10-16-2013

Ensemble Stream Model for Data-Cleaning in Sensor Networks

Vasanth Iyer

Florida International University, viyer002@fiu.edu

DOI: 10.25148/etd.FI13120409

Follow this and additional works at: <https://digitalcommons.fiu.edu/etd>

Recommended Citation

Iyer, Vasanth, "Ensemble Stream Model for Data-Cleaning in Sensor Networks" (2013). *FIU Electronic Theses and Dissertations*. 973.
<https://digitalcommons.fiu.edu/etd/973>

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact dcc@fiu.edu.

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

Ensemble Stream Model for Data-Cleaning in Sensor Networks

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Vasanth Iyer

2013

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Vasanth Iyer, and entitled Ensemble Stream Model for Data-Cleaning in Sensor Networks, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Deng Pan

Shaolei Ren

Shu-Ching Chen

Jean Andrian

Niki Pissinou, Co-Major Professor

S.S. Iyengar, Co-Major Professor

Date of Defense: October 16, 2013

The dissertation of Vasanth Iyer is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2013

© Copyright 2013. by Vasanth Iyer
All rights reserved.

DEDICATION

To my parents, wife, and our two children, Puja and Pranav and to the community of research scholars such as C. V. Ramamoorthy, University of California at Berkeley, Department of Computer Science, 1970. Also, like to dedicate this to two of my milestones of successful research, I was fortunate to be involved in. First, Jim Hensen puppeteer for his original show, "*The Muppets*" at Walt Disney World Florida's addition of the lights and sound show a Disney Imagineering featured with Genesis Control Series ©1990. Second, the "*Triumph of the Nerds*" a public documentary on Personal Computer revolution ©1996 (when I was at Canon Research, Palo, Alto, CA. working on document archiving for Picture in Computer -PIC).

ACKNOWLEDGMENTS

The continued support over the years of my supervisor, S.S. Iyengar, is highly appreciated as it has contributed in making greater improvements in this thesis. When I started my research it was with the sole aim of learning. However, as I went along, S.S. Iyengar along with R.R. Brooks motivated me to take up Ph.D at Miami and further Malcolm Atkinson helped to tackle future issues in petabyte volumes. This could not have been possible without the help of my co-advisor Niki Pissinou who had just started a Network track with Shaolei Ren and Deng Pan and helped me secure my research topic in data-cleaning of sensor streams with a Fellowship.

To be eligible for graduate funding I needed to complete my qualifiers in time after my first summer term. This period was extremely hectic and I was spending long hours for my courses with Tao Li and Wei Zing. During this time, my program chair Shu-Ching Chen was already petitioning for me and helping me secure the waivers. I am very grateful for this gesture.

I would also like to thank many others from LSU and UNH who have made my stay in the US a memorable one. At I learned many important things when I was a TA under Mathivanan Packiam and latter with Howard F. Okrent in Networks, which enabled me to extend my master's work. Moreover, there are many post-docs of the sensor network such as Winston Seah, JongHoon Kim and Dhananjay Sigh who collaborated with me and urged me to take up sensors instead of coursework. Funding for all the sensor projects was generously extended by S.S. Iyengar through many NSF grants.

I'm also thankful to many who lived overseas when I studied on the other side of the world in Australia. I'm thankful to Alistair Moffat and Raymond Wan for their introduction to latex and research. Palaniswami .M, Kotagiri Rao and Jim Bailey in mentoring me to cope with the pressure of academic publications and Ph.D. endurance. And last, but most of all, to my parents and my brother for supporting

and caring and having the patience, when I was not working due to my academic ambitions.

Back home, I like to thank my Institute for extending their help on my Ph.D. preparations specially G. Ramamurthy (ECE), M.B. Srinivas (ECE), Shatrunjay Rawat (Networks), P.J. Narayanan (Research), and Shailesh Kumar (Google India). Subhas Mukhopadhyay (IEEE Fellow), Vir Phoha (ACM Distinguished Scientist) and N. Balakrishnan (Editor) for their continued support during for my ongoing research.

ABSTRACT

ENSEMBLE STREAM MODEL FOR DATA-CLEANING IN SENSOR NETWORKS

by

Vasanth Iyer

Florida International University, 2013.

Miami, Florida

Professor S.S. Iyengar, Co-Major Professor

Professor Niki Pissinou, Co-Major Professor

Ensemble Stream Modeling and Data-cleaning are sensor information processing systems have different training and testing methods by which their goals are cross-validated. This research examines a mechanism, which seeks to extract novel patterns by generating ensembles from data. The main goal of label-less stream processing is to process the sensed events to eliminate the noises that are uncorrelated, and choose the most likely model without over fitting thus obtaining higher model confidence. Higher quality streams can be realized by combining many short streams into an ensemble which has the desired quality. The framework for the investigation is an existing data mining tool.

First, to accommodate feature extraction such as a bush or natural forest-fire event we make an assumption of the burnt area (BA^*), sensed ground truth as our target variable obtained from logs. Even though this is an obvious model choice the results are disappointing. The reasons for this are two: One, the histogram of fire activity is highly skewed. Two, the measured sensor parameters are highly correlated. Since using non descriptive features does not yield good results, we resort to temporal features. By doing so we carefully eliminate the averaging effects; the resulting histogram is more satisfactory and conceptual knowledge is learned from sensor streams.

Second is the process of feature induction by cross-validating attributes with single or multi-target variables to minimize training error. We use *F-measure* score, which combines precision and accuracy to determine the false alarm rate of fire events. The multi-target data-cleaning trees use information purity of the target leaf-nodes to learn higher order features. A sensitive variance measure such as *f-test* is performed during each node's split to select the best attribute. Ensemble stream model approach proved to improve when using complicated features with a simpler tree classifier.

The ensemble framework for data-cleaning and the enhancements to quantify quality of fitness (30% spatial, 10% temporal, and 90% mobility reduction) of sensor led to the formation of streams for sensor-enabled applications. Which further motivates the novelty of stream quality labeling and its importance in solving vast amounts of real-time mobile streams generated today.

DECLARATION

This is to certify that

- the thesis comprises only my original work towards the Ph.D except where indicated in the Preface,
- due acknowledgement has been made in the text to all other material used, and
- the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies, and appendices.

Vasanth Iyer

MSCIS '89, UNH, CT

Research Scholar (IIIT-Hyderabad, 2012)

PREFACE

Publications arising from this thesis.

The general concept of stream event processing using data mining of forest fire events was introduced as a full paper at the IEEE-ICDM Conference on Data Mining, 2011 [V.Iyer, and Iyengar, 2011], and subsequently expanded for a book chapter in *Intelligent Sensor Networks: Networks Signal Processing and Machine Learning* [V.Iyer, Iyengar and Niki Pissinou, 2013]. Data-cleaning and mobility using ensemble models, as reviewed in Chapter 6 of this study, was presented at the IEEE-PERCOM, 5th International Workshop on Information Quality and Quality of Service for Pervasive Computing 2013 [V.Iyer, Iyengar, Niki Pissinou, and Ren, 2013]. The qualitative results from the experiments of Chapter 8 of this study on power-aware simulations using SENSESIM and related MAC's was introduced at the poster presentation held during the Ph.D forum at the IEEE-ISSNIP, Melbourne Conference [V.Iyer and Iyengar, 2009]. The mobility research and coexisting of WSN [V.Iyer and Iyengar, 2010] was presented at the IEEE-IPCCC, 2010.

TABLE OF CONTENTS

CHAPTER		PAGE
I	Introduction	1
1	Introduction to Data-cleaning	2
1.1	Data-Cleaning	5
1.2	Quality of Data	7
1.3	Data Gathering in Sensor Networks	10
1.4	Issues Related to Mobile and Remote Sensing Applications	11
1.5	An Architecture for Feature Selection	11
1.6	Feature Creation	12
1.7	Spatial and Temporal Features	12
1.8	Classification	12
1.9	Types of Data	14
1.10	Classification of Data Streams	15
1.11	Structure of this thesis	16
 II	 Streams Processing 2	 19
	Stream Data Model	20
2.1	Model Definitions	20
2.2	Similarities and Dissimilarities	20
2.3	Distance	21
2.4	Base Classifier	22
2.5	Ensemble Model	22
2.6	Combining Static and Mobile Models	24
 3	 The Sub-problems in Clustering	 26
3.1	Introduction to Compressed Sensing	27
3.2	Computation Model	27
3.3	Multi-sensor Data Fusion	28

CHAPTER	PAGE
3.4 Pre-Processing	29
3.5 Compressive Sensing (CS) Algorithms	29
3.6 Distributed Compressive Sensing (DCS) Algorithms	31
3.7 Comparison of DCS and Data Aggregation Methods	33
3.8 Summary	36
4 Problems in Real-time Processing of Digital-Streams using Compression	38
4.1 Compressing Data Streams	38
4.2 Introduction	38
4.3 Trans-coding Model	40
4.4 Comparison of Current Source Formats	46
4.5 Related Work	46
4.6 Dynamic Range Control Using Fixed Step Quantization	52
4.7 Prototype Results of Step Quantization	54
4.8 Summary	54
5 Sub-problems in Quality of Sensor Logs	56
5.1 Supervised Classification of Skewed Sensor Logs	57
5.2 Background	58
5.3 State of the Art	60
5.4 Sensor Measurement and Fire Activity	63
5.5 Alarm Ranking Function for Accidental Small Fires	66
5.6 Performance of fire topics classification using temporal ranking	68
5.7 Alarm Ranking Function for Large Fires	69
5.8 Performance of Fire Topics Classification using FWI Ranking	70
5.9 Machine Learning Algorithms	72
5.10 Tree Classifier	74
5.11 Simulation	79
5.12 Correlation of Attributes	83
5.13 Better Model for Weather Data	84
5.14 Summary	85
6 Data Cleaning Trees	86
6.1 Introduction	86
6.2 Label-less Classification Problem Definition	89
6.3 Concept Learning	89
6.4 Related Work	93
6.5 Different Types of Noise in the Domain	96
6.6 Ensemble of Data Cleaning Trees	97
6.7 Results	101

