughput*, which impacts the maximum number of sensor nodes in a WSN. For instance, it may be possible to deploy nearly $100$ nodes with a density of five nodes per $1300$ meters$^2$ arranged in four rings.

Finally, Figure C.1b shows that larger WSNs may be deployed if sensor nodes are programmed to transmit once every $30$ minutes. For example, in this perfect scenario without control overhead, it would be possible to arrange TelosB motes in $6$ rings with a density of $6$ nodes, i.e., a total of $180$ sensor nodes.




(a) $1/f = 300$ seconds.

(b) $1/f = 1800$ seconds.

Figure C.1: The minimum required throughput for a WSN with sensor nodes transmitting packets with 15 bytes of payload.

# Appendix D

# MINIMUM ACCURACY

Let us assume a monitoring WSN with homogeneous sensor nodes periodically transmitting measurements every $1/f$ seconds, and a sensor node $i$ in ring $d$. According to (5.6), the number of transmissions at $i$ in a period of $1/f$ seconds is the sum $S_i + R_i$. If a DPS is adopted, the number of transmissions will be $S'_i + R'_i + X_{\text{DIS}}$, as defined in (5.14). Therefore, a DPS will reduce the number of transmissions during a period $T$ if and only if:

$$(S'_i + R'_i)fT + X_{\text{DIS}} \leq (S_i + R_i)fT. \tag{D.1}$$

Thus, we can define $\alpha_{\text{min}}$ as the minimum average between the accuracies of the children of $i$ that would reduce the number of transmissions in a DPS (without aggregation). It must satisfy the following equation:

$$(S'_i + R'_i)fT + X_{\text{DIS}} = (S_i + R_i)fT. \tag{D.2}$$

Based on (5.4), (5.5), (5.15), and (5.16), we can calculate it as

$$((K_d + 1)\,\alpha^c_{\text{min}} + (K_d\,\alpha^c_{\text{min}}))fT + X_{\text{DIS}} = ((K_{i,d} + 1) + K_d)fT. \tag{D.3}$$

Knowing this, we can work out the equation:

$$\begin{aligned}
((K_d+1)\,\alpha^c_{\text{min}} + (K_d\,\alpha^c_{\text{min}}))fT &= ((K_d+1)+K_d)fT - X_{\text{DIS}} \\
\alpha^c_{\text{min}}(D^2\ + (D^2-1))fT &= (D^2+(D^2-1))fT - X_{\text{DIS}} \\
\alpha^c_{\text{min}}(D^2\ + (D^2-1))fT &= (D^2+(D^2-1))fT - X_{\text{DIS}} \\
\alpha^c_{\text{min}} &= \frac{(D^2+(D^2-1))fT - X_{\text{DIS}}}{(D^2\ + (D^2-1))fT} \\
\alpha^c_{\text{min}} &= 1 - \frac{X_{\text{DIS}}}{(D^2\ + (D^2-1))fT} \\
\alpha_{\text{min}} &= \frac{X_{\text{DIS}}}{(D^2\ + (D^2-1))fT} \\
\alpha_{\text{min}} &= \frac{X_{\text{DIS}}}{(2D^2\ - 1)fT}.
\end{aligned} \tag{D.4}$$

Assuming no aggregation in the *dissemination* of prediction models, the value of $X_{\text{DIS}}$ is defined by (5.10):

$$X_{\text{DIS}} = S^* + R^* = (2D^2\ - 1). \tag{D.5}$$

Therefore,

$$\alpha_{\text{min}} = 1/fT. \tag{D.6}$$

# Appendix E

# DATA MODEL

A Normal distribution is characterized by its probability density function whose pattern is often encountered in several types of observations. According to the Central Limit Theorem, the sampling distribution of the mean of any independent random variable tends to be Normal, even if the distribution from which the average is computed is decidedly non-Normal. For example, it has been shown that environmental readings– such as temperature, light, and humidity–done by outdoor WSNs can be approximated to normal distributions if properly managed [51].

