

Fault-Tolerant Spatio-Temporal Compression Scheme for Wireless Sensor Networks

Vom Fachbereich Informatik der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades eines Doktor-Ingenieur
(Dr.-Ing.) vorgelegt von

M. Sc. Azad Ali

aus Sanghar, Pakistan

Referenten:

Prof. Neeraj Suri, Ph.D.

Prof. Christian Becker, Ph.D.

Datum der Einreichung: 07. Sep 2016

Datum der mündlichen Prüfung: 04. Nov 2016

Darmstadt 2016

D17

ABSTRACT

Wireless sensor networks are often deployed for environmental sampling and data gathering. A typical wireless sensor network consists, from hundreds to thousands, of battery powered sensor nodes fitted with various sensors to sample the environmental attributes, and one or more base stations, called the sink. Sensor nodes have limited computing power, memory and battery. Sensor nodes are wirelessly interconnected and transmit the sampled data in a multihop fashion to the sink. The sheer number of sensor nodes and the amount of sampled data can generate enormous amount of data to be transmitted to the sink, which subsequently can transform into network congestion problem resulting into data losses and rapid battery drain. Hence, one of the main challenges is to reduce the number of transmissions both to accommodate to the network bandwidth and to reduce the energy consumption. One possibility of reducing the data volume would be to reduce the sampling rates and shutdown sensor nodes. However, it can affect the spatial and temporal data resolution. Hence, we propose a compression scheme to minimize the transmissions instead of reducing the sampling.

The sensor nodes are vulnerable to external/environmental effects and, being relatively cheap, are susceptible to various hardware faults, e.g., sensor saturation, memory corruption. These factors can cause the sensor nodes to malfunction or sample erroneous data. Hence, the second biggest challenge in data gathering is to be able to tolerate such faults.

In this thesis we develop a spatio-temporal compression scheme that detects data redundancies both in space and time and applies data modeling techniques to compress the data to address the large data volume problem. The proposed scheme not only reduces the data volume but also the number of transmissions needed to transport the data to the sink, reducing the overall energy consumption. The proposed spatio-temporal compression scheme has the following major components:

Temporal Data Modeling: Models are constructed from the sampled data of the sensor nodes, which are then transmitted to the sink instead of the raw samples. Low computing power, limited memory and battery force us to avoid computationally expensive operations and use simple models, which offer limited data compressibility (fewer samples are approximated). However, we are able to extend the compressibility in time through our model caching scheme while maintaining simple models.

Hierarchical Clustering: The data sampled by the sensor nodes is often not only temporally correlated but also spatially correlated. Hence, the sensor nodes are

initially grouped into 1-hop clusters based on sampled data. Only a single model is constructed for one cluster, essentially reducing the sampled data of all the sensor nodes to a single data model. However, we also observed through experiments that the data correlations often extend beyond 1-hop clusters. Hence, we devised a hierarchical clustering scheme, which uses the model of one 1-hop cluster to also approximate the sampled data in the neighboring clusters. All the 1-hop clusters approximated by a given model are grouped into a larger cluster. The devised scheme determines the clusters that can construct the data models, the dissimilation of the model to the neighboring clusters and finally the transmission of the data model to the sink. The accuracy of data to the single sensor node level is maintained through outliers for each sensor node, which are maintained by the cluster heads of the respective 1-hop clusters and cumulatively transmitted to the sink.

The proposed spatio-temporal compression scheme reduces the total data volume, is computationally inexpensive, reduces the total network traffic and hence minimizes the overall energy consumption while maintaining the data accuracy as per the user requirements.

This thesis also addresses the second problem related to data gathering in sensor networks caused by the faults that results into data errors. We have developed a fault-tolerance scheme that can detect the anomalies in the sampled data and classify them as errors and can often correct the resulting data errors. The proposed scheme can detect data errors that may arise from a range of fault classes including sporadic and permanent faults. It is also able to distinguish the data patterns that may occur due to both the data errors and a physical event. The proposed scheme is quite light weight as it exploits the underlying mechanisms already implemented by spatio-temporal compression scheme. The proposed fault-tolerance scheme uses the data models constructed by the compression scheme to additionally detect data errors and subsequently correct the erroneous samples.

KURZFASSUNG

Für die Erfassung von Umweltdaten kommen häufig Sensornetzwerke zum Einsatz. Derartige Netzwerke bestehen typischerweise aus einigen hundert bis zu mehreren tausend batteriebetriebenen Sensorknoten (Quellen), welche die nötige Sensorik zur Erfassung der Umwelteigenschaften enthalten, sowie einer oder mehreren Basisstationen (Senken). Die Sensorknoten verfügen über eine begrenzte Rechenleistung, wenig Speicher und kleine Batterien. Sie sind untereinander drahtlos vernetzt und übertragen die gesammelten Daten über mehrere Sensorknoten, die als Zwischenstationen fungieren, zur Basisstation. Durch die enorme Zahl von Sensorknoten können riesige Datenmengen entstehen. In Folge kommt es zu Datenstaus oder gar zum Verlust von Messwerten und zu einer verkürzten Laufzeit der Sensorknoten durch erhöhten Stromverbrauch für die Übertragung. Eine zentrale Herausforderung bei Sensornetzwerken stellt somit die Optimierung der Datenübertragung hinsichtlich Bandbreite und Energieverbrauch dar. Eine einfache Methode, das Datenaufkommen zu reduzieren, bestünde in der Verringerung der Abtastrate und dem zeitweisen Abschalten von Sensorknoten. Allerdings würde dies die zeitliche und räumliche Auflösung der Ergebnisse beeinträchtigen. Wir schlagen daher stattdessen eine Komprimierung der aufgenommenen Daten vor. Aufgrund ihres kostenoptimierten Aufbaus sind die Sensorknoten störanfällig gegenüber externen Umwelteinflüssen. Auch interne Defekte wie zum Beispiel Sensorsättigung oder Speicherfehler können auftreten. Diese Faktoren führen zu verfälschten Messergebnissen und im schlimmsten Fall zum Ausfall von Sensorknoten. Eine weitere Herausforderung ist somit die Fehlertoleranz des Netzwerkes gegenüber derartigen Fehlern. In dieser Arbeit entwickeln wir ein Komprimierungsverfahren basierend auf der Erkennung von räumlichen und zeitlichen Redundanzen. Durch Anwendung von Datenmodellierungsverfahren reduziert sich die Datenmenge, die Anzahl der nötigen Übertragungen wird verringert und der Gesamtenergiebedarf gesenkt. Diese räumlich-zeitliche Komprimierung gliedert sich in 2 Bereiche:

Zeitliche Datenmodellierung: Die aufgenommenen Sensordaten werden vor dem Versenden in Modelle überführt und anstatt der Rohwerte zu den Basisstation übertragen. In Anbetracht der begrenzten Ressourcen auf den Sensorknoten verbietet sich die Nutzung komplexer Berechnungen. Zum Einsatz kommen daher einfache Modelle mit etwas geringerem Komprimierungspotential, deren Optimierung auf Basis einer geringeren Anzahl von Messwerten erfolgt. Es gelingt uns allerdings, die zeitliche Komprimierung durch effiziente Zwischenspeicherung der Modelle zu erhöhen. *Hierarchische Gruppierung:* Die gewonnenen Sensordaten

sind häufig nicht nur zeitlich, sondern auch räumlich korreliert. Dadurch bietet sich eine räumliche Gruppierung an. Unter Verwendung der aufgenommenen Daten werden die Sensorknoten initial in sogenannte '1-hop' Cluster zusammengefasst. Pro Cluster wird auf Basis der Rohwerte aller enthaltenen Sensorknoten jeweils nur ein einziges Datenmodell generiert. Experimente zeigten darüber hinaus oft auch übergeordnete Abhängigkeiten zwischen Messwerten außerhalb dieser ersten Ebene. Wir setzen daher auf eine hierarchische Gruppierung, welche die Daten benachbarter Cluster analog der Vorgehensweise der ersten Ebene kombiniert und somit Cluster höherer Ordnung bildet. Das beschriebene Verfahren beinhaltet die Ermittlung gruppierbarer Cluster, die Ausweitung des Datenmodells auf passende Nachbarn sowie die finale Übermittlung der komprimierten Daten an die Basisstationen. Um Fehler durch die Prozessierung zu vermeiden, wird die Güte der finalen Daten unter Einbeziehung von Ausreißern auf Einzelsensorebene gewährleistet. Die Hauptknoten der '1-hop' Cluster sammeln entsprechende Daten und senden diese kumuliert an die Basisstationen. Das vorgeschlagene räumlich-zeitliche Komprimierungsverfahren reduziert die Gesamtdatenmenge, benötigt wenig Rechenleistung, minimiert die Anzahl der Übertragungen und damit auch den Energieverbrauch des Netzwerkes. Dabei lässt sich die Genauigkeit der gewonnenen Daten flexibel an die Bedürfnisse des Nutzers anpassen. Neben der Komprimierung beschreiben wir in dieser Arbeit auch Methoden zur Erhöhung der Fehlertoleranz gegenüber internen und externen Einflüssen. Unregelmäßigkeiten in den Messwerten werden erkannt, als Fehler klassifiziert und können in vielen Fällen korrigiert werden. Das vorgeschlagene Verfahren deckt eine breite Palette möglicher Fehlerfälle ab und verarbeitet sowohl sporadische als auch permanente Abweichungen. Dabei ist es in der Lage, Fehler von realen, durch plötzliche Änderung der zu messenden physikalischen Größen hervorgerufenen, Abweichungen zu unterscheiden. Der verwendete Ansatz ist sehr effizient, da er sich der Methoden der räumlich-zeitlichen Komprimierung bedient und bereits existierende Datenmodelle zur Erkennung und Korrektur fehlerhafter Daten nutzt.

ACKNOWLEDGEMENTS

It is my pleasure to thank all the people who helped, guided and supported me in the successful completion of this thesis. This research would not have been possible without their contributions.

My greatest gratitude goes to my supervisor, *Prof. Dr. Neeraj Suri*, whose encouragement, guidance and support from the initial to the final stage of my work enabled me to develop an understanding of the subject. I would really like to appreciate his willingness to support me beyond my studies, specially in supporting me to bring my family to Germany, which allowed me to better concentrate on my research work. I would like to thank *Dr. Abdelmajid Khelil* for his valuable advice and inspiring discussions on my research work. A great many thanks also to *Prof. Christian Becker* for accepting to be my co-advisor.

Then, I would like to thank all DEEDS group's members. Many thanks to *Faisal, Ibrahim, Hamza, Ripon, Dinu, Dan, Marco, Peter, Matthias, Piotr, Stefan, Daniel, Thorsten, Kubilay, Mohammadreza* and my colleague *Martin*. Also, special thanks go to *Ute* and *Sabine* for helping me with various paperwork and all other circumstances related to living in Germany.

I would also like to thank all my teachers, advisors and friends, who supported me during my education in Pakistan. Had it not been for them, I would not have been able to accomplish what I have achieved so far.

My sincere and heartfelt thanks go to my father, *Muhammad Hassan*, and my mother, *Imamzadi*. I cannot possibly pay them back or even appreciate them enough for their love, encouragement and the role they played in making me what I am today. I would also like to thank my brother, *Mahboob*, and sisters, *Due-e-Shahwar, Sadaf* and *Nobahar*, for their endless love.

Finally, I would like to thank my beloved wife, *Sana*, and our beloved daughters, *Mariyam* and *Sophia*, for all the understanding, love, patience, support and happiness. They have always been the source of encouragement and inspiration for me to strive further. Thank you for believing in me.

CONTENTS

1	INTRODUCTION AND PROBLEM CONTEXT	3
1.1	Delay-Tolerant Spatio-Temporal Compression:	5
1.2	Fault-Tolerance for Data Compression	6
1.3	Fault-Tolerant Spatio-Temporal Data Compression Framework	7
1.4	Thesis Contributions	9
1.4.1	(C1) – Fault-Tolerant Spatio-Temporal Data Compression Framework	9
1.4.2	(C2) – Modularity of the Proposed Framework	9
1.4.3	(C3) – Delay- Tolerant Spatio-Temporal Compression Scheme	10
1.4.4	(C4) – Fault-Tolerance for Data Compression	11
1.4.5	(C5) – Event Prediction in WSN	12
1.5	Publications Resulting from the Thesis	12
1.6	Thesis Structure	13
2	STATE OF THE ART AND PRACTICE	15
2.1	Delay-Tolerant Spatio-Temporal Data Collection	16
2.1.1	Spatial Compression:	16
2.1.2	Temporal Compression:	16
2.1.3	Spatio-Temporal Compression:	17
2.2	Fault-Tolerance for Data Gathering Schemes	19
2.2.1	Handling Functional Faults in WSN	19
2.2.2	Handling Data Errors in WSN	20
2.3	Predictive Monitoring of WSN	21
2.4	Chapter Summary	23
3	SYSTEM, DATA AND FAULT MODELS	25
3.1	Design Requirements	25
3.2	System Model	26
3.3	Data Modeling	27
3.4	Fault Modeling - Fault Classes and Data Errors	31
3.4.1	NOISE	31
3.4.2	SHORT	31
3.4.3	CONSTANT	32
3.4.4	Transient Data Errors	32
3.4.5	Permanent Data Errors	32
3.5	Chapter Summary	32

4	DELAY-TOLERANT SPATIO-TEMPORAL DATA COMPRESSION	33
4.1	The Proposed Adaptive Spatio-Temporal Compression	34
4.1.1	A Guide through the Proposed Adaptive Hybrid Compression (AHC) Scheme	35
4.1.2	Stage 1: 1-Hop Cluster Formation	36
4.1.3	Stage 2: Temporal Modeling in 1-Hop Clusters	38
4.1.4	Stage 3: Merging 1-Hop Clusters Based on Master Cluster Model Cache	47
4.2	Efficiency and Compressibility Analysis	56
4.2.1	Message Payload	56
4.2.2	Maximum Theoretical Compressibility and Efficiency	58
4.3	Message and Computation Cost for Compression	59
4.3.1	Message Overhead	59
4.3.2	Computation Overhead	61
4.4	Experiments and Discussion	62
4.4.1	Simulation Settings	62
4.4.2	Experimental Performance Evaluation	65
4.4.3	Comparison to Related Work	71
4.4.4	Versatility - Phenomena Independence	74
4.5	Chapter Summary	74
5	FAULT-TOLERANT DATA COMPRESSION	77
5.1	Fault Abstraction and Modeling	77
5.1.1	The Fault Abstraction	78
5.1.2	Data Error Detection Models	79
5.2	Fault Vulnerability of Compression Schemes	81
5.3	Model-based Data-Fault Tolerance	81
5.3.1	Sporadic Data Errors	81
5.3.2	Complete Data Corruption	88
5.4	Performance Evaluation	89
5.4.1	Simulation Settings	90
5.4.2	Results	92
5.5	Chapter Summary	95
6	EFFICIENT PREDICTIVE MONITORING OF WSN	97
6.1	Introduction	97
6.2	System Model	99
6.3	Efficient Predictive Monitoring	99
6.3.1	Data Collection Phase	100
6.3.2	The Prediction Phase	100
6.3.3	The Event Detection Phase	103

6.4	Case Study: Network Partition Prediction	107
6.4.1	Problem Formulation	107
6.4.2	The Data Collection Phase	108
6.4.3	The Prediction Phase	108
6.4.4	The Event (Holes and Partition) Detection Phase	109
6.4.5	Simulation Settings	109
6.4.6	Simulation Results	110
6.5	Chapter Summary	115
7	CONCLUSIONS AND FUTURE RESAERCH	117
7.1	Overall Thesis contributions	118
7.1.1	Delay-Tolerant Spatio-Temporal Data Compression	118
7.1.2	Fault-Tolerant Data Compression	118
7.1.3	Efficient Predictive Monitoring	119
7.2	Future Work	120
	BIBLIOGRAPHY	123
	INDEX	136

LIST OF FIGURES

Figure 1.1	Phenomenon Distribution	4
Figure 1.2	Fault-Tolerant Spatio-Temporal Data Compression Framework	8
Figure 3.1	Original Signal and Liner Component	28
Figure 3.2	Original Signal and Liner Component	28
Figure 4.1	Phenomenon Distribution	34
Figure 4.2	Clusters, Master Clusters and Correlated Regions	36
Figure 4.3	1-Hop Clusters	37
Figure 4.4	Adaptive Modeling	40
Figure 4.5	Dynamic Approximation Window	43
Figure 4.6	Model Cache	45
Figure 4.7	Message Payload Format	57
Figure 4.8	Total Message Cost for Inter- and Intra-Cluster Communication, Cache Construction and Caches and Outliers Transportation of Temperature Data to the Sink	67
Figure 4.9	Message Cost for Inter- and Intra-Cluster Communication, Cache Construction and Transportation of Temperature Data to the Sink for Various $O_{\#max}$	69
Figure 4.10	Outliers Message Cost for Temperature Data Set to the Sink for Various $O_{\#max}$	70
Figure 4.11	Impact of ϵ on Message Cost on ASTC and AHC	71
Figure 4.12	Message Cost Comparison Between Various Compression Scheme for Different Error Thresholds ϵ	72
Figure 4.13	Mean Approximation Error on Sink for Temperature Data set	73
Figure 4.14	Message Cost for Compressing and Transporting Humidity Data to the Sink	75
Figure 4.15	Mean Approximation Error on Sink for Humidity Data Set	76
Figure 5.1	Reference Data vs Model Approximation	78
Figure 5.2	Mean Approximation Error Relative to Reference Data	92
Figure 5.3	FTDE-PAQ Message Cost	93
Figure 5.4	FTDE-ASTC Accuracy and Message Cost	95
Figure 6.1	Temporal stack of the grid maps	101
Figure 6.2	Max grid size	109
Figure 6.3	Mean square error accuracy measures	111

Figure 6.4	Misclassification cell count	111
Figure 6.5	Misclassification percentile error per region	112
Figure 6.6	Cumulative Cluster during Data Collection	114
Figure 6.7	Cumulative Outliners for DCF	114

LIST OF ALGORITHMS

4.1	Dynamic Model Order Selection	41
4.2	Model Cache Acceptance	50
4.3	Clique Expansion	53
4.4	Clique Joining by Neighboring Cluster	54
5.1	Error and Outliers Detection Algorithm	82
5.2	Classification of preliminarily faults into fault and phenomenon events	87
6.1	CRA: Centralized Regioning Algorithm (On the Sink)	106

LIST OF TABLES

Table 1	Cluster Notations	36
Table 2	Modeling Notations	39
Table 3	Efficiency and Compressibility Analysis Notations	57

1

INTRODUCTION AND PROBLEM CONTEXT

In Wireless Sensor Network (WSN) deployments, battery powered sensor nodes are often distributed over the area of interest for varied applications involving unattended environmental monitoring and supervisory functions. The sensor nodes regularly update a central station, termed as the sink, with the sampled environmental data for subsequent processing and analysis. The (multihop) data delivery, from the nodes to the sink, requires exchange of several messages thus depleting the batteries of involved sensor nodes, and reducing the residual network lifetime. Hence, the dual target of continuous collection of data while prolonging the network lifetime is a challenging problem.

Fortunately, WSNs naturally generate highly redundant spatial samples due to (a) the redundant sensor node deployment for connectivity and failure tolerance, and (b) the correlated values of the environmental attribute over larger spatial areas. This is also substantiated by our observation of available real world data [S. Madden, 2003](#), where we noticed that the trends/patterns are similar for a large number of nodes in a given physical neighborhood. [Fig. 1.1](#) depicts the sensor readings of two pairs of nodes, from [S. Madden, 2003](#), over four days time period. The nodes in each pair are 1-hop neighbors and both pairs of nodes are separated by several hops. The sampled attribute values also exhibit temporal correlation [Tulone and Samuel Madden, 2006](#) that can be exploited by constructing the models of the data and reporting the model parameters instead of raw sample values.

Applications requiring continuous data collection utilize the data for two prominent use cases of (a) live/real-time decision making, such as surveillance, or (b) offline/delay-tolerant processing such as modeling, analysis [Tolle et al., 2005](#) and inference [Ali, Khelil, Shaikh, et al., 2009](#). The focus of our work is on designing data compression scheme while exploit the delay-tolerant in data collection. Various WSNs deployed for scientific monitoring [Akyildiz et al., 2005](#); [Tolle et al., 2005](#); [Werner-Allen et al., 2005](#) continuously harvest data for modeling, analysis and simulations. They generally tolerate a certain data collection latency. Hence, for such applications real-time data acquisition does not have precedence over the quality and quantity of the data. This *delay-tolerance* in reporting the data to the sink opens up a fundamental design flexibility in data collection to significantly improve WSN energy efficiency. We propose an adaptive hybrid compression scheme that explicitly exploits the delay-tolerance in data collection to compress

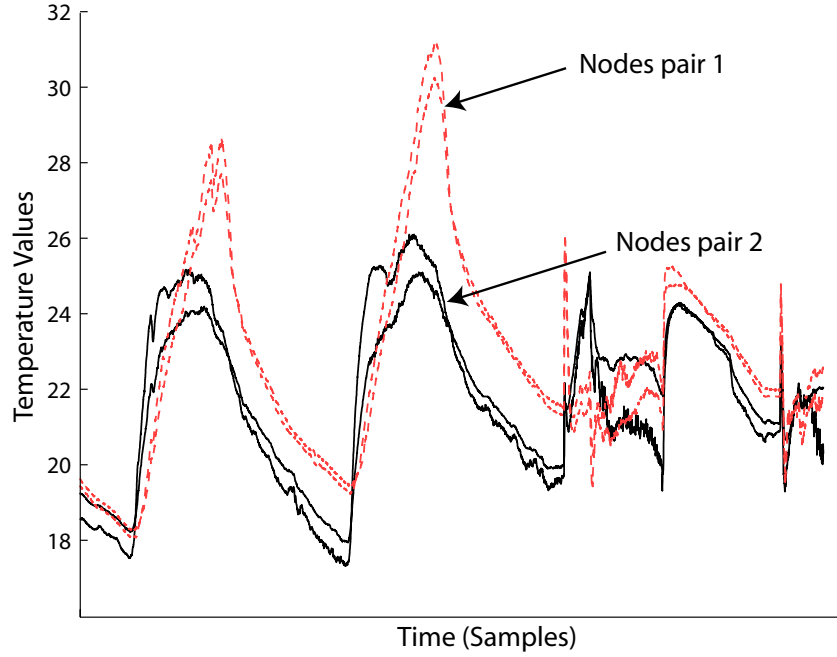


Figure 1.1: Phenomenon Distribution

the data both in time and space, which gives us considerable benefits over the real-time schemes that require frequent updates to be synchronized with the sink.

A number of approaches such as aggregation, temporal compression [Jindal and Psounis, 2006](#); [Yonggang J. Zhao et al., 2002](#) and spatial compression [Deshpande et al., 2004](#); [C. Zhang et al., 2006](#) have been proposed to reduce the data volume to be transported to the sink. However, only a few composite spatio-temporal approaches, such as [Ali et al., 2011a](#); [Gedik et al., 2007](#); [Tulone and Samuel Madden, 2006](#); [C. Wang, Ma, Y. He, and Shuguang Xiong, 2012](#); [Yoon and Shahabi, 2007](#), exist. These approaches exploit both spatial and temporal correlation in data, thus reaching higher data traffic reduction. However, the existing approaches are (a) partially centralized limiting local decision making for adaptability, (b) tailored to specific attribute dynamics, providing none or limited self-adaptability, or (c) limited in their scope of exploiting either spatial or temporal correlation reducing compressibility. [Ali et al., 2011a](#) is the only work to consider delay-tolerance in spatio-temporal data compression. However, it is limited due to the use of static models and limited scope for data compression.

WSNs, consisting of cheap and unreliable components, are prone to various kinds of faults, including faults that result in data errors. The data errors significantly impact the accuracy of the data and hence the performance of the compression schemes. Data compression schemes, like time-series based compression, is

prone to erroneous data as they use the sampled data to construct the models that are used for compression. However, if the reference data itself is corrupt, the resulting models will also be affected, often significantly affecting the accuracy of the data when reproduced at the sink. Hence, we propose a fault-tolerance scheme that extends from the compression scheme and can tolerate a wide range of faults by detecting and correcting the sampled data errors. We propose mechanisms that enable the sensor nodes to detect the data errors and often correct them at runtime. The proposed fault-tolerance scheme not only improves the accuracy of the reproduced data but also reduces the data traffic as it manages to remove the noise from the samples and the compression scheme is able to significantly improve its modeling of the data.

Next, we briefly discuss the proposed compression and fault-tolerance schemes and then present the proposed framework that consolidates the proposed schemes.

1.1 DELAY-TOLERANT SPATIO-TEMPORAL COMPRESSION:

We exploit temporal redundancies using simple, computationally inexpensive models considering the limited computation resources of the sensor nodes. The simple models can naturally only approximate a limited number of samples. Hence, in order to exploit the delay-tolerance in data collection, the nodes consecutively construct a batch of simple models to increase the number of approximated sample values. However, our scheme is optimized such that only a small number of nodes need to construct these batches of models to approximate the entire WSN. The spatial redundancy is exploited using a two-level hierarchical clustering. In contrast to existing literature [Gedik et al., 2007](#); [H. Jiang et al., 2011](#); [Pham et al., 2010](#), our proposed clustering is based on models instead of raw sample values. A model-based clustering allows us to evaluate the correlations across the nodes over a longer period of time in comparison to the sample value based clustering which determines correlation only at a given instance of time. Hence, the existing approaches require continuously maintaining large monolithic clusters, incurring additional energy overheads. Using our hierarchical clustering, we initially form 1-hop clusters by grouping nearby nodes with high correlation. The small 1-hop clusters require minimal maintenance. The model batches are constructed on a subset of the 1-hop clusters. The models constructed by these clusters are used to approximate the surrounding clusters based on the user defined error tolerance. The 1-hop clusters approximated by a few batches of models are merged to form the second clustering hierarchy. The second clustering hierarchy depicts the dynamic correlations between the 1-hop clusters and is reconstructed

in each iteration. Following this scheme, only a few model batches approximate the entire network sample values over a long duration of time (depending on the delay-tolerance limits). The model batches are sent to the sink, which uses these models to regenerate the sample values of each sensor node in the network.

The proposed compression scheme is fully decentralized that does not require any intervention from the sink apart from a few initial parameters for setup. It offers adaptive modeling and can approximate a wide variety of environmental attributes within the user defined accuracy requirements by dynamically adapting to tune the parameters. We propose a novel technique of hierarchical clustering based on adaptive models for spatial compression. The clustering adapts to the distribution of the environmental attribute over the sensor network to maximize the spatial compression. As the proposed scheme exploits delay-tolerance in data collection, hence it can enhance the compression both in time and space.

1.2 FAULT-TOLERANCE FOR DATA COMPRESSION

Wireless Sensor Networks (WSNs) encounter varied faults typically being built of low-cost, unreliable components and often deployed in harsh environments. The faults can be hardware, memory, sensor or transceiver defects that result in errors such as node crashes, message corruption, sampling errors, or communication failures such as message losses or network partitioning [Cinque et al., 2013](#)[de Souza et al., 2007](#)[M. Yu et al., 2007](#)[H. Liu et al., 2009](#)[Coronato and Testa, 2013](#). As the core objective of a WSN is to collect data of usable fidelity, hence effective fault handling should address the issue of erroneous/corrupt data. Existing works have documented that a substantial portion of aggregated data (up to 49%) collected in actual WSN deployments was faulty, with up to 3-60% incorrect data for a single sensor [Sharma et al., 2010](#). Such high data corruption rates undermine the usefulness of the data, thus motivating our approach to address data corruption.

A major challenge for battery-powered WSN deployments is the finite energy constraint along with the dominant energy consuming communication operations. Accordingly, our work proposes a fully distributed approach to handle data errors as close to the data source as possible, and to use the data models, instead of instantaneous sensor readings, for error detection and correction. This not only helps us reduce the unnecessary messages cost but also allows to detect and repair data errors efficiently. The proposed fault-tolerance scheme extends from the data compression scheme, hence it is very economical in terms of resources and energy consumption but it is still flexible enough to be adapted to various data collection schemes.

The proposed approach implicitly addresses a large number of potential faults by defining an abstract/composite data error that benefits from the knowledge of recently sensed data, either locally or in the immediate proximity. Using our proposed model based Fault-Tolerance scheme for transient and permanent faults Data Errors (FTDE), we detect the errors with minimal additional communication and computation overhead. Surprisingly, the effective message cost is often negative because (i) the additional message cost incurred is almost negligible, and (ii) our scheme detects the erroneous samples and discards them, which otherwise would result into noise that cannot be approximated by the models and need to be additionally transported to the sink, incurring considerable additional message costs.

1.3 FAULT-TOLERANT SPATIO-TEMPORAL DATA COMPRESSION FRAMEWORK

In this thesis, we propose a comprehensive framework for data compression that can exploit the data redundancy in space and time while using the delay tolerance in data collection. The proposed framework also takes into account underlying faults frequently found in WSNs that corrupt the sampled data and devises mechanism to improve the data accuracy by detecting and correcting the resulting data errors. Fig. 1.2 depicts the proposed framework comprising of various modules for data compression and fault-tolerance. In order to optimize the resource usage and minimize the memory and energy consumption, different modules reuse the mechanisms and network topology created by other modules.

The three main modules of the framework are 1. Temporal Data Compression 2. Spatial Data Compression 3. Fault-Tolerance. Temporal data compression module constructs the data models from the sampled data, as briefly described in Sec. 1.1. Spatial data compression module initially groups the nodes into small 1-hop clusters and then forms the hierarchical clusters in order to further merge the sensor nodes with similar sampled values. Fault-tolerance module consists of the data error detection and correction mechanisms to provide the fault-tolerance, as briefly described in Sec. 1.2. Temporal data compression module provides the foundations for the other two module, i.e., spatial data compression and fault-tolerance modules, with its data models. Accordingly, spatial data compression module exploits the data models from the temporal compression module in the construction of hierarchical clustering. Similarly, fault-tolerance module uses the same data models to detect data errors. It also exploits the existing 1-hop cluster topology, from the spatial data compression module to detect the correlations between the data errors and implements mechanism to correct the data errors.

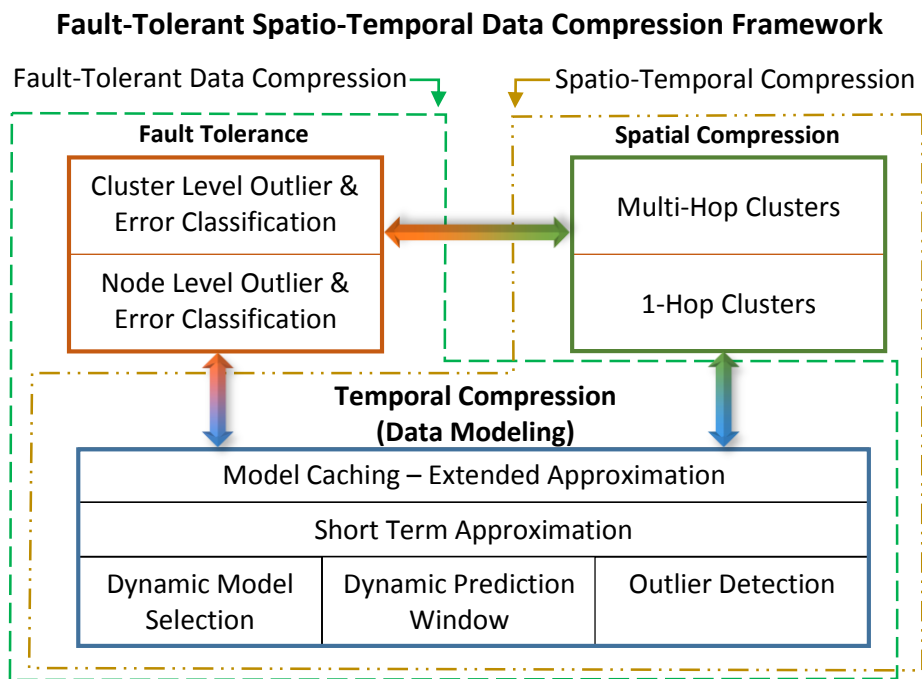


Figure 1.2: Fault-Tolerant Spatio-Temporal Data Compression Framework

1.4 THESIS CONTRIBUTIONS

The research work presented in this thesis makes several important contributions for the wireless sensor networks research community. Next, we briefly present major thesis contributions.

1.4.1 (C1) – Fault-Tolerant Spatio-Temporal Data Compression Framework

The major contribution of this thesis is the development of a comprehensive framework for fault-tolerant data compression in WSN. The proposed framework encompasses various aspects of the addressed problem from modeling, compression, fault identification, error correction to the efficient transportation of the data to the sink. The design criteria for developing the framework are efficiency (maximizing compression and reducing traffic) and accuracy (data corruption detection and correction). In order to increase the data compression and reduce the data transportation cost the modeling and compression schemes are designed to exploit the delay-tolerance. The proposed compression scheme exploits the redundancies inherent in the network, both in space and time, through exploitation of redundant node deployment in space and data correlation in time. The fault-tolerance scheme proposes mechanisms to tolerate the faults that corrupt the sampled data. The proposed fault-tolerance scheme is capable of detecting the data errors occurring due to the underlying faults and thus repair the data corruption. Through the proposed fault-tolerance scheme, we not only improve the data accuracy and meet the user requirements but also reduce data traffic further by removing the noise that incurs additional message cost.

1.4.2 (C2) – Modularity of the Proposed Framework

Continuously changing environmental and network conditions demand adaptability and flexibility for any scheme trying to model it. Accordingly, proposed framework has been designed in a modular fashion, in order to maximize flexibility and adaptability to various applications and environmental conditions. The framework consists of data modeling, spatial compression and fault-tolerance modules.

Modular approach for the framework allows us to be easily adapt it for different applications. Depending on the various network and environmental conditions each module can be tuned independently in order to best fit the data at hand. Different modules can also be used independently or combined together to extend their functionality. For example, application requiring only temporal modeling can skip the spatial compression module and use only temporal com-

pression module, which can be easily extended by combining the fault-tolerance module with the temporal compression module to provide fault-tolerant temporal compression. Similarly, temporal compression module can be combined with the spatial compression module to provide spatio-temporal compression, which can be further extended by combining them with the fault-tolerance module to provide fault-tolerant spatio-temporal compression.

The flexibility provided by the framework allows it to be used in variety of application ranging from simple sparse WSN deployments requirement simple temporal data compression to very large and dense networks requiring spatio-temporal data compression with fault tolerance.

1.4.3 (C3) – Delay- Tolerant Spatio-Temporal Compression Scheme

In the data compression module, we propose an *Adaptive Spatial-Temporal Compression* (ASTC) scheme. The propose scheme is described in detail in Chapter 4. The key idea is to develop an adaptive compression scheme that can exploit the temporal and spatial redundancies of the sampled environmental attribute.

The temporal redundancies are exploited by constructing models from the sampled data. The models are then used instead of raw sampled values, which results in reduced message cost. The spatial redundancy is exploited through two level hierarchical clustering. Initially 1-hop clusters are formed. A subset of the clusters, called the master clusters, are selected to construct the models. The models constructed in the master clusters are distributed to the neighboring clusters. The cluster heads in the neighboring clusters use the received model to approximate the sampled values in their respective clusters. If the sampled values collected by cluster members in a given cluster, can be approximated within the user defined error threshold, the model is accepted by the cluster head and the model is distributed further to the neighboring clusters. However, if the model cannot approximate the sampled values, it is rejected and a new model is constructed within the cluster and selected by the cluster head and the cluster becomes a master cluster itself.