CHAPTER	PAGE
III Mobility	103
7 QoD using SPOTLESS	104
7.1 Introduction	104
7.2 Motivation	106
7.3 Stream QoD Evaluation	107
7.4 Numerical Modeling of Sensor Data Matrix	109
7.5 Regression and Data Cleaning	111
7.6 Results	116
7.7 Summary	120
8 Managing Mobile Trajectories using HEAPS	122
8.1 Introduction	122
8.2 Sequence Classifier	125
8.3 Training the Ensemble HMMs	128
8.4 Summary	128
9 Effects of Noise Due to User Mobility	130
9.1 Introduction	131
9.2 Related Standards	132
9.3 Model Prediction	133
9.4 Modeling Mobility in Wireless Channels	137
9.5 Sensor Network Life-Time Vs Interference	140
9.6 Simulation	143
9.7 Summary	147
9.8 Interference vs Energy Model	149
IV Protocols	154
10 Research Methodologies used in WSN Simulations	155
10.1 Main contributions and organization of the work	155
10.2 Distributed Algorithms	157
10.3 Information Processing	158
10.4 Distributed source coding with side information	159
10.5 Event Detection	160
10.6 Reliability of Sensor Networks	160
10.7 Application Protocol Algorithms	162
10.8 Simulation	163
10.9 Bit Error Ratio (BER)	168

11 Renewable Energy	176
11.1 Renewable Energy: Qos Simulation of Wireless Sensor Networks . . .	176
11.2 Introduction	176
11.3 Motivation	178
11.4 Renewable Energy Model and Energy Harvesting	179
11.5 Lifetime Modeling-Syetsm Performance Using MAC Dependence . . .	180
11.6 Performance Analysis of WSN Data Aggregation Algorithms	182
11.7 Network Model	185
11.8 MAC Performance Using Ultra-low Duty Cycle	189
11.9 Summary	189
12 Experimental Setup	191
12.1 System Model and Notation	191
12.2 Power-aware Complexity	193
12.3 Scale Invariance Property	196
12.4 Fault Rate	197
12.5 Bayesian Classifier	198
12.6 Lifetime Calculation	200
12.7 Compression Rate	201
12.8 Probability Mass Function	204
12.9 Probability Density Function	205
V Summary	206
13 Summary	207
13.1 Putting it all together	210
13.2 Toward the Future	214
Bibliography	216
A Appendices	226
Index	248
VITA	250

List of Figures

FIGURES	PAGE
1.1 Data-cleaning for next generation wireless and mobile sensing using Quality of Data (QoD) of streams in high Bit Error Rate(s) (BER).	3
1.2 Ensemble trees streams with varying purity leaf nodes.	6
1.3 Correction of features and noise in event detection when sensor' are faulty used by by Canadian Forest Department.	10
2.1 Stream classification using multi-tree ensemble model.	22
2.2 Feature extraction and data-cleaning process (adapted from (Wu <i>et al.</i> , 2003)).	23
2.3 Reversible variable length codes	24
3.1 Bipartite graphs for distributed compressed sensing.	31
3.2 Sensor value estimation with aggregation and sensor fusion.	34
4.1 Reversible Stream Encoder (adapted from Girod (1999))	48
4.2 Reversible Stream Decoder (adapted from Girod (1999))	49
4.3 Block diagram of 4× variable bit channel mixer.	53
4.4 Bit masking for adaptive 8 to 16 bit.	53
4.5 PCM 8-bit to 16 bit decoding with variable volume selection [1..64].	55
5.1 Histograms of empirical samples.	58
5.2 Bayes net representation.	59

5.3	Twice the precision Vs relevance. Fire activity vs Burnt Area, yields a logarithmic relation.	67
5.4	Fire alarm days ranked using $F@(0.5)$	69
5.5	Lower-bound range of FWI = (blue) and higher-bound range FWI > (yellow), shows all large fires (red) fall into the yellow region.	71
5.6	Weka algorithm toolkit.	78
5.7	Plot of F-score for each model and its performance with WEATHER dataset training data.	83
6.1	A clustering tree of the first-order.	88
6.2	Data cleaning for labelless streams.	88
6.3	Multi-target and input combinations.	88
6.4	Cluster tree of 4 leafs for the WEATHER dataset.	91
6.5	Cluster tree with pruning enabled.	92
6.6	Boot strapping used in bagging during model training.	99
7.1	Empirical evaluation of the accuracy in the proposed hypothesis. . . .	112
7.2	The prototype of mobile probe sensor dataset being validated with SensorML's static ground truth estimates.	113
7.3	QR Factorization of a sample static measurement matrix X_{lboud} and its covariance matrices M and S is shown for $k = 3$ sensors. The estimated values of $\hat{x}$ are shown for $k=3,2,1$ with decreasing precision.	114
7.4	Fading and shadowing effects, clustering and without mobility. Figure 7.4(g) is a comparison of 802.15.4 using BPSK. Figure 7.4(h) Turbo codes with block length of 256.	117
8.1	Euler's tour and trajectory tour transforms.	124

8.2	Distribution of latitude and longitude from the trajectory dataset.	
	The variations can be captured in 2-States.	125
8.3	Trajectory Visualization from GEOLIFE Dataset Showing GPS States.	127
8.4	Hidden Markov model showing different activity recognition and its corresponding states.	127
9.1	Frequency and bandwidth of unlicensed user partially overlapping with primary user.	132
9.2	Temporal variations in time-domain with a single mobile user and fixed base stations.	138
9.3	Mobility framework and wireless channel for calculating pathloss. . .	143
9.4	Co-existing of WSN and node density	148
10.1	Simulation of a two-dimensional sensor placement showing routing tree.	156
10.2	On-line and sensor logs showing measured and non-measurable fields.	156
10.3	Two dimensional model for simulation of distributed clustering	163
10.4	Show energy dissipation node loading(with link layer abstraction). . .	164
10.5	Show energy dissipation node loading using multi-hop and hardware energy optimizations.	165
10.6	MAC performance comparison.	166
10.7	Data aggregation using SPEED/Directed Diffusion/LEACH	167
10.8	Show wireless channel simulation setup for minimizing transmission error due to transmission interference.	170
10.9	Using Huffman trees to compute min-entropy with $ l(i) - l(j) \leq 1$. .	172
10.10	Fault redundancy Huffman codes using codes with 2^{-b} codebook . . .	172
10.11	MAC performance in lifetime seconds	174
11.1	CSMA MAC performance with constant radio range	185

11.2	B-MAC performance with constant radio range	185
11.3	FARMS performance with constant radio range	185
11.4	Bayesian reliability bounds for Theorem 1, 2	185
12.1	(a) Cross Layer resource allocation and optimizer (b) WSN simulator	191
12.2	(a) Shows fixed energy overhead with distance in transmission, (b) Plot of the theoretical expected lifetime using Power Law, being used to demonstrate ranking of popularity. To the right is the long tail, to the left are the few that dominate (also known as the 80-20 rule). . .	195
12.3	Measurement metrics for node failures	198
12.4	Estimation of CH selection error & MAC layer routing using Bayesian P^*	200
13.1	The sensor data-matrix of the overall Data-Cleaning System, with relevant chapters for Static and Mobile Streams as indicated.	211

List of Tables

TABLE	PAGE
3.1 A typical random measurements from sensors showing variations in ranges.	33
4.1 CODEC and FPS requirements	41
4.2 Bandwidth requirements	41
4.3 Golomb/Rice Codes generated by the codec output for easy indexing of frames 1-30.	51
4.4 Adjusted PCM values for selected volumes (1 AND 128).	55
5.1 Confusion Matrix of Classifier.	60
5.2 Mean comparison of precision and recall.	61
5.3 Performance of ranking with false alarm rates.	69
5.4 FWI classes.	69
5.5 Cost function matrix for misclassification	71
5.6 Posteriors probabilities for background WEATHER dataset data for the peak month August.	72
5.7 Target variable occurrences.	73
5.8 Likelihood of fires for the month of August.	73
5.9 Posteriors probabilities for temporal feature day of the week.	74
5.10 Gain ratio calculation for tree using entropy.	79
5.11 Evaluation on training set for Naive Bayes.	80
5.12 Evaluation on training set for J48 tree classifier.	80

5.13	Evaluation on training set for SVM linear classifier.	81
5.14	Confusion matrix for Naive Bayes using training set.	81
5.15	Confusion matrix on training set for J48 Tree classifier.	81
5.16	Confusion matrix on testing set for J48 Tree classifier.	82
5.17	Confusion matrix on training set for SVM linear classifier.	82
5.18	Confusion matrix on testing set using SVM.	82
5.19	Confusion matrix on training set using Bayes network.	82
5.20	Confusion matrix on testing set using Bayes network.	82
5.21	Confusion matrix on training set for $FWI > 32$	83
5.22	Attribute selection 10 fold cross-validation (stratified)	84
5.23	Evaluation $FWI > 32$ on training set for J48 tree classifier.	84
5.24	F-measure performance for all tests compared with WEKA.	85
6.1	Datasets for static and mobile sensor streams.	97
6.2	Comparison of ensemble learning for multi and single target DC-trees.	98
7.1	Missing features (Kappa metric)	115
7.2	Missing labels (unsupervised label-less learning).	115
9.1	Peak SNR measured values of the diagonal elements from the complete data sets. (Method-1 Vs Method-2)	137
9.2	Cognitive Radio Channel Interference Simulation Setup	144
9.3	Simulation setup of 802.15.4 cognitive WSN network.	147
9.4	Energy Efficiency Comparison used by Radios	153
10.1	Summary of notations for analysis of routing fault-rate	164
10.2	Summary of notations for analysis of MAC performance.	175
12.1	Load simulation for lifetime routing when $N=100$ and $F=1$ (nano sec.)	200

12.2	Energy consumption in duty cycle when $N=100$ and $F=1$	200
12.3	Summary of Notations for Analysis of Huffman Coding.	202
A.1	Energy Efficiency Comparison used by Radios	247

List of Algorithms

ALGORITHMS	PAGE
3.1 DWT: Using a cost function for searching the best sparse representation of a signal.	35
4.1 ReversibleVLCStream(S)	48
6.1 DataCleaningTree DCT(E)	101
6.2 PurestTree(E)	102
A.1 K-means	226
A.2 Online-K-Means On-line Clustering	226
A.3 CONSISTENT HASHING <VALUE>: Sensor Measured values in the network	227
A.4 HUFFMAN ALGORITHM: Modified priority Q for merging probability	227
A.5 NORMAL SQL: node 1 to 100 will provide 100 network messages . .	227
A.6 DISTRIBUTED SQL: node 1 to 100 will aggregate 1 network messages and many local aggregation with hop < lg 100	227

Part I

Introduction

Chapter 1

Introduction to Data-cleaning

The primary motivation in stream data-cleaning is to correct missing values, outlier detection due to fault sensors, which amounts to 30% of all existing errors effecting precision of the estimated sensor data. Similarly, wireless errors are hard to quantify as these errors cause significant redundant retry messages which draining the sensor battery and reduces the overall network range. It has be shown that efficient net-coding can reduce channel errors significantly and increase affective life-time of sensor networks by a factor 4x. We have successfully devised a pre-processing algorithm to fix missing values which are present in the data stream. If net-coding is the 'woods' of data-cleaning, we will further see how to achieve 20%-50% better data-quality from in-networking processing using ensemble of feature trees known as data-cleaning 'trees' (see chapter 6).