We will assume that a sensor network is composed of a set of sensor nodes $S$ and each sensor node $i \in S$ is responsible for measuring a certain parameter from the environment, such that the set of observations follows a Normal distribution with mean $\mu_i$ and variance $\sigma_i^2$. By convention, this is represented as $Y_i = N(\mu_i, \sigma_i^2)$. A prediction $\bar{y}_i$ (for example, $\bar{y}_i = \mu_i$) can be calculated by the sensor node $i$ and the GW. We define the accepted threshold $\varepsilon_i$, i.e., the prediction is told to be correct if the real observation ($y_i$) is in the interval $[\bar{y}_i - \varepsilon_i,\ \bar{y}_i + \varepsilon_i]$.

Assuming that the data is normally distributed, the chances of observing a new value inside the accepted interval can be calculated by normalizing the value of $\varepsilon_i$, i.e., rewriting it in terms of the variance $\sigma_i^2$. The normalized value of $\varepsilon_i$ is represented by $z_i$ as

$$z_i = \frac{\varepsilon_i - \bar{y}_i}{\sigma_i}. \tag{E.1}$$

Thus, in this case, the accuracy of the predictions ($\alpha_i$) can be calculated based on the cumulative distribution function of the normal distribution:

$$\Phi_{\mu,\sigma}(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]. \tag{E.2}$$

Again, according to the Central Limit Theorem, we assume unbiased predictions and normally distributed errors. Therefore, the percentage of observations that will fall outside the accepted interval is represented by the two-tailed Z-test (i.e., $2\Phi(-|z_i|)$), and $\alpha_i$ is

$$\alpha_i = 1 - 2\Phi(-|z_i|). \tag{E.3}$$

By substituting (E.1) into (E.3), we can observe that

$$\alpha_i = 1 - 2\Phi\left(-\left|\frac{\varepsilon_i - \bar{y}_i}{\sigma_i}\right|\right), \tag{E.4}$$

which shows that the accuracy of the predictions depends on the accepted threshold, on the mean and on the variance of the data.

# Appendix F

# PROOF OF $1 - \alpha^X \leq X\,(1 - \alpha)$

Let us assume two values $\alpha$ and $x$ such that $\alpha \in [0, 1]$ and $x \geq 1$. We want to show that $1 - \alpha^x \leq x\,(1 - \alpha)$:

$$\begin{aligned} 1 - \alpha^x &\leq x\,(1 - \alpha) \\ 1 - \alpha^x &\leq x - \alpha\,x \\ -\alpha^x - 1 + &\leq x - \alpha\,x \\ \alpha^x &\geq 1 - x + \alpha\,x \\ \alpha^x &\geq 1 + x\,(\alpha - 1) \end{aligned} \tag{F.1}$$

When $\alpha = 0$ or $\alpha = 1$, we can easily observe that the affirmation is true because $x \geq 1$ by definition. For the other values of $\alpha$, we can use the Bernoulli's inequality:

$$(1 + i)^j \geq 1 + ij, \tag{F.2}$$

where $i > -1$, $i \neq 0$ is a real number and $j \geq 2$, an integer value. Substituting the values of $\alpha$ and $x$ in (F.1) respectively by $i + 1$ and $j$, the claim is proved.

# Bibliography


[1] A. L. Turnbull, "Ecology of the True Spiders (Araneomorphae)," *Annual Review of Entomology*, vol. 18, no. 1, pp. 305–348, 1973. [Online]. Available: http://dx.doi.org/10.1146/annurev.en.18.010173.001513

[2] Gartner, "Gartner says 6.4 billion connected "Things" will be in use in 2016, up 30 percent from 2015," 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3165317

[3] R. Jurdak, A. Elfes, B. Kusy, A. Tews, W. Hu, E. Hernandez, N. Kottege, and P. Sikka, "Autonomous surveillance for biosecurity," *Trends in Biotechnology*, vol. 33, no. 4, pp. 201–207, Apr. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.tibtech.2015.01.003

[4] D. J. McCorrie, E. Gaura, K. Burnham, N. Poole, and R. Hazelden, *Predictive Data Reduction in Wireless Sensor Networks Using Selective Filtering for Engine Monitoring*. New York, NY: Springer New York, 2015, pp. 129–148. [Online]. Available: http://dx.doi.org/10.1007/978-1-4939-2468-4_6