Accordingly, the proposed scheme exploits the spatial data redundancy through hierarchical clustering, consisting of master and standard clusters. The temporal data redundancy is exploited by approximating the sampled data through models and then reporting the models to the sink instead of the raw samples. The models are only constructed by a subset of clusters, the master clusters, which are then used to approximate also the neighboring clusters. The models are collected at the sink to reproduce the data sampled by all the sensor nodes in the network. Specific node discrepancies are adjusted through the outliers sent by the respective nodes to the sink.

1.4.4 (C4) – Fault-Tolerance for Data Compression

We propose a Fault-Tolerance scheme for transient and permanent Data Errors (FTDE) for data compression schemes. FTDE has been discussed in detail in Chapter 5.

The data compression scheme proposed in Chapter 4 is especially vulnerable to data errors, caused by sensor, memory or other faults. The compression scheme relies on the models used to approximate the sampled data. However, if the sampled data is corrupt, the resulting models will also be corrupted, because the model will try to approximate the corrupted data as close as possible. Accordingly, the reproduced data at the sink is also corrupted. Hence, we proposed a fault-tolerance scheme for a wide range of faults that result into data errors. The proposed scheme does not directly fix the faults itself, because often it cannot be fixed without a physical intervention like replacing the sensor or other hardware part. Rather, we focus on the consequence of the faults, i.e., the data errors that result due to the faults. Detecting and correcting the data error implicitly reverts the effect of a fault without needing to actually fix the fault itself. The major strength of the proposed fault-tolerance scheme is that it uses the data models and clusters already in place, being used by the compression scheme, to additionally detect and fix the data errors. Hence, the resulting overhead of the fault-tolerance scheme is minimal.

The proposed scheme is a two stage strategy. The models constructed by the compression scheme are initially used to detect outliers. If the outliers are found to be beyond a certain threshold, they are initially classified as sampling errors. The erroneous samples are then sent to the cluster head. The cluster head correlates the erroneous samples of various member nodes. Because the erroneous samples are generated by the independent random process, hence they generally are uncorrelated. Consequently, the erroneous samples that are shown to be correlated across multiple cluster members indicate an event instead of erroneous samples, as it is highly unlikely that multiple nodes will generate erroneous samples with similar amplitudes at the same time. Hence, the correlated samples are classified as events and are sent as outliers to the sink. Whereas, uncorrelated erroneous samples are finally classified as erroneous samples and are discarded and replaced with the model approximations. The data filtered and enhanced by the fault-tolerance scheme is then used by the compression scheme to construct the models. The resulting models enhance the approximation accuracy of the environmental attributes.

1.4.5 (C5) – Event Prediction in WSN

The proposed data compression scheme enables us to collect a large bulk of data and over a long period of time. We can use the collected data to forecast the state of the network for a given attribute. We propose a scheme based on the idea of maps to detect the events in the forecasted states of the network, essentially predicted the events that may take place, e.g., partitioning of the network.

A virtual map is constructed out of the collected data from the network. Successive rounds of data collection generate a time varying view of the network. The maps data over time is used to forecast the future state of the network, i.e., the future maps of the network. The forecasted maps are then used to detect the events. Forecasting of various events allows to proactively take certain actions if certain events are undesirable. For example, we can forecast if a certain part of the network will be partitioned and proactively take some precautionary measure, e.g., deploy more nodes or divert the traffic, to avoid the situation before it actually takes place.

1.5 PUBLICATIONS RESULTING FROM THE THESIS

The research work presented in this thesis is validated by several publications in international conference proceedings and journals:

- Azad Ali, Abdelmajid Khalil and Neeraj Suri, "Fault-Tolerance for Transient and Permanent Data Errors in Wireless Sensor Networks", Proceedings of Symposium on Reliable Distributed Systems (SRDS), pp. 140-145, 2015.
- Azad Ali, Abdelmajid Khelil, Mohammadreza Mahmudimanesh, Neeraj Suri, "Adaptive Hybrid Compression Scheme for Wireless Sensor Networks", ACM Transactions on Sensor Networks (TOSN), Volume 11 Issue 4, Article No. 53, 2015.
- Azad Ali, Abdelmajid Khelil, Faisal Karim Shaikh and Neeraj Suri, "Efficient Predictive Monitoring of Wireless Sensor Networks", International Journal Autonomous and Adaptive Communications Systems, vol. 5, 3, (IJAACS), 2012.
- Azad Ali, Abdelmajid Khelil, Piotr Szczytowski and Neeraj Suri, "An Adaptive and Composite Spatio-Temporal Data Compression Approach for Wireless Sensor Networks", Proceedings of ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), pp. 67-76, 2011.

- Azad Ali, Abdelmajid Khelil, Faisal Karim Shaikh, and Neeraj Suri, "MPM: Map based Predictive Monitoring for Wireless Sensor Networks", *Proceedings of Autonomics*, pp. 79-95, 2009.

1.6 THESIS STRUCTURE

Next, we give a brief description of the rest of the chapters in the thesis.

Chapter 2: In this chapter we discuss the state of the art regarding the work that we present here in the thesis. Initially, we discuss the existing work for spatial and temporal compression schemes and then explore in detail current work regarding the spatio-temporal compression schemes in WSN. Next, we explore the literature regarding fault-tolerance in WSN, how the faults are handled in WSN and discuss the novelty of our proposed scheme. Finally, we discuss the existing schemes regarding the predictive monitoring of WSN.

Chapter 3: This chapter discusses various models that we use in the development different scheme in this thesis. Initially, we discuss the system model assumed for the whole thesis. Next, we set forth the design requirements and goals that we are targeting to achieve during the development of various components of the thesis. Afterwards, we discuss in detail various fault class that we target to handle for the development of the fault-tolerance scheme and classify them into subclasses.

Chapter 4: In this chapter we propose our delay-tolerant spatio-temporal compression scheme [Ali et al., 2011a](#) [Ali, Khelil, Suri, and Mahmudimanesh, 2015](#). We initially introduce the proposed scheme and describe in details various stages of the scheme. Next, we perform a comprehensive analysis of the efficiency and compressibility of the proposed scheme. We also discuss in detail the message and computation cost of incurred by the scheme. Finally, we carry out comprehensive simulations and compare the proposed scheme's performance to the existing work.

Chapter 5: Here we propose our fault-tolerance scheme that extends the data compression scheme to make it resilient to the data errors [Ali, Khelil, and Suri, 2015](#). Initially, we discuss the abstraction and modeling of the faults and discuss the vulnerability of the times series to various faults. Next, we propose the fault-tolerance scheme and discuss in detail various types of resulting data errors that we can correct. Finally, we perform comprehensive performance evaluation of the proposed scheme.

Chapter 6: This chapter discusses our proposed long term predictive monitoring scheme for WSN [Ali et al., 2011b](#), [Ali, Khelil, Shaikh, et al., 2009](#). We discuss in detail various phases in the prediction of various events in the WSN. Next, we

discuss a case study of network partitioning and evaluate the proposed scheme with the help of comprehensive simulations.

Chapter 7: In this chapter we summarize our contributions in this thesis and give a brief account of the proposed mechanisms and the achieved results. We conclude the chapter with the future directions in which the work proposed in thesis can be extended further.

2 | STATE OF THE ART AND PRACTICE

In order to provide appropriate context to the work presented in this thesis, we discuss in detail the existing literature. Initially, we present the existing compression schemes, discussing the spatial only and temporal only schemes, and then we discuss in detail the spatio-temporal compression schemes. Next, we discuss existing work discussing the functional faults in WSN and how are they currently handled. Finally, schemes regarding the predictive monitoring of a WSN are presented. We look at the strengths and weaknesses of the existing schemes.

This chapter serves to provide the needed background and context to the research work carried out in the thesis. The chapter concludes with a discussion on the need for fault-tolerant compression scheme that can also exploit delay-tolerance inherent in various WSN application.

2.1 DELAY-TOLERANT SPATIO-TEMPORAL DATA COLLECTION

While discussing the data compression, our focus is on continuous data collection rather than event-based WSN design. Our primary objective is to reduce the data to be transported by spatio-temporally compressing the data during continuous data collection. We also aim to maintaining sensor node level granularity while exploiting delay-tolerance in data collection. There is a wide range of research work in WSN for data compression [Nakamura et al., 2007](#), e.g., simple aggregation [Blaß et al., 2008](#), suppression [Zhou et al., 2008](#), filtering, TinyDB [S. R. Madden et al., 2005](#) and Cougar [Gehrke and Samuel Madden, 2004](#), to name a few. However, there is very limited work in spatio-temporal compression. We summarize some of the important research work regarding spatial and temporal compression and discuss in detail the existing spatio-temporal techniques.

2.1.1 Spatial Compression:

The objective of spatial sampling is to collect interested attribute snapshots. The key idea behind spatial compression is to constrain neighboring sensor nodes with similar sensor readings from transmitting redundant data. [Jindal and Psounis, 2006](#); [Mahmudimanesh et al., 2010](#); [Solis and Obraczka, 2009](#); [Yonggang J. Zhao et al., 2002](#) are a few spatial compression techniques relying on compressive sensing, spatially correlated models, filtering and aggregation. The main weakness of all pure spatial compression approaches is their focus on the spatial redundancy while neglecting the temporal redundancy. Temporal resolution depends on the implemented snapshot update mechanism. Pure spatial compression techniques naively address temporal redundancy by creating a new snapshot or sending an incremental update.

2.1.2 Temporal Compression:

The driver of temporal compression is to exploit temporal correlation present in the attribute values. The key idea in these approaches is to let every sensor node create a prediction model for its sensor readings and send the model to the sink. The sensor node should send an updated model only if the model is not valid due to changes in the signal dynamics. The approaches in [Mini, Val Machado, et al., 2005](#) and [C. Zhang et al., 2006](#) construct models based on Markov-chains and time series, respectively. These temporal compression schemes mainly focus on the temporal redundancy. Some use limited forms of spatial compression (e.g.,

by constructing 1-hop clusters and allowing only cluster heads keep consistent models with the sink).

2.1.3 Spatio-Temporal Compression:

There is very limited research addressing spatio-temporal compression in WSN. Here, we have attempted to give an exhaustive account of the state of the art in this field. In [Vuran et al., 2004](#), the authors developed a theoretical framework to model the spatial and temporal correlations in WSN. This framework enables the development of efficient medium access and reliable event transport protocols, which exploit these advantageous intrinsic features of the WSN paradigm. This work does not focus on continuous data collection contrary to our target applications. [Gedik et al., 2007](#); [H. Jiang et al., 2011](#); [Pham et al., 2010](#) are value-based clustering mechanisms that form large monolithic clusters based on instantaneous values or aggregates of the values. Hence, they need to continuously collect data from the member nodes to check for consistency and maintain these clusters by breaking up and merging them.

We use hierarchical clustering based on models instead of instantaneous values, i.e., (1) 1-hop cluster rarely requiring maintenance, and (2) cliques that are not maintained rather reconstructed at a minimal message cost. [Tulone and Samuel Madden, 2006](#) proposes real-time data collection through repeated model construction and synchronizing them with the sink. [Tulone and Samuel Madden, 2006](#) uses 1-hop clusters, hence compression is limited both spatially and temporal. [C. Wang, Ma, Y. He, and Shuguang Xiong, 2010](#) and [C. Wang, Ma, Y. He, and Shuguang Xiong, 2012](#) relaxes the spatial constraints of [Tulone and Samuel Madden, 2006](#) to form larger than 1-hop clusters. Due to their focus on real-time updates they bounded to approximate only a limited amount of data in time, consequently limiting the temporal compressibility. On the other hand the focus of our proposed scheme are the applications that are delay-tolerant concerning the data collection, hence we can more efficiently compress the data both in space and time due to relaxed time constraints. [Villas et al., 2011](#) proposes values based scheme to exploit temporal and spatial correlation and explicitly focuses on the real-time reporting of the values. We on the other hand propose a model based scheme to exploit the temporal and spatial correlation, extending the compressibility both in space and time.

[Min and Chung, 2010](#) uses Kalman filter for modeling within 1-hop clusters, which incurs heavy computation cost on a sensor node. [Baek et al., 2004](#) proposes to reduce energy consumption through hierarchical aggregation. Both [Min and Chung, 2010](#) and [Baek et al., 2004](#) require the node location information, which could be prohibitively expensive. Our proposed scheme works independent of the location information and uses simple models that are easily computable on a

sensor node. [C. Liu et al., 2007](#) and [Gupta et al., 2008](#) use centralized heuristics for cluster formation and nodes correlation determination. [C. Liu et al., 2007](#) assumes all sensor nodes to be within communication range of the sink, or, the deployment of dedicated nodes, to act as cluster heads, to be able to communicate directly with the sink. Such assumptions limit the applicability of the approach [C. Liu et al., 2007](#) to a generic WSN. Our proposed scheme on the other hand assumes a generic WSN deployment with no additional requirement on communication range or of any dedicated sensor nodes. Moreover, it is completely distributed and does not require any global information other than a few static parameters initially required from the sink to setup the proposed distributed scheme as per the user requirements.

[Yoon and Shahabi, 2007](#) has two modes of operation, i.e., interactive mode is limited to spatial compression only and streaming mode performs both spatial and temporal compression. [L. Wang and Deshpande, 2008](#) and streaming mode of [Yoon and Shahabi, 2007](#) both construct Probability Density Function (PDF) for modeling the attribute. Consequently, these schemes require the system dynamics to be known in advance, and hence need manual setup for appropriate functioning. Additionally, correctness of such schemes cannot be guaranteed. A PDF constructed in a certain time window cannot be guaranteed to be valid after attribute dynamics change limiting the use of such schemes in long term continuous monitoring. They also require expensive long training time (e.g., 15 days for [L. Wang and Deshpande, 2008](#)).

We do not assume the system dynamics to be known in advance and construct the model dynamically based on the changes in the dynamics. Therefore, our technique cannot only track any changing attribute dynamic but can also adjust in case of unexpected changes in the dynamics. In our prior work [Ali et al., 2011a](#), we proposed spatio-temporal compression using models and clustering. In this article, we improve over [Ali et al., 2011a](#) with flexibility to dynamically select a range of models instead of one fixed model, dynamic temporal compression scope and relaxed time synchronization as compared to strict time synchronization.

In summary, the existing hybrid approaches require the signal and its statistical properties (dynamics) to be known, require location information, are partially centralized or use instantaneous values instead of models for clustering. The common factor among all the schemes is that they target real-time/immediate data collection. None of existing works, other than [Ali et al., 2011a](#), exploits the delay-tolerance of many applications, thus, loose efficiency potentials. In contrast, our approach is adaptive, does not require location information, is fully decentralized, uses simple easily computable models and exploits the delay-tolerance to maximize the data collection.

Another important class of compression methods for WSNs is based on *transform compression*. In transform compression, a linear transform is applied on the sensed signal that produces a more compressible version of the data and conse-

quently reduces the amount of data that needs to be transmitted to the sink [Duarte et al., 2012](#). Both transform compression and model-driven classes of data compression methods for WSNs take advantage of the compressibility of the sensed signal to reduce the amount of in-network transmissions, though using different approaches. Here, we study transform compression as an important related method for WSN data compression and compare our work with a representative implementation of transform compression. Distributed Transform Compression (DTC) methods (also known as Distributed Transform Coding techniques) are based on the fact that the raw data recorded from natural phenomena such as the data sensed by a WSN are compressible under certain linear transforms [Duarte et al., 2012](#). After transforming and compressing the raw data, the compressed data are sent to the sink. The sink then applies the inverse transformation to acquire the original data.

2.2 FAULT-TOLERANCE FOR DATA GATHERING SCHEMES

Fault handling in WSN, i.e., detection, prevention, isolation, identification and recovery, has been extensively targeted given that WSNs are built using low-cost failure-prone components. WSN relevant faults ¹ are typically classified into two broad classes: Functional faults and data errors where functional faults eventually lead to data errors. We present a brief synopsis to highlight the needed research gaps and refer the reader to [Cinque et al., 2013](#)[de Souza et al., 2007](#)[M. Yu et al., 2007](#)[H. Liu et al., 2009](#)[Coronato and Testa, 2013](#) for comprehensive details.

2.2.1 Handling Functional Faults in WSN

Functional faults include hardware faults (sensor, memory, unstable voltage, etc.) and software faults [Khan et al., 2014](#) (programming, algorithmic, bit flips, buffer overflows, etc.), which may lead to compromised functionality or node failures (transient or permanent crashes), communication failures, packet loss or link breakage.

Given this diversity, handling functional faults has been conducted on varied levels from hardware to software or physical, link and routing layers, etc. Accordingly, the different functionalities have been made robust to a subset of faults. However, most of approaches either fail to be effective across functionalities or

¹ A fault represents an anomalous condition; error is the observable deviance resulting from the activation of the fault, and failure represents loss of service.

across the varied classes of failures. As we focus on data errors, the following discussion of functional faults is non-exhaustive.

The diagnosis of functional faults [Mahapatro and Khilar, 2013](#) has been proposed at different levels of a WSN. Diagnostic schemes are run either by one node or in groups and address either node, region- or network-wide errors. Diagnosis can be either active or passive depending on whether the scheme generates its own network traffic or just uses the ongoing traffic, e.g., by marking packets [Y. Liu et al., 2010](#). Node self-diagnosis schemes [K. Liu et al., 2011](#) detect and identify node's own faults, e.g., by measuring the current battery voltage, monitoring the communication link quality to neighbors, etc. Cooperative diagnosis schemes involve monitoring of one node conducted by other nodes, e.g., by neighboring nodes. For instance, "consumer nodes" observe the behaviors of the "service provider nodes" (e.g., next-hop relay nodes). Hierarchical failure detection schemes rely on a well-defined fault tree and continuous monitoring of the network to detect failures and root their causes. They usually run on a powerful node (e.g., the sink). Examples are Sympathy [Nithya Ramanathan et al., 2005](#), Momento [Rost and Balakrishnan, 2006b](#), and SNIF [Ringwald et al., 2007](#) frameworks. Some approaches [Ringwald et al., 2007J. Zhao et al., 2002](#) require dedicated supplemental nodes for passive inspection and debugging. Region- and network-level diagnosis aims to detect errors on region level by computing global aggregates [J. Zhao et al., 2003](#) or monitoring global properties such as the residual energy map [Yonggang Jerry Zhao et al., 2002](#).

Common to the techniques addressing functional faults is that they are often driven by the network topology and less by sensed data content and semantics. Accordingly, they focus on topology anomalies such as corrupt packets [Kamal et al., 2013](#), node crashes, faulty links, or network partitioning [Barooah et al., 2012](#); [Dini et al., 2008](#); [Kuei-Ping Shih et al., 2007](#).

2.2.2 Handling Data Errors in WSN

Data errors result from internal (functional faults) and external influences, e.g., environmental interference and noise. There are four primary classes of sensor data errors: Sporadic irregularities/spikes, noisy values, constant/stuck values and constant drifts from the phenomena value. Most of data error detection schemes rely on a basic assumption that the sensor readings from the same region should have similar values [Vuran et al., 2004](#).

FIND [Guo, H. Zhang, et al., 2014Guo, Zhong, et al., 2009](#) ranks event suspecting nodes based on their sensor readings as well as their physical distances from the event. FIND works for systems where the measured signal attenuates with distance. A node is considered faulty if there is a significant mismatch between the sensor data rank and the distance rank. Consensus-based fault-tolerance ap-

proaches [Clouqueur et al., 2004](#) abstract the fault pattern by achieving a data-based agreement among the nodes that have detected a sudden reading change in order to detect an event on consensus basis. Some work propose to solve constant drifts through online calibration using a correction function [Balzano and Nowak, 2007](#)[Bychkovskiy et al., 2003](#)[Feng et al., 2003](#)[Miluzzo et al., 2008](#)[N. Ramanathan et al., 2006](#)[Whitehouse et al., 2002](#). The parameters of calibration function are obtained in different ways, assuming either a certain sensing model [Feng et al., 2003](#)[Whitehouse et al., 2002](#), dense deployment [Bychkovskiy et al., 2003](#)[Feng et al., 2003](#), similarity of readings among neighbors [Bychkovskiy et al., 2003](#)[Miluzzo et al., 2008](#), or availability of reference data (e.g., from highly accurate deployed nodes) [Miluzzo et al., 2008](#)[N. Ramanathan et al., 2006](#)[Hasenfratz, 2012](#).

Outlier detection [Y. Zhang et al., 2010](#)[Branch et al., 2013](#) is an approach related to data error handling. Outlier detection is used either to suppress or amplify selected outliers. Suppressing outliers (also known as data cleansing) improves the robustness of data analysis. Amplifying outliers helps to find rare patterns in domains such as event and intrusion detection. Outlier detection usually addresses non-faulty but extremal data.

As the key functionality of WSN is delivering accurate data to the user, we focus on the detection of data errors that can be rooted at arbitrary functional defects (sensor defect, calibration errors). Accordingly, our data focused approach is content/semantics driven, where the root cause is implicitly considered.

Unlike many existing approaches, which are based on instantaneous sensor data values, our approach is based on data models. The main limitation of contemporary instantaneous fault monitoring is that the detection scheme often encounters fluctuations and the overhead is high as the detection scheme needs to be executed for every sample on resource-constrained nodes. In our work, a model constructed from a set of sensed data, is used to detect individual data errors. This substantially improves our ability to detect corrupted data with high accuracy and reduces the vulnerability to data fluctuations.

Most data error handling approaches addressing generic data corruption fault are tailored to event-based WSN design. Our approach considers both event-based and continuous data collection designs to cover a wide spectrum of WSN applications, modeling the WSN data as a generalized time-series.

2.3 PREDICTIVE MONITORING OF WSN

In predictive monitoring our focus is on predicting the future states of the network and then attempting to detect events that might happen in future. In WSN literature a variety of work addresses event detection [[Yick et al., 2008](#)]. The most relevant work to our event detection strategy is [[Xue et al., 2006](#)], where the au-

thors investigate map based event detection. The approach requires the user to (a) specify the distribution of an attribute over space and (b) the variation of distribution over time incurred by the event. Three common types of events are defined namely pyramid, fault and island. In contrast, our detection technique is independent of event shape thanks to our generic regioning algorithm. Furthermore, we apply the detection technique on predicted profiles allowing to predict events rather than just detecting them. [Banerjee et al., 2008], presents a technique to detect multiple events simultaneously. They employ a polynomial based scheme to detect event regions with boundaries and propose a data aggregation scheme to perform function approximation of events using multivariate polynomial regression. Our work in addition to the capability of detecting multiple events, can predict events beforehand. Various other works exist that address specific event scenarios such as partition detection [Shrivastava et al., 2005], and fire detection [L. Yu et al., 2005]. These specific solutions do not feature portability to adapt to different application scenarios.

There is a variety of work for monitoring WSN's and prediction of a certain attribute. [Landsiedel et al., 2005] predicts the power consumption in WSN. [Mini, Nath, et al., 2002] proposes a network state model to predict the energy consumption rate and constructs energy map accordingly. In [X. Wang et al., 2007], authors focus on predicting multimedia networks energy efficiency. These works concentrate specifically on energy, also they do not provide any extension to predict other attributes. Authors in [Mamei and Nagpal, 2007] propose inference mechanism using Bayesian network to detect anomalies. We provide a generic framework to predict variety of events that might happen in future.

As we present a case study for network partition prediction, we discuss the related work in this respect. In [Shrivastava et al., 2005], partition detection has been addressed for a sub-class of linearly separable partitions, i.e., cuts. Memento [Rost and Balakrishnan, 2006a] continuously collects connectivity information at the sink to be able to detect network partitioning. The partition avoidance lazy movement protocol for mobile sensor networks [K.-P. Shih et al., 2007] is a decentralized approach, where a node periodically collects the position of all its neighbors and checks if at least one neighbor is located in a small angle towards the sink. If no neighbor is located in this "promising zone", the node suspects network partitioning and moves to avoid it. Based on our event prediction framework as an example we propose a solution that is generalized and not dependent on the shape, size or location of the partition. Moreover, our framework provides for prediction of network partitioning rather than just the detection.

2.4 CHAPTER SUMMARY

Extensive investigation of the existing literature in WSN shows that current literature does not exploit the potential of delay-tolerance for optimal compression. Considering delay-tolerance allows the proposed compression scheme to exploit the data redundancy not only in space but also in time. This design feature allows the data compression to be performed over an extended length of time while using very simple, inexpensive models. This enables the possibility to compress the data even further resulting in considerable message cost reductions.

We have also identified that compression schemes do not consider the WSN functional faults and the resulting data errors. Hence, various compression schemes inherently suffer performance issues due to corrupted data samples. Similarly, various fault-tolerance schemes are developed independent of any specific application (here specifically data compression), hence these schemes are not optimized for the specific application and cannot take advantage of the infrastructure already in place implemented for the compression scheme.

In this thesis we investigate how we can exploit delay-tolerance for optimal spatio-temporal data compression and couple it with our fault-tolerance scheme further increasing the approximation accuracy, while reducing the message cost. To the best of our knowledge, this is the first work that exploits delay-tolerance for data compression and couples it to the fault-tolerance scheme to further increase its effectiveness. Moreover, various components have been integrated into a framework that can be adapted to a variety of application scenarios.

3

SYSTEM, DATA AND FAULT MODELS

In this chapter we present various models that make the basis for the fault-tolerant spatio-temporal data compression scheme developed in this thesis. These models mainly consist of system model, data model and fault model. The system model describes various components of the considered sensor network. The data model refers to the mathematical models that we use to approximate the sampled data. The fault model is the abstraction of different data errors that result from various underlying faults. The system model is considered throughout the thesis in the development of various schemes. Whereas, the data and the fault model are applicable to the corresponding data compression and fault-tolerance components in the proposed framework.

While designing various components of the framework, we set forth certain design requirements that must be met. These requirements allow the developed framework to be adaptive and tunable by the end user by adjusting various design parameters. It also allows the developed framework to be applicable to large variety of application scenarios.

3.1 DESIGN REQUIREMENTS

For the temporal compression component and spatial compression component of the framework, our objective is to maximize the spatio-temporal data compression with accuracy guarantees for continuous delay-tolerant data collection (i.e., monitoring of phenomena or environmental attributes). In order to meet these objectives, we require the developed schemes to meet the following criteria

1. The proposed scheme must facilitate the reconstruction of the signal (sample values) of each sensor node from the collected models on the sink.
2. The reproduced data on the sink should be within the application-driven error bound.
3. The developed scheme must incur minimal bandwidth and energy overhead.
4. The proposed scheme should adapt to evolving attribute dynamics.

5. The developed scheme should be agnostic to the network properties such as node distribution, topology and routing protocols.
6. The data compression models should be efficiently computable on resource-limited sensor nodes.

The requirement to be able to reproduce the data of every single sensor node in the network allows us to cover the worst case situation. However, it is trivial to adapt the scheme to gather data from fewer sensor nodes by reducing the number of sensor nodes from which the data is gathered. Even more advanced schemes, like duty cycles, can be adapted to cycle between different nodes to sample the data. However, such schemes are beyond the scope of current work.

We extend our spatio-temporal compression scheme in Chapter 5 to make the scheme fault-tolerant. Accordingly, we extend the design requirements such that we require the developed scheme to be able to

1. detect data errors while being fault agnostic.
2. correct data corruption, hence, maintain data accuracy even in noisy conditions whenever applicable.
3. identify and exclude permanently faulty nodes.
4. adapt to various compression schemes.
5. have low overhead and be fully decentralized.

For the predictive monitoring scheme, developed in Chapter 6, the following design requirements must additionally be fulfilled

1. It should be lightweight, i.e., its creation, management and usage requires minimal resources with respect to energy.
2. We desire the scheme to long-term predict attribute profiles, hence the events. Depending on the context of the problem, long-term may mean hours, days or even months that should be enough to activate a self* mechanism to support autonomic actions.
3. We desire the scheme to be generic to adapt to prediction of varied event types.

3.2 SYSTEM MODEL

We consider a conventional WSN system model composed of N static sensor nodes $\{S_1, S_2, \dots, S_N\}$ and a static sink. We assume that each sensor node is

composed of a processor, sensor board, radio board, batteries, and software, all of which may behave erroneously either transiently or permanently. We assume that the clocks of nodes are synchronized, e.g., using [Faizulkhakov, 2007](#). Sensor nodes sample the environment attribute simultaneously and periodically every τ time units. This synchronized sampling allows nodes to run duty cycling for maximized energy efficiency, which is out of scope of this article. The sink is powerful enough to store large amount of data. Sensor nodes are battery powered and possess limited storage and processing capabilities. The WSN deployment follows an arbitrary node distribution with varying spatial node densities as per connectivity, coverage, fault-tolerance and sensing requirements. We assume the availability of a reliable end-to-end data transport service, such as [Shaikh, Khelil, Ayari, et al., 2010](#), to transport messages from sensor nodes to the sink, and acknowledgement schemes to ensure message delivery between neighboring nodes. Initially, all nodes including the sink are considered to be non-faulty.

In addition to these generic system model to be used trough out the thesis we make some further assumption for predictive monitoring scheme, presented in Chapter 6.

3.3 DATA MODELING

Generally, the sensor nodes, sampling the environmental attribute values, send the raw sampled value to the sink. However, sending raw sampled values is very expensive both in terms of energy and band width. Hence, we construct models based on collected raw samples. The constructed models are later sent to the sink to reproduce the data. In addition, they are also used by the fault-tolerance module to detect data errors and spatial compression module to construct cluster hierarchy. However, as given in the requirements Sec. 3.1, we require to use computationally inexpensive models to approximate generic patterns typically observed in sampled attributes. However, to model generic patterns using simple models is challenging to achieve on a sensor node due to (a) the limited nodes resources and (b) the unknown attribute dynamics. Hence, we will be using simple models that are easily computable on a sensor node but also implement additional mechanisms to cover the weaknesses of the models to maintain the user required accuracy.

In order to formalize the data modeling, we represent the sampled data of a sensor node as an infinite uni-variate time series that can be given by the expression

$$\dots, v(t), v(t-1), v(t-2), \dots \quad (1)$$

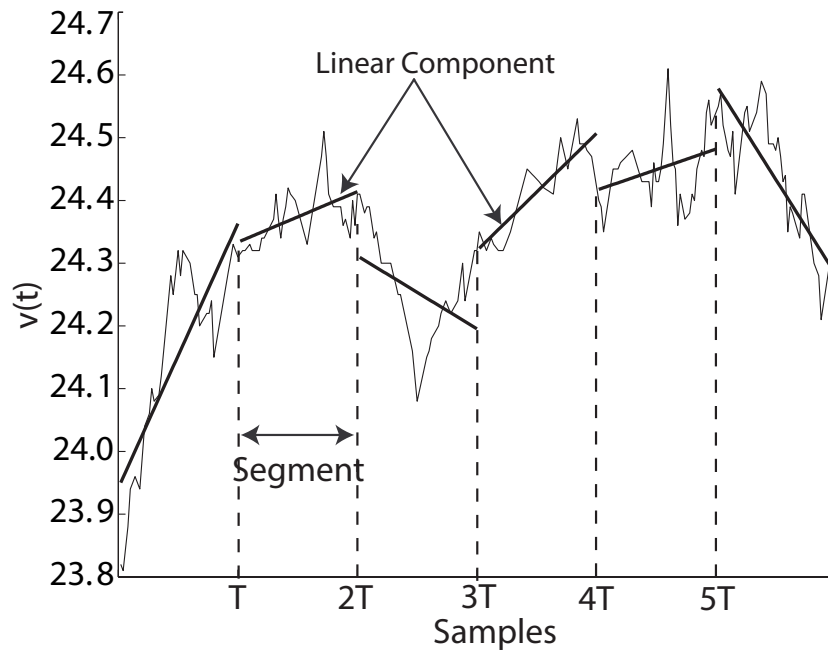


Figure 3.1: Original Signal and Liner Component

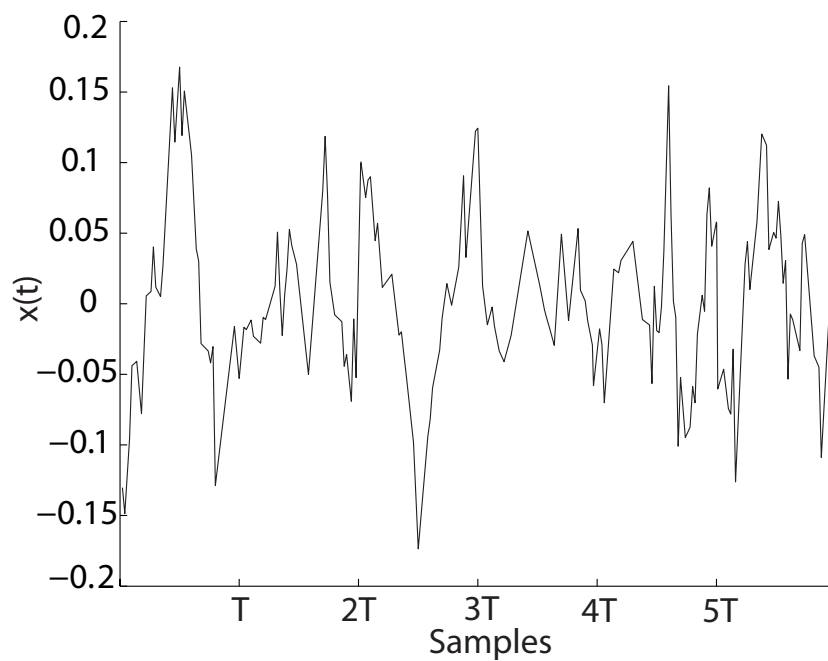


Figure 3.2: Original Signal and Liner Component

where $v(t)$ is data sampled by a sensor node at time t .

We decompose the sampled values (time series) into finite duration segments, so that the resource limited sensor nodes can process the data and compute the models. We term the data segments as training queue, denoted as $V(t)$ that are used to estimate model parameters

$$V(t) = (v(t), v(t-1), v(t-2), \dots, v(t-T+1)) \quad (2)$$

where $v(t)$ is data sampled by a sensor node at time t and T is the length of the training queue (Section 3.3).

Each segment can be modeled as a linear and a random component. Hence, $V(t)$ can be approximated as

$$V(t) = \mu(t) + x(t) \quad (3)$$

where $\mu(t)$ is the linear component and $x(t)$ is the random component as depicted in Fig. 3.1 and 3.2 respectively. The linear component can be estimated as

$$\mu(t) = a + b \times t \quad (4)$$

where a and b are real constants. Modeling of the random component is explained next.