Current data-cleaning algorithms are designed to handle wired sensors and do not consider channel noise and error corrections due to wireless transmission and node mobility. Even though there are many datasets available from traffic sensors using GPS reading, the data quality is still far from being addressed and how applications can fulfill its information needs. Present data-cleaning algorithms perform calibration and local estimation of noisy sensor data. To further accommodate errors due to random and bursty traffic the data-cleaning steps needs to address encoding of the sensor data for reliable wireless transmission and flag any data quality issues

Figure 1.1: Data-cleaning for next generation wireless and mobile sensing using Quality of Data (QoD) of streams in high Bit Error Rate(s) (BER).

effecting applications to perform end-to-end labeling.

Estimation and prediction of events are dependent on the number of training samples available. Due to unavailability of labeled training samples algorithms need to rely on unlabeled streams. Many of the new applications depend on streams generated by mobile sensors and to correlate it with existing static streams a time window is necessary. We know from static sensors processing that the errors can be corrected locally within one hop distance, we need to apply this to sensor networks where the decoder may be many hops away from the source. Current codes are designed to be resilient to bursty errors but they are not optimized for power consumption and are often complex to decode. The data-cleaning codes need to design have the same error resilient characteristics, but also needs be rely on sensing information which allows to decode accurately by using less number of bits. The use of sensing information reduces the number of possible codes for a given time window. The transmitting sensor needs not send the static value periodically, and only needs to transmit a subset of bits within the dynamic range to the correlated decoder. The proposed solution is described as shown in Figure 1.1, which reviews how the low-level algorithms reduces floor noise and calibrates the sensor using an

on-line time window in one-dimensional X-axis. Using the one-dimensional sensor hypothesis, the Y-axis quality labels for these streams are predicted. The use of multi-attributes to predict higher order labels (Y) using bagging ensures diversification minimizing possible sampling errors to a minimum from its observed values in X-axis. The data-cleaning ensemble (Y) has the quality label which has the lowest error for the estimated encoded streams (Z). The encoded stream ($\hat{Z}$) which can forward correct multi-hop Bit Errors (BER) carries the features (Y) and its current sensing range (X), and are updated only when measurement changes are observed within its interval range ($\hat{X}$), ensuring minimum communications between the sensors and the correlated decoder.

Longer codes gives the best performance (Shannon theorem (Jiang, 2013)) for error correcting, but at the same time increases the decoding complexity. Error handling needs to address random and bursty errors as in wireless channels are affected by fading. By concatenating smaller codes increases its performance without increasing the complexity.

The ensemble error probability average-bound for block codes and convolutional codes are shown below:

$$P_E \leq 2^{-nE_B(R)} \quad (1.1)$$

$$P_E \leq 2^{-(M+1)nE_c(R)} \quad (1.2)$$

Note that the rate R is defined as:

$$R = \frac{k}{n} \quad (1.3)$$

Where k is the message length in bits, n is the codeword length ($n > k$), and $(n - k)$ is the parity check bits. If $\frac{k}{n} < 1$, the effective data rate is reduced, which means

the sensor has to use more battery power to maintain the same data rate during routing K bit messages. Decoding dominates the computational complexity, it is exponential to the length of the code.

1.1 Data-Cleaning

$\mathcal{D}$ ata-cleaning (DC) of sensor stream is the study of methods to obtain information from noisy sensor generated data. Some of the previous algorithms are deployed in an off-line context as all data is present before the start of data-cleaning. There are times when data-cleaning has to be performed on-line as in the case of real-time data streams. The data streams can be labeled or label-less; typically, labeled streams are used for training of the Data-cleaning classifier.

Sensor Networks (Iyengar *et al.*, 2010) is one of the major contributors of sensor generated data for weather prediction. The expansion of mobility using smart phones constitutes a large portion of mobility and location based sensor data (Iyer *et al.*, 2013). Since the deployment of sensor network is unattended and has a fixed lifetime, sensors tend to become faulty or run out of battery power thereby compromising the data quality. In the case of mobile sensors the dimension of the problem is two-fold: the battery life and wireless connectivity. Due to limited range and bandwidth constraints and GPS errors the transmission is riddled with blackouts (Balakrishnan *et al.*, 2006) causing inconsistency in quality. Nevertheless this is an extraordinary source of information and has been featured by MIT as the best technology area for 2013.

The sub-problems of data-cleaning include feature extraction and sensor fault isolation. In a traditional sensor network to know an event, a query specifying a parameter of interest and its maximum value is sent across the network. The sensor,

Figure 1.2: Ensemble trees streams with varying purity leaf nodes.

which satisfies the distributed query parameters responds with location and the observed matching of evidence collected by the cluster using a distributed Bayesian algorithm(Krishnamachari & Iyengar, 2004).¹

The distributed sensor fault isolation and measurements are performed to give a consistent view of the parameter’s precision and accuracy in the sensor’s confidence intervals. When the sensor measurements for the same parameter are constantly varying, a weighted average (Brook & Iyengar, 1996) of the total overlapping number of sensors gives its accuracy.

The quality features complement the data-cleaning steps allowing not to overfit the model with unknown parameters such as noise, high or inconsistent sensor values and not to include too many parameters by seeking to use a realistic training dataset. Learning how to use ensemble climate model as shown in Figure 1.2 has many advantages such as it provides higher accuracy from the observed sensor measurements without over-fitting the data. Some disadvantages of maintaining large ensembles are higher memory and space requirements. Due to temporal nature of climate models an ensemble ensures it covers the whole range, which is a desirable factor. Determining which is a goodmodel is difficult.

★ Model Parameters

✓ Ensemble Bagging

¹(Krishnamachari & Iyengar, 2004) this related work was performed at LSU, Baton Rouge.

- ✓ Ensemble of Random Forest
- ★ Testing
 - ✓ Cross-Validation
 - ✓ Pruning

We reduce the bias of the sensor model by using bagging and random forest ensembles to learn and fit the datasets. As the training error is an optimistic estimate of the unseen observed samples (testing error), the validation uses concept learning model in the form of descriptive attributes and multi-objective clustering using numeric attributes. Overfitting is reduced by pruning during cross-validation of the training sets. These methods allow to better estimate the underlying process, the sample size, and accuracy of the ensemble model.

Extracting the features can be performed in many ways. The efficient and the simplest method is to aggregate data spatially and correct local errors and use the pre-processed stream to further classify temporal features. Paulo Cortez and Anibal Morais (Cortez & Morais, 2010) use metrological data from tropical forests to work for Forest Fire Index (FWI) calculations. Chapter 5 has more detailed discussion of the algorithms.

1.2 Quality of Data

The sensors allow aggregating all the parameters online, in particular, the aggregation of sensor values can rapidly accumulate errors, so either avoid, or use pre-processing to compensate for those errors in use. It is hard to measure and label stream quality on-line. To address these problems, QoD focuses on: (1) the detection and correction of data quality problems, and (2) the use of algorithms that can tolerate poor data quality. The pre-processing step is called data cleaning. As there

are many dependent issues such as measurement of data, mobile data collection and applications integration, we will focus on stream based data-cleaning algorithms, which provide meta-data and wireless channel noise.

1.2.1 Measurement and Data Collection Issues

Accuracy and completeness are close to impossible to measure. Incompleteness: interpretability, accessibility, meta-data, analysis, etc., are vague definitions in that there is no way to improve the data based on cookie-cutter/conventional definitions. Some of the aspects of data quality that are related to measurement and data collection are discussed in the following sections.

1.2.2 Noise and Artifacts

The characteristic of noise is that it is highly uncorrelated. Data that has noise can be divided into spatial and temporal components. Signal processing techniques can be used to discover signal patterns that might be lost in noise. Generally during pre-processing steps we can devise robust algorithms, which produce acceptable results even when noise is present.

1.2.3 Precision, Bias and Accuracy

The standard definitions of statistical methods used in experiments by underlying generating processes are

Definition 1.2.1 *Precision: The closeness of repeated measurements (of the same quality) to one another.*

Definition 1.2.2 *Bias: A systematic variation of measurements from the quality being measured. Bias can be determined for parameters, whose measured quantity is known.*

Bias can be determined for parameters, whose measured quantity is known.

Definition 1.2.3 *Accuracy: The closeness of measurement to the true value of the quality being measured. This depends on precision and bias but is largely defined by the sensitivity of the overall measurement system deployed.*

1.2.4 Outliers

Hard to classify patterns can be defined as outliers. Typically, outliers are of two types: (1) measurements that have characteristics, which are different from most of the other readings in the data set or (2) values that are inconsistent with respect to other sensor measurements. This is context-dependent such as spatial or temporal data sets. Climate and weather extreme events are well defined. The key challenge is to find significant events and explain the cause.

1.2.5 Missing Values

Missing values are found, when some of the attributes measured have zero or null values. One technique to eliminate missing values, is to estimate the measured parameter's values using spatial consistency and correlations to the observed values.

1.2.6 Inconsistent Values

Inconsistent values are found when due to hardware fault the sensor reading tends to become contradictory. It may be the case of a clock skew due to drift in its synchronization. The example data in Figure 1.2, shows Inconsistent values are

(a) Event detection when sensors' are faulty.

Parameter Estimation

(adapted from Krishnamachari & Iyengar (2004))

(b) As example of fire logging event.

Ground truth BA^* estimate

Figure 1.3: Correction of features and noise in event detection when sensor' are faulty used by by Canadian Forest Department.

found when due to hardware fault the sensor reading tends to become contradictory. It may be the case of a clock skew due to drift in its synchronization. The example data in Figure 1.3, shows inconsistency in actual events and the estimate of the region of interest for the query.