[5] J. A. Stankovic, "Research challenges for wireless sensor networks," *ACM SIGBED Review*, vol. 1, no. 2, pp. 9–12, Jul. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1121780

[6] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey,"

*Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, May 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1570870508000954

[7] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2014.03.027

[8] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 95–107, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?id=1031508

[9] G. M. Dias and T. Braun, "Implementing a Reliable Overlay Multicast Protocol on Wireless Sensor Nodes," Master's thesis, Universität Bern, 2011. [Online]. Available: http://rvs.unibe.ch/research/pub_files/Di11.pdf

[10] H. Luo, Y. Liu, and S. Das, "Routing correlated data in wireless sensor networks: A survey," *Network, IEEE*, no. December, pp. 40–47, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4395109

[11] M. Nurchis, R. Bruno, M. Conti, and L. Lenzini, "A self-adaptive routing paradigm for wireless mesh networks based on reinforcement learning," *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems - MSWiM '11*, p. 197, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2068897.2068932

[12] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 37–59, Jan. 2012. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1084804511000555

[13] Y. Yin, F. Liu, X. Zhou, and Q. Li, "An Efficient Data Compression Model Based on Spatial Clustering and Principal Component Analysis in Wireless Sensor Networks." *Sensors (Basel, Switzerland)*, vol. 15, no. 8, pp. 19 443–65, Jan. 2015. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/26262622

[14] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 3, pp. 443–461, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5522465

[15] L. Xie, Y. Shi, Y. T. Hou, and W. Lou, "Wireless Power Transfer and Applications to Sensor Networks," *IEEE Wireless Communications*, no. August, pp. 140–145, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6590061

[16] M. Le Thai, G. T. Chandran, R. K. Dutta, X. Li, and R. M. Penner, "100k Cycles and Beyond: Extraordinary Cycle Stability for MnO 2 Nanowires Imparted by a Gel Electrolyte," *ACS Energy Letters*, pp. 57–63, 2016. [Online]. Available: http://dx.doi.org/10.1021/acsenergylett.6b00029

[17] B. Bellalta, L. Bononi, R. Bruno, and A. Kassler, "Next generation IEEE 802.11 Wireless Local Area Networks: Current status, future directions and open challenges," *Computer Communications*, vol. 000, pp. 1–25, Nov. 2015. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0140366415003874

[18] S. Pal, S. Oechsner, B. Bellalta, and M. Oliver, "Performance optimization of multiple interconnected heterogeneous sensor networks via collaborative information sharing," *Journal of Ambient Intelligence and Smart Environments*, vol. 5, no. 4, pp. 403–413, Mar. 2013. [Online]. Available: http://arxiv.org/abs/1203.6316

[19] G. M. Dias, "Performance optimization of wsns using external information," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, Jun. 2013, pp. 1–2. [Online]. Available: https://arxiv.org/abs/1607.03408

[20] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer Communications*, vol. 30, no. 14-15, pp. 2826–2841, Oct. 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0140366407002162

[21] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," pp. 1–14, 2009. [Online]. Available: http://doi.acm.org/10.1145/1644038.1644040

[22] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1045297

[23] L. A. Villas, A. Boukerche, H. S. Ramos, H. A. B. F. De Oliveira, R. B. De Araujo, and A. A. F. Loureiro, "DRINA: A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks," *IEEE Transactions on Computers*, vol. 62, no. 4, pp. 676–689, Apr. 2013.

[24] F. Marcelloni and M. Vecchio, "An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks," *Computer Journal*, vol. 52, no. 8, pp. 969–987, Apr. 2009.