Random Component Estimation

One of the design goals is to support a wide range of applications. The key to increase the scope of the proposed technique is to design it independent of the statistical process of the sampled attribute values. Hence, we do not make any assumptions for the underlying process other than requiring the process to be weakly stationary, which is generally true for physical processes [Ljung, 1998](#). This allows to model the random component as a linear difference equation of Autoregressive Moving Average (ARMA) models as given by Eq. (5):

$$x(t) = \frac{\phi_1 x(t-1) + \dots + \phi_p x(t-p) + \theta_1 z(t-1) + \dots + \theta_q z(t-q)}{\theta_1 z(t-1) + \dots + \theta_q z(t-q)} \quad (5)$$

where ϕ and θ are model coefficients, $z(t)$ is white noise with mean zero and variance σ^2 , denoted as $WN(0, \sigma^2)$ and $p, q \in \mathbb{N}$. We denote the model constructed using Eq. 5 as $\Phi(x)$ (or Φ in short). The Moving Average (MA) part of the ARMA model is relatively expensive to solve, hence to reduce

the computation complexity we put $q = 0$ in Eq. (5) to reduce it to an Autoregressive (AR) model as follows

$$x(t) = \phi_1 x(t-1) + \dots + \phi_p x(t-p) + w(t) = \sum_{i=1}^p \phi_i x(t-i) + w(t) \quad (6)$$

where $w(t)$ is white noise series with mean zero and variance σ_a^2 .

AR model is a linear model hence to cope with the non-linearities we use two mechanism 1. We collect outliers, not approximated by the model, and send them to the sink so that the reproduced data can meet the user required accuracy 2. We construct a new model if the already constructed model cannot predicted the values within the defined error bounds. These mechanisms are described in detail in Chapter 4.

Model Construction

In order to construct the model, a node maintains a training queue $V = \{v_1, v_2, \dots, v_T\}$ of T sampled values. The AR model parameters are computed by (a) estimating the fitting error ($e(\Phi)$), and (b) minimizing the estimation error. The AR model fitting error $e(\Phi)$ is minimized in mean-square error sense as given by Eq. (7)

$$\frac{\partial}{\partial \phi_k} e(\Phi)^2 = \frac{\partial}{\partial \Phi} \left(\sum_{i=1}^T \left(x(i) - \sum_{j=1}^p \phi_j x(i-j) \right)^2 \right) = 0 \quad (7)$$

The estimated values as estimated by the model are given by

$$\hat{v}(t) = \mu(t) + \sum_{i=1}^p \phi_i (v(t-i) - \mu(t)) \quad (8)$$

where $\hat{v}(t)$ is the estimated sample value. We denote $\hat{V}$ as the approximated values set, where $\hat{V} = \{\hat{v}_1, \hat{v}_2, \dots, \hat{v}_{W_a}\}$, W_a is the number of estimated values referred to as approximation window (explained further in detail in section 4.1.3).

Data modeling scheme described so far is the basis for the compression scheme detailed in Chapter 4. However, this simplistic model is not adaptive/tolerant to changing dynamics. Hence, in Chapter 4 we extend the

basic data modeling to make it adaptive so that it can model changing attribute dynamics and extend its temporal compression scope to approximate more data samples.

3.4 FAULT MODELING – FAULT CLASSES AND DATA ERRORS

WSNs, because of their nature, have various faults ranging from hardware faults, such as sensor faults generating wrong readings, bit flips in memory affecting the sensor data, non-recommended deployment conditions resulting in wrong calibrations, to software faults, such as incorrect sampling algorithms wrongly transforming data. Many of these faults cannot be repaired and would generally require replacement of the sensor node. However, we are interested only in the faults that affect/corrupt the sampled data. Moreover, we do not always need to repair the fault itself, rather we can focus on the consequence, i.e., the data corruption, try to detect and correct the data corruption. Accordingly, we can implicitly tolerate the faults.

Data error types in sampled data from wireless sensor networks has been extensively studied in [Ni et al., 2009](#). Please note the term "data errors" in current work and "data faults" in [Ni et al., 2009](#) are equivalent. Authors in [Ni et al., 2009](#) classify the data errors as follows:

3.4.1 NOISE

The data corruption that occurs for a short duration but show high sample values variation are termed as NOISE. The faults causing these data errors are generally temporary and after a while the sampled values will return back to normal and the correct data will be sampled again. NOISE may occur due to low battery, hardware failure [Szewczyk et al., 2004](#), poor wire connections [Sharma et al., 2007](#) or in the case the environmental values lie out of the sensitivity range of transducer.

3.4.2 SHORT

A sharp sudden momentary changes between normal sampled values are termed as SHORT. These data errors can very frequently occur in low qual-

ity sensors. These data errors are temporary but recurring. Various conditions can cause SHORT data samples including low battery, hardware and poor connections or clipping, which refers to the condition when a sensor is maxed out.

3.4.3 CONSTANT

Data errors that occur for long duration or when the data errors are continuously correct and do not heal over time are termed as CONSTANT or "Stuck-at". CONSTANT data errors may also be caused by low battery, hardware and connection issues and clipping.

Depending on the persistence of the error, we classify these errors as follows:

3.4.4 Transient Data Errors

The data errors that occurring for a certain time duration but subside afterwards are termed as transient data errors. NOISE and SHORT are both transient errors. These are generally the most frequent type of data errors encountered during the operation of a WSN. The fault-tolerance scheme developed in this thesis, in Chapter 5, uses the data models described in Sec. 3.3 to identify such error and attempts to correct them.

3.4.5 Permanent Data Errors

Permanent data errors refer to the data errors where the data is continuously corrupted and the fault does not subside over time. CONSTANT are the permanent errors.

3.5 CHAPTER SUMMARY

We presented in this chapter various requirements and constraints that we impose on the schemes to be developed to make them viable for a real WSN. We also presented various models that form the basis for the development of various components of the framework. In the next chapters we will develop our compression and fault-tolerance schemes where we will be using the proposed models.

4

DELAY-TOLERANT
SPATIO-TEMPORAL DATA
COMPRESSION

Wireless Sensor Networks (WSN) are often deployed to sample the desired environmental attributes and deliver the acquired samples to a central station, termed as the sink, for processing as needed by the application. Many applications stipulate high granularity and data accuracy that results in high data volumes. However, sensor nodes are battery powered and sending the requested large amounts of data rapidly depletes their energy. Fortunately, environmental attributes (e.g., temperature, pressure) often exhibit spatial and temporal correlations. Moreover, a large class of applications such as scientific analysis and simulations tolerate high latency for sensor data collection. Hence, we exploit the spatio-temporal correlation of sensor readings while benefiting from possible data delivery latency tolerance to minimize the amount of data to be transported to the sink. Accordingly, we develop a fully distributed adaptive hybrid compression scheme that exploits both spatial and temporal data redundancies and fuses both temporal and spatial compression for maximal data compression with accuracy guarantees.

We present two main contributions: (i) *an adaptive modeling technique* that allows frugal and maximized *temporal compression* on resource-constraint sensor nodes by exploiting the data collection latency, and (ii) *a novel model-based hierarchical clustering technique* that allows for maximized *spatial compression* resulting into a hybrid compression scheme. Compared to the existing spatio-temporal compression schemes, our approach is fully decentralized and the proposed clustering scheme is based on sensor data models rather than instantaneous sensor data values, which allows merging nearby nodes with similar models into large clusters over a longer period of time rather than specific time instances. The analysis for computation and message overheads, the analysis for theoretical compressibility, and simulations using real world data demonstrate that our proposed scheme can provide significant communication/energy savings without sacrificing the accuracy of collected data.

4.1 THE PROPOSED ADAPTIVE SPATIO-TEMPORAL COMPRESSION

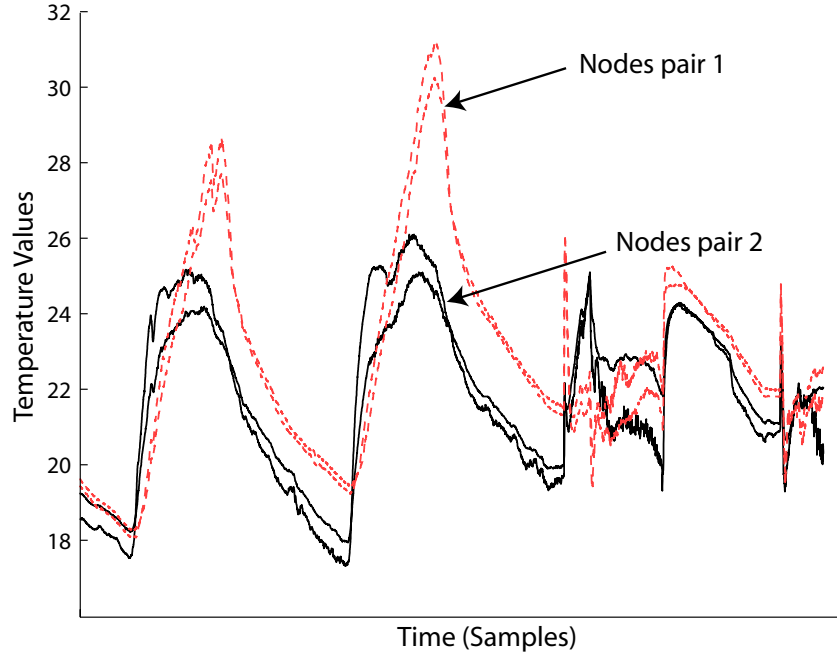


Figure 4.1: Phenomenon Distribution

We propose a decentralized adaptive hybrid compression technique that exploits application delay-tolerance. As observed in Fig. 1.1 (repeated here for convenience), nodes in close proximity of 1-hop distance generally exhibit persistently high correlations in sample values. However, correlation between the nodes farther away from each other, i.e., between 1-hop clusters, are generally non-persistent and dynamic when observed over a long time window. The asymmetry in correlations between the nodes in close proximity and the nodes farther away makes the spatio-temporal compression a challenging problem. The existing literature addresses this asymmetry by limiting the modeling to 1-hop clusters [Tulone and Samuel Madden, 2006](#), assuming the attribute dynamics to be constant [Chu et al., 2006](#), requiring assistance from the sink [Gupta et al., 2008](#); [C. Liu et al., 2007](#) or by clustering based on the instantaneous values rather than models [Gedik et al., 2007](#); [H. Jiang et al., 2011](#); [Pham et al., 2010](#).

We propose to exploit the spatial redundancy in two stages, i.e., (1) Proactive formation of 1-hop clusters that are usually highly correlated, (2) Merge 1-hop clusters to larger regions/clusters. Our spatial compression approach differs from contemporary approaches such as [Gedik et al., 2007](#); [Yoon and Shahabi, 2007](#). These schemes form large monolithic clusters making them vulnerable to frequent reconfiguration that incurs heavy maintenance overhead. We exploit temporal redundancy by constructing simple models on 1-hop clusters.

4.1.1 A Guide through the Proposed Adaptive Hybrid Compression (AHC) Scheme

Given the nature of WSN deployment redundancies and sensed attributes correlations, the proposed scheme performs spatio-temporal compression in three stages:

- Stage 1: 1-hop clusters formation
- Stage 2: Temporal modeling on clusters
- Stage 3: Merging 1-hop clusters

In Stage 1, neighborhoods of sensor nodes with correlated sensor readings form small 1-hop clusters based on a short history of the attribute values to exploit strong local correlation. Depending on the deployed sensor density the 1-hop cluster may consist of up to a dozen nodes.

In Stage 2, we exploit the temporal correlations by constructing the models on a small number of clusters, referred to as *master clusters* (depicted in Fig. 4.2 bearing crown). Each constructed model is initially limited to the respective master cluster and approximates the sampled values of all member sensor nodes of the master cluster.

In Stage 3, we propose mechanisms to utilize the models constructed on the master clusters to also approximate the sampled values of nodes in surrounding 1-hop clusters. The master cluster sends the model to its neighboring clusters. The cluster members fit the received model to their sampled values and accordingly either accept the model or reject it. The clusters accepting the model merge to form a correlated region (a larger cluster) and further propagate the model to the 1-hop clusters on the border of this region.

Following this scheme, only a small set of the models constructed on master clusters can approximate the entire network both in space and time.

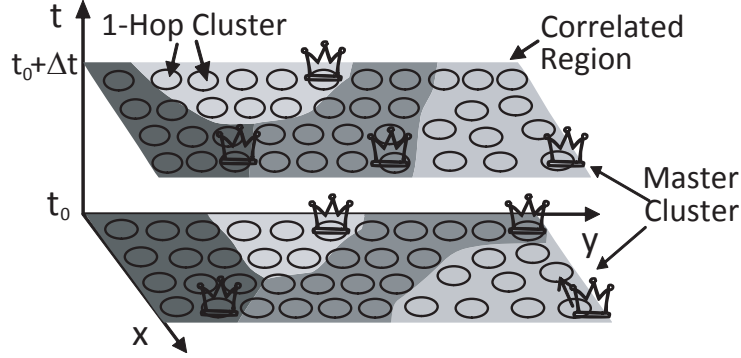


Figure 4.2: Clusters, Master Clusters and Correlated Regions

The resulting spatial compression is a two level hierarchical clustering. The first and second hierarchy levels are formed during Stages 1 and 3. Stage 1 is executed only once, while Stages 2 and 3 are repeated to continuously model the sampled value and adapt to the changing dynamics. In the following, we detail these three stages.

4.1.2 Stage 1: 1-Hop Cluster Formation

Notation	Description
S_i	i^{th} Sensor node
S_{CH}	Cluster head
C_i	i^{th} Cluster
r	Cluster members count
C_N	Neighboring clusters list

Table 1: Cluster Notations

In contrast to existing approaches, we initially form small 1-hop clusters instead of large monolithic clusters to exploit the strong local spatial correlations. The 1-hop clusters are subsequently (Stage 3) merged to form larger clusters to model the dynamic correlations between the 1-hop clusters. Constructing and maintaining large monolithic clusters, in comparison to smaller 1-hop clusters, would incur heavy maintenance costs as we explain further in Section 4.1.4. As cluster formation is a very well-studied topic in WSN [Abbasi and Younis, 2007](#), we only briefly describe the formation of 1-hop clusters.

1-hop clusters are formed based on the similarity of short history of the attribute values that are transmitted by the nodes that candidate to be the cluster head. All the nodes in the network are initially candidates for cluster heads. The sink issues the task of sampling the environment and sending the compressed sampled values back to the sink. Nodes wait for a random time t_{1h} . On the expiry of t_{1h} a node assumes the role of a cluster head and issues a "join request" along with a set of its sampled values to the 1-hop neighbors. Meanwhile, nodes receiving the join request cannot issue the join requests. If more than one node issue the join request within each other's 1-hop neighborhood, the node with lower id withdraws its request. Additionally, nodes may still receive join requests from other nodes as a given node might be 1-hop member of many nodes that are not 1-hop neighbors of each other.

Nodes receiving the join request compare the received values with their own sampled value. Nodes join the cluster for which the difference between their sampled values and received values is within the accuracy requirement set by the application/user. Occasionally, the requests sent by the candidate cluster might not be received by its 1-hop cluster members due to collisions or hidden terminal problem [Tobagi and Kleinrock, 1975](#). Nodes unable to join a cluster, either because they did not receive the request or the error was too high, initiate their own cluster formation requests.

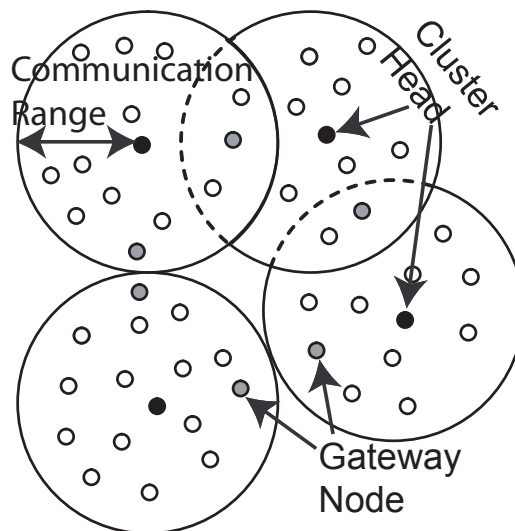


Figure 4.3: 1-Hop Clusters

The later requesting nodes can be claimed by the already existing cluster heads if the cluster head finds the values of the requesting node to be within the given threshold to compensate for the lost request message. Accordingly, we define a 1-hop cluster (C_i) to consist of one cluster head node (S_{CH}) and 'r' member nodes (S_i) such that they are 1-hop away from the cluster head, i.e., $C_i = \{S_{CH}, S_1, S_2, \dots, S_r\}, \wedge \forall S_i \in C_i \sqcup \text{hopdist}(S_{CH}, S_i) = 1$ (where $\text{hopdist}()$ returns the hop distance between two nodes). Members of one cluster might be able to listen to more than one cluster head but exclusively belong to one cluster, i.e., $C_i \cap C_j = \emptyset, i \neq j$.

Each 1-hop cluster head discovers immediate 1-hop neighboring clusters (C_N) around it. In the discovery process, the cluster heads exchange the cluster ids (same as cluster head id) and hop distances of cluster head from the sink. They also identify the nodes that will be used to communicate between the two neighboring cluster heads referred to a gateway nodes as depicted in Fig. 4.3. The 1-hop cluster formation process is performed only once. We do not explicitly refresh 1-hop clusters in order to maintain the correlations between the nodes within these clusters. Instead, we design Stages 2 and 3 in an adaptive manner such that 1-hop cluster rearrangement takes place dynamically in response to the changes in the physical phenomenon, as detailed further in Section 4.1.4.

4.1.3 Stage 2: Temporal Modeling in 1-Hop Clusters

We now elaborate a temporal compression scheme to model the sampled values of a sensor node. We will use the developed temporal compression approach in Section 4.1.3 to model the sampled values of nodes in 1-hop clusters initially and extend the modeling beyond 1-hop clusters in Section 4.1.4. The scheme has been developed such that the model construction is carried out only by master clusters (selection of master clusters is explained in Section 4.1.4), while the rest of the clusters use the models constructed by the master clusters to approximate their sampled values.

Temporal compression scheme is based on the data modeling introduced in Sec. 3.3. The data modeling scheme models the sampled data using autoregressive models based on the raw attribute values gathered by the sensor nodes. We repeat the equation to compute values approximated by the model here again for ease of reference

$$\hat{v}(t) = \mu + \sum_{i=1}^p \phi_i(v(t-i) - \mu) \quad (9)$$

Notation	Description
$v(t)$	Sample value at time t
$\hat{v}(t)$	Approximated values at time t
$V(t)$ or V	Sampled values queue for training models (training queue)
$\hat{V}(t)$ or $\hat{V}$	Estimated values
T	Training queue length
p	Model order of AR models
ϕ	Model parameters
Φ_i	i^{th} Model
W_Φ	Approximation window of a model
Ψ	Model cache (set of models)
W_Ψ	Approximation window of a model cache
$\hat{V}_\Psi$	Set of estimated samples by a model cache
$m_\#$	Number of models in Ψ
ϵ	Error threshold
$O_{\#_{\max}}$	Maximum allowed number of outliers per model per node

Table 2: Modeling Notations

where $\hat{v}(t)$ is the estimated sample value and μ is the series mean. We denote $\hat{V}$ as the approximated values set, where $\hat{V} = \{\hat{v}_1, \hat{v}_2 \dots \hat{v}_{W_\Phi}\}$, W_Φ is the number of estimated values referred to as approximation window.

Piece-Wise Adaptive Modeling (AM) through Outlier Detection

As we assume the statistical process of sampled values to be unknown, the natural precondition is to have adaptability of the model according to the changing statistical dynamics. The adaptability can be achieved by providing feedback to the model to adapt to the new dynamics. This is expensive due to continuous updates transmitted to the sink with new model parameters elevating the message cost. Hence, we use an adaptive update algorithm based on outliers rather than the model parameters.

Outlier Detection: The sample values that cannot be approximated by the model can be tolerated and classified as outliers. If α is the maximum tolerable estimation error then the estimated value must lie between $[v - \alpha, v + \alpha]$. If an estimated value does not lie between the bounds, the node replaces the value with the original sampled value and classifies it as an outlier value. The outlier values should be reported to the sink separately

for accurate signal regeneration. Nodes can determine an estimated value to be an outlier using Eq. (10):

$$\hat{v}(t) = \begin{cases} v(t), & \text{if } |\hat{v}(t) - v(t)| > \alpha; \\ \hat{v}(t), & \text{otherwise.} \end{cases} \quad (10)$$

The 1-hop cluster head gathers and reports the outliers of its cluster members to the sink to maintain accuracy within α . Fig. 4.4 shows the block diagram for adaptive modeling based on Eq. (10) (z^{-1} represents the time delay).

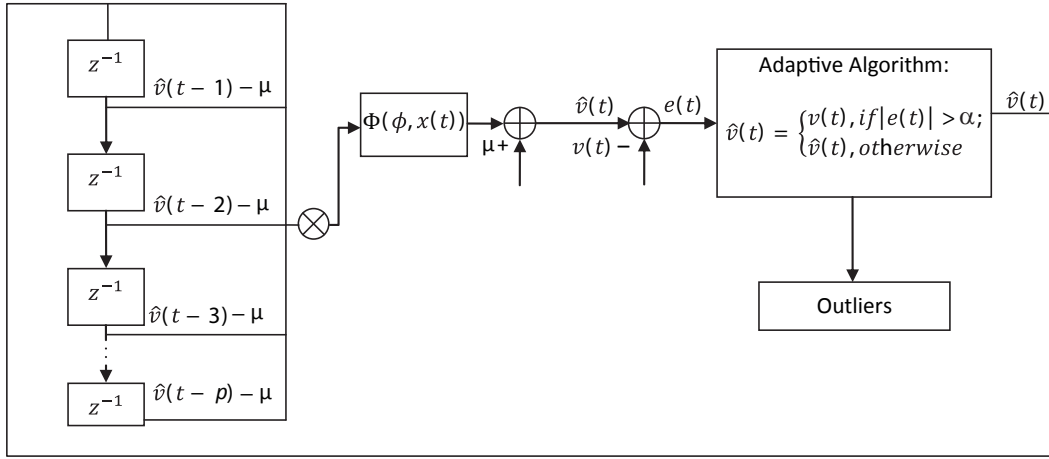


Figure 4.4: Adaptive Modeling

The error upper bound and error probability associated with the estimated values can be given by Lemma 4.1.1 (the detailed proof can be found in [Tulone and Samuel Madden, 2006](#)).

Lemma 4.1.1. *Let $\alpha = \nu\sigma$, where ν is an application specified real constant larger than 1, the actual sampled value $v_i(t)$ is contained in $[\hat{v}(t) - \alpha, \hat{v}(t) + \alpha]$ with error probability of at most $1/\nu^2$.*

Model Invalidation and Outliers Upper Bound: A model becomes invalid if it can no longer approximate the sampled values within the defined accuracy bounds. Dissatisfying the accuracy bounds results into outliers that increase the message cost. We consider a model to be valid until it results into a maximum number of outliers ($O_{\#max}$). The nodes approximate the sampled values based on the constructed model using Eq. (9) and outliers are counted using Eq. (10). If the generated outliers for a given model

are more than $O_{\#max}$ the model becomes invalid and a new model must be constructed. By adapting this scheme, we control the maximum number of the generated outliers, satisfy the accuracy requirement and avoid the unnecessary reconstruction of the model.

We have discussed the standard modeling technique, how to make it adaptive and proposed to use simple AR models to meet the minimal computation and memory requirements. However, we do not know what order of AR model will best approximate the sample values. Hence, in the next section we discuss the selection of the model that approximates the sample values with minimal error, generating the least outliers, while still satisfying the resource constraints.

Dynamic Adaptive Modeling (DAM) through Dynamic Model Order Selection

We have developed an adaptive modeling scheme that the sensor nodes use to dynamically select an appropriate model order that best approximates the sampled values and reduces the message overhead. It also avoids unnecessary higher order computation.

Algorithm 4.1 Dynamic Model Order Selection

```

1: modelOrder  $\leftarrow$  1;
2:  $\Phi_{Current} \leftarrow \text{train}(V, \text{modelOrder})$ ;
3:  $\text{Outliers}_{Current} \leftarrow \text{AM}(\Phi_{Current}, x(t))$ ;
4: for modelOrder  $\leftarrow$  2 to maxOrder do
5:    $\Phi_{New} \leftarrow \text{train}(x(t), \text{modelOrder})$ 
6:    $\text{Outliers}_{New} \leftarrow \text{AM}(\Phi_{Current}, V)$ ;
7:   if  $\text{Outliers}_{New} < \text{Outliers}_{Current}$  then
8:      $\Phi_{Current} \leftarrow \Phi_{New}$ ;
9:      $\text{Outliers}_{Current} \leftarrow \text{Outliers}_{New}$ ;
10:  else
11:    break; ;
12:  end if
13: end for

```

Sensor nodes have limited computational capabilities, and higher order models (e.g., AR(4) and higher) are typically computationally expensive. Hence, we allow the nodes to choose the model from AR(1) to AR(3). A node initially constructs the AR(1) and AR(2) models as described in Section 3.3 and counts outliers for each model (Alg. 4.1, L. 1-6). Both models have low computation cost as AR(1) solves a linear equation and AR(2)

solves two linear equations. The outliers for lower order model (AR(1)) are compared with those for higher order model (AR(2)) (Alg. 4.1, L. 7). If a higher order model does not outperform a lower order model in terms of outliers, i.e., it does not have lower number of outliers, then it is discarded and no further search is carried out (Alg. 4.1, L. 10-11). However, if the higher order model outperforms a lower order model then the higher order model is considered and the lower order model is discarded (Alg. 4.1, L. 8-9). Similarly, the next higher order model is evaluated and compared until no further improvement is observed. Consequently, the nodes dynamically select a model order that optimally matches the approximated sampled data. The resultant scheme can adaptively model the attribute samples and dynamically selects its order, and is termed as Dynamic Adaptive Modeling (DAM). Conceptually, our scheme is neither bound to the maximum AR(3) model nor to the AR model type. If sensor nodes would possess better computation resources, then the choice for the model order can be adaptively increased or other model types can be used exclusively or in addition to AR models.

With our adaptive scheme, the number of samples approximated by a certain model is also dynamic and hence the compression scope of each model may be different at different time in order to adapt to the changing signal dynamics. In the next section, we detail the adaptability process to approximate the varying number of samples.

Dynamic Approximation Window

A model, in general, can approximate only a limited number of sampled values within the accuracy bounds. We refer to the number of samples that a model can approximate as the approximation window and define it as:

Definition 1. *Approximation window (W_Φ) for a model is the number of samples that a model (Φ) can approximate within the required accuracy bounds while allowing $O_{\#max}$ outliers.*

The approximation windows depends on accuracy bounds, allowed outliers, model order and the attribute dynamics. Increasing the tolerated accuracy bounds, maximum allowed outliers and the model order usually increases the approximation window.

Whereas, the increasing non-linearity in the signal generally decreases the approximation window because of the use of linear models, the accuracy bound is fixed based on the user requirements. The maximum

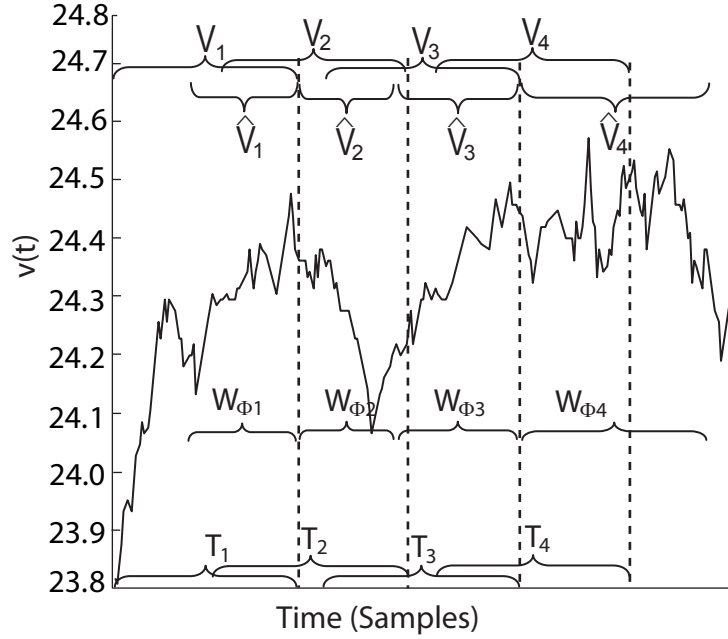


Figure 4.5: Dynamic Approximation Window

allowed number of outliers is fixed by design to limit the message cost. The model order is dynamically selected for optimal compression (Section 4.1.3). Hence, given the accuracy requirements and outliers bounds the attribute dynamics determine the approximation window for a certain model.

The constructed model is used to approximate the values that it was trained with, i.e., the training queue (V). The approximated values set is denoted by $\hat{V}$ ($\hat{V} = \{\hat{v}(t), \hat{v}(t-1) \dots \hat{v}(t-W_\Phi+1)\}$). Due to the dynamic nature, the number of samples for an approximation window may vary. In Fig. 4.5, we illustrate the idea of dynamic approximation window. The tailing sample values are estimated using the model (Φ) constructed from the training sampled values. The number of estimated values (model approximation windows) is denoted as W_Φ . W_Φ can vary in number of estimated values from approximated sample short of the training length to samples beyond it depending on the underlying process dynamics, error threshold and maximum allowed outliers.

Fig. 4.5 depicts various cases, e.g., $\hat{V}_1$ estimates samples to the length of training vector $\hat{V}_1$, $\hat{V}_2$ runs short of V_2 and $\hat{V}_4$ runs pass the training length of V_4 . In the case of $\hat{V} \subsetneq V$, where the estimated values in $\hat{V}$ run short of the training samples length, the values not estimated in a training set are

passed to the next training set. For example, in Fig. 4.5 the samples of V_2 not estimated by $\hat{V}_2$ are passed to V_3 to train the next model. Occasionally, a model may estimate all the sample values in the training set (e.g., $\hat{V}_4$) and still the number of outliers is less than $O_{\#max}$. In this case, the model keeps on approximating the next values (i.e., predicting) until the number of outliers are less than $O_{\#max}$.

We now develop a mechanism to construct a batch of models so that instead of sending individual models, a batch of models may be sent to the sink in just one message.

Maximal Temporal Compression and Model Caching

We assume that data delivery can be delayed as specified by the delay-tolerance for the application. We denote the application delay-tolerance in terms of samples that can be collected and used to construct the model cache and then report to the sink. The delay is represented as the total number of samples approximated by the model cache, given as W_Ψ . The tolerated latency, and hence the length of $\hat{V}_\Psi$, are generally many orders longer than the approximation window of a simple AR model. In general the nodes will require more than one model to approximate the entire length of the sampled data.

The master cluster heads construct a consecutive batch of models referred to as the *model cache*. A sensor node collects training data (V) with T samples that are used to train the model as described in Section. 3.3. The constructed model is used to approximate the samples ($\hat{V}$). Subsequent samples are used to construct a new training queue. The new training queue is again used in the same manner to construct the next model. This process keeps on repeating until W_Ψ samples have been approximated. We define the set of $m_\#$ models, that are used to approximate W_Ψ samples, as a model cache denoted by Ψ and defined as $\Psi = \{\Phi_1, \Phi_2, \dots, \Phi_{m\#}\}$. The construction of a model cache has also been illustrated in Fig. 4.6. Due to delay-tolerance and presence of the model cache, we do not repeatedly need to construct a model, send it to the sink and send a new model once the previous model is invalid. Such a scheme would require repeated transmission of the model parameters to the sink. By constructing the model caches the master clusters refrain from reporting each model and avoid such repeated re-transmissions, decreasing the message cost considerably. Consequently, we increase the temporal compression window, handle non-linearities, decrease message cost, save energy and still use simple computationally inexpensive models. Next, in 4.1.2 we show that

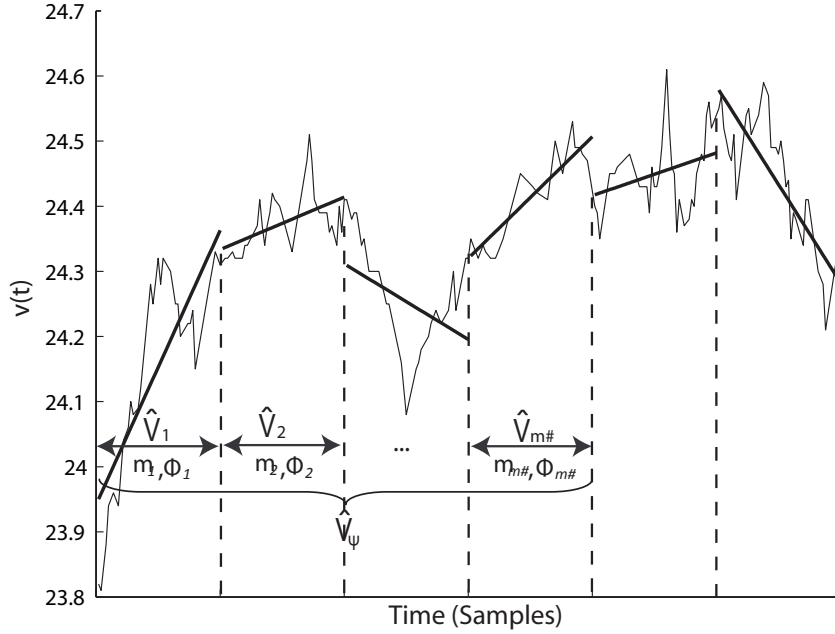


Figure 4.6: Model Cache

the samples estimated by a model cache error lie within user defined error threshold.

Lemma 4.1.2. *For a model cache Ψ , the actual sampled valued $v(t)$ belonging to the approximated model cache value $\hat{v}_\Psi(t)$ ($\wedge \hat{v}_\Psi(t) \in \hat{V}_\Psi(t)$) is contained in $[\hat{v}_\Psi(t) - \alpha, \hat{v}_\Psi(t) + \alpha]$.*

Proof. A model cache is set of $m_\#$ AR models, i.e., $\Psi = \{\Phi_1, \Phi_2, \dots, \Phi_{m\#}\}$. The sampled values for each model $\Phi_i \in \Psi$ are contained in $[\hat{v}_{\Phi_i}(t) - \alpha, \hat{v}_{\Phi_i}(t) + \alpha]$ (where $\hat{v}_{\Phi_i}$ is the approximated sampled value using model Φ_i) due to Lemma 4.1.1. Hence, the sampled values approximated by model cache Ψ_i are contained in $[\hat{v}_\Psi(t) - \alpha, \hat{v}_\Psi(t) + \alpha]$. $\square$

The temporal compression scheme developed so far can be used by any sensor node to dynamically determine a model cache that best approximates its sampled values. However, we have developed our proposed scheme such that only a few nodes in the network belonging to the master cluster actually execute the proposed scheme to construct the models. Next, we describe how a cluster head of a master cluster uses the developed technique to construct a model. The selection of the master clusters and how other clusters can be approximated using the model cache constructed on master cluster are discussed later in Stage 3.

Master Cluster Model Construction

The fundamental basis of our compression scheme is that only one node (or a set of a few nodes) should construct a model that carefully approximates the largest number of nodes in its surrounding clusters, i.e., within the defined accuracy bounds. Accordingly, we construct the models on master clusters that are used to approximate the sampled values of the nodes not only in the master cluster but also in clusters surrounding the master clusters. The selection of master clusters is discussed in Section 4.1.4. Once, the master cluster is selected the model cache is constructed as described earlier.

For a model cache constructed on cluster head to approximate the sampled values of other sensor nodes, we need to define a measure for the similarity of the sampled values between two sensor nodes.