1.3 Data Gathering in Sensor Networks

It has been recognized that energy savings can be obtained by pushing computation within the network in the form of localized and distributed algorithms. A related work (Iyer *et al.*, 2009f) compares the density of sensor nodes and how it affects the sensor faults during the lifetime of the network.

1.4 Issues Related to Mobile and Remote Sensing Applications

This relates to the fitness to the enabled sensor application. Data should be consistent even when used in a mobile database application. Also, the relevance of the information should be pre-processed at the lower-level for the application's use. Due to this unique abstraction we need a classification without bias and capture its variance. To further improve quality and reduce cost of data streams', we need to use the following pre-processing methods.

★ Aggregation	★ Feature subset selection
★ Sampling	★ Discretization and binarization
★ Dimensionality reduction	★ Variable transformation

These pre-processing methods fall into two categories: (1) selecting feature attributes (2) using higher level features for the task in hand.

1.4.1 Feature Subset Selection

Depending on the data set and its domain certain rules can be applied to pre-process the initial feature selection. This step eliminates irrelevant and redundant features.

1.5 An Architecture for Feature Selection

Many of the learning algorithms use training sets and cross-validation sets to minimize training errors. This is the basic goal of feature selection as the best combination of labeled features produces the least training error for a given data set.

Feature selection process uses the search space of all the available attributes, which best fits the model. A typical example is tree classifier, which will be discussed in Chapter 6.

1.6 Feature Creation

Fusing two or more base line features to represent a new feature vector is called feature extraction. Chapter 6.2 will offer more details on Data Cleaning Trees. This will enable us to learn a descriptive concept using clustering of label-less streams. This suggests that the accuracy of the model is enhanced by generalizing on the patterns or features. Such features selection and generation are termed *novel* patterns (Duda *et al.*, 2000).

1.7 Spatial and Temporal Features

Sensor network is organized into clusters, which allow us to efficiently compute spatial correlated measurements by using minimum battery drain. Many of the applications are based on remote sensing and cover a large enough area, which captures temporal variations. Typically spatial measurement are concerned with accuracy of the sensors and the parameters it is measuring. On the other hand temporal measurements are concerned with the model and network aggregation characteristic.

1.8 Classification

To convert some of the current algorithms to measure quality on-line, we use the technical basis of newer learning algorithms from disciplines such as Statistics, Ma-

chine Learning and Neural Networks, which are opening the horizon of pattern recognition when applied to stream learning. Computer Vision, Object tracking and Virtualization of sensors, (Iyer *et al.*, 2013.), (Iyer & Iyengar, 2011) which use pre-recorded video streams rely heavily on these techniques. In all the supervised learning methods the underlying densities are assumed, the parameters estimated rarely fit the distribution generated by sensors. In practice, one method which works well is the Parzen Window (Duda *et al.*, 2000) approach, especially in the case of the hardware implementation of the Probabilistic Neural Network (PNN). PNN representation for this implementation needs d inputs units connected to the n pattern units. Each feature extractor unit is then connected to one of the c category units. The learning function (setting $w_k = X_k$ (1)) in the PNN is very fast and requires only one pass ($z_k = w_k^t X$ (2)) is inner product) of the input. The space complexity can be calculated by the number of wires $O(n+1)d$ needed in the model. As n and d can grow to large numbers this can be a limiting factor for hardware implementation. This can be a severe constraint when the input complexity grows exponentially with increasing dimensions - "*curse of dimensionality*". The time complexity can be achieved by the virtue of parallelization since the inner product can be implemented in $O(1)$. When it comes to on-line training of streams, new patterns (vectors representation) can be easily incorporated into the previously trained classifier due to the fact that the kernalized (Bishop, 2006) implementation needs access to all inputs during classification.

★ Supervised, reinforced
and unsupervised	★ Deterministic, stochastic and fuzzy
	★ Discrete and continuous
★ Off-line and on-line	★ Criteria for learning

1.9 Types of Data

★ Categorical (Mixed)	★ Spatial
✓ Fraud Detection	✓ GIS
✓ Cyber Networks	✓ Image analysis
★ Time Series	★ Spatio-temporal
✓ Sensor Networks	✓ Remote sensing
✓ Healthcare	✓ Climate
★ Discrete Sequences	★ Graphs
✓ Genomic	✓ Social networks
✓ System Calls	✓ Epidemiology

1.9.1 Sequential Data

Unsupervised Learning

Clustering is the choice of learning when the underlying data is non-parametric, where the distinct number of clusters is unknown. As there could be mixtures in the observed distribution, we need to maximize the likelihood from the limited number of samples available. This type of learning is called Bayesian learning. It is a better model, when compared to maximum likelihood estimation, which assumes a fixed value for the parameter but unknown. In a Bayesian non-parametric learning, the prior and post distributions are both stochastic processes. This type of clustering algorithms can be studied when data is observed in sequential manner and when Bayesian learning happened when the distribution was at the maximum. In the sensor model we use concept learning, which uses a cluster tree of descriptive attributes and leaves have multi-target attributes that form efficient clusters.

Streams measured by mobile sensors and time varying spatio-temporal streams need sequential data to be clustered. The ensemble of such streams can be studied using many Hidden Markov Models (HMMs) to learn latent features from the observed data. Temporal learning of useful events from data generated by sensor applications, such as mobile sensing can be described by a framework called HEAPS (HMMs Ensembles Aggregating Patterns from Streams). Similarly we use and mine different datasets, which are prone excessively to outliers and noise.

For example, some of the data-cleaning algorithms work well with GPS tagged data as it has context available and others need to be visually viewed to further remove outliers when clustering. In fact, some of the test sets have included very hard samples, which have high occultation and how an algorithm can categorize such level of noise.

1.10 Classification of Data Streams

Data streams can be broadly classified into:

- ★ Static sensor streams
- ★ Mobile sensor streams

Static and mobile streams are both generated by wireless sensors and are susceptible to data corruption due to external noise and channel interference. Static sensor can be time-series data or i.i.d samples, but the mobile sensors are always sequential in nature. In this thesis we use UCI- Machine Language repository datasets **WEATHER** (Frank & Asuncion, 2010), **FOREST FIRES** (Frank & Asuncion, 2010) and Microsoft Research provided trajectory data **GEOLIFE**(Zheng *et al.*, 2012). The formats and attributes are discussed in greater detail in Sections 6.4 and 8.3. For label-less

testing we use the same UCI-Machine Language repository datasets and remove the labels, which allow label-less learning from the observed attributes only.

1.11 Structure of this thesis

This thesis is structured into 12 chapters with an Appendix, each of which includes sections on related work and its summary. The overall structure is as follows: Chapter 2 introduces the reader to the Stream data model and to the notion of unsupervised classification using similarity measures and various types of algorithms used in training feature selection. Chapter 3 explains the sub-problems in clustering and how spatial features are estimated (Brook & Iyengar, 1996) by using Compressed sensing (Wavelets, 2000) and sparse sensor data matrix completion methods (Optimization methods) . Chapter 4 describes the notion of small, medium to large data sets and explains how temporal features in the form of events are estimated using data mining algorithms (HMMs, Random forest, Bagging, 1995). A UCI Machine Learning data set is used to compare the performance of data mining algorithms. Experimental results are presented that validate using audio music data (PCM) and compression, the analysis of detecting rare events such as **FOREST FIRES** using labeled and label-less datasets. Chapter 5 explains the sub problems in quality of sensor logs and introduces ensemble-based tree learning using current base classifiers, which come from noisy label-less streams and need bagging and advanced tree algorithms to further classify them with accuracy. Chapter 6 describes the data cleaning trees. Chapter 7 presents data collection using mobile wireless devices, and a framework called SPOTLESS for missing data is discussed in the context of spatial data-cleaning. It also offers details of the data-cleaning steps, and how the algorithms are divided into on-line and off-line to perform pre-processing, fea-

ture extraction (Krishnamachari & Iyengar, 2004) and post-processing of static and mobile streams generated by sensor enabled applications. The abstraction for a bottom up quality measure (Iyer et al., 2013.) using stream classification is proposed addressing mobile and sensor enabled applications. Chapter 8 discusses ensemble based time-series data in the form of GPS trajectories and the underlying HMM framework called HEAPS are used to classify quality of mobile sensor sequential trajectories. Chapter 9 discusses the effects of noise due to user mobility. Chapters 10, 11 and 12 review past work on Mobility, computational QoS, and renewable energy protocols using battery model. The cross-layer distributed node topologies are modeled using k-means, knn and Bayes Rule error bounds (Duda et al., 2000) for unsupervised clustering. The wireless 802.11 and its extensions to mobility and coexistence with emerging 802.15.4 standards, are simulated and compared for WSN-QoS performance. Chapter 13 puts it all together in the Summary of the thesis.

This thesis is structured into individual chapters and which have inclusive sections on related work and its summary. The overall structure is as follows. Chapter 2 introduces to the notion of unsupervised classification using similarity measures and various types of algorithms used in training feature selection. Estimating spatial features (Brook & Iyengar, 1996) is explained using Compressed sensing (Wavelets, 2000) and sparse sensor data matrix completion (Optimization methods) in Chapter 3. The notion of small, medium to large data sets are discussed in Chapter 4 and how temporal features in the form of events are estimated using data mining algorithms (HMMs, Random forest, Bagging, 1995). A UCI Machine Learning data set is used to compare the performance of data mining algorithms. Experimental results are presented that validate using audio music data (PCM) and compression, the analysis of detecting rare events such as **FOREST FIRES** using labeled and label-less datasets.

Ensemble based tree learning is introduced using current base classifiers in Chapter 5, which come from noisy label-less streams and needs bagging and advanced tree algorithms to further classify them with needed accuracy. Data collection using mobile wireless devices are introduced in Chapter 6, and a framework called SPOT-LESS for missing data is discussed in the context of spatial data-cleaning. Details of the data-cleaning steps, and how the algorithm are divided into on-line and off-line to perform pre-processing, feature extraction (Krishnamachari & Iyengar, 2004) and post-processing of static and mobile streams generated by sensor enabled applications. The abstraction for a bottom up quality measure (Iyer *et al.*, 2013.) using stream classification is proposed addressing mobile and sensor enabled applications. Ensemble based time-series data in the form of GPS trajectories are discussed in Chapter 7, and the underlying HMM framework called HEAPS are used to classify quality of mobile sensor sequential trajectories. Chapter 8 and 9 reviews past work on Mobility, computational QoS and renewable energy protocols using battery model. The cross-layer distributed node topologies are modeled using *k-means*, *knn* and *Bayes Rule* error bounds (Duda *et al.*, 2000) for unsupervised clustering. The wireless 802.11 and its extensions to mobility and coexistence with emerging 802.15.4 standards, are simulated and compared for WSN-QoS performance.