[25] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014. [Online]. Available: https://www.otexts.org/fpp

[26] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "Convergence of MANET and WSN in IoT Urban Scenarios,"

*IEEE Sensors Journal*, vol. 13, no. 10, pp. 3558–3567, Oct. 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6552998

[27] S. Sridharan and J. M. Patel, "Profiling R on a contemporary processor," *Proceedings of the VLDB Endowment*, vol. 8, no. 2, pp. 173–184, Oct. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2735471.2735478

[28] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, *WISEBED: An Open Large-Scale Wireless Sensor Network Testbed*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 68–87. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11870-8_6

[29] S. Ezdiani, I. S. Acharyya, S. Sivakumar, and A. Al-Anbuky, "An IoT Environment for WSN Adaptive QoS," *Proceedings - 2015 IEEE International Conference on Data Science and Data Intensive Systems; 8th IEEE International Conference Cyber, Physical and Social Computing; 11th IEEE International Conference on Green Computing and Communications and 8th IEEE Inte*, pp. 586–593, 2016. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7396561

[30] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7123563

[31] G. M. Dias, B. Bellalta, and S. Oechsner, "A survey about prediction-based data reduction in wireless sensor networks," in *ACM Computing Surveys*, 2016. [Online]. Available: https://arxiv.org/abs/1607.03443

[32] F. A. Aderohunmu, G. Paci, D. Brunelli, J. D. Deng, L. Benini, and M. Purvis, "An Application-Specific Forecasting Algorithm for Extending WSN Lifetime," *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pp. 374–381, May 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6569459

[33] R. Askari Moghadam and M. Keshmirpour, "Hybrid arima and neural network model for measurement estimation in energy-efficient wireless sensor networks," in *Informatics Engineering and Information Science*, ser. Communications in Computer and Information Science, A. Abd Manaf, S. Sahibuddin, R. Ahmad, S. Mohd Daud, and E. El-Qawasmeh, Eds. Springer Berlin Heidelberg, 2011, vol. 253, pp. 35–48. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-25462-8_4

[34] A. Mahmood, K. Shi, S. Khatoon, and M. Xiao, "Data mining techniques for wireless sensor networks: A survey," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013. [Online]. Available: http://www.hindawi.com/journals/ijdsn/2013/406316/abs/

[35] G. M. Dias, S. Oechsner, and B. Bellalta, *A Centralized Mechanism to Make Predictions Based on Data from Multiple WSNs*. Cham: Springer International Publishing, Aug. 2015, pp. 19–32. [Online]. Available: https://arxiv.org/abs/1407.0981

[36] Crossbow Technology Inc., *TelosB - TelosB Mote Platform*, 2004, revision B. [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf

[37] G. M. Dias, C. B. Margi, F. C. P. Oliveira, and B. Bellalta, "Cloud Empowered Self-Managing WSNs," *IEEE Communications Magazine*, Dec. 2016. [Online]. Available: https://arxiv.org/abs/1607.03607

[38] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1061322

[39] T. Minh, B. Bellalta, and M. Oliver, "DISON: A Self-organizing Network Management Framework for Wireless Sensor Networks," *Ad Hoc Networks*, vol. 111, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36958-2_11

[40] H. Silva, A. Hahn Pereira, Y. Solano, B. T. de Oliveira, and C. B. Margi, "WARM: WSN application development and resource management," in *XXXIV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2016) (SBrT 2016)*, Santarém, Brazil, Aug. 2016.

[41] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel, "Tinysdn: Enabling multiple controllers for software-defined wireless sensor networks," in *2014 IEEE Latin-America Conference on Communications (LATINCOM)*, Nov. 2014, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7041885

[42] G. M. Dias, T. Adame, B. Bellalta, and S. Oechsner, "A self-managed architecture for sensor networks based on real time data analysis," in *2016 IEEE Future Technologies Conference*. IEEE, Dec. 2016. [Online]. Available: https://arxiv.org/abs/1605.09011

[43] M. McNeil and contributors, *Sails.js — Realtime MVC Framework for Node.js*, 2016. [Online]. Available: http://sailsjs.org

[44] AngularJS, "Angularjs - superheroic javascript mvw framework," 2016. [Online]. Available: https://angularjs.org

[45] I. Riverbed Technology, “Riverbed modeler,” jun 2016. [Online]. Available: http://www.riverbed.com/products/steelcentral/steelcentral-riverbed-modeler.html