Definition 2. *If the approximated sampled values from a sensor s_i and s_j are given by $\hat{v}_i(t)$ and $\hat{v}_j(t)$ respectively, then approximation loss for approximating sample values of sensor node s_j using model cache (Ψ_i) of sensor node s_i can be given by $L_{ij}(t) = |\hat{v}_i(t) - \hat{v}_j(t)|$.*

In order to meet the user's accuracy requirements, we define two attribute values to be similar only if their approximation loss is bounded by β . Accordingly, if a sensor node s_i and s_j satisfy the approximation loss bound, i.e., $L_{ij}(t) \leq \beta$, then the approximation error between the two nodes can be given by Lemma 4.1.3.

Lemma 4.1.3. *The maximum approximation error for sensor node s_i using model cache Ψ_j from nodes s_j while satisfying the approximation loss bound, is at the most $\alpha + L_{ij} = \alpha + \beta$, with the error probability less than $1/\nu^2$.*

Proof. Suppose sensor nodes s_i constructs a model cache Ψ_i and sensor node s_j uses Ψ_i to estimate its sampled values such that s_i and s_j satisfy the approximation loss bound at any given time t for the length of the model cache. The approximation error between the two sensors can be given by $|v_i(t) - \hat{v}_j(t)| = |v_i(t) - \hat{v}_i(t) + \hat{v}_i(t) - \hat{v}_j(t)| \leq |v_i(t) - \hat{v}_i(t)| + |\hat{v}_i(t) - \hat{v}_j(t)| = |v_i(t) - \hat{v}_i(t)| + L_{ij}(t)$. Using lemma 4.1.1, $|v_i(t) - \hat{v}_i(t)|$ can be at the most α with the probability $1/\nu^2$, additionally $L_{ij}(t)$ has an upper bound of β . Hence, the approximation error can be at the most $\alpha + \beta$ with a probability of $1/\nu^2$. $\square$

In order to simplify the application of the developed scheme, we define a single parameter for error threshold, ϵ , instead of using two separate parameters α and β . Accordingly, we will use ϵ to define the error threshold

for the sensor nodes constructing their own model caches and the sensor nodes that use the model caches from other sensor nodes (e.g., master cluster heads). Where necessary, the original error threshold parameters can be used to maintain the necessary level of granularity.

The model cache constructed on the master cluster is used to approximate a larger number of nodes. Hence, the constructed model cache should produce minimal approximation error. However, the sampled data may naturally contain spurious and noisy values. The sampled values are used to train the models and the noisy values may introduce modeling errors. Hence, the model cache constructed on a particular node, e.g., cluster head, may not necessarily be the model cache with minimal approximation error. In order to build the minimal error model cache, the cluster head along with a few randomly selected member nodes build the model caches and the best model cache amongst them is selected. Accordingly, the master cluster head broadcasts a request to its member nodes to construct the model cache. A few randomly selected cluster members build the model cache using DAM (Section 4.1.3) and send it to the cluster head. The cluster head accumulates all the model caches and broadcasts them to the member nodes. Each member node (including cluster head) fits each model cache to its sampled values using AM scheme (Section 4.1.3) and reports back outliers for each model. The cluster head selects the model cache that generates the least number of total outliers for the cluster member nodes. This criteria allows us to select the model cache that best approximates the sampled values of all sensor nodes in the cluster but also results into minimal message cost as we have to spend fewer messages to send the outliers to the sink.

The search for least error model cache incurs some additional cost in terms of computations and message exchange. However, this single model cache is used to approximate large number of nodes in the surrounding clusters. Hence, this additional cost is compensated with the saving of large number of outliers that would otherwise incur large message penalties.

4.1.4 Stage 3: Merging 1-Hop Clusters Based on Master Cluster Model Cache

In Stage 1, we exploited limited but strong spatial correlations by forming 1-hop clusters. In Stage 2, we constructed the model caches to exploit the temporal correlation on the master clusters, achieving spatio-temporal

compression. Now, we detail our scheme to extend the spatio-temporal compression beyond the master clusters. Various existing techniques use monolithic clustering based on instantaneous values [Abbasi and Younis, 2007](#); [Gedik et al., 2007](#); [Yoon and Shahabi, 2007](#) to exploit the spatial correlation. They generally are very costly as they incur continuous cluster maintenance cost. We use models instead of instantaneous values to form the clusters. Large monolithic clusters based on models would require tracking of all sensor nodes that fit the entire model cache length. Moreover, the attribute dynamics may often change beyond the 1-hop cluster range. Hence, active tracking of the sensor nodes to form and maintain large clusters can require continuous message exchanges resulting in high energy consumption. Therefore, instead of forming large monolithic clusters we use a two level hierarchical clustering. The first level is 1-hop clusters (Stage 1) where (typically) the nodes are highly correlated. The second hierarchical clustering level is constructed by merging the 1-hop clusters that can be approximated by the same model cache to form larger clusters referred to as model cache cliques or simply cliques (Def. 4). The schemes described next (including the schemes in Stage 1 and 2) are executed on sensor nodes without assistance from the sink highlighting the fully decentralized nature of our approach. Before describing clique formation, we discuss how the master clusters are selected and how a model cache constructed on the master cluster can be used to approximate the sampled values of the nodes in surrounding clusters.

Master Cluster Head Selection

A master cluster merges with its surrounding clusters based on its model cache to form cliques or regions. We are interested in determining the smallest possible number of regions in the network or smallest set of model caches that best approximates the largest network parts in order to achieve maximal spatio-temporal compression, i.e., transport maximum data in least number of messages to the sink. We formulate the problem of defining the minimum number of spatio-temporally correlated regions in terms of cliques in a network graph.

Definition 3. *Given a sensor network consisting of a set of sensor nodes S , the topology of the sensor network can be modeled as an undirected graph $G(D, E)$, where D is the set of vertices and E is the set of edges. An edge (S_i, S_j) is included in the edge set E if Nodes S_i and S_j can communicate directly with each other.*

The network sub-graph induced by a subset S_M , of set S , is the sub-graph of G involving only the vertices and nodes in S_M .

Following the formation of 1-hop clusters, we get a virtual network consisting of a set of cluster heads S_{CH} that are modeled as an undirected graph $G_C(D_C, E_C)$, where D_C is the set of vertices and E_C is the set of edges based on cluster heads. (C_i, C_j) is included in the edge set E_C if Cluster Heads C_i and C_j can communicate (through a gateway node $S_G \in D$) with each other.

Definition 4. *We define a clique as a network sub-graph Q_C , a subset of the cluster heads of set D_C , such that a model cache Ψ approximates the samples values within the given error bounds for all the member nodes of each cluster belonging to cluster heads in Q_C .*

Finding the master clusters that can construct the model caches that forms the largest coverage clique (approximates largest number of clusters) is NP complete [Abello et al., 1999](#). Hence, we utilize a heuristic based on the requirement of minimizing the message overhead. As the information flow direction is from the network to the sink, we propose the farthest cluster to be selected as the master cluster to initiate the clique formation and expand the clique in the direction of the sink by sending its model cache to the neighboring clusters. Looking at the alternate possibility if the clusters nearer to the sink initiates the clique formation, the clique might expand in the direction away from the sink. We will have to spend additional messages to transport the accumulated information back to the sink. Hence, the heuristic biased to expand in the direction of the sink generally reduces the message cost to transport the information to the sink. Each cluster head knows its hop distance and the neighboring cluster heads hop distance from the sink as explained in Section 4.1.2. The cluster heads use this information to figure out the farthest cluster and hence the master cluster locally. In case of cluster heads having the same number of hops, the cluster head with higher id has the precedence. Next, we describe how a model cache constructed on master cluster (Section 4.1.3) can be used to approximate the sampled values of the nodes in the surrounding 1-hop clusters

Model Cache Acceptance by 1-Hop Clusters

The master cluster constructs the model cache as described in Section 4.1.3 and sends it to the neighboring clusters. A model cache received by a

neighboring cluster must approximate the values of its member nodes within the defined error bounds (Section 4.1.3). Because of similar attribute distribution and redundant deployment a model constructed in a master cluster can approximate the sensor nodes in the surrounding clusters.

Algorithm 4.2 Model Cache Acceptance

```

1: MSG.type='Ψfit';
2: MSG.Ψ ← Ψ;
3: timer.start();
4: broadcast(MSG)
5: function msgReceive(MSG)
6: if MSG.type=='Ψfit' then
7:   vote ← (solve Eq. 12 for Φ, Φ ∈ Ψ)?reject:accept
8:   if vote == reject then
9:     unicast(vote,outliers[],CH);
10:  else if count(outliers > 0) then
11:    unicast(outliers[],CH);
12:  end if
13: end if
14: add Ψ to Ψ_List;
15: function timer.expired()
16: Ψ.accepted ← (Ψ_List total accepts > %k, k ∈ C)
17: broadcast(Ψ.acceptance);
18: if Ψ.accepted then
19:   remove member that rejected Ψ & add the new requesting member
20: end if

```

To evaluate whether the received model cache approximates the sampled values, each member node approximates its samples values as:

$$\hat{v}(t) = \mu_{local} + \sum_{i=1}^p \phi_i \cdot (v(t-i) - \mu_{local}) \quad (11)$$

where $v(t)$ are the sample values of the node, μ_{local} is the sample mean of the local cluster head and model parameters are as received from the master cluster. Each cluster uses μ_{local} as we observed μ changes very quickly from one cluster to the other, while model parameters remain similar.

The nodes in the surrounding clusters approximate their sampled values using received model parameters as given in Eq. (11), calculate the error $\hat{v}(t) - v(t)$ and count the outliers. As discussed in Section 4.1.3, we allow

a certain number of outliers for a model to be accepted by a sensor node. Total count of the outliers for each model Φ in the model cache Ψ should be less than the maximum number of allowed outliers ($O_{\#max}$) for model acceptance (Section 4.1.3). Hence, for each model in the model cache to be accepted, the node counts the outliers for each model as:

$$\text{count } |v_j(t) - \hat{v}_j(t) > \epsilon| \leq O_{\#max}, \hat{v}_j(t) \in \hat{V}_i, i = 1 \dots m_{\#} \quad (12)$$

If all models in the cache satisfy the criteria in Eq. (12) the model cache is accepted by the node.

In Alg. 4.2, we describe the model cache acceptance by the cluster head and model cache evaluation by the sensor nodes in a neighboring cluster. When a neighboring cluster head receives the model cache from the master cluster, it broadcasts the model cache with local means, calculated from cluster receiving cluster head's sample values, to its cluster members (Alg. 4.2, L. 1-4). The cluster head starts a timer to wait for the responses to be collected (Alg. 4.2, L. 3). Each cluster member prepares a vote, using Eq. (12), by counting outliers to either reject or accept the model cache (Alg. 4.2, L. 7). Each member node must accept all the models in the model cache to accept it for the entire 1-hop cluster. In order to reduce the message cost we use the negative acknowledgement scheme. Accordingly, the cluster head is not notified of the acceptance of the model cache (implicit vote), rather when the timer expires on the cluster head it assumes the model cache to be accepted by its member nodes. The negative votes are, however, explicitly reported along with the outliers to be reported to the sink (Alg. 4.2, L. 9). Withholding report for positive votes saves messages because most of the nodes in the 1-hop cluster usually accept the model cache. The member nodes send outliers to the cluster head instead of sending them to the sink (Alg. 4.2, L. 10). The cluster heads report the outliers efficiently to the sink by concatenating multiple outliers in one message.

The cluster head counts the votes for model cache when the timer expires (Alg. 4.2, L. 15-16). A model cache is accepted as a model cache for the cluster if the cluster head received acceptance from at least $k\%$ member nodes (Alg. 4.2, L. 16).

We discussed the construction of model cache on a master cluster and how a model constructed on a master cluster can be used to approximate the sample values of the nodes in other clusters. However, it is required by the clusters to be synchronized in time so that other clusters can use the model cache of the master cluster to approximate nodes sample values. If

the clusters are not synchronized, the sample values might have been collected at different time instances (or time segments). Consequently, master cluster model cache may not approximate the other cluster sample values as the model construction time segment and the time segment of the samples to be approximated might be entirely different.

Self-adaptive Clique Formation

So far, we have described the formation of the 1-hop clusters, the construction of the model cache on the master clusters and how a model cache constructed on the master cluster can be used to approximate the values of sensor nodes in the surrounding clusters. We now describe the clique formation based on the model caches of the master clusters. For clique formation the master cluster sends its model cache to its immediate neighboring 1-hop clusters. Each neighboring cluster evaluates the model cache for acceptance. If the neighboring cluster head accepts the model cache, it joins the clique and sends the model cache to its neighboring clusters (except the neighbor that sent the model cache) and so forth. Hence, the clique formation is essentially a controlled flooding of model cache over the cluster heads around the master cluster. The flooding stops once the model cache is not accepted anymore by the surrounding cluster heads or it reaches the boundary of the network. We now detail this scheme.

Alg. 4.3 describes the expansion of the clique and the functionality of the clusters that constitute the clique and send the model caches to their neighboring clusters to further expand the clique. Initially, only master cluster constitutes the clique as it initiates the clique. Alg. 4.4 describes the functionality of a neighboring cluster receiving the clique 'join' request to merge into the clique.

Model Cache Dispersion and Clique Expansion: A master cluster initiates the clique formation by constructing the model cache. Fig. 4.7 shows the format of the *JOIN message* payload that is used by the cluster heads to propagate the list of constituting clusters and the model cache. The master cluster head adds its cluster id, the model parameters and the sample mean values for each model in the model cache to the message payload. It sends the JOIN message to all the neighboring clusters (C_N) through the gateway sensor nodes (S_G) (Alg. 4.3, L. 1). Each neighboring cluster head executes the 1-hop cluster model acceptance according to Alg. 4.2 (Alg. 4.4, L. 4). If the model cache is accepted, the cluster joins the clique by appending its cluster id and the sample mean values to the message payload. The cluster joining the clique considers itself on the boundary of

the clique and executes Alg. 4.3 to further propagate the JOIN message to its neighboring clusters, hence expand the clique (Alg. 4.4, L. 8-9). The neighboring clusters receiving JOIN message always notify the requesting cluster whether it is joining the clique (Alg. 4.4, L. 11). Each cluster head maintains a local record of its neighboring cluster with respect to their status regarding the clique. The receiving cluster heads update their local record regarding the neighboring cluster status from the clique JOIN messages during each message exchange (Alg. 4.4, L. 7). Once the responses from all the neighboring cluster C_N are received by the requesting clusters, it uses its local record to check if all C_N around it belong to the same clique. If all the surrounding clusters belong to the same clique, it implies that this cluster is not on the clique boundary. It notifies all cluster heads in C_N that it is not on boundary anymore through a "Not Boundary Cluster" message (NBC) and transfers its local list of clusters in the clique to the neighboring clusters (Alg. 4.3, L. 6-9). Each cluster head knows the status of each neighboring cluster whether it has joined the clique and whether it is on the clique boundary by continuously maintaining the local record and forwarding it to its neighbors.

Algorithm 4.3 Clique Expansion

```

1: 1HSend(JOIN,  $C_i \wedge \forall C_i \in C_N$ )
2: function receive(MSG)
3:   if MSG.type=='RESP' then
4:     clique.add(MSG.accepted, MSG.CHR); R#++;
5:     if R#==CN# then
6:       if  $\forall$  clique.acctance then
7:         NBC.clique  $\leftarrow$  clique;
8:         1HSend(NBC,  $C_i \wedge \forall C_i \in C_N$ )
9:       end if
10:    end if
11:  else if MSG.type=='NBC' then
12:    clique.CHi.boundary  $\leftarrow$  'false'  $\wedge$  CHi == MSG.CH;
13:    clique.remove((NBC.clique));
14:  end if

```

Model Cache Flood Control: The clique expansion continues until the new clusters return positive join responses. A cluster head then knows that the clique expansion has stopped locally if it received the responses from all the neighboring clusters and one or more responses are negative. This cluster now stays on the clique boundary. As the join message is

flooded from the master cluster to the final boundary of the clique, each boundary cluster possesses the partial list of clusters forwarded during clique expansion by the clusters from the clique body. The border is traversed to accumulate the clusters list. The border traversal is initiated by a cluster possessing special token termed as the Boundary Traversal Token (BTT). The master cluster initially assigns itself the BTT. The BTT possessing cluster forwards it to its neighboring cluster with least hops from sink if the BTT possessing cluster is no longer on the boundary. The boundary cluster possessing BTT transfers its partial list of the clique constituting cluster to its neighboring cluster on the boundary. The neighboring cluster merges its partial list with the received list and repeats the process until all the boundary clusters are traversed. The last cluster on the edge of the boundary forwards the aggregated list to the sink.

Algorithm 4.4 Clique Joining by Neighboring Cluster

```

1: function receive(MSG)
2:   if MSG.type  $\leftarrow$  'join' then
3:     RESP.accept  $\leftarrow$  'false';
4:     Call Alg. 4.2 to evaluate MSG. $\Psi$ 
5:     if MSG. $\Psi$ .accepted then
6:       RESP.accept  $\leftarrow$  'true';
7:       clique.add(MSG.CRequestCH);
8:       me.Boundary  $\leftarrow$  true;
9:       Execute Alg. 4.3
10:    end if
11:    1HSend(RESP, CHR)
12:  end if

```

The clusters beyond the clique boundary follow the normal procedure of determining the farthest cluster to initiate and form a new clique. The cluster adjacent to the boundary of the clique exclude their neighbors that are already part of another clique in determining the farthest cluster. Using our proposed scheme the nodes dynamically group to create a region that is spatially and temporally correlated for a given attribute for the modeled time duration. Hence, we have one model cache to be reported to the sink that represents the behavior of the spatio-temporally correlated region. During the clique formation each node in each cluster takes part in model cache acceptance, hence, the data regenerated from the model caches on the sink are accurate to the level of single node. The particular discrepancies are corrected through the outliers sent by the cluster heads

for their respective members. We do not assume any particular distribution of the sensor nodes. Therefore, there can be multiple 1-hop clusters in different parts of the network that may have the maximum number of hops in the local neighborhood. We also set an upper bound on the time ($T_{\max}$) that a cluster head can wait for the larger hop number cluster (clusters farther from sink) to initiate the following round of clique formation. On the expiration of the wait time period the cluster head initiates clique formation. Hence, there can be multiple instances of clique formation executing in parallel. Two or more clique formations execute mutually exclusively, i.e., a growing clique stops at the boundary of the other growing clique. We put this restriction because the two cliques are growing based on two different model caches.

Iterations and Dynamic Adaptability

The second level of cluster hierarchy or the clique is a temporary entity to determine the correlated region based on the model cache and is not strictly a larger cluster in conventional sense. It does not have a cluster head and we do not maintain it. Maintaining such a large cluster body may be very costly, because it is built using the model cache, which requires the clusters to agree for a longer duration of time. Hence, cliques are reconstructed rather than being maintained. We show in Section 4.3.1 that message overhead to construct a clique is very low. The reconstruction of the cliques allows to continuously adapt to the changing dynamics. The 1-hop cluster that were part of one clique may be part of another clique or even make their own clique in the next iteration of clique formation depending on the dynamics of the phenomenon. After reporting the model caches, the cluster heads wait for enough data to be collected to construct the next model cache as explained in Section 4.1.3. The process of the model cache construction and the clique formation is repeated and the sink receives accurate continuous data.

1-Hop Cluster Dynamic Rearrangement

The 1-hop cluster members usually stay correlated. However, due to changes in the physical phenomenon, the correlations even at the 1-hop cluster level may change and the clusters require rearrangement. We do not implement an explicit mechanism to detect such changes as it would require further message overhead. We determine such changes in the node correlations in Section 4.1.4 when $k\%$ of nodes agree with the model cache

and the model cache is accepted by the cluster head due to majority vote. The cluster head evicts the remaining $r - k\%$ nodes that send the negative votes (reject the model cache). Using overlapping nature of the clusters the evicted sensor nodes wait and snoop the model caches sent by the rest of surrounding clusters. An evicted sensor node checks the model cache as if it were part of this cluster and updates the cluster head accordingly. If the model cache is accepted by the cluster head, the evicted sensor node joins the cluster. If the evicted node cannot join any surrounding cluster it forms a new cluster and becomes the cluster head as described in Section 4.1.2. Such rare condition arises due to a new phenomenon developing in local region. The sensor nodes in this region (around the evicted sensor node) will start leaving their current cluster (as the evicted sensor node did) and will eventually form a cluster with the evicted sensor node. Consequently, the sensor nodes rearrange the 1-hop cluster to self-adapt to the environmental changes.

4.2 EFFICIENCY AND COMPRESSIBILITY ANALYSIS

In this section, we carry out the cost analysis of our proposed scheme in terms of number of messages required for compression and transport of the information. We discuss the efficiency of our proposed scheme in terms of theoretical compressibility that we can achieve. Finally, we also discuss the computational overhead incurred by the proposed scheme.

4.2.1 Message Payload

The analysis and structure of the message payload that carries the compressed data is very important, because it directly impacts the message cost. The general structure of the model cache message (Msg_{Ψ}) payload is as depicted in Fig. 4.7. The complete information about the model cache and the clusters taking part in the formation of the clique are contained in the message payload. The payload consists of three parts: 1) The model type of each model in the model cache, 2) all the model parameters of each model in the model cache, and 3) the cluster ids (C_{id}) and the local means of each cluster.

We denote the bytes required to denote a certain parameter by a "B" in subscript, e.g., model type_B denotes bytes required for model type, or ϕ_B denotes bytes required by a model parameter. If there are $m_\#$ number of models in the model cache and p denotes the model order then the total size of the message payload can be given as:

$$\text{payload}_B = \text{model type}_B + \sum_{\Phi \in \Psi} \phi_B \times \Phi_p + \sum_{C \in Q} (C_{\text{id}_B} + \mu_{C_B} \times m_\#)$$

The three parts of the equation correspond to the three parts of the message payload.

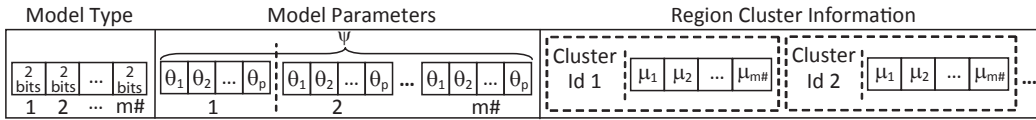


Figure 4.7: Message Payload Format

Notation	Description
r	Cluster members count
μ_C	Mean value for Cluster C
Q	Clique
v_B	Bytes per sample value
$S_{\#C_i}$	Sensors in cluster C_i
Msg_{Req}	Model cache request
Msg_Ψ	Model cache message
Msg_{Out}	Outliers message
$S_{\#rand}$	Random number of nodes
$S_{\#G}$	Number of gateway nodes

Table 3: Efficiency and Compressibility Analysis Notations

The size of payload in comparison to the amount of data that is compressed using our proposed scheme is fairly small. For typical values of 3 models per model cache and 2 bytes to store each model parameter, the bytes required for modeling will be $2\text{bytes} + 2\text{bytes} \times 3(\text{parameters}) \times 3(\text{models}) = 20\text{ bytes}$. If we reserve 2 byte for cluster id and 2 bytes for mean value, then we require 8 bytes (2 bytes for cluster id + 3 models $\times$ 2 bytes for mean per model) for each new participating cluster. Hence, for the given case, we typically require 28 bytes to store the complete in-

formation for one cluster and 8 bytes for each additional cluster in the clique.

TinyOS [Levis et al., 2004](#) (an established OS for WSN) has a default payload size of 28 bytes. However, radios on various popular platforms, such as telos [Polastre et al., 2005](#), support message lengths of up to 128 bytes. Hence, we can easily transport the model cache with multiple cluster information in a single message by changing the default payload size. In contrast to TinyOS, Contiki [Dunkels et al., 2004](#) (another popular platform) does not have such limitations.

4.2.2 Maximum Theoretical Compressibility and Efficiency

In order to keep the nodes synchronized, we limit the number of samples that a node can compress using a single model cache to $\hat{V}_\psi$. Hence, each node in each cluster in the clique compresses $\hat{V}_\psi$ samples using the model cache. If $S_{\#C_i}$ denotes the number of nodes in cluster C_i accepting the model cache (minimum is $k\%$, otherwise model cache is rejected), then the maximum theoretical compression that our proposed scheme can achieve is given by Eq. (13):

$$\text{Total}_{\text{bytes}} = \sum_{C_i \in Q} \hat{V}_{\psi_B} \times (S_{\#C_i}) \quad (13)$$

Eq. (13) shows that the maximum theoretical compression in a given cluster is equal to the number of bytes representing the sample values (max: $\hat{V}_\psi$) of nodes accepting the model cache. To determine the total achieved compression for the clique, this factor should be summed for all the clique member clusters. Assuming a conservative value for $\hat{V}_\psi = 75$, average cluster size of 7 members and 2 bytes per sample, the total data that we compress is $2 \times 75 \times 7 = 1050$ bytes for one cluster. Interestingly, we require only 28 bytes to represent the complete data in the cluster as shown in Section 4.2.1. Consequently, we would achieve a compression ratio of more than 37 times for the assumed parameter values. For each next cluster, we require only 8 bytes in message payload to compress an additional 1050 bytes of data. In terms of message we require only one message. We present the analytic details of message costs and elaborate it further in Section 4.3.1.

The actually achieved compression, however, falls slightly short of the theoretical maximum compressibility. The reduction in compressibility is

due to the outlier values that cannot be approximated by the models to satisfy the accuracy requirements imposed by the user. However, the decrement is limited. The worst case compression that we can really achieve can be given by the following equation

$$\text{Total}_{\text{bytes}} = \sum_{C_i \in Q} (V_{\Psi_B} \times (S_{\#C_i}) - v_B \times O_{\#max} \times m_{\#}) \quad (14)$$

Eq. (14) gives the compression when we assume each node in the cluster has outliers equal to the maximum number of outliers allowed. If we assume maximum allowed outliers value to be 5, the achieved compression decrements by $2 \text{ bytes} \times 5 \times 3 \times 7 = 210$ bytes. Still we are able to compress 840 bytes after fulfilling the accuracy requirements for the given parameters.

4.3 MESSAGE AND COMPUTATION COST FOR COMPRESSION

We now detail the cost of message exchange to achieve the data compression.

4.3.1 Message Overhead

In Section 4.2.2, we discussed the maximum theoretical and practically achieved compression in terms of the number of bytes reduced using our scheme. However, for WSN the notion of number of messages is the true measure of compression to be achieved. Hence, in WSN reduction in number of bytes is only an indirect measure and may be eclipsed or become completely irrelevant if the compression is achieved at the cost of high number of messages exchanged. Hence, we have systematically broken down each stage of the proposed scheme to account for the message cost incurred to achieve the spatio-temporally compression and transportation of information to the sink.

Master Cluster Model Cache Construction

Model cache Ψ is constructed on a few master clusters. The cluster head broadcasts the request (Msg_{REQ}) to the cluster members to send their

model caches (Msg_{Ψ}). A few random members ($S_{\#rand}$) send their model caches. The cluster head broadcasts back all the collected model caches to the members. Finally the cluster members respond back with the number of outliers for each model cache (Msg_{OUT}). If the master cluster has r members then the total number of messages exchanged in a master cluster to construct a model cache can be estimated by the following equations:

$$Msgs = Msg_{REQ} + Msg_{\Psi} \times S_{\#rand} + Msg_{\Psi} + Msg_{OUT} \times r \quad (15)$$

The cluster head and the nodes constructing the model cache in the master cluster transmit two messages and the rest of the nodes in the cluster transmit just one message to construct a model cache. Hence, in worst case a node in the master cluster needs to transmit two messages to compress the data and construct the model cache.

Intra-Cluster Agreement

The clusters other than the master cluster use the model cache constructed by the master cluster to estimate the sample values of the nodes in the cluster. To evaluate whether the model cache estimates the sample values of the sensor nodes, the cluster head broadcasts the model cache. The sensor nodes evaluate the models and respond with the outlier values to the cluster head. The count of messages for this stage can be given by the following equation:

$$Msgs = Msg_{\Psi} + Msg_{OUT}$$

Accordingly, in the clusters other than master cluster the worst case count of messages to decide a model cache is one for any node in the cluster.

Model Cache Dispersion

Once the master cluster has constructed a model cache or a normal cluster has accepted a model cache, it disperses the model cache to its neighboring cluster through the gateway nodes. If S_G denotes a gateway node and $S_{\#G}$ is the count of gateway nodes then the total number of messages required by each cluster to disperse the model cache is given by the following equation:

$$Msgs = Msg_{\Psi} + Msg_{\Psi} \times S_{\#G}$$

This equation represents the worst case situation when the cluster are non-overlapping (which is not typically the case as in Fig. 4.3) and all gateway nodes transmit without optimization, like avoid transmitting to the neighboring clusters that already are part of clique.

Joining Clique

If the model cache is accepted by the cluster head, it joins the clique. The joining cluster head sends a message to the requesting cluster head to report for joining the clique through the gateway node, accounting to one message for cluster and the corresponding gateway node.

Clique Border Expansion

During the model cache dispersion the clique expands and the border of the clique extends further. If a cluster head finds out, after receiving the responses from the neighboring cluster, that it no longer is on the border of the clique it sends this confirmation to the neighboring clusters so that it may no longer be considered on the border of the clique.

$$\text{Msgs} = \text{Msg}_\psi + \text{Msg}_\psi \times S_{\#G}$$

The clique border expansion accounts for one message broadcast from the cluster head and further transmission to neighboring cluster by the gateway node.

In summary the number of transmissions range from one to four messages depending on their role and whether they are in the master cluster or the non-master cluster. In the worst case and with very large clique formation, more than one packets may be required to contain the complete clique information. The above calculations assume the roles of the nodes, from functional point of view, to be mutually exclusive. However, in reality they may not necessarily be exclusive, e.g., a gateway node may also be the same node as the node participating in model cache construction.

4.3.2 Computation Overhead

The computation cost arises mainly during the model learning phase where the model parameters are evaluated (Section 3.3). During the training phase the intent is to avoid unnecessary model construction as proposed in Section 4.1.3. Initially only the first and second order models are constructed. The third order model (or any higher order model in general) is

estimated, and used only if an improvement is observed with increasing order. The parameter estimation consists of solving $AX = B$ in p unknowns and accordingly computing the matrices A and B . For a third order model, which represents the worst case situation, we need to carry out $12(T - 3)$ sum and production operations and additionally the cost to solve the linear system of equations. As we compute a higher order model only if it is needed, hence generally we need to carry out even fewer computations. Additionally, the linear set of equations resulting due to higher order contains terms already computing for the lower order system. Hence, computing a higher order model is not same as the computation of each term in the linear system of equation rather only a few new terms in the higher order system. In addition to the optimization mechanisms that we have in place to reduce the computations as much as possible, we also limit the model construction only to a small sub-set of the clusters, i.e., the master clusters. Other clusters only use the estimated model parameters to determine the model's estimation accuracy, which is a fairly inexpensive operation. The nodes use Eq. (11) for this purpose, which comprises of 3 multiplication and 4 addition operations to estimate a sample value for a third order model.

4.4 EXPERIMENTS AND DISCUSSION

In this section, we evaluate the proposed spatio-temporal compression technique.

4.4.1 Simulation Settings

In order to carry out comprehensive simulations for the evaluation, we used publicly available real-world data set [S. Madden, 2003](#), containing traces for temperature, humidity, light and voltage. The network simulations are performed in TOSSIM. The signal reconstruction at the sink is conducted using MATLAB.

For extensive evaluation of the proposed scheme the simulations were performed on temperature and humidity data as they are not monotonic and continuously change during day and night and hence provides a good opportunity to test the adaptability to the changing dynamics.

The considered network consists of 52 nodes. We selected the AR models which are fairly inexpensive to evaluate. The order of the models is dynamically selected as described in Sec. 4.1.3. The model training length has been fixed to $T = 75$. [Tulone and Samuel Madden, 2006](#) shows that long training windows do not necessarily improve either accuracy or efficiency. We also conducted initial studies to determine optimal value (range) for T and came to the same conclusion as [Tulone and Samuel Madden, 2006](#) that accuracy and efficiency almost stagnates beyond $T = 75$, while the cost to train the model keeps increasing. Hence, in our simulations we deliberately did not vary this parameter and used the specific value of $T=75$.

AHC has been thoroughly evaluated with a wide array of various parameter values. The maximum allowed outliers ($O_{\#max}$) is set to the values of 15, 30 and 45 outliers per model cache per node. Model cache approximation window size W_ψ is assigned values of 60, 75, 90, 105 and 120 samples. The desired error threshold ϵ is simulated for the values of 0.01, 0.05 and 0.1. Comprehensive simulations have been carried out to evaluate and analyze the impact and role of each parameter. As we discussed in Section 4.2.1, we require a larger payload size than the default size of 28 bytes in TinyOS. Hence, we have set the message payload size to 90 bytes. [Haas and Wilke, 2011](#) shows that increasing the payload size to some extent does not increase the energy consumption, which we exploit here.

Comparison with the State of the Art Techniques

In order to put the performance of the proposed scheme in perspective, we compare the performance of AHC to that of our prior delay-tolerant spatio-temporal data collection scheme, i.e., ASTC [Ali et al., 2011a](#), that of the time-series based real-time data spatio-temporal collection scheme PAQ [Tulone and Samuel Madden, 2006](#) and that of another class of in-network compression methods, namely *transform compression* that is based on signal compression. This comparison gives us an even more comprehensive study of the performance of AHC alongside other data compression methods that are based on a different paradigm. The dataset that is being used by our simulation shows high compressibility of the temporal data under Discrete Cosine Transform (DCT) [Ahmed et al., 1974](#). We evaluate the performance of representative Distributed Transform Coding or Distributed Transform Compression (DTC) method that uses DCT as its compressive basis [Duarte et al., 2012](#). The sensor nodes first compute the DCT of the temporal data and select the largest DCT coefficients. The

magnitude and location of the most significant DCT coefficients are then communicated to the sink. The sink reconstructs the original data by applying the inverse DCT transform on the received data while setting the value of non-significant coefficients to zero. We set the accuracy requirements for ASTC, AHC, PAQ and DTC the same and compare the amount of in-network transmissions required by each method to fulfill the accuracy prerequisite. For DTC, we observed that a high enough accuracy is attainable by looking for the most significant DCT coefficients in the first 20 values of the DCT transform. Accordingly, we limit the size of the DCT transform in order to avoid too much memory requirements as the memory of a sensor node is quite limited.

Design Parameters and Trade-offs

Various parameters were introduced while developing AHC. Each parameter influences the performance achieved by AHC. In Section 4.1.3 we have already discussed the effect of certain parameters, such as model order, on the performance of AHC. Accordingly, an automated mechanism was developed to optimally select the best values for these parameters. However, we have to explicitly define other parameters, such as error threshold (ϵ) and approximation window (W_Φ), user accuracy requirements and delay-tolerance level. Hence, before discussing the results, we describe here the implications of various parameters and how they can affect compressibility, efficiency and accuracy.

User defined error threshold (ϵ) defines the maximum error desired in the reproduced data at the sink. It plays a key role in the degree of compression that we can achieve both temporally and spatially. In terms of temporal compressibility, stringent accuracy requirement (lower values of ϵ) can increase the number of resultant outliers. It can decrease the number of sample values that can be approximated by a given model, as a model is valid only for a limited number of outliers ($O_{\#max}$).