Part II

Streams Processing

Chapter 2

Stream Data Model

2.1 Model Definitions

Stream classification algorithm uses higher dimension transformation of the sensor measured input attributes, which allow the use of distance functions to compare its similarities in 2-D, 3-D and multi-dimension. Typically if this measure is low then the attributes are close and similar. The same concept of similarity can be extended to multiple attributes, which are represented as vectors in the transformed space. Some of the distance measures are Euclidian distance (Duda *et al.*, 2000), Mahalanobis distance, Minkowski distance, Hamming distance and Cosine similarity. In this chapter we will briefly introduce different algorithms of classification used in data mining. Sections 2.3 and 2.4 describes models which help accuracy.

2.2 Similarities and Dissimilarities

A comparison metric (Iyer *et al.*, 2008g) is used by classification algorithms let it be similarity based or discriminative model. In a probability model or when learning weights are in a neural network, the total sum of probabilities must be equal to 1. In the case of non-probability measures such as distance measures, a score is provided, which can be used as a metric for comparison. As the scores get smaller,

the samples become similar. The smaller the score gets then the samples are very similar.

2.3 Distance

The Euclidian distance, d , between two points, x , and y , in one-, two-, three-, or higher-dimensional space, is given by the following formula:

$$d(x, y) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2} \quad (2.1)$$

Mahalanobis distance is given by the formula:

$$Mahalanobis(x, y) = (x - y) \sum^{-1} (x - y)^T \quad (2.2)$$

Cosine similarity is given by the formula:

$$\cos(x, y) = \frac{x}{||x||} \cdot \frac{y}{||y||} = x' \cdot y' \quad (2.3)$$

Minkowski distance is given by the formula:

$$d(x, y) = \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{\frac{1}{r}} \quad (2.4)$$

2.3.1 Streams

When classifying streams can be divided into two major categories:

- ★ Labeled streams
- ★ Label-less streams

Figure 2.1: Stream classification using multi-tree ensemble model.

Labeled streams use supervised learning techniques and in this research we mostly mean label- less streams as the sensor data is unattended and unsupervised.

2.4 Base Classifier

A base classifier can be running any algorithm such as k-nearest neighbor, Decision Trees, Naive Bayes, (Duda *et al.*, 2000) etc. The framework calls it a base classifier as it can further provide stacking, voting and ensembles.

2.5 Ensemble Model

A set of base classifiers and their outcomes become the input to an ensemble, which uses a product, sum, mean and other voting methods to enhance the accuracy of the base classifier. Figure 2.1 shows the possible paths of input permutations and how these affect the complexity of the decision tree.

Figure 2.2: Feature extraction and data-cleaning process (adapted from (Wu *et al.*, 2003)).

2.5.1 Sub-Problems in Stream Classification

The subproblems can be categorized according to pre-processing, estimation and post-processing

★ Stream noise removal	★ Stream Segmentation into sequential data
★ Stream model overfitting	
★ Stream model selection	★ Streaming mobile content
★ Stream missing attribute values	★ Streaming in real-time

2.5.2 Sub-Problems in Forward Error Correction

We are seeking in applying stream processing in the context of power-aware sensor network applications. Due to this constraint the design needs to have coded streams with which allows reliability in routing sensor data payloads. The network layer performs aggregation into a single stream at every hop from routing its path between source to destination and is susceptible to data corruption due to wireless channel noise. The receiver's decoder has some built-in error recovery for basic decoding, but as the BER error rate (Bit Error Rate) increases SNR decreases. Due to this multiple bits are corrupted and the receiver is not able to correct them and requests a re-transmission. Due to communication overhead in redundant processing the lifetime of the sensor node is depleted due its fixed battery capacity. To further

Figure 2.3: Reversible variable length codes

include a step in data-cleaning of sensor network stream we need to device forward correction codes which are resilient to high BER error rates so that the decoders do not need redundant retries. In the PHY implementation schemes we discuss reversible codes for compressed streams in the context of sensor networks.

2.6 Combining Static and Mobile Models

Currently many of the climate modeling domains participating in sensing are doing a better job in gathering data that explain why a particular event or a phenomenon is explainable. Many of the data-cleaning algorithms as shown in Figure 2.2 deals with data which do not capture the underlying variation over time. In this thesis, we use several models which allow dealing with missing values in spatial, temporal and trajectory based sensing. The overall affects is to combine static and mobile sensing models to enhance the quality labeling of unsupervised streams. We propose a simple yet effective strategy to induce task dependent feature representation using ensemble model. These tasks can be descriptive attribute or regression based on multi-objectives for stream quality labeling. The use of quality labeling is further extended to sensor network applications that allow multi-hop forward error corrections. Finally, we provide concrete study and learning tasks for the following models.

- Spatial data-cleaning of sensor streams

- Temporal data-cleaning and labeling of sensor streams
- Mobile trajectory classification

Chapter 3

The Sub-problems in Clustering

Clustering and routing data are routine tasks in sensor networks. Most of the work ¹ presented in this chapter was done under sensor fusion and its applications to clustering. These projects were performed at International Institute of Information Technology, Hyderabad, India and National Institute of Mathematical Sciences, South Korea. Results were obtained with the datasets from the work by (Brook & Iyengar, 1996), and the participating authors. In this Chapter we explore the redundancy aspects of data and sampling which are primarily due to spatial deployment of sensors. In (Brook & Iyengar, 1996), the authors proposed an overlap function for noisy sensor measurement. The computational aspects were funded by a DoD grant from Louisiana State University² to the application of Compressed Sensing (Baron *et al.*, 2005) and hardware data path optimizations. Sections 4.5 and 4.6 in Chapter 3 describes pre-processing of spatial streams. Sections 4.7 and 4.8 in Chapter 3 compares how distributed sensing can be accomplished with the same accuracy using overlap functions of the original paper.

¹Sensor Fusion and its Applications, Chapter: Vasanth Iyer and Dhananjay Singh. "Distributed Compressed Sensing of Sensor Data." Edited by Ciza Thomas. Publisher: Sciyo, Chapters published August 16, 2010.

²New Developments and Applications in Sensing Technology. Chapter: Iyer, Vasanth and Iyengar, S.Sitharama and Murthy, Garmiela Rama and Srinathan, Kannan and Srinivas, MandaliikaB. and Govindarajulu, Regeti. "STACK: Sparse Timing of Algorithms Using Computational Knowledge. Lecture Notes in Electrical Engineering Springer Berlin Heidelberg, 2011"

3.1 Introduction to Compressed Sensing

Intelligent Information processing in distributed wireless sensor networks has many different optimizations by which redundancies in data can be eliminated, and at the same time the original source signal can be retrieved without loss. The data-centric nature of sensor network is modeled, which allows environmental applications to measure correlated data by periodic data aggregation. In the distributed framework, we explore how Compressed Sensing can be used to represent the measured signals in its sparse form, and model the framework to reproduce the individual signals from the ensembles. The processed signals are then represented with their common component, which is represented by its significant coefficients, and the variation components. These are sparse and projected using scaling wavelet functions of the correlated component. The overall representation of the basis preserves the temporal (intra-signal) and spatial (inter-signal) characteristics. All these scenarios correspond to measuring properties of physical processes that change smoothly in time, and in space, and thus are highly correlated. We show by simulation that the framework using cross-layer protocols can be extended using sensor fusion, and data-centric aggregation, to scale to a large number of nodes.

3.2 Computation Model

The sensor network model is based on network scalability of the total number of sensors N , which can be very large and go up to many thousand nodes. Due to this fact an application needs to find the computation power in terms of the combined energy it has, and also the minimum accuracy of the data it can track and measure. The computation steps can be described in terms of the cross-layer protocol messages in the network model. The pre-processing needs to accomplish the minimal number

of measurements needed, given by $x = \sum \vartheta(n) \Psi_n = \sum \vartheta(n_k)$, where Ψ_n^k is the best basis. The local coefficients can be represented by 2^j different levels, the search for best basis can be accomplished, using a binary search in $O(\lg m)$ steps. The post processing step involves efficient coding of the measured values, if there are m coefficients, the space required to store the computation can be accomplished in $O(\lg_2 m)$ bits. The routing of data using the sensor network needs to be power-aware, so they use a distributed algorithm with cluster head rotation, which enhances the total lifetime of the sensor network. The computation complexity of routing in terms of the total number of nodes can be shown as $OC(\lg N)$, where C is the number of cluster heads and N total number of nodes. The computational bounds are derived for pre- and post processing algorithms for large data-sets, and is bounds are derived for a large node size in Chapter 10.4, simulation plot.

3.3 Multi-sensor Data Fusion

By using the cross-layer protocol approach, we reduce the communication cost, and derive bounds for the number of measurements necessary for signal recovery under a given sparsity ensemble model, similar to Slepian-Wolf rate (Slepian & Wolf, 1975) for correlated sources. At the same time, using the collaborative sensor node computation model, the number of measurements required for each sensor must account for the minimal features unique to that sensor, while at the same time features that appear among multiple sensors must be amortized over the group.

3.4 Pre-Processing

As different sensors are connected to each node, the nodes have to periodically measure the values for the given parameters which are correlated. The inexpensive sensors may not be calibrated, and need processing of correlated data, according to intra and inter sensor variations. The pre-processing algorithms allow us to accomplish two functions, one, to use minimal number of measurement methods at each sensor, and two to represent the signal in its loss-less sparse representation.