[46] I. Google, “Prediction api - pattern matching in the cloud,” 2016. [Online]. Available: https://cloud.google.com/prediction

[47] I. Amazon Web Services, “Amazon machine learning - predictive analytics with aws,” 2016. [Online]. Available: https://aws.amazon.com/machine-learning

[48] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org

[49] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, “Evaluating probabilistic queries over imprecise data,” in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’03. New York, NY, USA: ACM, 2003, pp. 551–562. [Online]. Available: http://doi.acm.org/10.1145/872757.872823

[50] F. Emekci, S. E. Tuna, D. Agrawal, and A. E. Abbadi, “Binocular: A system monitoring framework,” in *Proceeedings of the 1st International Workshop on Data Management for Sensor Networks: In Conjunction with VLDB 2004*, ser. DMSN ’04. New York, NY, USA: ACM, 2004, pp. 5–9. [Online]. Available: http://doi.acm.org/10.1145/1052199.1052201

[51] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, “Model-driven data acquisition in sensor networks,” in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB ’04. VLDB Endowment, 2004, pp. 588–599. [Online]. Available: http://dl.acm.org/citation.cfm?id=1316689.1316741

[52] L. B. Yann-Ael, B. Gianluca, and G. Bontempi, "Round Robin Cycle for Predictions in Wireless Sensor Networks," in *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. IEEE, 2005, pp. 253–258. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1595588

[53] D. Tulone and S. Madden, "Paq: Time series forecasting for approximate query answering in sensor networks," *Proceedings of the Third European Conference on Wireless Sensor Networks*, pp. 21–37, 2006. [Online]. Available: http://dx.doi.org/10.1007/11669463_5

[54] H. Malik, A. S. Malik, and C. K. Roy, "A methodology to optimize query in wireless sensor networks using historical data," *Journal of Ambient Intelligence and Humanized Computing*, vol. 2, no. 3, pp. 227–238, Jun. 2011. [Online]. Available: http://link.springer.com/10.1007/s12652-011-0059-x

[55] G. M. Dias, M. Nurchis, and B. Bellalta, "Adapting sampling interval of sensors using reinforcement learning," in *2016 IEEE World Forum on Internet of Things*. IEEE, Dec. 2016. [Online]. Available: https://arxiv.org/abs/1606.02193

[56] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[57] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel lab data," Jun. 2004, online dataset. [Online]. Available: http://db.lcs.mit.edu/labdata/labdata.html

[58] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[59] A. Varga, "The OMNeT++ discrete event simulation system," *Proceedings of the European Simulation Multiconference (ESM'2001)*, no. S 185, p. 65, 2001. [Online]. Available: https://labo4g.enstb.fr/twiki/pub/Simulator/SimulatorReferences/esm2001-meth48.pdf

[60] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. K. Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin, "Simulating wireless and mobile networks in omnet++ the mixim vision," pp. 71:1–71:8, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1416222.1416302

[61] G. M. Dias, B. Bellalta, and S. Oechsner, "Towards information-centric wsn simulations," in *Proceedings of the 1st OMNeT++ Community Summit 2014*, Sep. 2014. [Online]. Available: http://arxiv.org/abs/1409.1001

[62] Libelium, *Waspmote - Wireless Sensor Networks 802.15.4 ZigBee Mote - Open Source Sensor Device — Libelium*. [Online]. Available: http://www.libelium.com/v11-files/documentation/waspmote/waspmote-datasheet_eng.pdf

[63] S. Makridakis and M. Hibon, "The M3-Competition: results, conclusions and implications," *International Journal of Forecasting*, vol. 16, no. 4, pp. 451–476, Oct. 2000. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0169207000000571

[64] F. A. Aderohunmu, G. Paci, D. Brunelli, J. D. Deng, and L. Benini, "Prolonging the lifetime of wireless sensor networks using light-weight forecasting algorithms," in *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. IEEE, Apr. 2013, pp. 461–466. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6529834

[65] G. M. Dias, B. Bellalta, and S. Oechsner, "On the importance and feasibility of forecasting data in sensors," in *Transactions*