In terms of temporal compression, smaller values of $O_{\#max}$ will invalidate a model quickly and we may require more models per model cache to approximate the same number of samples. For spatial compression fewer neighboring clusters may accept a model cache from a master cluster for fewer number of outliers, forming smaller cliques and generating more model caches. Increasing the allowed outliers may increase the temporal compressibility and form larger clique (hence fewer model caches). But larger $O_{\#max}$ also means transporting larger number of raw sample values, which may in turn increase the message cost again.

In terms of temporal compression the larger the value of W_ψ , less number of model caches are required to transport the data. However, in terms of spatial compression, increasing W_ψ may negatively impact the spatial compressibility, because larger W_ψ requires the clusters to agree for a larger number of samples (hence longer duration of time). Hence, larger W_ψ may result in smaller cliques and may consequently require more model caches to be constructed. Increasing W_ψ also means increasing the latency/delay in reporting the samples to the sink. Hence, the upper bound for W_ψ is already defined by the tolerated latency in collecting the sampled values.

The discussion about the parameters and their impact on the degree of compression provides the basic understanding of their behavior while discussing them independently. However, they do influence each other. For example, ϵ influences the resultant number of outliers or $O_{\#max}$ influences the number of a models in a model cache. We will further discuss these parameters and their inter-dependencies in the results section next.

4.4.2 Experimental Performance Evaluation

Our performance evaluation is based on two key metrics: Accuracy of collected data and message efficiency. Accuracy measures how closely the sampled data is approximated after being approximated through AHC and reproduced on the sink. Whereas, message efficiency measures the number of messages required during the whole operation of AHC. Less message efficiency (more number of messages) would imply less energy efficient, as more energy would be consumed with more messages. Similarly, more message efficiency (fewer number of messages) would imply more energy efficient.

Efficiency

We define efficiency in terms of message overhead, i.e., the total message transmissions that account for all messages transmitted during all three stages of AHC. Message overhead comprises of intra-cluster and inter cluster message exchange for model caches construction and verification, and then reporting of the model caches and outliers to the sink.

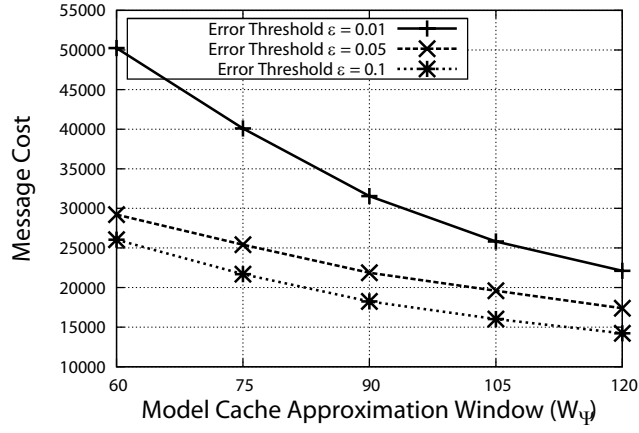
TOTAL MESSAGE OVERHEAD Fig. 4.8 shows the total message cost for intra and inter-cluster communication and model caches and outliers trans-

port to the sink. Fig. 4.8 (a), (b) and (c), show message cost for maximum allowed outliers ($O_{\#max}$) of 5, 10 and 15 per model per node respectively. Each figure depicts the variation in message cost for increasing approximation window (W_ψ) for different error thresholds (ϵ). From Fig 4.8, we make the following notable observations:

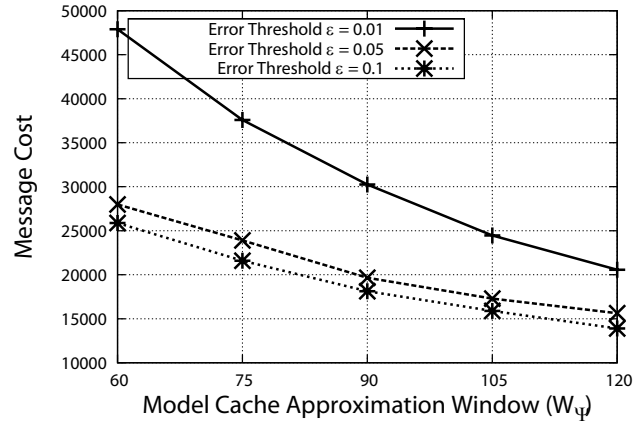
1. For stringent accuracy requirement, i.e., $\epsilon = 0.01$, the message overhead is highest and the message cost decreases rapidly with decreasing ϵ , i.e., to $\epsilon = 0.05$ and $\epsilon = 0.1$, resulting in fewer outliers and more clusters accepting the model caches forming larger cliques resulting in fewer messages to be transported.
2. The message overhead decreases with increasing approximation window because, as explained in Section 5.4.1, the increasing approximation window requires fewer rounds to complete the complete length of data to be compressed and transported. It results in lesser number of model caches to be constructed, saving the intra-inter cluster communication costs and model cache transportation costs.
3. Message overhead drops when $O_{\#max}$ is increased to 10 (Fig. 4.8(b)) but the total message overhead increases when $O_{\#max}$ is increased further to 15 in Fig. 4.8(c). It happens due to drop in inter- and intra-cluster messages and increase in outliers message overhead.

MESSAGE OVERHEAD FOR INTER- AND INTRA-CLUSTER COMMUNICATION :
 We now dig deeper in the total message transportation costs results and explore further why we observe various trends while discussing Fig. 4.9 and 4.10, depicting the total message cost excluding the outliers cost and only outliers cost respectively. In Fig. 4.9 (a), (b) and (c), similar to Fig. 4.8, we depict the inter- and intra-cluster message exchange for various maximum allowed outliers (excluding the outliers cost). The interesting results to be observed here are:

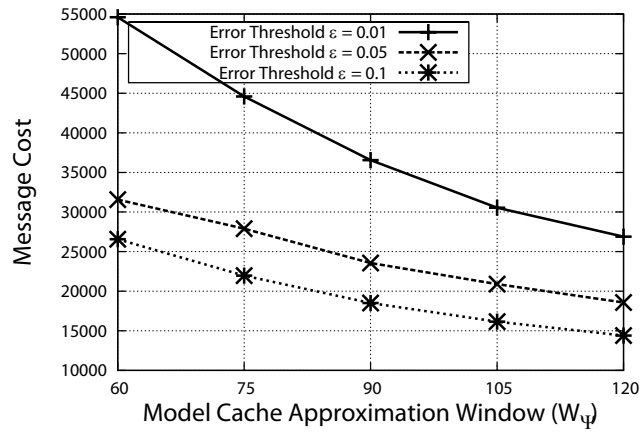
- We observe a considerable drop in the inter- and intra-cluster message overhead when the error threshold is relaxed from $\epsilon = 0.01$ to $\epsilon = 0.05$ in Fig. 4.9 (a). The relaxed error threshold results in larger differences between the approximated value and the sensed values to be tolerated. Hence, we have larger cliques, fewer model cache rejections, fewer new model cache constructions, fewer model caches to be transported to the sink and consequently fewer overall messages to be exchanged.



(a) Message Cost ($O_{\max} = 5$)



(b) Message Cost ($O_{\max} = 10$)



(c) Message Cost ($O_{\max} = 15$)

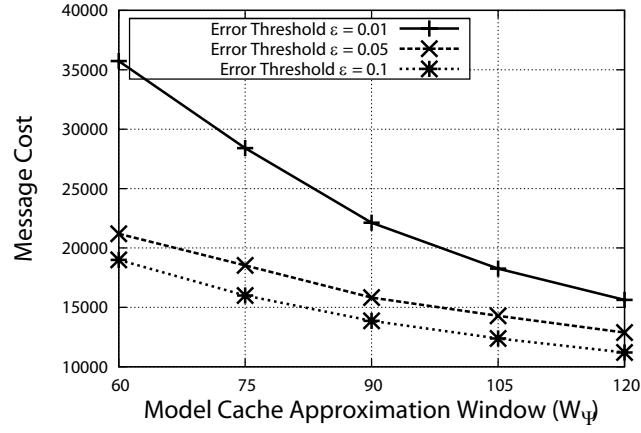
Figure 4.8: Total Message Cost for Inter- and Intra-Cluster Communication, Cache Construction and Caches and Outliers Transportation of Temperature Data to the Sink

- Reduction in message overhead is observed with increasing $O_{\#max}$. More outliers relax the requirements for the acceptance of a model cache by a neighboring cluster and allow it to accept a model cache in spite of more outliers. Hence, we also observe a significant drop in the message overhead for $\epsilon = 0.01$ when $O_{\#max}$ is increased from 15 to 30 as depicted in Fig. 4.9 (a) and (b). However, because of relaxed error threshold that results in wider model cache acceptance by the neighboring clusters in general, we do not observe a similarly significant drop in Fig. 4.9 (c).

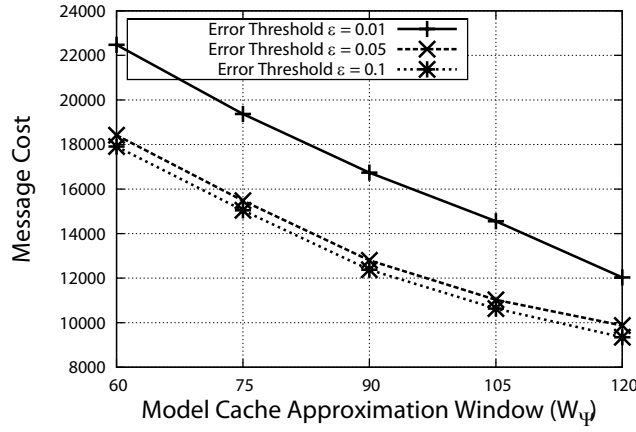
MESSAGE OVERHEAD FOR REPORTING OUTLIERS Fig. 4.10 depicts only the message cost associated with the outliers for various $O_{\#max}$ and ϵ . The notable observations are:

- Similar to the cost of model cache messages, we observe very small changes in message cost for relaxed error threshold when maximum outliers are increased.
- However, contrary to the model cache, we see a significant increase in message overhead as we increase the maximum outliers for $\epsilon = 0.01$. The outliers are expensive to transport not only because they are raw sample values but also because they carry extra overhead of value indices and the value owner id. Whereas, for $\epsilon = 0.05$ and $\epsilon = 0.1$ the message cost remains almost similar because relaxed accuracy requirements do not dictate the data to be more accurate and does not produce any significantly greater number of outlier messages.
- Though we observe a consistent increase in message overhead with increasing $O_{\#max}$, we initially observe a reduction in the total message overhead in Fig. 4.8, because the reduction in the inter- and intra-cluster message cost is greater than the increase in the outliers message overhead. However, when we increase $O_{\#max}$ even further, the increase in the outliers message cost exceeds the reduction in the inter- and intra-cluster message cost and the total message starts to increase again.

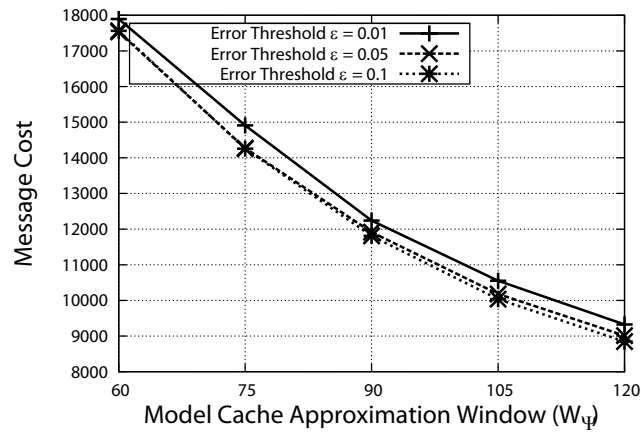
In Fig. 4.11, we compare the message cost of AHC with ASTC with extended approximation window W_Ψ from 35 to 120 samples and error threshold ϵ from 0.01 to 0.2 for AHC. Whereas, ASTC has been simulated at $\epsilon = 0.2$. We can see that AHC consistently performs better than ASTC



(a) Message Cost ($O_{\#max} = 5$)

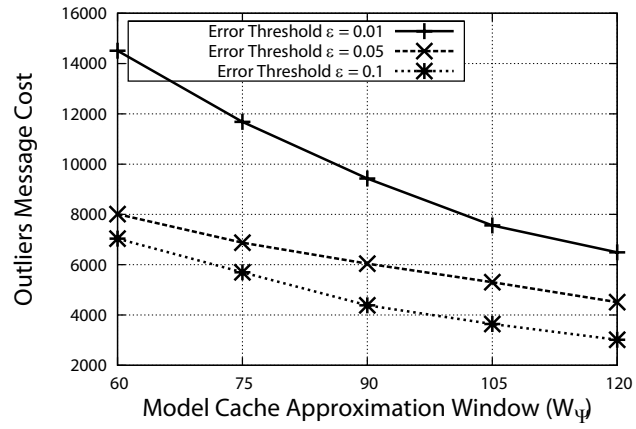


(b) Message Cost ($O_{\#max} = 10$)

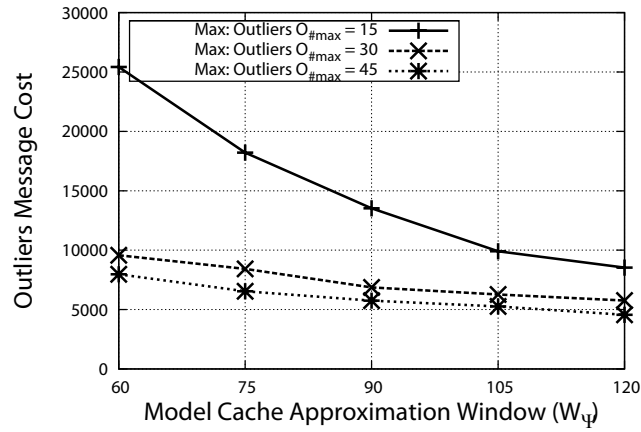


(c) Message Cost ($O_{\#max} = 15$)

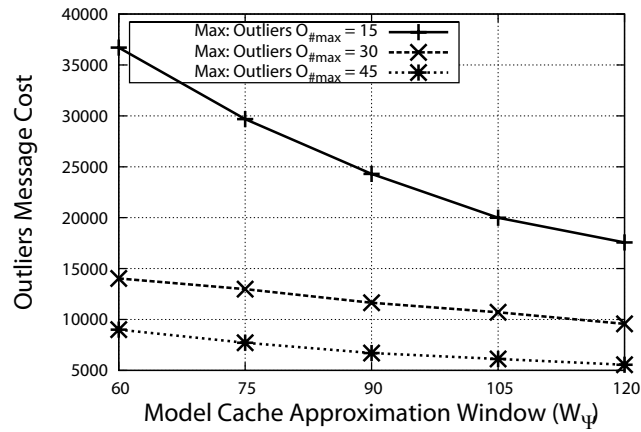
Figure 4.9: Message Cost for Inter- and Intra-Cluster Communication, Cache Construction and Transportation of Temperature Data to the Sink for Various $O_{\#max}$



(a) Message Cost ($O_{\#max} = 5$)



(b) Message Cost ($O_{\#max} = 10$)



(c) Message Cost ($O_{\#max} = 15$)

Figure 4.10: Outliers Message Cost for Temperature Data Set to the Sink for Various $O_{\#max}$

by a large margin not only for the same error threshold (i.e., $\epsilon = 0.2$) but even for lower error threshold thanks to adaptive modeling (Section 4.1.3) and dynamic approximation window (Section 4.1.3) that considerably reduce the inter- and intra-message cost and yield considerably fewer outliers.

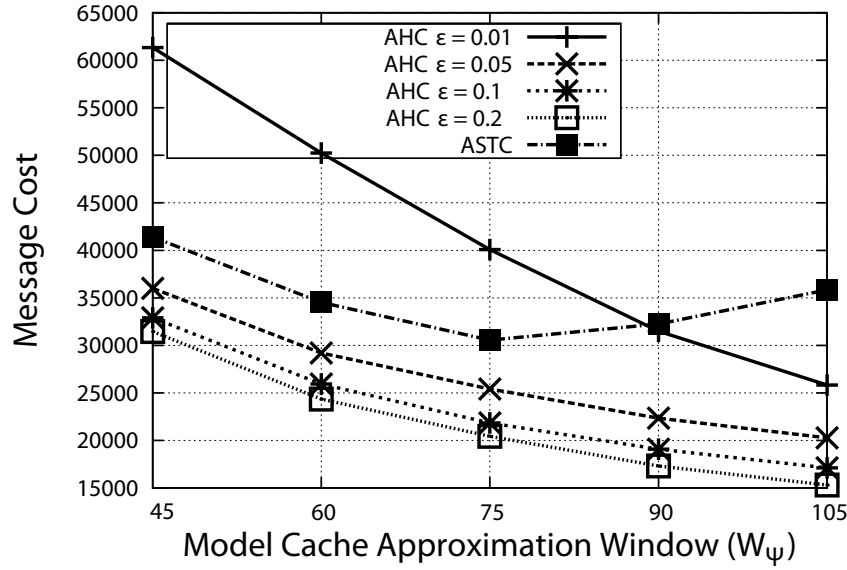


Figure 4.11: Impact of ϵ on Message Cost on ASTC and AHC

4.4.3 Comparison to Related Work

EFFICIENCY Fig. 4.12 depicts the message overhead comparison between AHC, ASTC, PAQ and DTC. We observe that AHC clearly outperforms the other techniques in terms of message cost, which was the design goal for AHC. The large gap between PAQ and AHC is due to the design choices, as PAQ targets realtime monitoring applications, hence in contrast to AHC it cannot exploit the temporal redundancies. Whereas, ASTC similar to AHC not only exploits the spatial and temporal redundancies but also exploits the delay-tolerance. Hence, it also successively collects the data samples, compresses them and then the compressed models are sent to the sink collectively. However, adaptive model and dynamic prediction window of AHC give it an edge over ASTC that reduce the message cost further. Consequently, we can conclude that if the application can tolerate delays in data collection then AHC could be a better choice, as its design

takes into consideration the application delay tolerance. Hence, for delay tolerant applications AHC can compress the data even further and reduce the message cost.

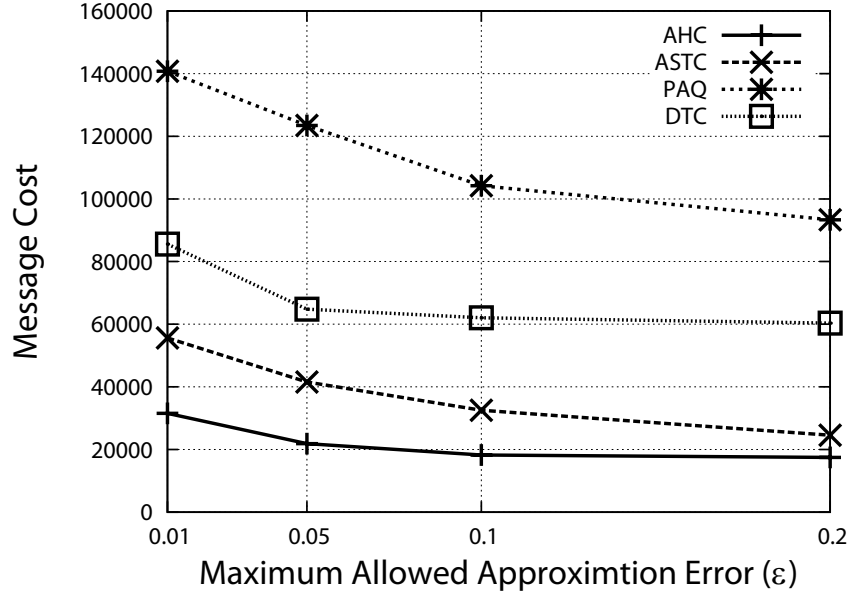


Figure 4.12: Message Cost Comparison Between Various Compression Scheme for Different Error Thresholds ϵ

ACCURACY In Fig. 4.13, we depict the mean error of all sensor nodes in the network for the entire length of monitored data for AHC, ASTC and PAQ. The figure depicts for AHC the attained error versus the maximum error thresholds for various prediction windows. The error thresholds set in the simulations mean very high accuracy requirement. The error threshold of $\epsilon = 0.01$, $\epsilon = 0.05$ and $\epsilon = 0.1$ represent maximum error of 0.044%, 0.22% and 0.43% of the mean sample value. In Fig. 4.13, we observe an increase in the attained error but it always remains well below the defined maximum threshold. We can observe that all three schemes easily attain the user accuracy requirement. However, the message cost incurred to attain the desired accuracy level shows the efficiency of the scheme. We can put the results from Fig. 4.13 in perspective by looking at the message cost (Fig. 4.12) incurred by each scheme to achieve the accuracy level. By considering both figures, we observe that AHC is manifolds more efficient than ASTC and PAQ in terms of message cost in order to meet the same user accuracy requirement. Hence, AHC is able to meet the user

requirement with fewer number of messages that effectively reduces the energy consumption and can prolong the life of the individual nodes and the network overall.

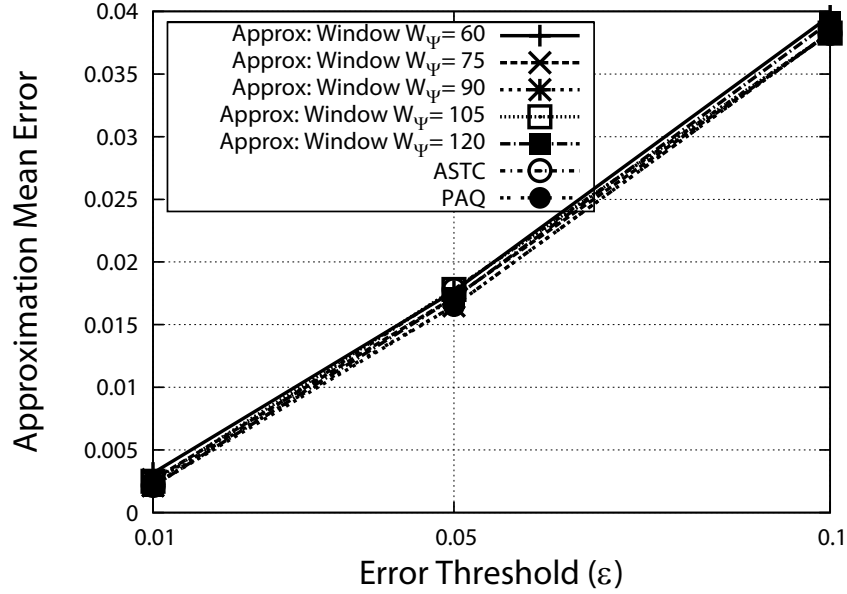


Figure 4.13: Mean Approximation Error on Sink for Temperature Data set

Note: We cannot include DTC in our comparison w.r.t. accuracy, since there is no equivalent parameter for error threshold ϵ in DTC. Nevertheless, Fig. 4.12 still gives a good comparison of the discussed compression methods. In Fig. 4.12, both axes of the diagram correspond the parameters that exist in all classes of the compression schemes mentioned here. Our implementation of DTC involves transmitting the largest DCT coefficients of the transformed temporal data. The number of transmitted coefficients is determined before data gathering such that a certain level of accuracy is achieved when the DCT coefficients are sent to the sink. Determining the outliers for a given error threshold ϵ requires inverse DCT computation at the sensor nodes which is beyond the hardware capabilities of these nodes. Therefore, it is not possible to compare DTC with ASTC, AHC and PAQ based on the error threshold ϵ as such an implementation of DTC that finds outliers depending on a given ϵ is not realistic.

4.4.4 Versatility - Phenomena Independence

In order to evaluate the versatility of the proposed scheme, we also carried out extensive simulations (however, to save space we show here only limited results), similar to temperature, for the humidity data that is also available in the real-world data set [S. Madden, 2003](#).

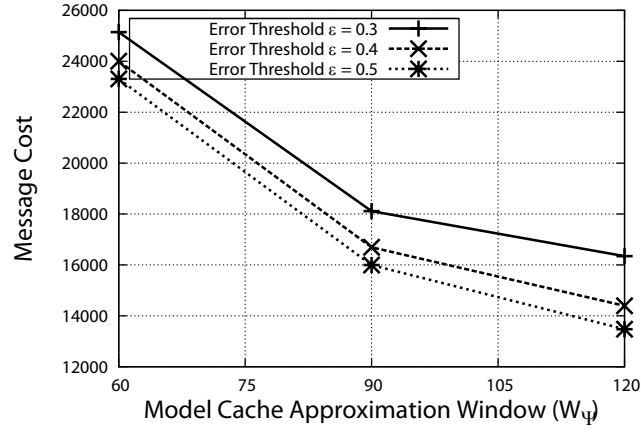
EFFICIENCY Similar to temperature data, Fig. 4.14 depicts the message cost results for humidity. Fig. 4.14 (a) shows the total message cost incurred in inter-intra cluster communication, model cache and outliers transportation. Fig. 4.14 (b) and (c) further break down the message cost. Fig. 4.14 (c) shows only the message cost related to transporting the outliers to the sink, whereas Fig. 4.14 (b) depicts the rest of the message cost related to inter-intra cluster message exchange and model caches transportation. The trends observed in Fig. 4.14 are similar to the ones discussed related to the temperature data, hence we do not discuss them again here. However, these results signify the versatility, consistency and adaptability of the proposed scheme to various data types.

ACCURACY Fig. 4.15 depicts the mean approximation error for the humidity data set reproduced from its model caches on the sink. Similar to temperature data set we can observe that the proposed scheme not only meets the accuracy requirement but easily exceeds the desired accuracy level. We can further observe that the reproduced data error remains well below the error threshold even for one standard deviation around the mean.

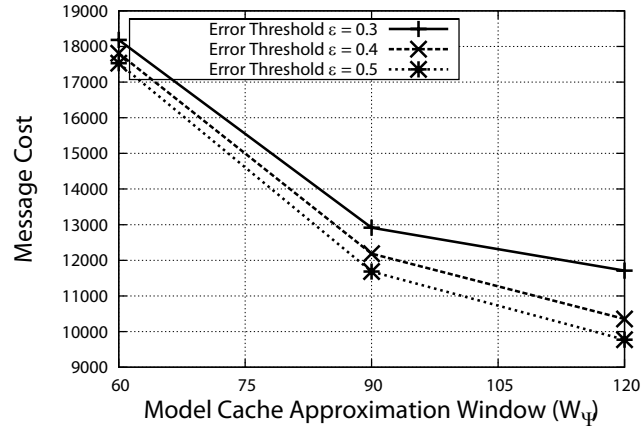
4.5 CHAPTER SUMMARY

In this chapter, we have presented Delay-Tolerant Spatio-Temporal Data compression scheme for of wireless sensor networks corresponding to Temporal Compression and Spatial Compression modules of the proposed framework. We provided adaptive mechanisms that allows the proposed scheme to be applicable to wide variety of applications. The proposed schemes maximizes the data compression both in time and space. We have also demonstrated the efficiency and accuracy achieved by our proposed spatio-temporal compression scheme.

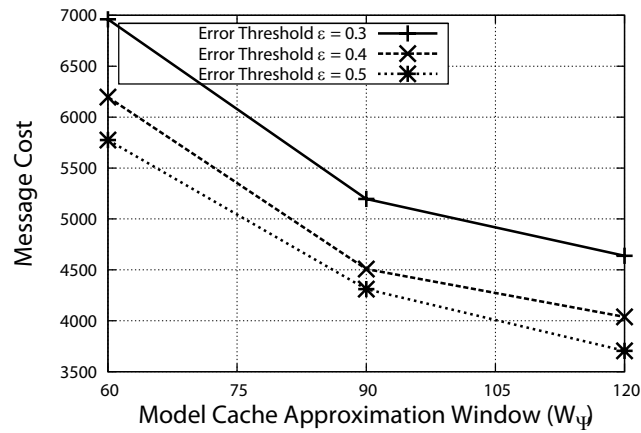
The proposed scheme is, however, vulnerable to data errors. The data modeling scheme uses the sampled data to construct models. If the sam-



(a) Total Message Cost



(b) Inter-intra Message Cost



(c) Outliers Message Cost

Figure 4.14: Message Cost for Compressing and Transporting Humidity Data to the Sink

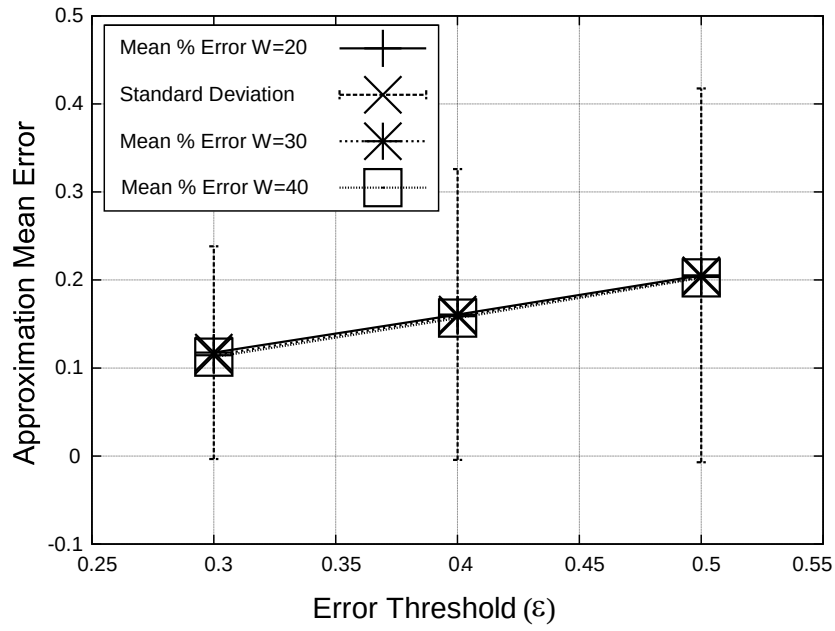


Figure 4.15: Mean Approximation Error on Sink for Humidity Data Set

pled data would contain corrupted samples, the resulting models will also be corrupted, as the model tries to correspond closest to the sampled data. Hence, the data reproduced on the sink is also corrupted. In the next chapter we address the data corruption issue and extend the proposed scheme to be resilient to a large class of faults that effectively result into data errors.

5

FAULT-TOLERANT DATA COMPRESSION

Wireless Sensor Networks (WSNs), typically comprising of hundreds of battery-powered sensor nodes, get deployed for monitoring environmental attributes and transporting the data to a central computer (sink) for processing. Given the constrained bandwidth and energy resources, WSNs often employ data compression schemes to efficiently collect and transfer the sensed data. However, various factors such as low node reliability (node crashes), harsh operating environments and communication faults can compromise the core objective of accurate collection and delivery of the sensor data. However, most current data acquisition and compression schemes fail to consider either the source faults or the resulting data errors. Consequently, the data collated at the sink is also often erroneous.

Here, we propose a novel and efficient fault-tolerance technique that extends from the compression schemes proposed in Chapter 4. In contrast to existing schemes that utilize instantaneous sensor readings, we propose a model-based scheme to provide fault-tolerance. Being fully distributed, our scheme corrects data errors at the point of origin (faulty sensor nodes), and thus avoids costly transmissions of corrupt data. The proposed scheme is flexible and scalable to ensure easy integration also with the varied available compression techniques and to also enhance compression effectiveness by correcting the erroneous samples. The overhead of the proposed scheme, in terms of computation and memory, is minimal. In terms of messages, the scheme not only exploits the background traffic to avoid new messages but also reduces the existing data collection traffic through the removal of erroneous samples.

5.1 FAULT ABSTRACTION AND MODELING

We refer to the environmental data that needs to be collected (e.g., temperature, pressure) as *reference* data. The data actually collected by the sensor nodes is the *sampled* data. The sampled data may not necessarily be same as the reference data because of the faults causing data errors. Hence, the

sampled data is often not good enough to be used as the reference data. Moreover, the instantaneous (sampled) values are not suitable for fault detection because of the erroneous samples. Hence, we need a time spread (i.e., time-series) data approximating mechanism to approximate the reference data. Consequently, we propose a scheme where we first construct a model out of the sampled data and use the model to represent the reference data. The model is then used to detect and correct the erroneous samples. The model does not depend on a single instantaneous value, hence it is relatively less prone to the faults and the resulting data errors. Fig. 5.1 illustrates the reference data and the model approximation of the reference data (based on model proposed in Sec. 5.1.2) with 10% and 20% corrupted samples. We observe that the model approximation of the corrupted data is very close to the reference data. Hence, we assume that the selected model provides a good approximation of the reference data.

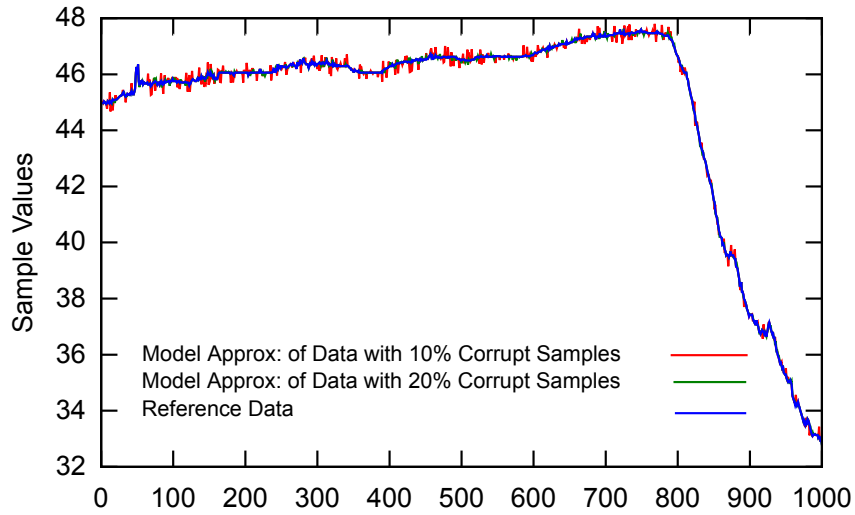


Figure 5.1: Reference Data vs Model Approximation

5.1.1 The Fault Abstraction

Various functional faults result into data errors. Examples include sensor faults generating wrong readings, bit flips in memory affecting the sensor data, non-recommended deployment conditions resulting in wrong calibrations, incorrect sampling algorithms wrongly transforming data, etc. However, we do not detect the faults explicitly, rather we focus on the detection and correction of erroneous data produced by the underlying faults.

Consequently, we do not require any specific knowledge of a given fault but still implicitly correct it by correcting sampling errors. Hence, in the rest of the manuscript, we will be dealing with data errors (or erroneous samples) instead of faults directly. In order to detect an erroneous sample, we define it as follows:

Definition 1. *An erroneous sample $v_e(t)$ at time t deviates at least ξ units from the reference value $v(t)$, i.e., $|v_e(t) - v(t)| > \xi$.*

As the sensor nodes do not have the reference value, the functional faults causing erroneous samples deviating less than ξ from the reference values are not discernible from the non-erroneous sampled data (as detailed in Sec. 5.3.1).