3.5 Compressive Sensing (CS) Algorithms

This technique is called the sparse basis as shown in equation (3.1), if the signal measured shows that it can be represented as a sparse (Baron *et al.*, 2005) representation. The technique of finding a representation with a small number of significant coefficients is often referred to as Sparse Coding. Many techniques have been implemented for sensing locally such as the Nyquist rate (Baron *et al.*, 2005), which defines the minimum number of measurements needed to faithfully reproduce the original signal. Using CS algorithms makes it possible to reduce the number of measurements for a set of sensors with correlated measurements (Krishnamachari & Iyengar, 2004).

$$x = \sum \vartheta(n)\Psi_n = \sum \vartheta(n_k)\Psi_{n_k}, \quad (3.1)$$

Consider a real-valued signal $x \in R^N$ indexed as $x(n)$, $n \in 1, 2, \dots, N$. Suppose that the basis $\Psi = [\Psi_1, \dots, \Psi_N]$ provides a K -sparse representation of x ; that is, where x is a linear combination of K vectors chosen from, Ψ_{n_k} are the indices of those vectors, and $\vartheta(n)$ are the coefficients; the concept is extendable to tight frames (Baron *et al.*, 2005). Alternatively, we can write in matrix notation $x = \Psi\vartheta$, where

x is an $N \times 1$ column vector, the sparse basis matrix is $N \times N$ with the basis vectors Ψ_n as columns, and $\vartheta(n)$ is an $N \times 1$ column vector with K nonzero elements. Using $\|\cdot\|_p$ to denote the ℓ_p norm, we can write that $\|\vartheta\|_p = K$; we can also write the set of nonzero indices $\Omega_1, \dots, N$, with $|\Omega| = K$. Various expansions, including wavelets (Baron *et al.*, 2005), Gabor bases, curvelets, are widely used for representation and compression of natural signals, images, and other data.

3.5.1 Sparse representation

Sparse representation is a single measured signal of finite length, which can be represented in its sparse representation, by transforming it into all its possible basis representations. The number of basis for each level j can be calculated from the equation (3.2) as,

$$A_{j+1} = A_j^2 + 1 \quad (3.2)$$

As an example, starting at $j = 0$, $A_0 = 1$ and similarly, $A_1 = 1^2 + 1 = 2$, $A_2 = 2^2 + 1 = 5$ and $A_3 = 5^2 + 1 = 26$ different basis representations, making its complexity exponential.

Let us define a framework to quantify the sparsity of ensembles of correlated signals $x_1, x_2, \dots, x_j$ and to quantify the measurement requirements. These correlated signals can be represented by its basis from equation (3.2). The collection of all possible basis representation is called sparsity model.

$$x = P\theta \quad (3.3)$$

Where P is the sparsity model of K vectors ($K \ll N$) and θ is the non zero coefficients of the sparse representation of the signal. The sparsity of a signal is defined by this model P , as there are many factored possibilities of $x = P\theta$. Among

factorization, the unique representation of the smallest dimensionality of θ is the sparsity level of the signal x under this model.

3.6 Distributed Compressive Sensing (DCS) Algorithms

Figure 3.1: Bipartite graphs for distributed compressed sensing.

DCS enables distributed coding algorithms to exploit both intra-and inter-signal correlation structures. In a sensor network deployment, a number of sensors measure signals that are each individually sparse in some basis and also correlated from sensor to sensor. If the separate sparse basis are projected onto the scaling and wavelet functions of the correlated sensors (common coefficients), then all the information is already stored to individually recover each of the signal at the joint decoder. This does not require any pre-initialization between sensor nodes.

3.6.1 Joint Sparsity representation

For a given ensemble X , we let $P_F(X) \subseteq P$ denote the set of feasible location matrices $P \in P$ for which a factorization $X = P\Theta$ exists. We define the joint sparsity levels of the signal ensemble as follows. The joint sparsity level D of the signal ensemble X is the number of columns of the smallest matrix $P \in P$. In these models each signal x_j is generated as a combination of two components: (i) a common component z_C , which is present in all signals, and (ii) an innovation component z_j , which is unique to each signal. These combine additively, giving

$$x_j = z_C + z_j, j \in \forall \quad (3.4)$$

$$X = P\Theta \quad (3.5)$$

We now introduce a bipartite graph $G = (V_V, V_M, E)$, as shown in Figure 3.1, that represents the relationships between the entries of the value vector and its measurements. The common and innovation components K_C and K_j , ($1 < j < J$), as well as the joint sparsity $D = K_C + \sum K_j$.

The set of edges E is defined as follows:

- ★ The edge E is connected for all K_C if the coefficients are not in common with K_j .
- ★ The edge E is connected for all K_j if the coefficients are in common with K_j .

A further optimization can be performed to reduce the number of measurements made by each sensor. The number of measurement is now proportional to the maximal overlap of the inter sensor ranges and not a constant as shown in equation (3.1). This is calculated by the common coefficients K_C and K_j , if there are common coefficients in K_j then one of the K_C coefficient is removed and the common Z_C is

Sensors	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
Value	4.7 ± 2.0	1.6 ± 1.6	3.0 ± 1.5	1.8 ± 1.0	4.7 ± 1.0	1.6 ± 0.8	3.0 ± 0.75	1.8 ± 0.5
Group	-	-	-	-	-	-	-	-

Table 3.1: A typical random measurements from sensors showing variations in ranges.

added. These change does not affect the reconstruction of the original measurement signal x .

3.7 Comparison of DCS and Data Aggregation Methods

In Sections 3.4 and 3.5, we have seen various data processing algorithms. In terms of communication cost they are comparable. In this section, we will look into four design factors of the distributed framework:

1. How well can the individual sensor signal sparsity be represented.
2. What are the re be the minimum measurement possible if we use joint sparsity model from equation (3.5).
3. What are the maximum possible basis representations for the joint ensemble coefficients.
4. Determining a cost function search which allows to represent the best basis without overlapping coefficients.

The design framework allows us to pre-process individual sensor sparse measurement, and uses a computationally efficient algorithm to perform in-network data fusion.

To use an example dataset, we will use four random measurements obtained by multiple sensors. This is shown in Table 3.1. Here there are two groups of four

sensors each. The mean values are the same for both the groups and the variance due to random sensor measurements vary with time. The buffer is created according to the design criteria (1), which preserves the sparsity of the individual sensor readings. This takes three values for each sensor to be represented as shown in Figure (3.2).

Figure 3.2: Sensor value estimation with aggregation and sensor fusion.

In the case of post-processing algorithms, which optimizes on the space and the number of bits needed to represent multi-sensor readings, the fusing sensor calculates the average or the mean from the values to be aggregated into a single value. From our example data, we see that both the datasets give the same end result, in this case $\mu = 2.7$ as shown in the output plot of Figure (3.2 a). The design criteria (1),

specifies the that sparse representation is not used by post-processing algorithms. Due to this dynamic features are lost during data aggregation step.

The pre-processing step uses Discrete Wavelet Transform (DWT) on the signal, and may have to recursively apply the decomposition to arrive at a sparse representation. This pre-process is shown in Figure (3.2 b). This step uses the design criterion (1), which specifies the small number of significant coefficients needed to represent the given signal measured. As seen in Figure (3.2 b), each level of decomposition reduces the size of the coefficients. As memory is constrained, we use up to four levels of decomposition with a possible of 26 different representations, as computed by equation (3.2). These uses the design criterion (3) for lossless reconstruction of the original signal. The next step of pre-processing is to find the best

Algorithm 3.1 DWT: Using a cost function for searching the best sparse representation of a signal.

- 1: Mark all the elements on the bottom level
 - 2: Let $j = J$
 - 3: Let $k = 0$
 - 4: Compare the cost v_1 of the element k on level $(j - 1)$ (counting from the left on that level) to the sum v_2 of the cost values of the element $2k$ and the $2k + 1$ on the level j .
 - 5: if $v_1 \leq v_2$, all marks below element k on level $j - 1$ are deleted, and element k is marked.
 - 6: if $v_1 > v_2$, the cost value v_1 of element k is replaced with v_2 $k = k + 1$. If there are more elements on level j (if $k < 2^{j-1} - 1$), go to step 4.
 - 7: $j = j - 1$. If $j > 1$, go to step 3.
 - 8: The marked sparse representation has the lowest possible cost value, having no overlaps.
-

basis. For this we use a vector Basis of the same length as cost values representing the basis. This method uses Algorithm 1. The indexing of the two vector is the same and are enumerated in Figure of (3.2 b). In Figure (3.2 b), we have marked a basis with shaded boxes. This basis is then represented by the vector. With the basis search, which is part of design criteria (4), we can represent the best coefficients for

inter and intra sensor features. It will be noticed that the values are not averages or means of the signal representation. These preserves the actual sensor outputs. It is an important design criterion (2), which calibrates the minimum possible sensitivity of the sensor. The output in Figure (3.2 b), shows the constant estimate of S_3, S_7 which is $Z_C = 2.7$ from equation (3.4).

To represent the variance in four sensors, a basis search is performed which finds coefficients of sensors that match the same columns. In this example, we find $Z_j = 1.6, 0.75$ from equation (3.4), which is the innovation component.

$$Basis = [001010011000000000000000000000000000]$$

$$Correlated\ range = [000010010000000000000000000000000000]$$

3.8 Summary

In this Chapter, we have discussed a distributed framework for correlated multi-sensor measurements and data-centric routing. The framework, uses compressed sensing to reduce the number of required measurements. With the joint sparsity model we can, define the system accuracy in terms of the lowest range, which can be measured by a group of sensors. The sensor fusion algorithms allows us to estimate the physical parameter, which is measured without any inter sensor communications.

The complexity model that is developed describes the encoding of the data. The model is easy for encoding, and grows more complex at the joint decoding level. To able to have enough resources these decoder nodes have more resources for decoding the signals.

Post processing and data aggregation are discussed with cross-layer protocols at the network and the MAC layer, as is their implication to data-centric routing

using DHT. This is then compared with the DCS model. Even though these routing algorithms are power-aware, the model does not scale in terms of accurately estimating the physical parameters at the sensor level, making sensor driven routing more reliable for such data driven applications.

Chapter 4

Problems in Real-time Processing of Digital-Streams

using Compression

*T*emporal features in sensor networks is a sub-problem in stream processing of multi-media data. Most of the work is from the paper ¹. The real-time frame-work was provided by Microsoft Direct-X team and the project funding was from Ardtully Technologies, CA. Section 4.6 shows how quality of audio streams can be preserved using dynamic range control using quantization (codec compression).

4.1 Compressing Data Streams

Compression is a mechanism to eliminate redundancy and we will apply different principles to stream compression for multi-media 16-bit formats.