*on Mobile Computing Journal*, 2016. [Online]. Available: https://arxiv.org/abs/1604.01275

[66] S. Santini and K. Römer, "An Adaptive Strategy for Quality-Based Data Reduction in Wireless Sensor Networks," in *3rd International Conference on Networked Sensing Systems*, 2006, pp. 29 – 36. [Online]. Available: http://vs.inf.ethz.ch/publ/papers/santinis_inss2006.pdf

[67] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *Data Engineering, 2003. Proceedings. 19th International Conference on*, Mar. 2003, pp. 429–440. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1260811

[68] B. R. Stojkoska, D. Solev, and D. Davcev, "Data prediction in WSN using variable step size LMS algorithm," in *Proceedings of the 5th International Conference on Sensor Technologies and Applications*, 2011.

[69] M. Wu, L. Tan, and N. Xiong, "Data prediction, compression, and recovery in clustered wireless sensor networks for environmental monitoring applications," *Information Sciences*, vol. 329, pp. 800–818, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2015.10.004

[70] Y.-A. Le Borgne, S. Santini, and G. Bontempi, "Adaptive model selection for time series prediction in wireless sensor networks," *Signal Process.*, vol. 87, no. 12, pp. 3010–3020, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2007.05.015

[71] G. Li and Y. Wang, "Automatic ARIMA modeling-based data aggregation scheme in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 85, 2013. [Online]. Available: http://jwcn.eurasipjournals.com/content/2013/1/85

[72] D. J. McCorrie, E. Gaura, K. Burnham, N. Poole, and R. Hazelden, "Predictive data reduction in wireless sensor networks using selective filtering for engine monitoring," in *Wireless Sensor and Mobile Ad-Hoc Networks: Vehicular and Space Applications*. New York, NY: Springer New York, 2015, pp. 129–148. [Online]. Available: http://dx.doi.org/10.1007/978-1-4939-2468-4_6

[73] S. Goel and T. Imielinski, "Prediction-based monitoring in sensor networks: taking lessons from MPEG," *ACM SIGCOMM Computer Communication Review*, vol. 1, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=1037117

[74] C. Liu, K. Wu, and M. Tsao, "Energy efficient information collection with the ARIMA model in wireless sensor networks," in *IEEE Global Communications Conference*, 2005, pp. 2470–2474. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1578206

[75] H. Jiang, S. Jin, and C. Wang, "Prediction or Not? An Energy-Efficient Framework for Clustering-based Data Collection in Wireless Sensor Networks," *IEEE Transactions on Communications*, vol. 39, no. 12, pp. 1721–1725, 1991. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5601711

[76] "Scale of measurement," in *Encyclopedia of Public Health*, W. Kirch, Ed. Dordrecht: Springer Netherlands, 2008, pp. 1279–1279. [Online]. Available: http://dx.doi.org/10.1007/978-1-4020-5614-7_3099

[77] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128608001254

[78] J.-K. Min and C.-W. Chung, "EDGES: Efficient data gathering in sensor networks using temporal and spatial correlations," *Journal*

*of Systems and Software*, vol. 83, no. 2, pp. 271–282, Feb. 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0164121209001964

[79] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: Out-of-the-Box Environmental Monitoring," in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*. IEEE, Apr. 2008, pp. 332–343. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4505485

[80] S. Bhatti and J. Xu, "Survey of Target Tracking Protocols Using Wireless Sensor Network," *2009 Fifth International Conference on Wireless and Mobile Communications*, pp. 110–115, 2009. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5279476

[81] G. L. Baker and J. A. Blackburn, *The pendulum: A case study in physics*. Oxford University Press, 2005. [Online]. Available: http://www.maa.org/publications/maa-reviews/the-pendulum-a-case-study-in-physics

[82] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, Mar 1996. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=485891

[83] S. Haykin, *Neural networks and learning machines*. Pearson Upper Saddle River, NJ, USA:, 2009, vol. 3.