5.1.2 Data Error Detection Models

As discussed earlier, we derive a model from sampled data for reference data approximation that we can afterwards use to detect data errors. The types of models that are most suitable for data modeling and tracking, and that are also commonly used in WSN, include Kalman filter [Min and Chung, 2010](#), Probability Density Function (PDF) [Yoon and Shahabi, 2007](#) and Auto-regressive (AR) models [Tulone and Samuel Madden, 2006](#). Kalman filters are computationally expensive for resource constrained sensors. PDF requires expensive training routines (up to 15 days [L. Wang and Deshpande, 2008](#)) and expects the data patterns to remain consistent. AR models, on the other hand, are adaptive [Tulone and Samuel Madden, 2006](#) and computationally inexpensive [Borgne et al., 2007](#). Moreover, these models are already used by a large body of existing work in our target domain of data compression. Examples include [Tulone and Samuel Madden, 2006](#) [Ali et al., 2011a](#) [C. Wang et al., 2012](#) [Borgne et al., 2007](#) [Miranda et al., 2013](#) [P. Jiang and S.-Q. Li, 2010](#) [G. Li and Y. Wang, 2013](#). In order to detect the errors and eliminate them, we are interested in reducing the noise as much as possible. Accordingly, AR models are well suited acting as low pass noise filters. We also prefer AR models because they are readily available and already computed by the proposed compression scheme as discussed in Section. 3.3. The reuse of the available models allows us to save resources both in terms of memory and computation. It additionally reduces the overall complexity of the scheme.

The proposed Fault-Tolerant Data Compression (FTDC) scheme provides mechanisms of evaluating whether a certain sample, that is being

approximated, is erroneous. Similar to standard schemes, FTDC also uses the same sampled data and constructs the models using the same data. However, because model is constructed from a large set of sampled values, depending on the rate of data corruption, the model is only partially affected by the data corruption. Hence, the model can still predict values close to reference values even for the corrupted samples. For standard compression scheme, if model cannot approximate a sample, as per Eq. 10, it is due to outliers. But, for FTDC it could be due to outliers or corrupted samples. Accordingly, if the model can't approximate a certain sample, FTDC does not always replace the data points with the sampled data (standard technique will always do, considering it an outlier). Instead, FTDC uses various heuristics to decide whether it is an outlier or a corrupted sample to choose either the model predicted data or the sampled data. If FTDC detects a sample to be erroneous, it discards the collected sample and uses the model approximation instead. The main difference between FTDC and standard time-series compression scheme, such as PAQ/ASTC, is that PAQ and ASTC trust the sampled data to be the reference data and try to approximate it as close as possible.

However, FTDC also has its limitations due to the fact that the model is constructed from sampled data, which can potentially be corrupted. Hence, the model itself will be biased to some or greater degree, depending on the extent of sample data corruption. However, to cope with this situation we keep the training length long enough to mitigate the corruption of model parameters. But, if the noise is uniform with high corruption rate or concentrated around the training samples, the model will be accordingly affected by the noise. As, we demonstrate in our simulations in Sec. 5.4.2 that FTDC is able to detect most of the erroneous samples, even when the models are constructed from corrupted data. But, if more than 50% of the samples are corrupted, the performance of FTDC starts to deteriorate, though it still is better than the base performance of the compression scheme without FTDC.

Before, we develop the fault-tolerance scheme, we discuss briefly how time-series compression schemes are vulnerable to faults that cause data errors.

5.2 FAULT VULNERABILITY OF COMPRESSION SCHEMES

Some compression schemes are more vulnerable to faults than others, e.g., Compressive Sensing [Luo et al., 2009](#) schemes are less vulnerable to faults than time-series based schemes because of the nature of the underlying mechanisms. The time series based compression, while very efficient for data compression, are particularly vulnerable to the sampling errors as they fundamentally rely on the correctness of the sampled data. Standard time series compression schemes do not have a mechanism to determine if the collected samples are erroneous. Hence, they try to approximate the sampled data as close as possible. In the absence of a suitable mechanism capable of detecting the erroneous data, these schemes end up approximating the erroneous data! Overall, given the high vulnerability of time-series schemes to data errors but with their extensive application in WSN [Tulone and Samuel Madden, 2006](#)[Ali et al., 2011a](#)[C. Wang et al., 2012](#) [Borgne et al., 2007](#) [Miranda et al., 2013](#) [P. Jiang and S.-Q. Li, 2010](#) [G. Li and Y. Wang, 2013](#), we focus on this class of compression schemes. Though, the developed fault-tolerance scheme extends from the compression scheme proposed in chapter 4, it can easily be adapted to be used with other times series schemes [Tulone and Samuel Madden, 2006](#)[Ali et al., 2011a](#)[C. Wang et al., 2012](#) [Borgne et al., 2007](#) [Miranda et al., 2013](#) [P. Jiang and S.-Q. Li, 2010](#) [G. Li and Y. Wang, 2013](#). Hence, next we will be presenting FTDC in a relatively broader context of being applicable to the class of time-series compression schemes, instead of being just an extension of AHC.

5.3 MODEL-BASED DATA-FAULT TOLERANCE

We now detail the proposed fault-tolerance scheme for time-series compression schemes. We elaborate two classes of data errors, i.e., sporadic and permanent errors, and devise the mechanisms to detect and also correct them whenever possible.

5.3.1 Sporadic Data Errors

Sporadic data errors refer to SHORT data errors and also NOISE when it lasts for less than one fourth the training queue length (T). For sporadic

data errors we first propose a mechanism to detect the anomalies in the sampled data and then describe a mechanism to correct these anomalies. The detection process is divided into two stages, (a) local node level detection, and (b) cluster level detection. Most of the time series schemes such as PAQ [Tulone and Samuel Madden, 2006](#) and ASTC [Ali et al., 2011a](#) already have a clustering scheme implemented. Hence, for FTDE we do not have to additionally implement any clustering scheme utilizing available clusters.

Node Level Detection

Each node initially detects sporadic data errors. However, as shown later, individual nodes can misclassify some legitimate samples as data errors. Hence, a second cluster-level stage is needed to scrutinize the classification in the first stage (node level) and make final classification.

Sensor nodes first construct a model out of the sampled data as described in Sec. 5.1.2. Using the constructed model, the estimated reference data is calculated using Eq. (11). The sampled values not appropriately approximated by the model within the user defined accuracy bounds (ϵ), but still are termed as *outliers*. Accordingly, if the difference between the sampled value and the model estimated value exceeds ϵ , it is classified as an outlier. However, during *normal operation* of the WSN, the value of the physical attributes generally do not significantly vary around their mean, hence the outliers are expected to be within a specified valid range. Hence, as described in Def. 1, we have a threshold ξ to detect data errors, and if the approximation error is beyond ξ , it is classified as a data error.

Algorithm 5.1 Error and Outliers Detection Algorithm

```

1: for  $\hat{v}(t)$  in  $V(t)$  do
2:    $e := |\hat{v}(t) - v(t)|$ 
3:   if  $e > \epsilon$  and  $e < \xi$  then
4:      $\hat{v}(t) \leftarrow v(t)$ ;
5:     Append  $v(t)$  to  $O(t)$ ;
6:   else if  $e \geq \xi$  then
7:     Append  $v(t)$  to  $E(t)$ ;
8:   end if
9: end for

```

Alg. 5.1 describes the process on a sensor node for the classification of sampled values into outliers or data errors. The set V refers to the approximated reference values $\hat{v}(t)$ as estimated by the model. Alg. 5.1 tracks

data errors and outliers by constructing two sets, namely $\mathcal{O}(t)$, set of samples classified as outliers and $\mathcal{E}(t)$, set of samples classified as data errors. Based on the outlier and data error threshold (ϵ and ξ), the estimated value is classified either as an outlier (Alg. 5.1, Line 5), or as a data error (Alg. 5.1, Line 7), or it is correctly estimated by the compression scheme. The lists of outliers (the set $\mathcal{O}$) and preliminary errors set (data errors) (the set $\mathcal{E}$) are then sent to the cluster head for further processing. The data errors are further processed to identify if any sample values originating from an event were incorrectly classified as data errors.

Cluster Level Detection

A physical event, i.e., sudden change in physical attribute such as temperature, pressure, etc. may also exhibit behavior similar to sporadic errors and consequently be misclassified as a data error during stage 1. Hence, we consider the node level sporadic data error detection only as an initial/temporary classification, because a single node does not have enough information to determine if the classification of the sample as a data error is due to corruption of the sample by an underlying functional fault or it is a result of a physical event. Hence, we have devised a second stage cluster level sporadic data error detection mechanism that can reclassify the sample values classified as data errors in the first stage to either data errors or events. FTDE requires simple 1-hop clusters that are typically available in different compression schemes [Ali et al., 2011a](#) [Tulone and Samuel Madden, 2006](#). The cluster head collects the data errors from all the member nodes in the cluster, hence it can use this information to verify if the initial classification by a node was incorrect and then reclassify the incorrectly classified samples as outliers.

In order to differentiate a data error from a physical event, we develop a formal notation for data errors and then develop a logical basis for differentiating the data errors from physical events. We now define the terms related to the probabilistic random error events and physical events. The classification of a sample value as an error (hence, detection of a sporadic fault or an event) is a random event.

Definition 2. *We define the random event that a sensor node classifies a sample as a data error due to corruption by an underlying functional fault as a data error event.*

Definition 3. We define the random event that a sensor node classifies a sample as an event arising due to a change (sudden, extreme, etc) change in the physical phenomenon as a phenomenon event.

Note that a single node cannot differentiate the two events and in both cases it classifies the samples as data errors, because both the data error event and phenomenon event exhibit similar behavior at the level of a single node.

Next, we discuss both events, i.e., *data error event* and *phenomenon event* and analyze how they can be differentiated using the samples of multiple sensor nodes in the cluster.

Distinguishing Data Error Events from Phenomenon Events: We now develop the basis to differentiate the data error events from phenomenon events and then use it to classify them on the cluster head.

Axiom 1. *Data error events taking place on two separate sensor nodes are independent.*

Based on the discussion in Sec. 5.3.1, the underlying cause of the data errors are predominantly random hardware faults resulting in random data errors. As each node has its own underlying fault generating the errors and the fault of one node does not influence the sampling of another node. The underlying fault in each sensor nodes can be caused by various factors, such as, build quality, faulty sensors and/or external parameters such as very high/low temperature, corrosion and other unexpected damages. These factors affect each sensor node independently. Moreover, faults of one node cannot cause errors on another node. Hence, a faulty sensor on Node 1 will not corrupt the samples of Node 2. Accordingly, if sporadic data errors on Nodes 1 and 2 are given by $E_1(t)$ and $E_2(t)$, then *data error events* occurring on two different nodes are independent from each other, i.e., $P(E_1(t) \wedge E_2(t)) = P(E_1(t))P(E_2(t))$.

As data error events are random events (Def. 2):

Theorem 5.3.1. *Data error events occurring on two different sensor nodes are independent random variables.*

Proof: A data error event being detected on a sensor node is a random stochastic event. Accordingly, we assume R_1 and R_2 represent random variables for data error events on sensor node S_1 and S_2 . For R_1 and R_2 to be independent, each data error event (random event) of R_1 must be independent of data error events of R_2 and vice-versa. From Axiom

1, each data error event $E_1(t)$ and $E_2(t)$ belonging to random variables R_1 and R_2 is independent because of physically independent hardware faults that cannot influence the other sensor node, hence R_1 and R_2 are also independent random variables.

Corollary 5.3.2. *The random variables for data error events on a set of n sensor nodes are independent.*

Using Theorem 5.3.1, each pair of data error event random variables R_i and R_j belonging to sensor pair S_i and S_j , $i \neq j$ is independent. Hence, all data error event random variables for n sensor nodes are independent of each other.

Axiom 2. *An environmental event sensed separately by two sensor nodes at time t is not independent.*

Unlike data error events, where each event is generated by a separate stochastic process (hardware fault), the environmental events are essentially replication of the same random variable, generated by a single stochastic process (e.g., suddenly change in temperature/pressure) that is sampled by multiple sensor nodes. Hence, an environmental event happening on two sensor nodes at time t is not independent.

Axiom 2 can be extended, similar to Axiom 1, to show that the environmental random variables (random variables related to environmental events) are not independent.

Corollary 5.3.2 and Axiom 2 set the foundation for differentiating data error events from environment events. Accordingly, the cluster head can determine if event random variables (initially reported by all the sensors as data error events) of two sensor nodes are independent then these are due to data error events, otherwise they are caused by environment events. However, cluster head needs to know the joint probabilities and the probabilities of the events to determine independence of the events, as given in Axiom 1. Moreover, sensor nodes have very limited computational, memory and energy resources. Given that we have shown that (a) the error events are independent, while (b) phenomenon events are not independent, and (c) the evaluation on the sensor nodes is not computationally feasible, we have devised a simpler measure to infer the independence of the events to detect the data error events.

Detection of Data Error and Phenomenon Events: The cluster head collects the preliminary errors set (which contains both the data error events and phenomenon events but all are initially classified as data error events)

from all the cluster members. Due to the collected preliminary error sets from the member nodes, the cluster head possesses a broader localized view on the events. The cluster head uses the results from Corollary 5.3.2 to detect data error events. However, as discussed earlier, using Corollary 5.3.2 is complex as the cluster heads do not have the joint probabilities. Hence, we propose a simple two stage mechanism to detect data error events.

The cluster head (S_H) constructs a cumulative preliminary error set ($\mathbb{E}_{S_H}(t)$, Alg. 5.2, Line 1) by merging preliminary error sets of its member nodes ($\mathbb{E}_{S_i}(t)$, where $\mathbb{E}_{S_i}(t)$ is the preliminary error set from sensor node S_i belonging to the cluster C_j). The cluster head detects the error events in two stages using the cumulative preliminary errors set.

In the first stage, the cluster head determines if more than one sensor node detected a data error event for a given time index (Alg. 5.2, Line 3). As the data error events are independent (Axiom 1), it is highly unlikely that more than one node will detect the data error events at the same time index. Even if they do detect data error events at the same time, they will still be independent. Whereas, for a phenomenon event we can easily observe more than one nodes reporting a data error event (due to mis-classification during in the preliminary error set), which actually will be indicative of a phenomenon event, rather than a data error. However, we cannot fully rule out the possibility of data error events taking place at more than one nodes at a given time index as a coincidence, depending on the cluster size and node density. Hence, if the cluster head detects a data error on more than one sensor node, it needs to process further to verify and finally classify the concerned sample either as a data error event or a phenomenon event.

In the second stage, the cluster head processes the values detected as data error events in the first stage. It computes the standard deviation of these error values for all cluster member nodes that reported these data error events (Alg. 5.2, Line 4). If the error values were generated by a phenomenon event, the sampled values collected by various nodes will be highly correlated (as they are not independent) and the standard deviation is expected to be low. Whereas, for error events, even if they happen to occur at the same time index, it is highly unlikely that the values sampled due to the data errors are correlated. Hence, we define a tolerance threshold for the standard deviation around the mean for the errors (ρ) to qualify as phenomenon event, i.e., if the standard deviation of the error values are beyond the defined threshold then they are classified as data

error events (Alg. 5.2, Line 5), otherwise, they are classified as the phenomenon event (Alg. 5.2, Line 6-8). In Alg. 5.2, $\mathbb{I}(t)$, $\mathbb{P}(t)$ refer to finally classified data error events and phenomenon events sets respectively. The sample values related to final phenomenon event set $\mathbb{P}(t)$ are passed to the sink as outliers, so that the sink may appropriately approximate the phenomenon anomalies, whereas, the data errors classified as data error events are suppressed and not sent to the sink. Instead model is used for sample estimation .

Algorithm 5.2 Classification of preliminarily faults into fault and phenomenon events

```

1: Construct  $\mathbb{E}_{S_H}$  from  $\mathbb{E}_{S_i}$  for  $S_i \in C_j$ 
2: for  $t_i \in T$  do
3:   if  $\text{COUNT}_{S_j \in C_k}(\mathbb{E}_{S_H}(S_j, t_i)) > 1$  then
4:     if  $\text{SD}_{S_j \in C_k}(\mathbb{E}_{S_H}(S_j, t_i)) > \rho$  then
5:        $\mathbb{I}(t) \leftarrow \mathbb{I}(t) \cup \left( \bigcup_{S_j \in C_k} \mathbb{E}_{S_H}(S_j, t_i) \right);$ 
6:     else'
7:        $\mathbb{P}(t) \leftarrow \mathbb{P}(t) \cup \left( \bigcup_{S_j \in C_k} \mathbb{E}_{S_H}(S_j, t_i) \right);$ 
8:     end if
9:   else
10:     $\mathbb{I}(t) \leftarrow \mathbb{I}(t) \cup \left( \bigcup_{S_j \in C_k} \mathbb{E}_{S_H}(S_j, t_i) \right);$ 
11:   end if
12: end for
```

Reducing False Positive Errors and False Negative Outliers: We demonstrate later in simulations (Sec. 5.4) that we can also use the mechanism developed in stage 2 (Sec. 5.3.1) to reduce the false positive errors and false negative outliers. We can keep ξ lower than desired value and let the outliers be classified as errors in stage 1 (Sec. 5.3.1) and exploit the correlations analysis of Stage 2 (Sec. 5.3.1) to correctly classify them back to outliers. However, it will reduce the probability of the errors to be incorrectly classified as outliers.

Event duration and sampling rate: We have, so far, used only the spatial sampling information to detect error events, However, we can also use temporal information to aid in such classification, i.e., a phenomenon event generally happens for a certain time duration and successive samples might actually have the same phenomenon event detected. However, detection of a phenomenon event for successive time indexes is highly dependent on the sampling frequency [Hempstead et al., 2008](#) and the du-

ration of the physical events. If the sampling frequency is lower than the duration of the physical event then the successive samples may not indicate any phenomenon event. In typical WSN deployments, targeting long term monitoring of a physical phenomenon, the sampling frequency is ranges from many minutes or even hours. Hence, temporal sampling information may not be as reliable as the spatial information.

5.3.2 Complete Data Corruption

Complete data corruption refers to CONSTANT data errors and NOISE when the number of erroneous samples is more than at least quarter of the training length. As we focus on time-series based compression schemes, hence we exploit the underlying model construction and acceptance mechanism of compression scheme for fault detection. Time-series based data compression schemes construct data approximation models. These models are constructed by one or more nodes in a 1-hop cluster that are used to approximate the sampled values of the other sensor nodes in the cluster. However, for a model to approximate the samples of a given node it should be accepted by the given node. Hence, in the acceptance phase, the data approximation model is sent to the node whose sampled data is to be approximated. A model is only considered to be valid and accepted by a sensor node if it can approximate the sampled data with at the most $O_{\max}$ number of outliers. Otherwise, the model is rejected. Based on data approximation model acceptance mechanism we propose Theorem 5.3.3 to detect a faulty sensor node in a 1-hop cluster head, generating constantly corrupted samples. The following discussion is in the context of 1-hop clusters, as the model construction process is performed in a 1-hop cluster.

Theorem 5.3.3. *A faulty sensor node S_F with complete data corruption neither accepts data approximation models from other sensor nodes, nor do other nodes accept the data approximation model constructed by S_F .*

Proof: Assume n data samples $v(t)$ are collected between time t_m and t_n by the k non-faulty sensor nodes S_i and a faulty-sensor node S_F in a 1-hop cluster C ($\forall S_i, i \in \{1 \dots k\}$ and $S_F \in C$). If $v_{S_F}(t)$ and $v_{S_i}(t)$ represent the data sampled by faulty and non-faulty sensor nodes respectively, then for any data sample $|v_{S_F}(t_p) - v_{S_i}(t_p)| > \xi$ and $t_p \in \{t_m, \dots, t_n\}$ as given by Def. 1. Accordingly, if a model constructed by a faulty sensor node (Φ_F) is used to approximate the sample values of non-faulty sensor nodes, then $|v_{S_i}(t_p) - \hat{v}_{(S_i, \Phi_F)}(t_p)| > \xi$ and $t_p \in \{t_m, \dots, t_n\}$ where $\hat{v}_{S_i, \Phi_F}(t_p)$ is

the approximated data sample of sensor node S_i using model Φ_F of faulty sensor node S_F . Hence, for each sensor node $S_i \in C$, the approximation of sampled data using Φ_F results in more $O_{\max}$ number of outlier. Accordingly, Φ_F is rejected by all the sensor nodes $S_i \in C$. Similarly, if a model constructed by a non-faulty sensor node Φ_{S_i} is used to approximate the sampled values of faulty sensor node then $|v_{S_F}(t_p) - \hat{v}_{(S_F, \Phi_{S_i})}(t_p)| > \xi$ for $t_p \in \{t_m, \dots, t_n\}$. Accordingly, the faulty sensor node rejects the model from all the sensor nodes used to approximate its sampled values.

Using Theorem 5.3.3, a cluster head can detect if a sensor node is faulty. Accordingly, if the cluster head detects if a certain node is neither accepting the models from any other sensor node nor is the model from the given node being accepted by any other node, the cluster head can isolate the concerned sensor node as faulty sensor node.

In specific circumstances, a sensor node may incorrectly be identified faulty. One such situation is the change in phenomenon distribution such that one or more nodes from the cluster may sense different data samples than the rest of the sensor nodes. For example, in a sensor network sensing light levels, the nodes in the shadow may group themselves into a cluster and the ones in sun light group themselves in another cluster. However, over time the position of sun changes and one or more of the sensor nodes in the shade may start sensing more light than the rest of the sensor nodes in the cluster. The sensor nodes newly sensing more sun light will be rejected by the rest of the sensor nodes in the cluster. Hence, in order to cope with such conditions, we do not immediately classify a sensor node as faulty. Rather, when a sensor node is identified as faulty using Theorem 5.3.3, the node attempts to join the other surrounding clusters by requesting the model from the neighboring cluster. If the sensor node was initially rejected from its native cluster because of the changing phenomenon then it can find a model from the surrounding cluster that can fit its sample values.

5.4 PERFORMANCE EVALUATION

On this analytical basis, we now describe the experimental setting supporting our evaluations.

5.4.1 Simulation Settings

In order to conduct comprehensive evaluation simulations, we used the publicly available real-world data set [S. Madden, 2003](#) as a reference. We use TOSSIM to perform network simulations and MATLAB to reconstruct the signal at the sink. We implemented FTDE for PAQ and ASTC to evaluate the FTDE's adaptability to different schemes. We conduct simulations on humidity data as it varies steadily over time, hence, it allows to additionally evaluate the adaptability of the proposed scheme to changing dynamics.

We selected third order AR models with user defined desired approximation threshold for compression scheme $\epsilon = 0.1$, inter-node error threshold $\rho = 0.5$, data error threshold $\xi = 4$ and a training length of 75. In order to intensively evaluate FTDE, we performed controlled corruption of the original data set at the rate of 5%, 10%, 20%, 30%, 40%, 50%, 60%, and 70% by adding noise. As discussed earlier, we know from the literature that the aggregated data in WSN can be corrupt up to 49% and up to 60% for an individual node [Sharma et al., 2010](#). Hence, we successively corrupted data at the rate of 5% to 50%. However, we increased the corruption rate even further to 70%, as it allows us to evaluate the behavior of the proposed scheme when the amount of corrupt data is more than the real data. We observe very interesting results when FTDE starts to degrade for error rates beyond 50%, as the models constructed out of majority of corrupted samples actually match closely to the corrupted samples than the reference data. The added data errors are the combination of SHORT, NOISE and CONSTANT at varying degrees, as various real world data from sensor nodes contain these data errors at varying degrees [Sharma et al., 2010](#). The values for ξ was estimated by computing the standard deviation of the reference data. ξ is selected to be two times the standard deviation. Twice the standard deviation threshold causes some of outliers to be misclassified as errors in stage 1 (Sec. [5.3.1](#)). However, it reduces the chances of the erroneous samples to be classified as outliers. Due to our multistage error classification mechanism the misclassified outliers samples in stage 1 (Sec. [5.3.1](#)) are later correctly classified back to outliers in stage 2 (Sec. [5.3.1](#)).

Interestingly we do not have to explicitly adapt to the rate of data corruption to detect the faults. If the faults rate is lower and there are predominantly SHORT or short duration NOISE data errors they are corrected by the transient error detection mechanism. However, if the transient error

detection fails, the permanent error detection mechanisms runs in parallel to detect the CONSTANT and long duration NOISE data errors.

Scheme Specifics and Trade-offs

Different schemes have different underlying mechanism. Accordingly, FTDE adapts to exploit the underlying mechanisms in order to further enhance the performance. For example, ASTC selects a model by choosing the best fitted model from multiple sensor nodes. Hence, the performance of FTDE with ASTC considerably outperforms FTDE with PAQ concerning the accuracy of collected models and the reconstructed data at the sink. Interestingly, no special adaptations are needed in this case to take the advantage of the ASTC model selection mechanism.

Methodology and Performance Metrics

Our performance evaluation is based on two key metrics: *Accuracy* of collected data and *message efficiency* as an implied measure of the energy consumption. The humidity data set [S. Madden, 2003](#) is considered as the reference. The corrupted data set is used in the simulations. Please note that in reality we will not have the reference data, rather only the sampled data (potentially containing corrupted samples). The corrupted data used for the simulations is equivalent to the sampled data. We collect the data models and outliers at the sink through PAQ and ASTC, both enhanced by FTDE to detect and correct the sample errors in the network. The collected data (approximations using the received models and outliers) is then compared with the reference data to evaluate the effectiveness of FTDE. We define accuracy as the mean square error between the data collected assuming corrupted data and the reference data. The plots depict the aggregated mean square error of all the sensor nodes over the complete duration of approximated data as accuracy measure. We measure the message efficiency as the message overhead created by the selected enhanced compression scheme, i.e., either PAQ with FTDE or ASTC with FTDE. The message cost accounts for all the stages of both compression scheme and FTDE, i.e., modeling, outliers, errors and events detection, reporting to the cluster head and finally transporting the models and the outliers to the sink. In the first simulation study, we compare the performances of PAQ, FTDE-PAQ (PAQ enhanced with FTDE) and FTDE-PAQ-CH_{EF} (PAQ enhanced with FTDE and where the data collected by the cluster head (CH)

data is Error Free (EF)). In a second study, we compare the performance of ASTC to FTDE-ASTC (ASTC enhanced with FTDE).

5.4.2 Results

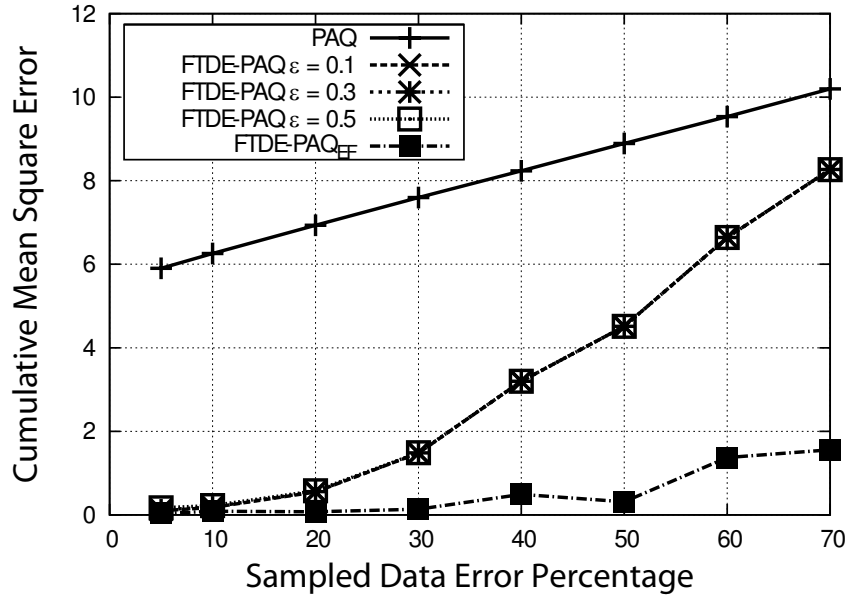


Figure 5.2: Mean Approximation Error Relative to Reference Data

Fig. 5.2 depicts the approximation error relative to the reference data for PAQ, FTDE-PAQ and FTDE-PAQ-CH_{EF}. We can easily observe that PAQ cannot detect the erroneous samples and blindly tries to fit to the corrupted data. Hence, relative to the reference data the resulting error is considerably larger than the user required error threshold $\epsilon = 0.1$. FTDE, on the other hand, can easily detect the erroneous samples and stays close to the user defined error threshold up to 10% of data corruption. The impact of error threshold is negligible on FTDE in terms of achieved cumulative mean square error, hence the results for various error thresholds overlap indistinguishably on the graph (though, accuracy values are distinguishable for individual nodes, which are not shown here). However, the performance of FTDE also starts to degrade with increasing data corruptions. As the sampled data (here the corrupted data) is used for constructing the models, hence the models also get corrupted (as discussed in Sec. 5.2). However, we can clearly observe that FTDE performance is con-

sistently better than the standard PAQ scheme. FTDE-PAQ-CH_{EF} depicts a hypothetical scenario, where the cluster head data is not corrupted and the data for the rest of the nodes is corrupted as usual. Due to error free data on the cluster head, the constructed model is error free. Accordingly, FTDE can easily detect the erroneous samples and the resulting error is considerably reduced. Though, this is a hypothetical scenario, it gives a very important hint that the performance of FTDE can be significantly improved by constructing the model on nodes that have the least corrupted data. This potential of FTDE is exploited by ASTC, as discussed later.

Fig. 5.2 effectively highlights capability of FTDE to detect erroneous samples, discard them and approximate the data in noisy conditions. The accuracy achieved by FTDE is within the user defined threshold for low error rates and considerably higher than standard compression scheme.

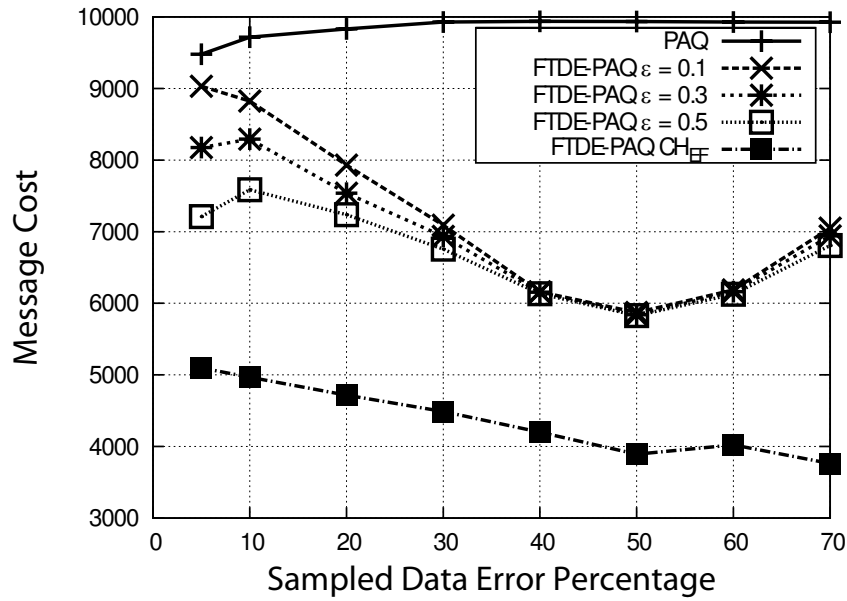


Figure 5.3: FTDE-PAQ Message Cost

Fig. 5.3 depicts the message cost incurred by PAQ, FTDE-PAQ and FTDE-PAQ-CH_{EF}. As the errors and outliers are indifferent to PAQ, hence it classifies all as outliers and accordingly transport them to the sink to maintain the accuracy. The large number of outliers entail a high message cost. In contrast, FTDE-PAQ can detect the erroneous samples and discard them using the proposed two-stage model based scheme. Due to true distributed nature of the scheme, we are able to detect and discard the

erroneous sample at the origin. Eliminating erroneous samples reduces the number of outliers to be reported, considerably reducing the message cost. We also observe that more the data is corrupted, more the samples are classified as erroneous and discarded and not transported to the sink, instead they are approximated through compression scheme models. This results in steady decline in message cost.

In order to investigate these results further, we simulated the FTDE-PAQ with additional error threshold of $\epsilon = 0.3$ and $\epsilon = 0.5$. These additional simulations show that we do actually observe a slight increase in message cost initially when the data corruption is increased from 5% to 10%. However, then message cost drops with increasing data corruption until 50%. It is noteworthy that $\epsilon = 0.1$ represents a very high level of accuracy requirement for the given data set, as it requires the model approximation to be within 0.2% of the original data. Hence, (a) there are many outliers produced due to high accuracy requirement, and (b) the model is not accurate, as it was constructed from the corrupted data. However, as we increase the data corruption rate, it reduces outliers and instead introduces more erroneous samples (as more outlier samples are also replaced by the erroneous samples). Once, data corruption rate crosses 50%, the corrupted samples become more abundant than the actual data. The constructed model built from these corrupted sampled and tries to approximate the corrupted samples instead, which results in mis-classification of errors as outliers increasing message cost.

Fig. 5.3 gives the interesting and surprising result. As the message cost includes all the message exchanges, including the message overhead of FTDE, hence we can observe that the effective message cost of FTDE is negative in comparison to the standard compression scheme. This is because the overhead introduced by the FTDE is negligible in comparison to the message overhead it reduces by detecting the erroneous samples and avoid to report them to the sink as outliers (that the standard compression scheme does). Hence, FTDE not only increases the approximation accuracy but additionally reduces the message cost.

FTDE has been designed such that it can fit to different compression schemes and exploit their capabilities to further enhance the data error detection. In contrast to PAQ, ASTC is an adaptive scheme that selects the best suitable models in the cluster. Hence, in order to test capability of FTDE to take advantage of ASTC's best adaptive mechanism, we designed the data sets that were not uniform as in PAQ simulations. Accordingly, the data corruption was carried out at different rates for different nodes,

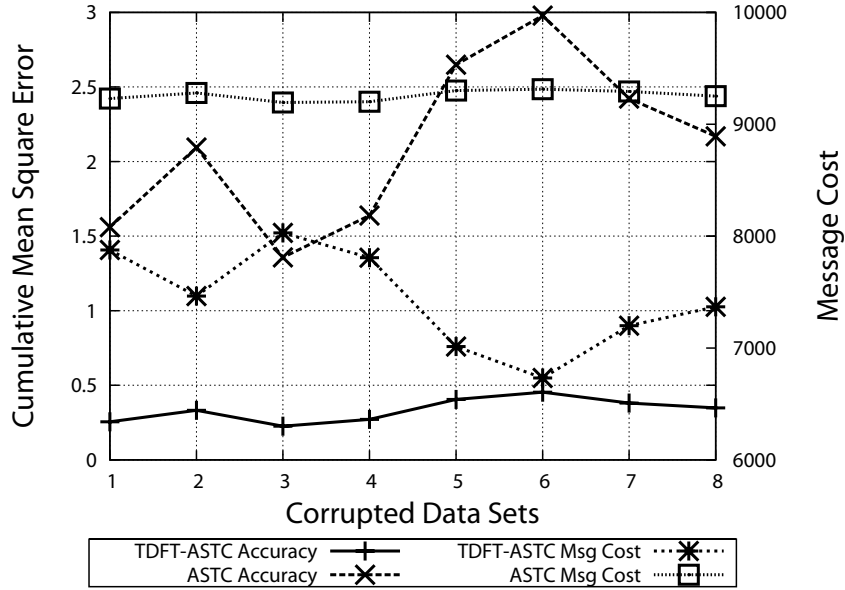


Figure 5.4: FTDE-ASTC Accuracy and Message Cost

ranging from 10% to 50%, which approximates the natural distribution of data errors better in contrast to uniform error rate across all nodes. In Fig. 5.4, we illustrate the performance results for FTDE-ASTC and plain ASTC w.r.t. accuracy and message efficiency. We observe that FTDE-ASTC easily maintains a mean square error of maximum 0.4 most of the time. This is because FTDE can exploit the best model selecting scheme of ASTC to detect the erroneous values and discard them to maintain low mean errors, which results in fewer message transmissions. Here also, FTDE-ASTC outperforms standard ASTC in all aspects.