4.2 Introduction

GPP packet-switched streaming service (PSS) is a standardized packet-based mobile streaming service, which is based on IETF RTSP/SDP standards. PSS can be

¹Iyer, V. and Garimella, R. M. and Srinivas, M. B. Entropy based variable rate compression for low-bandwidth multi-media streams. Proc. First International Conference on Distributed Framework and Applications DFmA, 2008.

implemented over GPRS networks; however these cannot usually guarantee any data rates or delay bounds, but allow sufficient bandwidth for mobile streaming. As these are server based technologies here we concentrate on the content of existing media such as source encoded formats for digital quality using combined bandwidth of all the smart devices available to multiply non-used bandwidth for high-quality streaming. The Source coded media are available in MP3, WMA and other formats. Their sampling rates and other desired parameters are stored in the file in the form of tags. These tags help to sort a given collection based on the bit rate, audio or video before it is trans-coded into the embedded player. The Source formats are like typical headers which identify the type of format and the offsets where the actual streams (Vasanth Iyer, 2005) are located and any type of compression model (Witten *et al.*, 1999.) it uses. As this type of formats is based on off-line storage format we here explore the scalability of this model to sterilizable variable bit stream (VLC) based on the optimal playing capacity of the embedded wireless player. The first pass takes into account the baseline tags such as the sampling rate, and type of media based on the lossy-model. It then calculates the trans-coding parameters such as desirable and the instantaneous R steam rate, CBR or VBR and serializes it into variable packet size with variable bit sizes. Once this is completed the first frame is sent to the player who is receiving the last bit. The initial fullness is complete allowing the player to start playing. The encoder then indexes the serial stream with send times based on the arrival time to the players active play stream (send time is earlier than and takes into account the delivery over head). Once the indexing is complete the second and the following packets to the n^{th} packet request the pre-calculated compressed stream according to the current players as shown in Figure 4.3, decoder's context without ever overrunning the player buffer and hence adapting the upstream source format to an variable bit rate trans-coded streamable

format for network delivery. The initial fullness of the decoder stream's bucket is the fullness of the buffer at the instant before the first bit in the stream is dumped in (in most implementations, this value is 0, but other values are valid as well).

The send time value in the stream packet is an indication of when the payload data inside the packet would be leaked according to the leak rate R . This, of course, has to be no later than the presentation time of the payload (otherwise the data arrives too late to present). Due to the fact that the instantaneous bit rate as determined by presentation time values fluctuates around R , though, when the buffer is not completely full, send time will have to be earlier than the presentation time to ensure that when the buffer does get close to being full, every payload is still leaked before or at its presentation time. Therefore, the send time values should be seen following the leak rate R relatively constantly, even if the presentation time values in the presentation are not. Another way to think of it is that the Send Time values allow for a head start so that the objects will arrive at the destination before their presentation time values, even when the presentation time values are advancing at an instantaneous rate that is higher than R .

4.3 Trans-coding Model

4.3.1 Constraints of Speed Versus Quality

The multi-media bandwidth requirements for High quality Video and audio are determined by the number of frame rates (*fps*) needed per second. The values are dependent as seen in Table 4.1, based on the current speed of the wireless channel. Due to this compatibility some standards have assigned desirable sampling rates which are for best performance and wireless connectivity. In Figure 4.3 various different smart wireless devices form an ad-hoc network which can provide upcoming

Table 4.1: CODEC and FPS requirements

Source	Resolution	Quality	Codec	FPS
Audio	44.1kbps	11,050samp/sec	MP3/FM	20
Video	160X120	Trans-coding	ASF	24
	320X240	Trans-coding	MP4	24
	640X600	Trans-coding	XVid	26
	800X600	Trans-coding	H.263	30

Table 4.2: Bandwidth requirements

Source	Dedicated Bandwidth	CODEC
Audio	96 kbit/s	FM quality
	192 kbit/s	MP3
	1,411 kbit/s	CD-Quality
Video	2.0 Mbit/s	VHS
	8.0 Mbit/s	DVD
	27 Mbit/s	HDTV

streaming technologies. By aggregating all the bandwidth of each device an active device in the network can use multiplex streams by resource sharing off a single remote server.

Tables (4.1 and 4.2) shows popular source formats that allows content to be streamed in a compatible device. These audio and video samples are 8-bit or 16-bit raw values which are available in the existing source format. These are source encoded and are given by

$$Source_{bps} = \frac{\text{Total no.of bytes} \times 16}{Bandwidth} \quad (4.1)$$

4.3.2 Trans-coding Quality

From Table 4.1 and heterogeneous network of emerging wireless standards like Bluetooth, Zigbee and on-demand streaming servers can generate variable bit-rate ac-

cording to the matching hardware capabilities.

$$H(S) = \sum_{i=0}^m P(X_i) \log P X_i \text{bits/sample} \quad (4.2)$$

Equation (4.4) gives the source entropy for a set of m samples were $P(X_i)$ is the probability of each symbol occurring in the source file. In a quantization based codec the number of symbol it can be sensitive to equals the max number of quantization steps (it uses rounding for matching the raw samples). But the actual number of symbols needed to encode may vary according to the content of a file. Equation (4.5) is the lower bound of the minimum number of bits needed to encode the current source.

$$\textbf{Trans-coding rate} \geq H(S) + 1, \text{bit per symbol} \quad (4.3)$$

In equation (4.4) the samples are now referred to as symbols, these are optimally assigned by each coding algorithm such as Huffman or Arithmetic coders. The lower bound is still greater than the entropy of the source as it can never be less in the case of lossless compression

$$2^{\text{samples}} \xrightarrow{\text{Trans-coding}} m^{\text{symbols}}, \text{for } < \text{Low Bandwidth} \quad (4.4)$$

$$2^{\text{samples}} \xrightarrow{\text{Trans-coding}} m^{\text{symbols}}, \text{for } < \text{High Bandwidth} \quad (4.5)$$

Equation (4.4) can now be re-written as

$$\textbf{Trans-coding rate} = \sum P(a_i) n(a_i), \text{bits/symbol} \quad (4.6)$$

where a_i is the number of bits for the codeword (a_i)

$$\text{Trans-coding rate}_{bps} = \frac{\text{No.of encoded bits in stream}}{\text{Bandwidth}} \quad (4.7)$$

To calculate the trans-coding rate which needs to be optimized for the available bandwidth as shown in Table 4.2, we have

$$\text{Trans-coding rate}_{bps} \leq MP4 \leq MP3 \leq CD \quad (4.8)$$

Equation (4.7) is for lossless model where the decoded output is preserved without any distortion. In the case of wireless devices the bandwidth requirements are still lower than many source formats. So to design a transcoder we need to have a lossy model which brings down the total number of distinct symbols to a lossy number k without losing the quality.

$$\text{Trans-coding rate}_{lossy} \leq GPRS \leq ZigBee \quad (4.9)$$

where $samples = n_{max} = R_{lossless} \leq R_{lossy}$ and R is the on-demand rate requested by the decoder.

$$\text{Trans-coding rate}_{lossy} \leq \text{device look ahead buffer} \quad (4.10)$$

Equation (4.9) has to be modeled in the embedded device as it is a hardware limitation. The streaming can be further divided into blocks according to the decoding efficiency available at individual players. The rest of the chapter is divided into describing the popular digital formats and the design of a software codec for existing hardware codec's for smart devices.

4.3.3 Trans-repairing Lossless Algorithm

In the previous definitions we used a constant source symbol count which needs at least 2^n symbols where n = total number of source symbols. From equation (4.9) we get the average trans-coding rate R given in terms of bits/symbol (not bits/second as in the case of bandwidth). Suppose we use a new pre-processing step which defines new codeword by re-pairing for every n symbols. As there are m^n patterns of the n symbols, we need m^n codewords in Huffman coding. We could generate this code by viewing the m^n symbols as letters of the extended alphabet.

$$S_{re-paired} = a_1a_1...a_1,a_1a_1,...a_2,a_1a_1...a_m,a_1a_1...a_2a_1,...a_m \quad (4.11)$$

From a source of $S^{((n))}$. Let us denote the new rate, R , is given by $R = \frac{1}{n}R^{(n)}$. Equation (4.2) can be rewritten as

$$H(S) \leq R \leq H(S) + \frac{1}{n} \quad (4.12)$$

Practically comparing the new trans-coding rate given in equation (4.3) to the source entropy in equation (4.2) we can say that a good coding scheme can limit the rate very close to the source entropy.

$$H(S) = H(S)_{repaired} \quad (4.13)$$

As an example if the normal trans-coding rate given by equation (4.12) has been 1.782 bits/symbol if we re-pair (Witten *et al.*, 1999.) creating a sequence of two symbols a_1, a_2 then the average length would be $\frac{1.782}{2} = 0.89$ bits/second, a redundancy of about 0.045 bits/symbol without any loss of information. This allows better throughput when streaming over a low-bandwidth line. The algorithm has some limitations as it is sensitive to the number of possible sequences m_n codebook size. If the block size is increased the codebook size grows exponentially and makes Huffman coding impractical, but works efficiently if n is small. Latter in the Chapter we will study different formats which have expected ranges and this is normally used more for video based streaming as it has lots of symbols.

4.3.4 Managing Re-pair Dictionary Size

For the proposed algorithm to be efficient it needs to address the dictionary size which is m^n . One choice is to match a appropriate coding algorithm, here the obvious choice is Arithmetic coding as it does not need all the possible combinations of the dictionary to be generated and only the ones it uses. In our example of media files it can be 2^{16} distinct levels which needs 65535 codewords. As this is a general case this pre-processing needs to find the actual number of active levels present which will vary for each and every file. Then the dictionary size will be m^L where L is the active levels.

$$Trans - coding_{bps} = \frac{Dictionary_{size}}{\text{No. of source samples}} \quad (4.14)$$

4.4 Comparison of Current Source Formats

MP3 (Rao, 1996b) standards and WMA format (Microsoft, 2008) allow to source code to the standards needed in Table 4.1. Each of these formats use digitization to convert the samples and store them after encoding the header tags for the codec to select the desired streams. As both the models use a conversion this comes under a lossy-model in terms of data compression. This lossy-model is developed more in depth by the WMA format as it is currently available with RTSP (Schulzrinne, 1998) to be playable as you stream. Analyzing the current digitized files it is seen that most of them can be recompressed by 70%. As the media files have average entropy of $H(S) = 30\%$ of the original file size. This shows that recompressing can be done in a lossless manner. In video some adaptive lossy model can be implemented as the initial frame could be lossless and contain about 30% compressed content and the following can be a delta difference of the initial content which can be trans-coded at even lesser rate of 30%.