[84] Crossbow Technology Inc., *MICA2DOT datasheet*, 2002. [Online]. Available: https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2dot.pdf

[85] Sensirion AG, *Datasheet SHT1X*, Dec. 2011. [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/SHT1x_datasheet.pdf

[86] Crossbow Technology Inc., *Wireless Sensor Networks Product Reference Guide*, 2007.

[87] Shockfish SA, *TinyNode 584 / Standard Extension Board*, 2005. [Online]. Available: http://www.btnode.ethz.ch/pub/uploads/Projects/TinyNode_Users_Manual_rev11.pdf

[88] Schmitt Industries Inc., *AR3000 Distance Measurement Sensor*, 2010. [Online]. Available: http://www.naic.edu/~phil/hardware/lr/accurange3000/ar3000-data-sheet.pdf

[89] G. Xu, *GPS*, 2nd ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 1. [Online]. Available: http://link.springer.com/10.1007/978-3-540-72715-6

[90] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks*. IEEE (Comput. Soc.), Nov. 2004, pp. 455–462. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1367266

[91] G. M. Dias, B. Bellalta, and S. Oechsner, "The impact of dual prediction schemes on the reduction of the number of transmissions in sensor networks," in *Computer Communications Journal*, 2016. [Online]. Available: https://arxiv.org/abs/1509.08778

[92] K. Langendoen and A. Meier, "Analyzing mac protocols for low data-rate applications," *ACM Trans. Sen. Netw.*, vol. 7, no. 2, pp. 19:1–19:40, Sep. 2010. [Online]. Available: http://doi.acm.org/10.1145/1824766.1824775

[93] J. S. Armstrong, *Principles of forecasting: a handbook for researchers and practitioners*. Springer Science & Business Media, 2001. [Online]. Available: https://www.gwern.net/docs/predictions/2001-principlesforecasting.pdf

[94] A. Genz, "Comparison of Methods for the Computation of Multivariate Normal Probabilities," *Computing Sciences and Statistics*, vol. 25, pp. 400 – 405, 1993. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.9631&rep=rep1&type=pdf

[95] G. M. Dias, B. Bellalta, and S. Oechsner, "Predicting occupancy trends in barcelona's bicycle service stations using open data," in *SAI Intelligent Systems Conference (IntelliSys), 2015*, Nov 2015, pp. 439–445. [Online]. Available: https://arxiv.org/abs/1505.03662

[96] N. H. Timm, *Applied multivariate analysis*, 2002. [Online]. Available: http://link.springer.com/content/pdf/10.1007/b98963.pdf

[97] H. Akaike, "A new look at the statistical model identification," *Automatic Control, IEEE Transactions on*, 1974. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1100705

[98] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, 1978. [Online]. Available: http://projecteuclid.org/euclid.aos/1176344136

[99] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, Oct. 2006. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0169207006000239

[100] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. John Wiley & Sons, 2015.

[101] S. G. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting: Methods and Applications*, 3rd ed., 1998.

[102] R. J. Hyndman, A. B. Koehler, J. Ord, and R. D. Snyder, *Forecasting with Exponential Smoothing: The State Space Approach*, 2008.

[103] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, no. 2, pp. 188–228, Jun. 2002. [Online]. Available: http://doi.acm.org/10.1145/568518.568520

[104] S. Y. Kang, "An investigation of the use of feedforward neural networks for forecasting," Ph.D. dissertation, Kent, OH, USA, 1992, uMI Order No. GAX92-01899.

[105] W. R. Tim Hill, Marcus O'Connor, "Neural network models for time series forecasts," *Management Science*, vol. 42, no. 7, pp. 1082–1092, 1996. [Online]. Available: http://www.jstor.org/stable/2634369

[106] C. Pham, "Communication performances of IEEE 802.15.4 wireless sensor motes for data-intensive applications: A comparison of WaspMote, Arduino MEGA, TelosB, MicaZ and iMote2 for image surveillance," *Journal of Network and Computer Applications*, vol. 46, pp. 48–59, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2014.08.002

[107] MEMSIC Inc., *MICAz datasheet: 6020-0065-05 rev*, San Jose, CA, California, 2003.