5.5 CHAPTER SUMMARY

In this chapter, we have improved the spatio-temporal compression to be fault tolerant. We have shown that the proposed fault-tolerance scheme can detect the resulting data errors and correct them. The proposed scheme is applicable to large variety of faults and is agnostic to underlying faults as it detects the resulting data error. The reduced data errors results into improved data accuracy and reduced message cost because the models can approximate the environmental attribute better and fewer outlier sam-

ples need to be reported to the sink. The proposed fault-tolerance scheme is efficient both in terms of memory and computation time as it reuses the existing mechanisms of the underlying compression scheme. Simultaneously, it is flexible enough to be portable and adaptable to a class of time-series compression schemes, instead of being limited to one single compression scheme.

6

EFFICIENT PREDICTIVE MONITORING OF WSN

Wireless Sensor Networks (WSN) are deployed to monitor physical events, such as fire, or the state of physical objects, such as bridges, in order to support appropriate reaction to avoid potential damages. However, many situations require immediate attention or long reaction plan. Therefore, the classical approach of just detecting the physical events may not suffice in many cases. We present a WSN level event prediction scheme to forecast the physical events well in advance. Our approach is generic and allows to predict also the network events such as network partitioning, thus supporting proactive self* actions. For example, by monitoring and subsequently predicting trends on network load or sensor nodes energy levels, the WSN can proactively initiate self-reconfiguration to meet its desired operational requirements. The scheme collects the state of a specified attribute on the sink for a certain duration of time. Based on the collected history of attribute values, the future state of the targeted attributes are predicted using time series modeling. The forecasted future state is then used to detect generalized future events through our proposed event detection algorithm. Additionally, the proposed scheme is adaptable to cover multiple application domains.

6.1 INTRODUCTION

Wireless Sensor Networks (WSN) typically entail an aggregation of both sensing and communicating sensor nodes to result in an ad hoc network linking them to the base station or sink. The sensor nodes typically possess limited storage and computational capabilities and require low-energy operations to provide longevity of operational time.

WSN's are deployed for monitoring different environmental attributes. Based on the sensed data, the corresponding decisions are made to carry out the appropriate reaction if needed. The decisions to take actions are triggered by some events happening in the network. For example, while monitoring the physical pressure in a certain facility an event could be trig-

gered to indicate either high or low pressure. Similarly, we can have many events related to numerous attributes. In addition to the environmental events, there are also network events to be considered such as network partitioning. Various works exist for detecting different discrete events [Yick et al., 2008] such as, fire detection [L. Yu et al., 2005] and network partitioning [Rost and Balakrishnan, 2006a; K.-P. Shih et al., 2007; Shrivastava et al., 2005]. Most of these efforts develop excellent foundations, however, are tailored for specific scenarios. Other works do consider generic scenarios [Xue et al., 2006], but they suppose the event to take specific shapes and patterns. Also, all of these efforts focus on detecting the events *after* the events have already occurred. It could be already too late to react to many such events if the traditional approach of detecting and reacting to the events is followed. Consequently, it is either insufficient or inefficient just to react to the events. For example, if we detect network partitioning, the repair might require a long time and the required resources may not be available. Meanwhile, the functionality of network and hence the monitoring, which is the main objective of deployment, will be lost. Thus, reporting of such events beyond simple monitoring becomes highly useful if these events can be predicted in advance. The ideological shift from detecting the events to predicting them provides enough time window to take appropriate autonomic actions. Consequently, we could avoid or delay events from happening. Multiple efforts exist for forecasting in WSN [Landsiedel et al., 2005; Mini, Nath, et al., 2002; X. Wang et al., 2007]. However, most of them are either limited to predicting specifically a certain attribute such as energy or provide only node level short-term prediction for data compression to minimize data to be reported from the network.

It is very useful to combine generic prediction techniques with generalized event detection to predict the events and to carry out self* actions well in advance. To the best of our knowledge there exists no work that proposes generic event prediction. Here, we develop a generic scheme to predict the events. It predicts the future states of the network for the attribute of interest, e.g., temperature or residual energy. The developed generic event detection technique is used on "predicted future state of attributes" to effectively forecast the events. We target long-term predictions that require the history of the attribute to be long enough to contain sufficient system dynamics. However, a sensor node does not possess the computational resources required to model the complex dynamics in attribute values to accurately predict the future states. Hence, we collect multiple profiles of the considered attribute and conduct modeling on the sink. We

refer to the collection process of attribute values from network as *profiling*. Accordingly, a profile is the state of the attribute in the network at a specified instance of time. We use the technique, we developed in Chapter 4, to efficiently and accurately collect such a history of the attribute from the network. On this background this work makes three specific contributions, namely

- Generalized scheme design for sink-aided attribute profile prediction, allowing to predict varied physical and network events.
Generic event detection technique to detect events from the predicted profiles.
- Case study of network partition as validation for our efficient predictive monitoring scheme.

6.2 SYSTEM MODEL

The basic system model assumptions are the same as presented in Chapter 3. In addition, we also assume that the sensor nodes know their geographic position either using distributed localization methods [T. He et al., 2005] or GPS. A typical WSN deployment may contain hundreds or thousands of sensor nodes with varying densities according to the coverage requirements. We consider an arbitrary node distribution, provided the network is connected at deployment time. We assume all sensor nodes to be homogeneous. Hence, the sensor nodes have the same transmission range R and same initial battery capacity. We consider that nodes crash due to energy depletion only. We assume the events for predictions to be happening over a longer period of time, e.g., events that may take hours, days or even months to develop. We consider that events are (a) not spontaneous, (b) spatially correlated, (c) do not depend discretely on a single node, and (d) display attribute trends that can be predicted.

6.3 EFFICIENT PREDICTIVE MONITORING

We develop the proposed scheme in a modular manner for it to be applicable to a variety of scenarios. The scheme consists of three phases, i.e., *data collection phase*, *prediction phase* and *event detection phase*. In the data

collection phase, attribute values related to an event are periodically but efficiently fetched from the network on the sink. The prediction phase is used for predicting future states of the network for the interested attribute using the previous history of attribute fetched from the network during data collection phase. The main objective in event detection phase is to detect events in the predicted state of the network, obtained in prediction phase, essentially predicting the events. The techniques we propose in each phase are independent of the attribute to be monitored, thus fulfilling generality requirements of our scheme. It is important to highlight that we do not limit the proposed scheme to only these techniques. Rather, for a particular implementation, specialized additional techniques can be easily accommodated due to the proposed scheme's modular structure. These phases are individually detailed in the following sections.

6.3.1 Data Collection Phase

From the nature of the problem we can expect an efflux of data (sensed attribute values) from the network towards the sink. A simplistic periodic approach to collect data from each sensor node on the sink would lead to high communication and energy overhead on sensor nodes rendering the proposed scheme impracticable. Hence, for fetching the data an efficient data compression scheme, like AHC as discussed in Chapter 4, can be used.

6.3.2 The Prediction Phase

The prediction phase takes place on the sink. We regenerate the profiles for the desired attribute using the collected models of clusters. Generic time series modeling techniques are then used to model the complete history of each node to predict the future profiles.

The models received on the sink in data collection phase for each cluster are used to regenerate the attribute history through reverse transformation. It is rather a simple procedure as sink has the models for each cluster, so reverse transformation comprises of solving the cluster model equation for each node in each cluster constituting each cluster. Hence, the reverse transformation generates many complete profiles of the WSN. The reverse transformation forms a temporal stack of such profiles as shown in Fig. 6.1. The regenerated history of each node contains all the complex dynamics.

On the sink we can now take the complete history of each node and model its complete behavior using time series analysis for predictions as opposed to on node piecewise modeling. Individual models of each node can then be used to predict future values by fitting a prediction model, effectively predicting future profiles. The time series can be modeled in different ways [X. Wang et al., 2007]. Here, we use the widely used time domain modeling because of its general applicability.

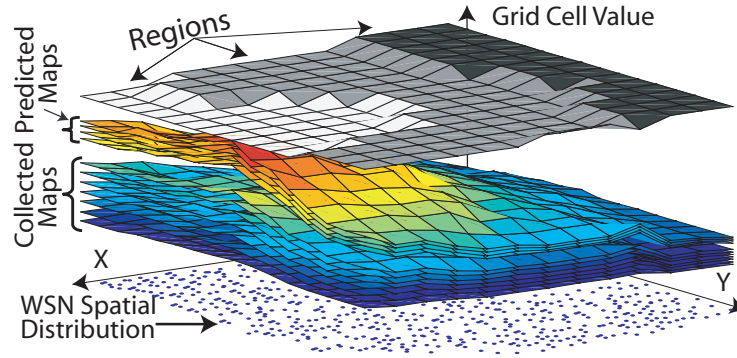


Figure 6.1: Temporal stack of the grid maps

Modeling Time Series:

A time series $X(t)$ can be modeled as a process containing following components

$$X(t) = T_t + S_t + R_t \quad (16)$$

where T_t is a trend, S_t is a function of the seasonal component with known period, and R_t is the random noise component. To keep the notion of generality valid for the proposed scheme we use a well known generalized technique termed Box-Jenkins Model to model a time series containing any of these components.

Box-Jenkins Model:

Box-Jenkins (BJ) model predicts a time series by fitting it an Autoregressive Integrated Moving Average (ARIMA) process. To fit an ARIMA process the model and the order of the model needs to be specified. The BJ model

provides a guideline to select the appropriate model, i.e., either Autoregressive (AR, Eq. (17)) or Moving Average (MA, Eq. (18))

$$X(t) = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} \quad (17)$$

$$X(t) = \theta_1 Z_{t-1} + \cdots + \theta_p Z_{t-p} \quad (18)$$

or combination of both, i.e., ARMA process as given in Eq. (5). It also gives the guideline for the model order selection. BJ modeling is a four steps procedure:

i.) *Data Preparation*: BJ model requires a time series to be stationary [Ljung, 1998](#). Therefore, if it contains trends and seasonal components then these should be appropriately removed. This can be achieved by either Least Square Polynomial Fitting (LSPF) or differencing as $X(t) = X(t) - X(t + u)$. For a simple linear trend, u is 1. For higher order trends or seasonal component of period s , u equals s . This operation is repeated until stationarity is achieved.

ii.) *Model Identification*: At this stage run-sequence plot or Autocorrelation Function (ACF) can be used to identify the stationarity of the time series and the order of the AR model. ACF for k lag is given by

$$\rho_k = \frac{\sum_{i=1}^{N-k} (X_i - \bar{X}) (X_{i+k} - \bar{X})}{\sum_{i=1}^N (X_i - \bar{X})^2} \quad (19)$$

where $\bar{X}$ is the mean value. Non-stationarity is often indicated by an ACF plot with very slow decay. Order of the AR and MA models are determined with the help of ACF and Partial Autocorrelation Function (PACF) [[Montgomery et al., 2008](#)]. To automate the model selection process either Akaike's Information Criterion (AIC) or Akaike's Final Prediction Error (FPE) [[Ljung, 1998](#)] can be used. Various models can be computed and compared by calculating either AIC or FPE. The least value of AIC or FPE ensures the best fit model.

iii.) *Parameter Estimation*: In this step the values of the ARMA model coefficients that give the best estimate of the series are determined. Iterative techniques are used for model parameter estimation [[Ljung, 1998](#)].

iv.) *Prediction*: Once the modeling is complete, it is simple to predict the series values using the estimated model. It comprises of calculating the

future values at next time instances and reversing all the transformations applied to the series in phase 1 for data preparation.

6.3.3 The Event Detection Phase

We now develop a generic event detection technique. Subsequently, using this technique we detect events in the predicted profiles of the attribute obtained in prediction phase, effectively predicting upcoming events in the network.

The main objective of our proposed scheme is to predict events. However, it is a very generic term in WSN. It may refer to an event as simple as sending a message to very complex events like network partitioning. We do not refer to a more generic definition of event, rather to a certain class of events that qualify for certain attributes. We target the events that either rely on physical or network attribute, do not depend on a single node and take relatively long time for development. In the system model (Section 6.2) we have described the domain of the events that we are targeting in our work. These can be exemplified by detection of temperature above a certain threshold in a certain part of network that is indicative of fire. Similarly, there are many other events associated with each physical attribute such as pressure, humidity etc. Our challenge here is to design an event detection mechanism that is generic and can be ported to wide range of scenarios. To cope with this problem we use here an abstraction of maps for WSN. For a WSN an eMap is an energy map that represents the current residual energy of the network [Yonggang Jerry Zhao, 2002], or tMap for temperature etc. Once a WSN is converted to a map for a certain attribute such as temperature, pressure etc., the events appear as regions in these maps. For example, in a tMap of WSN the part of the network that is beyond the given threshold of temperature will appear as a region in a tMap. Consequently, in our proposed scheme we define an event as a region of map whose values fall in the range of attribute values for which the event is defined. Using the abstraction of maps for WSN and regions for events we are able to keep the proposed scheme generic enough to be portable to different scenarios. Thus, the quantification of WSN space and the conversion of a WSN to a map abstraction is the key to detect generic events.

To quantify the continuous space of WSN profile and construct the map a grid is virtually placed over the WSN profile and each grid cell represents

the aggregated attribute of all the nodes located within the grid cell. We define the resultant quantification as *Grid Map* or simply *Map*.

Map Abstraction

In order to reach an acceptable spatial resolution with higher level abstraction of network as a map, we considered virtual grids and Voronoi diagram [Aurenhammer, 1991] techniques to segment WSN profile. Voronoi-based segmentation depends only on sensor node distribution and is static for a given node distribution. However, we require a segmentation strategy that allows variable spatial sampling to accommodate both the physical and network parameters. Such variability also allows us to detect events from a single node to a region of the network. Grid allows such flexibility therefore, we base our map construction on grid. The virtual grid or simply *grid* divides the WSN profile into fixed size squares or *grid cells* as shown in Fig. 6.1. Thus, nodes that fall within a cell are grouped. For the grid map construction, two parameters must be specified. The first parameter is the grid cell size γ , which is a spatial sampling or resolution parameter. The second parameter is the aggregation value ξ that a grid cell represents. Both parameters are essential for event detection. γ defines the geographic area covered by the grid cell. The number of nodes being grouped in a grid cell is dependant on γ . It can also be seen as a zooming parameter. Hence, it can be used to decide at which level the user intends to detect the event, i.e., very detailed (zoomed-in) level of node or an overview at the level of regions. The grid cell value ξ is an aggregate of the attribute values of the set of nodes in a cell. The choice of the exact function depends on the application. For example, for temperature or pressure, it is most appropriate to average the values of the nodes in the grid cell. If ξ_{ij} is the grid cell value in the $(i, j)^{\text{th}}$ grid cell g_{ij} and v_n represents attribute value of node n in g_{ij} then ξ_{ij} is an aggregation function such as average, min, max of v_n :

$$\xi_{ij} = f(v_n) \forall n \in g_{ij} \quad (20)$$

We do not impose assumptions on the selection of γ and $f(\cdot)$, highlighting the generality of our proposed scheme (requirement on our scheme). An illustration for the selection of both parameters is given in the case study in Section 6.4.

Centralized Regioning Algorithm

As the events appears as regions in a map, we propose here a centralized regioning algorithm (CRA) that can detect the regions and their borders in WSN map, which leads to generic event detection. CRA is conceptually the same as its distributed counterpart DCF. However, it has been used here for entirely a different purpose of detecting events. The parallel between the two applications of conceptually same algorithm is that in DCF the *models* build correlated regions based on similarity of models and in CRA the *events* build regions based on similar values of attribute for an event.

The regions are formed because the attribute values fall into a certain class of values. For example, we normally classify the temperature as freezing, low, normal, high or very high. These classes also contain event class (range of values belonging to event, e.g., temperature above 500°C for fire). This gives us more acceptable abstraction than the exact values themselves. In the context of events, we do not care about the continuous values of physical parameters rather classes in which these values appear. It is necessary to maintain the exact values till prediction phase so that we can model these values to accurately predict the future values. However, the classification of the values becomes important when we consider events like fire or network partitioning. Therefore, thresholding of values into classes becomes logical representation for event detection. Thus, to detect these events we define the *class maps* that thresholds the exact values of the cells in grid map with their class denominations. If we define class map values K as $k_1, k_2, \dots$ for the range of the values of grid cell g_{ij} between $(\xi_2, \xi_1]$ and $(\xi_3, \xi_2] \dots$ respectively, then a class map value is defined by

$$K = \begin{cases} k_1 & \text{if } \xi_2 < \xi_{ij} \leq \xi_1 \\ k_2 & \text{if } \xi_3 < \xi_{ij} \leq \xi_2 \\ \dots & \end{cases} \quad (21)$$

CRA (Alg. 6.1) takes the grid map as input and determines border and regions belonging to different classes and hence events. We refer to the resultant output as the *regions map*. CRA essentially needs a class map to group all the same class cells and determine the boundary. The process of converting to class map and determining the regions boundary are both carried out concurrently. In order to merge the cells into regions, we define attribute classes as in Eq. (21). Neighboring cells are merged to form the same region if they belong to the same class. The definition of attribute

classes and fusion of same class grid cells makes CRA independent of the shape that a region takes or the number of regions (hence the number of events) in the map.

Algorithm 6.1 CRA: Centralized Regioning Algorithm (On the Sink)

```

1: Var: rB= regionBorder, mB= mapBorders, nRB= newRegionBorder,
   rM= regionsMap, rId= regionId, nL= neighborList, Gcxy= Grid cell
   at (x,y)
2: rM[][]=-1;
3: mB[][];
4: while rM(i, j) = -1 do
5:   rB[]=(Gcij  $\wedge$  rM(i, j) = -1)
6:   dilateRegion(map, rB[], rM[], rId)
7:   mB[rId][]=rB;
8:   rId++;
9: end while
10: Function dilateRegion(map, rB[], rM[], rId)
11: repeat
12:   changeInBorder=0;nRB[]=0;
13:   for Gcij  $\wedge$   $\forall$ Gcij  $\in$  rB do
14:     nL[] = 8Nbrs(Gcij);
15:     for Gckl  $\wedge$   $\forall$ Gckl  $\in$  nL[] do
16:       if (class(Gcij) = class(Gckl) & rM(k, l) = -1 then
17:         rM(k,l)=rId;
18:         nRB.add(Gckl);
19:         changeInBorder=1;
20:       end if
21:     end for
22:     if class(Gcij)  $\neq$  class(Gckl)  $\wedge$   $\forall$ Gckl  $\in$  nL[] then
23:       nRB.add(Gcij);
24:     end if
25:   end for
26:   rB[]=nRB[];
27: until changeInBorder

```

The algorithm starts by defining all the cells as *not assigned to a region* by initializing the variable $rM[][]=-1$. Next it searches a grid cell that has not been assigned a class yet and lists it as the border of the region, as the region itself and region border at this moment consists of a single cell (Alg. 6.1:L 5), cell with -1 in rM is not assigned a region yet). CRA then starts expanding/*dilating* the region (Alg. 6.1:L 6). To expand the

region, the neighboring eight cells around this region cell are checked if they already belong to any class (Alg. 6.1:L 14-16), if not then they are also classified according to Eq. (21). If they belong to the same region they are assigned the same region ID and the new qualifying cells are listed as the region border, otherwise the previous cells retain their status as region border (Alg. 6.1:L 17-18). To further expand the region neighboring cells of each cell in the border cells are searched iteratively until no change occurs in the border of the region (Alg. 6.1:L 11,27), which implies the completion of the construction of a single region with its boundary. The whole process repeats again by searching a new cell that has not been assigned a region yet. It keeps on repeating until all the cells in the map are classified into their corresponding regions (Alg. 6.1:L 4,9).

We maintain the generality of the proposed scheme by devising a technique that does not assume any shape, size or number of events occurring in the WSN.

6.4 CASE STUDY: NETWORK PARTITION PREDICTION

We need to formulate the problem according to the abstractions (maps, classes, etc.) of the proposed scheme, to use our scheme for network partition prediction.

6.4.1 Problem Formulation

Partition detection is a complex problem as physical and network parameters are coupled, i.e., energy level of the nodes and communication range necessary to maintain connectivity. Given that sensor nodes are resource constrained, eventually a WSN has to consider the depletion of node batteries leading to the partitioning of the network. The energy dissipation, however, is generally spatially correlated. Therefore, groups of nodes form hotspots that deplete to coverage holes. A hole can be defined as a part of the network, which due to the energy depletion is no longer covered. These holes, when grow, can disconnect a part of network from accessing the sink, defined as a partition. If the network energy state can be modeled and predicted, then we can predict the occurrence of the holes and consequently the partitions. The holes and partitions appear as regions in an

energy map. Our scheme has all the necessary tools to profile the energy dissipation patterns, predict the network future energy state and detect the regions formed due to partitioning. Therefore, partition prediction becomes a natural candidate problem to be solved using our scheme.

We can now define the problem according to the abstraction of our proposed scheme. A grid cell (cell in a grid map) gets disconnected from the network if it has energy below a minimum threshold so that it cannot communicate anymore. These depleted grid cells form a region that represents a hole in an eMap. Partition, however, is a group of non-depleted grid cells that cannot access sink due to the holes. It is therefore sufficient to profile the energy status of the network during its lifetime by collecting the energy profiles in order to predict network partitioning. As per definition the adaptation of the scheme to predict network partitioning consists of three phases (Section 6.3) that we discuss as follows.

6.4.2 The Data Collection Phase

The nodes start the formation of 1-hops clusters as in [C. Wang et al., 2012]. During the 1-hop cluster formation cluster heads are selected, the cluster heads learn about their neighboring clusters. All cluster heads start aggregating the energy values. The models for regions are aggregated and periodically sent to the sink.

6.4.3 The Prediction Phase

The sink regenerates the time series (energy values of cluster and hence the nodes) by applying reverse transformation. The data regeneration of the reporting nodes actually generates the energy profiles. The energy profiles of each node are modeled and predicted as described in Section 6.3.2. Energy dissipation is a decaying process so the time series contains trends but no seasonal components. The trends are removed by fitting polynomials. ARMA models are fitted to random components, selecting the best fit model using AIC criteria. After completion of modeling the node energy values are predicted and hence the future WSN energy profiles.

6.4.4 The Event (Holes and Partition) Detection Phase

The first step towards the abstraction of the WSN profile as a grid map (eMap in this case) is the selection of resolution, i.e., grid cell size at which this event (network partitioning or holes) is to be detected. From the formulation of the problem we know that we have two coupled parameters, i.e., energy and communication range. Therefore, an upper bound for γ is the communication range (R). To accommodate a worst case scenario of two nodes lying on opposite corners of two grid cells, γ is given by $\gamma < R/2\sqrt{2}$, as shown in Fig. 6.2. The lower bound can be obtained from the node density, it should be selected such that the network area is not over sampled, as we show in simulation Section 6.4.5. A cell is connected to the neighboring cells until at least a single node has enough energy to communicate. The node having the highest energy level is selected as reporting node and Eq. (20) becomes $\xi_{ij} = \max(v_n) \forall n \in g_{ij}$.

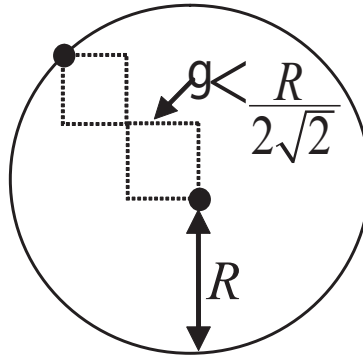


Figure 6.2: Max grid size

The predicted energy profiles are converted to the eMaps. CRA developed in Section 6.3.3 is used for both partition and hole detection on these eMaps. As per the given scheme we define two energy classes at 10% and below as the partition (or hole) class, and above 10% as non-partition class. This definition of energy classes gives the areas that are vulnerable to partitioning because of low energy.

6.4.5 Simulation Settings

As two phases of the proposed scheme are carried out on the sink, therefore we performed our simulations in Matlab. It is a very well-known simulation tool and suits our work as it facilitates to model energy dissipation

patterns of very huge number of nodes. The network that we used in our simulations is generated as a random non-uniform distribution of nodes. For energy dissipation modeling the common hotspot model [Yonggang Jerry Zhao, 2002] was used. The energy dissipates in a spatially correlated manner around the hotspot. The nodes nearest to the hotspot are more active and hence dissipate more energy. The parts of the network that act as the coverage-bridge between two parts of the network and around the sink show relatively high energy dissipation rates. Subsequently these areas are modeled as hotspots.

We used a network containing 5000 sensor nodes that span in an area of 50×100 unit², each node having a communication range $R = 2$ units. For $R = 2$ the upper bound for grid cell size is 0.7 units. We found 0.3 as the lower bound because if we take a grid size smaller than 0.3 then we have more occupied grid cells than the number of nodes that over samples the network area. We therefore selected three grid sizes between upper and lower bounds 0.3, 0.5 and 0.7 units. Energy dissipation history of 164 profiles was collected from all the nodes. To evaluate the statistics we divided the history of profiles into two parts. 139 profiles were used for modeling purposes and 25 used for validation. 164 profiles represent the network lifetime history. If we scale 164 lifetime profiles to 164 days then 139 days of network operation are used to predict the next 25 days network status. 139 profiles of WSN were used to predict next 25 profiles. Each predicted profiles of the network was transformed to grid map.

6.4.6 Simulation Results

Fig. 6.3 shows the mean square sum of error for 25 prediction steps for 3 different grid sizes. The low values of mean square error show that the predictions are very accurate. The increasing trend is natural, as an increasing number of prediction steps makes the prediction model less accurate.

Fig. 6.4 shows the misclassified cells count. The mean square error results (Fig. 6.3) imply that we cannot expect much inaccuracy in misclassification. The highest count is naturally in the case of grid size 0.3, which reaches 88 at the peak. The total number of occupied grid cells at this resolution is 4196, so a worst case misclassification of 88 cells accounts to around 2% of the total cells. We also see an increasing trend in the misclassification for each prediction step because of the increasing error between model approximation and the actual data. The oscillations in the graph give interesting insight. We have defined two classes of en-

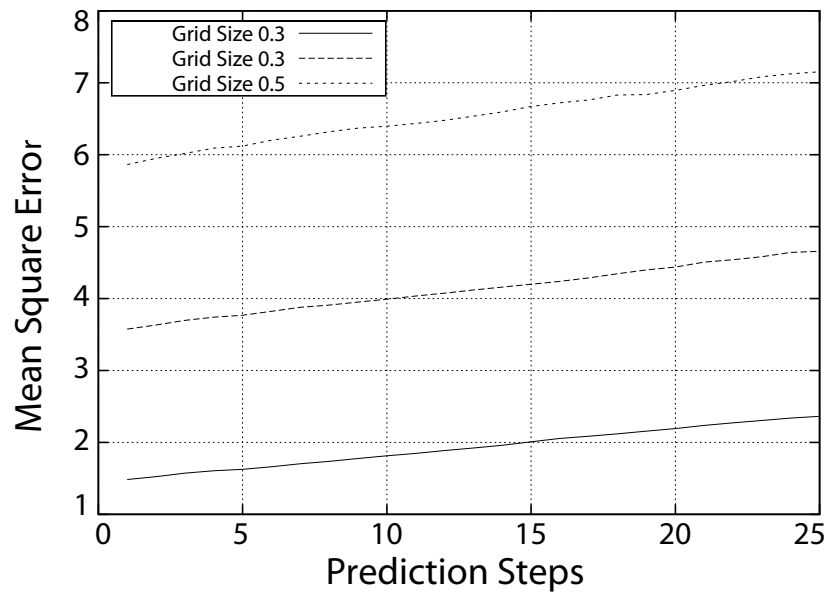


Figure 6.3: Mean square error accuracy measures

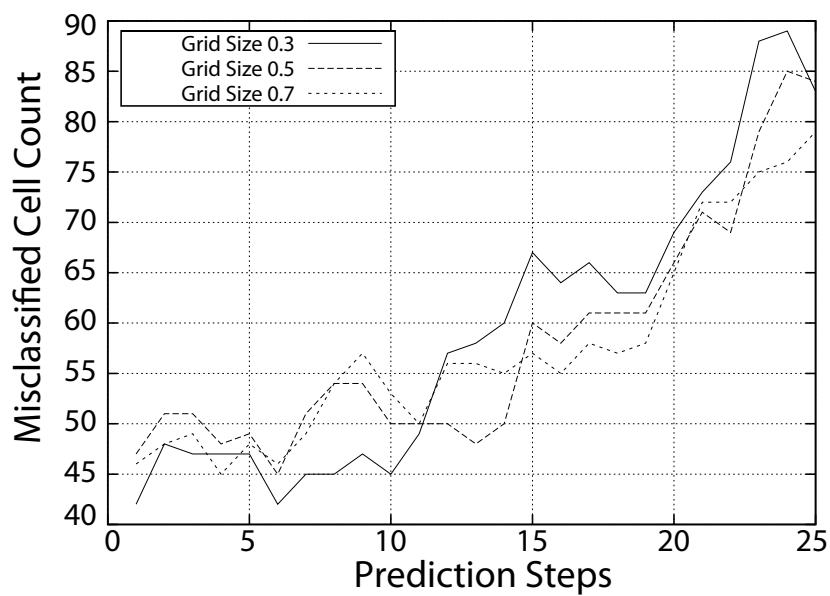


Figure 6.4: Misclassification cell count

ergy and as soon as the grid cells cross the class threshold (10% of energy) they are classified into the partitioned class. The crests appear when cells in the actual data (K_r) cross the threshold of 10% but the cells from the modeled data (K_t), due to the lag in value, do not cross the threshold at the same time. Therefore, cells from the reference class are classified in the partitioned class but corresponding cells in the test class are still in the non-partitioned class, which increases the count of difference cells. As soon as the modeled data crosses the threshold the error decreases and troughs appear but a clear trend in increase of error continues.

Fig. 6.5 gives account of the error in the detected regions predicted through profiles of the WSN. To summarize the results we have selected prediction maps separated by five prediction steps. On first prediction step there are only three regions with less than 2% max percentile error. With each next prediction step the number of regions increases and errors distribute between the different regions. In the worst case scenario a region has a maximum percentile error of less than 2.7%. The results however show that each region is very accurately detected. Misclassification per region on the average is less than 3% which shows the accuracy of our approach to detect the regions and their boundaries.

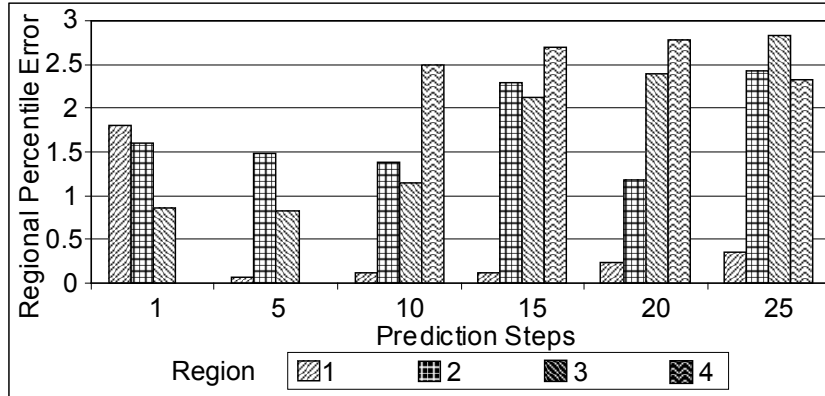


Figure 6.5: Misclassification percentile error per region

Now, we summarize the results w.r.t. efficiency metric, i.e., number of packets needed to create energy profile of the whole network. To profile the whole network of 5000 nodes and to collect 164 profiles over the entire lifetime requires nearly 1 million data points. This overhead is reduced dramatically by using the spatio-temporal compression. Models are constructed for clusters and regions. We fix the length of history that can

be represented by a model. We chose 8, 14 and 20 values to be modeled by a single model. The graphs in Fig. 6.6 and 6.7 represent cumulative sum of region formed and outliers respectively that we require over the lifetime of network. To observe the impact of maximum allowed outliers for a model, we chose values from 0 to 4 to allow the clusters to form regions. As a model does not approximate 100% accurately all the data, therefore in addition to outliers per model, we also allow outliers for each cluster head to build its model and fit to its data. Therefore, we will observe outliers even when we set outliers per model to zero. From Fig. 6.6 we can conclude that the number of regions being formed decreases as we allow more outliers. Allowing more outliers relaxes the neighboring clusters to fit to a given model. Similarly from Fig. 6.7 the number of outliers to be reported naturally increases also. The length of data to be represented by a model also affects the regions to be formed. The shorter length (8) forms more cumulative regions over the lifetime as it has to report more often than the longer length models that manage to fit more data within one model. However, shorter length model has to report very few outliers first, but shoots immediately as shorter length has more regions. Longest length models (20) have least regions initially again because it reports the complete data in less cycles of sending the models but increases to maximum as its very hard for the neighboring clusters to agree for a very long length of attribute. These long length models also have the maximum number of outliers to be reported even outliers per model are zero. Therefore, maximum number of regions are formed but decreases afterwards as it has to make less updates. The data length to be modeled is a very important factor in this whole compression scheme. We found from our analysis that there is a compromise between the two extremes, which in our case was around 14. At this value the number of regions formed and the number of outliers to be reported is balanced between the two extremes.

Each region in Fig. 6.6 is equivalent to sending a message to the sink as a region is represented by this model. Three outliers are grouped in one packet, the number of packets for outliers is equivalent to 1/3rd of the outliers. With the settings of modeled data length=14 and $\Delta Nc_i=1$, we have to send 40499 packets, which is 4.93% of the total raw data to be reported otherwise.

The results obtained in the evaluation are in accordance with the design requirements of the scheme. The achieved predictions are long-term and accurate, represented by the maximum prediction error of approximately

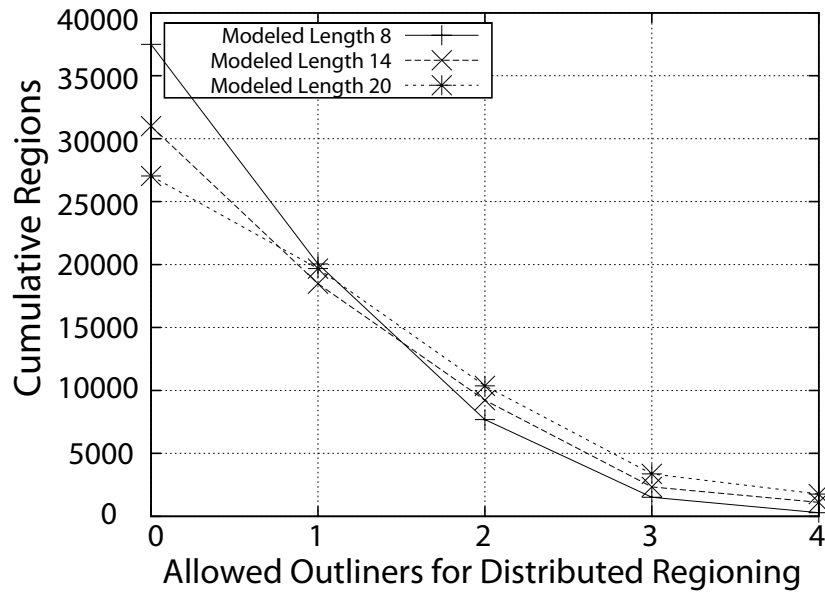


Figure 6.6: Cumulative Cluster during Data Collection

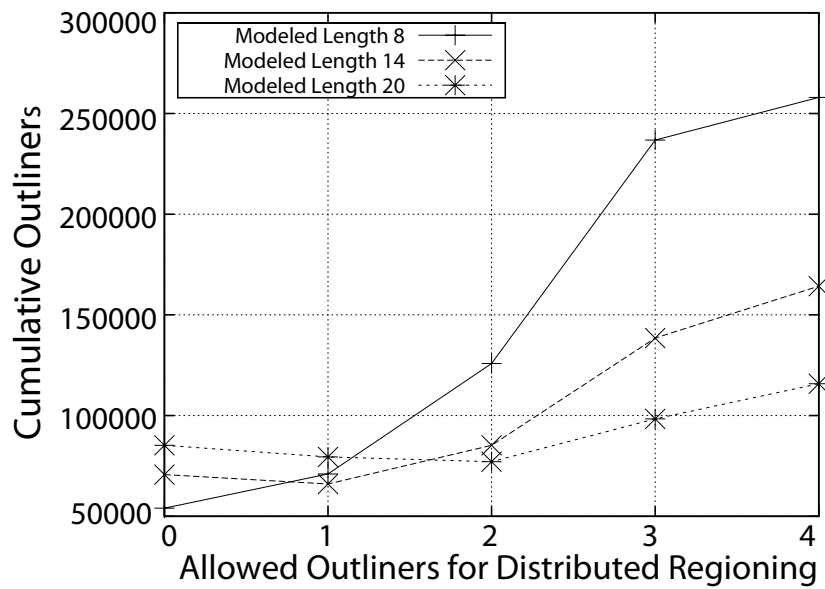


Figure 6.7: Cumulative Outliers for DCF

3% in misclassification of the regions of the map for 25 prediction steps. The proposed scheme detects energy holes that may partition the network in 22nd days (in scaled time as explained in Section 6.4.5). From Fig. 6.5, we conclude that the partition prediction is more than 97% reliable (because of 97% region accuracy).

6.5 CHAPTER SUMMARY

In this chapter, we have presented a generic scheme for event forecasting in WSN. We presented an abstraction mechanism for the sampled attribute in the form of maps. Periodically collected attribute values are used to construct the maps, which in turn are used to forecast the future maps. The future maps represent the state of the network at a later point of time in the network. We also proposed a generalized mechanism to detect events using the maps. We demonstrated the proposed scheme using the example of network partitioning.

7

CONCLUSIONS AND FUTURE RESEARCH

In this thesis we have investigated long term data compression and gathering schemes, specially time-series based compression schemes. The existing compression schemes are suitable for some application scenarios, however, they rapidly start to deteriorate in the presence of noise. In this thesis we develop a framework for fault-tolerant compression scheme to be employed in the noisy environments that explicitly exploits the long term monitoring nature of WSN.

This chapter concludes the thesis by presenting the main contributions and then we discuss the extensibility of our proposed work. We discuss the possible extensions of our work to compression schemes beyond the time series based compression schemes and possibility of extending the scheme to simultaneous multi-variate fault-tolerant compression. We are very positive that the research work presented in thesis opens up new and interesting research directions.

7.1 OVERALL THESIS CONTRIBUTIONS

The main objective of this thesis was to study long term data gathering and compression schemes that are resilient to underlying faults that corrupt the sampled data. Our research work was driven by the current need for an efficient solution for long term data gathering that may also be resilient to the faults. Accordingly, here we discuss the key contributions made by the research presented in this thesis. Driven by the research problem introduced in Sec. 1 and grouped by topic, the thesis contributions are surveyed and their relevance is discussed.

7.1.1 Delay-Tolerant Spatio-Temporal Data Compression

We have developed Adaptive Hybrid Compression (AHC) , a fully distributed spatio-temporal data compression technique for accurate continuous sensor data collection in WSN. AHC dynamically self-adapts to approximate the monitored attribute both in space and time. In order to achieve the adaptability, AHC proposes (a) automated mechanisms to determine optimal models to best approximate the observed attribute, (b) dynamic hierarchical clustering based on models, and (c) automated mechanisms to adjust the compression scopes both in time and space. The use of 'simple models batches' instead of complex monolithic models was the key idea to design a technique that adapts to the dynamic changes of the monitored attribute. This key design choice has allowed for the first time to delay data collection rounds as tolerated by the application, thus, maximizing compression ratio and significantly reducing the message cost. In our experiments, we were able to reconstruct the signal from spatio-temporally compressed data on the sink with granularity of a single node and mean error less than 0.04%. In order to put the performance of AHC in perspective, we compare it to the relevant state of the art work PAQ, ASTC and DTC. The simulation results demonstrate that AHC easily outperforms the rest for delay-tolerant applications that it specifically targets.

7.1.2 Fault-Tolerant Data Compression

We proposed Fault-Tolerant Data Compression (FTDC), a fully distributed, fault-agnostic data error detection and correction scheme. FTDC exploits the existing logical infrastructure already in place, created and used by

the compression scheme, to implement proposed fault-tolerance mechanisms. For example, it uses the data models, created by compression scheme, to identify the sampling errors and uses 1-hop cluster infrastructure to further filter out noisy samples from samples resulting from physical events. Due to reuse of the resources, the proposed scheme is very lightweight in terms of memory, computation and message cost. In spite of the ability to exploit existing resources, FTDC is still flexible to be easily posted to various time-series based compression schemes. This has also been demonstrated in chapter 5 through simulations. The simulation results clearly demonstrated that established time-series based compression schemes, when enhanced by FTDE, are able to isolate erroneous data and maintain high approximation accuracy relative to the non-enhanced schemes. In addition, FTDE actually reduced the message overhead since it allows to suppress the transmission of erroneous data. We demonstrated through the simulations that FTDE can adapt to different time-series based schemes and accordingly further improves its own performance by exploiting the specific enhancement of a given scheme.

7.1.3 Efficient Predictive Monitoring

We developed a generalized framework for efficient predictive monitoring to forecast events in order to support an autonomic self* system for WSN. We presented an abstraction mechanism for the sampled attribute in the form of maps. Periodically collected attribute values are used to construct the maps, which in turn are used to forecast the future maps. The future maps represent the state of the network at a later point of time in the network. We also proposed a generalized mechanism to detect events using the maps. We demonstrated that it can be effectively used to predict events related to different attributes. We described it as a three phase strategy. In the data collection phase we proposed efficient algorithms to spatio-temporally compress the attribute values and transport them to the sink. In the prediction phase, we long-term predicted the attribute states. In the event detection phase, we proposed a generalized event detection algorithm. We demonstrated the feasibility and validity of approach by predicting the network partitioning as a case study. We were able to predict multiple holes and the resulting partitioned area of the network; information necessary to initiate proactive self* actions. Simulations support the practicality of our approach by showing its high accuracy and low monitoring overhead on the network. In order to further increase the efficiency,

we propose to adapt the spatio-temporal data compression to the occurrence probability of events. For instance sensor nodes should increase data model accuracy if they are located in areas where events frequently occur or when an event is suspected. We demonstrated the proposed scheme using the example of network partitioning.

7.2 FUTURE WORK

AHC is our first step to efficiently transport large volumes of wireless sensing data with accuracy guarantees in extremely computation/energy constrained devices and networks. Despite this effort, additional work to further extend the network lifetime is needed. The proposed scheme provide various optimizations in order to increase the energy efficiency. Accordingly, it provides varied dynamic adaption mechanisms to achieve that goal. However, the current work seriously lacks on the dynamic adaptability of the roles of the nodes and does not explicitly perform any load balancing. For example, once a certain node adapts a certain role, e.g., cluster head, it stays in that role, unless a change in the phenomenon forces the nodes to rearrange to better model the change in the phenomenon. Hence, the current work can be extended to incorporate the concept of load balancing so that no a specific node or set of nodes are adversely affected over a duration range of time. Studying the repercussions of such a load balancing scheme is important and it should be made sure that it does not have detrimental effects on the performance of the performance of the scheme. For instance integrating a duty cycling technique or intelligent selection of subset of nodes while maintaining the spatial resolution to save energy without sacrificing the required data accuracy.

One major weakness of AHC is that, currently, it cannot handle multi-variate data. Theoretically, AHC is not limited to just single variable. However, it still is a challenging problem to extend it to a multi-variate situation. A multi-variate extended version of the scheme will be able to simultaneously compression and transport multiple physical attributes being gathered by the sensor nodes. Hence, such an extension can further reduce the data transportation costs.

The proposed fault-tolerant scheme is very well suited to our proposed compression scheme and is flexible enough to be ported to other time-series based compression schemes. It can also be further extended beyond the domain of data compression to more generic time-series based WSN

operations such as data aggregation and data prediction with the aim of designing a generalized data fault tolerance scheme. It would also be very interesting and challenging to extend to schemes beyond time-series based schemes and investigate how well does it extend to compression scheme other than time-series. The proposed fault-tolerance scheme also does not cope well against *drifting* sensor values, i.e., the sampled data that drift over time away from the actual values. Such error arise specially when the physical characteristics of the sensor change, e.g., if a sensor is being heated due to sun light. FTDC can identify can easily identify and isolate a sensor node if such an error occurs in a single sensor node. However, if a group of nodes shows such behavior, FTDC will not be able to detect such errors. Extending the scheme to also identify such errors will make the proposed scheme even more robust.

Similarly to AHC, it would be really interesting to study and investigate how fault-tolerance in FTDC can be further increased by extending it to a multi-variate fault-tolerance problem. Often sampled values from different sensors vary in tandem, e.g., light and temperature both tend to increase if exposed to sun light. Because different sensors have different physical characteristics, thresholds and saturation values, when investigated together as a multi-variate problem they can be used to identify the sensors with drifting values.

BIBLIOGRAPHY

Abbasi, Ameer Ahmed and Mohamed Younis

- 2007 "A survey on clustering algorithms for wireless sensor networks," *Computer Communication*, vol. 30 (14-15 Oct. 2007), pp. 2826-2841, ISSN: 0140-3664.

Abello, J., P. Pardalos, and M. G. C. Resende

- 1999 "On maximum clique problems in very large graphs," *External memory algorithms*, pp. 119-130.

Ahmed, Nasir, T Natarajan, and Kamisetty R Rao

- 1974 "Discrete cosine transform," *IEEE Trans. on Computers*, vol. 100, 1, pp. 90-93.

Akyildiz, Ian F., Dario Pompili, and Tommaso Melodia

- 2005 "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, vol. 3 (May 2005), pp. 257-279.

Ali, Azad, Abdelmajid Khelil, Faisal Karim Shaikh, and Neeraj Suri

- 2009 "MPM: Map based Predictive Monitoring for Wireless Sensor Networks," in *3rd Int. Conf. on Autonomic Computing and Comm. Sys.* Pp. 79-95.

Ali, Azad, Abdelmajid Khelil, and Neeraj Suri

- 2015 "FTDE: Distributed Fault Tolerance for WSN Data Collection and Compression Schemes," in *34th IEEE Symposium on Reliable Distributed Systems, SRDS 2015, Montreal, QC, Canada, September 28 - October 1, 2015*, pp. 140-145.

Ali, Azad, Abdelmajid Khelil, Neeraj Suri, and Mohammadreza Mahmudi-manesh

- 2015 "Adaptive Hybrid Compression for Wireless Sensor Networks," *ACM Transaction on Sensor Networks*, vol. 11, 4 (May 2015), pp. 1-36, ISSN: 1550-4859.

- Ali, Azad, Abdelmajid Khelil, Piotr Szczytowski, and Neeraj Suri
- 2011a "An adaptive and composite spatio-temporal data compression approach for wireless sensor networks," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, Miami, Florida, USA, pp. 67-76, ISBN: 978-1-4503-0898-4.
 - 2011b "An adaptive and composite spatio-temporal data compression approach for wireless sensor networks," in *Proceedings of the 14th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, MSWiM*, pp. 67-76.
- Aurenhammer, Franz
- 1991 "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure," *ACM Comput. Surv.*, vol. 23, 3, pp. 345-405.
- Baek, Seung Jun, Gustavo de Veciana, and Xun Su
- 2004 "Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 1130-1140.
- Balzano, Laura and Robert Nowak
- 2007 "Blind Calibration of Sensor Networks," in *Proc. of ACM/IEEE IPSN'07*, pp. 79-88.
- Banerjee, Torsha, Bin Xie, and Dharma P. Agrawal
- 2008 "Fault tolerant multiple event detection in a wireless sensor network," *J. Parallel Distrib. Comput.*, vol. 68, 9, pp. 1222-1234.
- Barooah, Prabir et al.
- 2012 "Cut Detection in Wireless Sensor Networks," *IEEE Trans. on Parallel and Dist. Sys.*, vol. 23, 3, pp. 483-490.
- Basu, S. and M. Meckesheimer
- 2007 "Automatic Outlier Detection for Time Series: An Application to Sensor Data," *Knowledge and Information Systems Journal*, vol. 11, 2, pp. 137-154.

Blaß, Erik-Oliver, Joachim Wilke, and Martina Zitterbart

- 2008 "Relaxed authenticity for data aggregation in wireless sensor networks," in *Proceedings of the 4th international conference on Security and privacy in communication networks*, SecureComm '08, Istanbul, Turkey, 4:1-4:10, ISBN: 978-1-60558-241-2.

Borgne, Yann-Aël Le, Silvia Santini, and Gianluca Bontempi

- 2007 "Adaptive model selection for time series prediction in wireless sensor networks," *Signal Process.*, vol. 87, 12, pp. 3010-3020, ISSN: 0165-1684.

Branch, Joel W. et al.

- 2013 "In-network outlier detection in wireless sensor networks," *Knowledge and Information Systems Journal*, vol. 34, 1, pp. 23-54.

Brockwell, Peter J. and Richard A. Davis

- 2002 *Introduction to Time Series and Forecasting*, 2nd, Springer, ISBN: 0387953515.

Bychkovskiy et al.

- 2003 "A Collaborative Approach to In-place Sensor Calibration," in *Proc. of ACM/IEEE IPSN'03*, pp. 301-316.

Chu, David, Amol Deshpande, Joseph M. Hellerstein, and Wei Hong

- 2006 "Approximate Data Collection in Sensor Networks using Probabilistic Models," in *Proc. of the 22nd Int. Conf. on Data Engineering*, pp. 48-, ISBN: 0-7695-2570-9.

Cinque, Marcello et al.

- 2013 "A survey on resiliency assessment techniques for wireless sensor networks," in *Proc. of MOBIWAC'13*, pp. 73-80.

Clouqueur, T., K.K. Saluja, and P. Ramanathan

- 2004 "Fault tolerance in collaborative sensor networks for target detection," *Computers, IEEE Transactions on*, vol. 53, 3 (Mar. 2004), pp. 320-333, ISSN: 0018-9340.

Coronato, Antonio and Alessandro Testa

- 2013 "Approaches of Wireless Sensor Network Dependability Assessment," in *Proc. of IEEE FedCSIS*, pp. 881-888.

De Souza, Luciana Moreira Sá et al.

- 2007 *A survey on fault tolerance in wireless sensor networks*.

- Deshpande, Amol, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong
 2004 "Model-driven data acquisition in sensor networks," in vol. 30, pp. 588-599.
- Dini, Gianluca et al.
 2008 "An Algorithm for Reconnecting Wireless Sensor Network Partitions," in *Proc. of EWSN'08*, pp. 253-267.
- Duarte, Marco F., Godwin Shen, Antonio Ortega, and Richard G. Baraniuk
 2012 "Signal compression in wireless sensor networks," *Phil. Tran. of the Royal Society A: Mathematical, Phy. and Eng. Sciences*, vol. 370, 1958, pp. 118-135.
- Dunkels, A., B. Gronvall, and T. Voigt
 2004 "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks*.
- Faizulkhakov, Ya. R.
 2007 "Time synchronization methods for wireless sensor networks: A survey," *Programming and Computing Software*, vol. 33 (4 July 2007), pp. 214-226, ISSN: 0361-7688.
- Feng, J. et al.
 2003 "Model-based calibration for sensor networks," in *Proc. of IEEE Sensors'03*, vol. 2, pp. 737-742.
- Gedik, Bugra, Ling Liu, and Philip S. Yu
 2007 "ASAP: An Adaptive Sampling Approach to Data Collection in Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, 12, pp. 1766-1783.
- Gehrke, Johannes and Samuel Madden
 2004 "Query Processing in Sensor Networks," *IEEE Perv. Comp.*, vol. 3, pp. 46-55, ISSN: 1536-1268.
- Guo, Shuo, Heng Zhang, Ziguo Zhong, Jiming Chen, Qing Cao, and Tian He
 2014 "Detecting Faulty Nodes with Data Errors for Wireless Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 10, 3 (May 2014), 40:1-40:27, ISSN: 1550-4859.

Guo, Shuo, Ziguang Zhong, and Tian He

- 2009 "FIND: Faulty Node Detection for Wireless Sensor Networks," in *Proceedings of ACM Conference on Embedded Networked Sensor Systems, SenSys '09*, ACM, Berkeley, California, pp. 253-266, ISBN: 978-1-60558-519-2.

Gupta, Himanshu, Vishnu Navda, Samir Das, and Vishal Chowdhary

- 2008 "Efficient gathering of correlated data in sensor networks," *ACM Transaction on Sensor Networks*, vol. 4 (1 Feb. 2008), 4:1-4:31, ISSN: 1550-4859.

Haas, Christian and Joachim Wilke

- 2011 "Energy evaluations in wireless sensor networks: a reality check," in *Proc. of the 14th ACM int. conf. on Modeling, analysis and simulation of wireless and mobile systems*, ACM, New York, NY, USA, pp. 27-30.

Hasenfratz, David others

- 2012 "On-the-Fly Calibration of Low-cost Gas Sensors," in *Proc. of EWSN'12*, pp. 228-244.

He, Tian, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek F. Abdelzaher

- 2005 "Range-free localization and its impact on large scale sensor networks," *Trans. on Embedded Computing Sys.*, vol. 4, 4, pp. 877-906.

Hempstead, Mark et al.

- 2008 "Survey of Hardware Systems for Wireless Sensor Networks," *Journal of Low Power Electronics*, vol. 4, 1, pp. 11-20.

Jiang, Hongbo, Shudong Jin, and Chonggang Wang

- 2011 "Prediction or Not? An Energy-Efficient Framework for Clustering-Based Data Collection in Wireless Sensor Networks," *IEEE Tran. on Parallel and Dist. Sys.*, vol. 22 (6 June 2011), pp. 1064-1071, ISSN: 1045-9219.

Jiang, Peng and Sheng-Qiang Li

- 2010 "A Data Compression Algorithm for Wireless Sensor Networks Based on an Optimal Order Estimation Model and Distributed Coding," *Sensors*, vol. 10, 10, pp. 9065-9083, ISSN: 1424-8220.

Jindal, Apoorva and Konstantinos Psounis

- 2006 "Modeling spatially correlated data in sensor networks," *ACM Transaction on Sensor Networks*, vol. 2 (4 Nov. 2006), pp. 466-499, ISSN: 1550-4859.

Kamal, Abu Raihan M., Chris Bleakley, and Simon Dobson

- 2013 "Packet-level Attestation (PLA): A Framework for In-network Sensor Data Reliability," *ACM Trans. Sen. Netw.*, vol. 9, 2 (Apr. 2013), 19:1-19:28, ISSN: 1550-4859.

Khan, MMH et al.

- 2014 "Troubleshooting interactive complexity bugs in wireless sensor networks using data mining techniques," *ACM Trans. on Sensor Networks*, vol. 10, 2.

Landsiedel, O., K. Wehrle, and S. Gotz

- 2005 "Accurate prediction of power consumption in sensor networks," in *IEEE Workshop on Embedded Networked Sensors*, pp. 37-44.

Levis, Philip, Sam Madden, Joseph Polastre, Robert Szewczyk, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler

- 2004 "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*.

Li, Guorui and Ying Wang

- 2013 "Automatic ARIMA modeling-based data aggregation scheme in wireless sensor networks." *EURASIP J. Wireless Comm. and Networking*, vol. 2013, p. 85.

Liu, Chong, Kui Wu, and Jian Pei

- 2007 "An Energy-Efficient Data Collection Framework for Wireless Sensor Networks by Exploiting Spatiotemporal Correlation," *IEEE Tran. on Parallel and Dist. Systems*, vol. 18 (7 July 2007), pp. 1010-1023, ISSN: 1045-9219.

Liu, Hai, Amiya Nayak, and Ivan Stojmenovic

- 2009 "Fault-Tolerant Algorithms/Protocols in Wireless Sensor Networks," English, in *Guide to Wireless Sensor Networks*, pp. 261-291, ISBN: 978-1-84882-217-7.

Liu, Kebin et al.

- 2011 "Self-diagnosis for large scale wireless sensor networks," in *IEEE INFOCOM'11*, pp. 1539-1547.

Liu, Yunhao, Kebin Liu, and Mo Li

- 2010 "Passive Diagnosis for Wireless Sensor Networks," *IEEE/ACM Trans. on Networking*, vol. 18, 4 (Aug. 2010), pp. 1132-1144, ISSN: 1063-6692.

Ljung, Lennart

- 1998 *System Identification: Theory for the User (2nd Edition)*, ISBN: 0136566952.

Luo, Chong et al.

- 2009 "Compressive data gathering for large-scale wireless sensor networks," in *Proc. of Mobicom'09*, pp. 145-156.

Madden, S.

- 2003 *Intel lab data*.

Madden, Samuel R., Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong

- 2005 "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transaction on Database Systems*, vol. 30 (1 Mar. 2005), pp. 122-173, ISSN: 0362-5915.

Mahapatro, A. and P.M. Khilar

- 2013 "Fault Diagnosis in Wireless Sensor Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, 4, pp. 2000-2026.

Mahmudimanesh, Mohammadreza, Abdelmajid Khelil, and Neeraj Suri

- 2010 "Reordering for Better Compressibility: Efficient Spatial Sampling in Wireless Sensor Networks," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pp. 50-57, ISBN: 978-0-7695-4049-8.

Mamei, Marco and Radhika Nagpal

- 2007 "Macro Programming through Bayesian Networks: Distributed Inference and Anomaly Detection," in *PERCOM '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications*, pp. 87-96.

Miluzzo, Emiliano et al.

- 2008 "CaliBree: A Self-calibration System for Mobile Sensor Networks," in *Proc. of IEEE DCOSS'08*, pp. 314-331.

Min, Jun-Ki and Chin-Wan Chung

- 2010 "EDGES: Efficient data gathering in sensor networks using temporal and spatial correlations," *Journal of Systems and Software*, vol. 83 (2 Feb. 2010), pp. 271-282, ISSN: 0164-1212.

Mini, Raquel A. F., Badri Nath, and Antonio A. F. Loureiro

- 2002 "A probabilistic approach to predict the energy consumption in wireless sensor networks," in *In IV Workshop de Comunicacao sem Fio e Computao Mvel. So Paulo*, pp. 23-25.

Mini, Raquel A. F., Max do Val Machado, Antonio A. F. Loureiro, and Badri Nath

- 2005 "Prediction-based energy map for wireless sensor networks," *Ad Hoc Netw.*, vol. 3 (2 Mar. 2005), pp. 235-253, ISSN: 1570-8705.

Miranda, K., V.M. Ramos R, and T. Razafindralambo

- 2013 "Using efficiently autoregressive estimation in wireless sensor networks," in *CITS*, pp. 1-5.

Montgomery, D. C., C. L. Jennings, and M. Kulahci

- 2008 *Introduction to Time Series Analysis and Forecasting*, Wiley.

Nakamura, Eduardo F., Antonio A. F. Loureiro, and Alejandro C. Frery

- 2007 "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Computer Surveys*, vol. 39 (3 Sept. 2007), ISSN: 0360-0300.

Ni, Kevin, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava

- 2009 "Sensor Network Data Fault Types," *ACM Trans. Sen. Netw.*, vol. 5, 3 (June 2009), 25:1-25:29, ISSN: 1550-4859.

Pham, Ngoc Duy, Trong Duc Le, KwangJin Park, and Hyunseung Choo

- 2010 "SCCS: Spatiotemporal clustering and compressing schemes for efficient data collection applications in WSNs," *Int. Journal of Comm. Sys.*, vol. 23, 11, pp. 1311-1333.

Polastre, Joseph, Robert Szewczyk, and David Culler

- 2005 "Telos: enabling ultra-low power wireless research," in *Proc. of the 4th int. symposium on Information processing in sensor networks, IPSN '05*, IEEE Press, Los Angeles, California, ISBN: 0-7803-9202-7.

Ramanathan, N. et al.

- 2006 "Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks," in *Technical Report CENS-TR-62, Center for Embedded Networked Sensing*.

Ramanathan, Nithya et al.

- 2005 "Sympathy for the Sensor Network Debugger," in *Proc. of ACM SenSys'05*, pp. 255-267.

Razzaque, M. A. et al.

- 2013 "Compression in Wireless Sensor Networks: A Survey and Comparative Evaluation," *ACM Trans. on Sensor Networks*, vol. 10, 1, 5:1-5:44.

Ringwald, Matthias et al.

- 2007 "Demo abstract: Passive inspection of deployed sensor networks with SNIF," in *Adjunct Proc. of EWSN'07*, pp. 45-46.

Rost, S. and H. Balakrishnan

- 2006a "Memento: A Health Monitoring System for Wireless Sensor Networks," in *Sensor and Ad Hoc Communications and Networks, 2006. Third International Conference on*, vol. 2, pp. 575-584.
- 2006b "Memento: A Health Monitoring System for Wireless Sensor Networks," in *Proc. of IEEE SECON'06*, vol. 2, pp. 575-584.

Shaikh, Faisal Karim, Abdelmajid Khelil, Brahim Ayari, Piotr Szczytowski, and Neeraj Suri

- 2010 "Generic Information Transport for Wireless Sensor Networks," in *Proceedings of the 2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, SUTC '10*, pp. 27-34, ISBN: 978-0-7695-4049-8.

Shaikh, Faisal Karim, Abdelmajid Khelil, and Neeraj Suri

- 2009 "AReIT: Adaptable Reliable Information Transport for Service Availability in Wireless Sensor Networks," in *International Conference on Wireless Networks*, pp. 75-81.

Sharma, Abhishek B. et al.

- 2010 "Sensor Faults: Detection Methods and Prevalence in Real-world Datasets," *ACM Trans. on Sensor Networks*, vol. 6, 3 (June 2010), 23:1-23:39, ISSN: 1550-4859.

- Sharma, Abhishek B., Leana Golubchik, and Ramesh Govindan
 2007 "Sensor Faults: Detection Methods and Prevalence in Real-world Datasets," *ACM Trans. Sen. Netw.*, vol. 6, 3 (June 2007), 23:1-23:99.
- Shih, K.-P., H.-C. Chen, J.-K. Tsai, and C.-C. Li
 2007 "PALM: A Partition Avoidance Lazy Movement Protocol for Mobile Sensor Networks," in *Proc. of the IEEE Wireless Communications and Networking Conference*, pp. 2484-2489.
- Shih, Kuei-Ping et al.
 2007 "PALM: A Partition Avoidance Lazy Movement Protocol for Mobile Sensor Networks," in *Proc. of IEEE WCNC'07*, pp. 2484-2489.
- Shrivastava, Nisheeth, Subhash Suri, and Csaba D. Tóth
 2005 "Detecting cuts in sensor networks," in *Proc. of the 4th int. symposium on Info: processing in sensor networks*, pp. 210-217.
- Solis, Ignacio and Katia Obraczka
 2009 "Isolines: efficient spatio-temporal data aggregation in sensor networks," *Wireless Communications and Mobile Computing*, vol. 9 (3 Mar. 2009), pp. 357-367, ISSN: 1530-8669.
- Szewczyk, Robert, Joseph Polastre, Alan M. Mainwaring, and David E. Culler
 2004 "Lessons from a Sensor Network Expedition," in *Wireless Sensor Networks, First European Workshop, EWSN 2004, Berlin, Germany, January 19-21, 2004, Proceedings*, pp. 307-322.
- Tobagi, F. and L. Kleinrock
 1975 "Packet Switching in Radio Channels: Part II—The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution," *Communications, IEEE Transactions on [legacy, pre - 1988]*, vol. 23, 12, pp. 1417-1433.
- Tolle, Gilman, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong
 2005 "A macroscope in the redwoods," in *Proceedings of the 3rd international conference on Embedded networked sensor systems, SenSys '05, San Diego, California, USA*, pp. 51-63, ISBN: 1-59593-054-X.

Tulone, Daniela and Samuel Madden

- 2006 "PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks," in *European Conference on Wireless Sensor Networks*, pp. 21-37.

Villas, Leandro A., Azzedine Boukerche, Daniel Guidoni, Regina B. Araujo, and Antonio A. F. Loureiro

- 2011 "An energy-aware spatial correlation mechanism to perform efficient data collection in WSNs," in *Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks*, LCN '11, pp. 882-889, ISBN: 978-1-61284-926-3.

Vuran, Mehmet C., Özgür B. Akan, and Ian F. Akyildiz

- 2004 "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 45, 3, pp. 245-259, ISSN: 1389-1286.

Wang, Chao et al.

- 2012 "Adaptive Approximate Data Collection for Wireless Sensor Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 23, 6 (June 2012), pp. 1004-1016, ISSN: 1045-9219.

Wang, Chao, Huadong Ma, Yuan He, and Shuguang Xiong

- 2012 "Adaptive Approximate Data Collection for Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, 6, pp. 1004-1016, ISSN: 1045-9219.

Wang, Chao, Huadong Ma, Yuan He, and Shuguang Xiong

- 2010 "Approximate Data Collection for Wireless Sensor Networks," in *Proc. of the 2010 IEEE 16th Int. Conf. on Parallel and Distributed Systems*, ICPADS '10, pp. 164-171, ISBN: 978-0-7695-4307-9.

Wang, Lidan and Amol Deshpande

- 2008 "Predictive modeling-based data collection in wireless sensor networks," in *Proceedings of the 5th European conference on Wireless sensor networks*, EWSN'08, Bologna, Italy, pp. 34-51, ISBN: 3-540-77689-3, 978-3-540-77689-5.

Wang, Xue, Jun-Jie Ma, Liang Ding, and Dao-Wei Bi

- 2007 "Robust Forecasting for Energy Efficiency of Wireless Multimedia Sensor Networks," *Sensors*, vol. 7, 11, pp. 2779-2807.

- Werner-Allen, G., J. Johnson, M. Ruiz, J. Lees, and M. Welsh
 2005 "Monitoring volcanic eruptions with a wireless sensor network,"
 in *Wireless Sensor Networks, 2005. Proceedings of the Second Euro-
 pean Workshop on*, IEEE, pp. 108-120, ISBN: 0-7803-8801-1.
- Whitehouse, Kamin et al.
 2002 "Calibration as parameter estimation in sensor networks," in *Proc.
 of ACM WSNA'02*, pp. 59-67.
- Xue, Wenwei, Qiong Luo, Lei Chen, and Yunhao Liu
 2006 "Contour map matching for event detection in sensor networks,"
 in *Proceedings of the 2006 International Conference on Management of
 data*, pp. 145-156.
- Yick, Jennifer, Biswanath Mukherjee, and Dipak Ghosal
 2008 "Wireless sensor network survey," *Comput. Netw.*, vol. 52, 12, pp. 2292-
 2330.
- Yoon, Sunhee and Cyrus Shahabi
 2007 "The Clustered AGgregation (CAG) technique leveraging spatial
 and temporal correlations in wireless sensor networks," *ACM
 Transaction on Sensor Networks*, vol. 3 (1 Mar. 2007), ISSN: 1550-4859.
- Yu, Liyang, Neng Wang, and Xiaoqiao Meng
 2005 "Real-time forest fire detection with wireless sensor networks," in
*Proceedings of International Conference on Wireless Communications,
 Networking and Mobile Computing*. Vol. 2, pp. 1214-1217.
- Yu, M. et al.
 2007 *A Survey on Fault Management in Wireless Sensor Networks*, vol. ISBN
 1-9025-6016-7.
- Zhang, Chongqing, Minglu Li, and Min-You Wu
 2006 "Model-Aided data collecting for wireless sensor networks," in
*Proc. of the Second international conference on High Performance Com-
 puting and Comm. HPCC'06*, Munich, Germany, pp. 692-699, ISBN:
 3-540-39368-4, 978-3-540-39368-9.
- Zhang, Yang et al.
 2010 "Outlier Detection Techniques for Wireless Sensor Networks: A
 Survey," *IEEE Communications Surveys & Tutorials*, vol. 12, 2, pp. 159-
 170.

Zhao, Jerry et al.

- 2002 "Sensor Network Tomography: monitoring wireless sensor networks," in *SIGCOMM Comp. Comm. Review*, 1, vol. 32, pp. 64-64.

Zhao, Jerry, Ramesh Govindan, and Deborah Estrin

- 2003 "Computing aggregates for monitoring wireless sensor networks," in *Proc. of IEEE SNPA'03 Workshop*, pp. 139-148.

Zhao, Yonggang J., Ramesh Govindan, and Deborah Estrin

- 2002 , in *IEEE Wireless Communications and Networking Conference (WCNC 02)*.

Zhao, Yonggang Jerry

- 2002 "Residual energy scan for monitoring sensor networks," in *Wireless Communications and Networking Conference*, pp. 356-362.

Zhao, Yonggang Jerry et al.

- 2002 "Residual energy scan for monitoring sensor networks," in *Proc. of IEEE WCNC'02*, pp. 356-362.

Zhou, Xi, Guangtao Xue, Chen Qian, and Minglu Li

- 2008 "Efficient Data Suppression for Wireless Sensor Networks," in *Proceedings of 14th IEEE International Conference on Parallel and Distributed Systems*, pp. 599-606, ISBN: 978-0-7695-3434-3.

INDEX

- Data Compression
 - Fault-Tolerance, [11](#), [118](#)
 - Spatial, [16](#)
 - Spatio-Temporal, [10](#), [17](#), [118](#)
 - Temporal, [16](#)
- Data Errors, [20](#)
- Data Model, [27](#)
 - Box-Jenkins Model, [101](#)
 - Linear Component, [27](#)
 - Master Cluster, [46](#)
 - Model Cache, [44](#), [52](#)
 - Model Caching, [49](#)
 - Random Component, [29](#)
 - Spatial Modeling, [47](#)
 - Clique, [52](#)
 - Cluster Head, [48](#)
 - Temporal Compression, [38](#)
 - Adaptive Modeling, [41](#)
 - Approximation Window, [42](#)
 - Model Construction, [30](#), [46](#)
 - Model Invalidation, [40](#)
 - Model Order, [41](#)
 - Outlier Detection, [39](#)
- Event
 - Data Error, [84](#)
 - Phenomenon, [84](#)
- Event Prediction, [12](#)
- Fault Model, [31](#), [77](#)
 - Data Error Detection, [79](#)
 - Permanant Errors, [32](#)
 - Constant, [32](#)
 - Transient Errors, [32](#)
 - Noise, [31](#)
 - Short, [31](#)
- Functional Faults, [19](#), [78](#), [83](#)
- Predictive Monitoring, [21](#), [99](#), [119](#)
- WSN, [3](#)