4.5 Related Work

Bandwidth standardization has been used extensively in the fields of wireless standards design of NIC and power management. Some of the new devices are compatible with WIFI, ZigBee and GPRS based network services. These are emerging services and affects mostly content management services, and allow their subscriber base to have on-line access to dynamic content. Some of the application developers have provided applications on top of streaming networks, which are very similar to the current approach in this chapter. One other enhancement is DRM (digital rights management) which allows the user to an original copy and copy it many times into specific wireless devices. All the above work does take into consideration the

complete bandwidth QOS, which is needed to make a successful on-line streaming over low bandwidth wireless networks such as GPRS, ZigBee and Sensor networks. The current work uses a content centric trans-coding algorithmic approach, which takes care of quality parameters such as dynamic range, recompression, frames-per-second, and matching characteristics of the end device, which is very specific to the new smart phones and its build-in ability to do power management. Also the same can be scaled to a content delivery engine with multiple subscribers using a lossless model or on-demand lossy model (IETF, 2008).

4.5.1 Reversible Codes

Huffman codes are variable length pre-fix codes and can only be forward decodable. In a sensor network application a data stream is compressed to minimize the number of bits transmitted. Even though this encoding achieves power efficient codes the decoder may not be able to correct errors due to channel noise. As shown earlier, the pre-fix codes are only forward decodable, due to this if some of the bits are corrupted the serial decoder needs to retry the whole transmitted stream again, wasting more battery power. A reversible code will allow decoding in forward and backwards directions allowing to quickly synchronize during an event of an error within the next block. By the virtue of reversible decoding of the bitstream avoids further transmission retries.

Figure 4.1, shows a serial reversible Huffman encoder, where symbols $S_1...S_n$ and their prefix codes $B_1...B_n$ are concatenated with their reversed suffix codeword as shown in the Algorithm 4.1. The bars shown are concatenation of prefix and suffix codes which are denoted by B_1 and B'_1 , and $\oplus$ performs EXOR of bits of the bitstream. The L offset bits are padded to synchronize the prefix and suffix codes

Figure 4.1: Reversible Stream Encoder (adapted from Girod (1999))

in both the directions, which amounts to an overhead given by an upper-bound:

$$L \geq l_{max} = \max_n l(n)$$

Where the decodable stream is delayed by l bits and offset by l padded zeros.

Algorithm 4.1 ReversibleVLCStream(S)

- 1: $B = B(1)|B(2)|\dots|B(N)$
 - 2: $B' = B'(1)|B'(2)|\dots|B'(N)$
 - 3: $C = \left(B(1)B(2)\dots B(N) \underbrace{0|\dots|0}_{L \text{ zeros}} \right) \oplus \left(\underbrace{0|\dots|0}_{L \text{ zeros}} |B'(1)B'(2)|\dots|B'(N) \right)$
Forward Decoding
 - 4: $B(1)B(2)\dots B(N) \underbrace{0|\dots|0}_{L \text{ zeros}} = C \oplus \left(\underbrace{0|\dots|0}_{L \text{ zeros}} |B'(1)B'(2)|\dots|B'(N) \right)$
Backward Decoding
 - 5: $\underbrace{0|\dots|0}_{L \text{ zeros}} B(N)B(N-1)\dots B(1) = C \oplus \left(B'(N)B'(N-1)|\dots|B'(1) \underbrace{0|\dots|0}_{L \text{ zeros}} \right)$
-

During an error, the decoder will synchronize by decoding the bitstream in the reverse direction until it finds a valid codeword which can be decodable with in the next block. Doing so, the decoder achieves high power efficiency by not going all the way back to start of the bitstream header everytime a few bits are corrupted. Figure 4.2 shows the process of decoding a reversible stream. Forward decoding

Figure 4.2: Reversible Stream Decoder (adapted from Girod (1999))

is achieved by Exclusive-or EXOR of the bitstream C by the reversed codewords with offset bits. Similarly, the reverse decoding is accomplished by EXOR of the bitstream C by backward decoding of the

4.5.2 Normalizing to a Standard FPS for Portable Player Applications

In the case of high quality such as real-time video a desired 29.97 *fps* is needed in the current implementation. We will use PCM audio samples only such that the underlying codec has a fixed algorithm and codes generated are for any processor that can handle approximately 30 *fps*. Design of codes (Witten *et al.*, 1999.) has two parts: the first part is the time index and the second is the fixed part, which is the pointer to the bucket of corresponding values as shown in Figure 4.3. Using Golomb code (Gallager, 1978) to generate a range of 1-30 we can calculate the number of bits for 2^k

$$\frac{n}{2k} + (k+1)bits \quad (4.15)$$

For PCM values we have a fixed step quantized 0-255 levels which needs 8 bits. Totaling the variable bit part and the constant part we get from equation (4.15) for

n=30,k=4

$$max = 8.75 + 8 = 16.75bits \quad (4.16)$$

$$min = 1.125 + 8 = 9.175bits \quad (4.17)$$

4.5.3 Lossy Compression for Constraint Portable Player Applications

Most of the codec algorithms use one of the prefix codes like the popular Huffman minimum redundancy codes. These codes perform efficiently for large amounts of binary data which is not applicable in our case. To match the current requirements the codec needs to limit the quantization step making it a fixed step quantized codec. To generate the corresponding 16-bit values for a dynamic range between 1-64 levels, we use 8-bit samples. This gives a selection from 0-256, which is used by the encoder and the decoder who shares the same settings and uses it to translate the value back to 16-bits. A comparison is shown in Figure 4.5 where the dynamic range scale for high data samples are adapted to damped data samples during mixing operation.

$$\text{matching 16bit sample value} = \frac{volume \times \mathbf{8bit\ samples} \times 256}{(VOLUME_{Levels} - 1)} \quad (4.18)$$

As we have already addressed the matching of the input codec we need to provide a way for the decoder to lookup the corresponding 16-bit values. The input samples are limited to the range -128 0 +128 the lookup Table 4.4 needs to be an array of 255 16-bit integers.

$$[volume][\mathbf{8-bit\ sample+128}] \quad (4.19)$$

The offset bits which is the fixed part of the code allows to index from 0 to 255 into the lookup array as defined in integer array equation (4.19) for a fixed volume

Table 4.3: Golomb/Rice Codes generated by the codec output for easy indexing of frames 1-30.

Pilot Frame	2nd Frame	3rd Frame	4th Frame	5th Frame
0	01	100	101	1100
Samples				
$0-00X$	$01-00X$	$100-00X$	$101-00X$	$1100-00X$
$0-01X$	$01-01X$	$100-01X$	$101-01X$	$1100-01X$
$0-10X$	$01-10X$	$100-10X$	$101-10X$	$1100-10X$
$0-11X$	$01-11X$	$100-11X$	$101-11X$	$1100-11X$

....			n-1	30th Frame
....				$1111111111-00X$
....				$1111111111-01X$
....				$1111111111-10X$
....				$1111111111-11X$

level to decode the 16 bit stereo sample. The sample and the corresponding decoded values are tabulated in Table 4.4. In this sample we use a distortion measure $R(D)$ which is the lowest rate at which the output can be encoded while keeping distortion less than or equal to D . Re-writing equation (4.6) for lossy trans-coding we have

$$D = \sum \sum d(x_i, y_j) P(x_i) P(\frac{y_j}{x_i}) \quad (4.20)$$

In our case $P(\frac{y_i}{x_i}) = \frac{1}{512}$ Substituting in equation (4.20)

$$H(X) = \sum \frac{1}{512} \log_2 \frac{1}{512} = 9 \text{ bits/sample} \quad (4.21)$$

This satisfies equation (4.15) and (4.16)

$$H(D) = H(S)_{\text{repaired}} < H(S) \quad (4.22)$$

By again using trans-repairing on lossy model we can optimize to the new distorted entropy, thereby increasing the trans-coding rate to be compatible with constrained bandwidths from an original 15 bits/sample to a mere 9 bit/sample. Further some of the content based media streams can use cost functions which are designed with scalar transformation using **basis** functions.

4.6 Dynamic Range Control Using Fixed Step Quantization

Audio mixing is done in real-time to combine separate output channels to play in a timely and synchronized fashion to a multimedia channel. As the typical audio data is sampled from different source codec's the mixing algorithm needs to evaluate a least common mixing volume to clearly combine the multi-channel effects. This chapter emphasis using up-sampling and down-sampling of the raw values as shown in equations (4.4,4.5) to avoid pops and cracks in mixing. This also examines matching compression codes for effective channel transmission and replaying of the mixed output.

4.6.1 Implementation of CODEC for PCM Stream

Typically the mixing player hardware has limited dynamic range and capabilities to match the output of different sampling rates and codec specifications. Simple addition of the inputs would cause out-of-range errors and serious distortion of the audio quality and undermine the mixer design. The raw data from a codec is uncompressed PCM (Pulse Code Modulation) 8-16 bits, the mixer output is fixed to 16-bit PCM stereo in most of the embedded players as shown in Table 4.3. To

Figure 4.3: Block diagram of 4× variable bit channel mixer.

Figure 4.4: Bit masking for adaptive 8 to 16 bit.

get a better quality audio output all the mixing is done after the completion of the corresponding volume translation using the converted 8-bit values in Table 4.4. If a particular sample has predominantly high values and if it is mixed with a softer sample then the output may not be distinguishable. Therefore to correct this phenomenon the appropriate lower level of the volume is tried for the channel, which has high values till a break even point is reached such that the softer channel has a better audible range in the mix generated. This correction is sometimes called the dynamic range adjustment. Typically the difference of the recorded lowest value to the highest value is taken into account which holds good in the case of two different channels as well. By using this setting and ignoring any other out of range values will not degrade the quality of audio and this setting could be fixed on a codec basis.

4.7 Prototype Results of Step Quantization

This algorithm was implemented using C++ classes into Linux cell phone running at 285 MHz using an arm processor and a DSP audio buffer capable of handling 65535 bytes per stream of CD quality. The low level driver was adapted to play long PCM files (20 seconds as shown in Table 4.4) by using memory management techniques in the real-time player buffer by using the DSP hardware as a file handle and synchronization of end of play was handled by waiting on the DSP hardware handle. The raw values after dynamic quantization are shown in Table 4.4.

4.8 Summary

As these families of codes allow up to 30 frames per second on a low end processor it would work for both audio(which is our test case) but also NTSC video (Kumar *et al.*, 1985) requirements. Most of the hardware vendors for portable processor have a built-in player, which implements buffering streams based on arrival time from an on demand server. This real-time compression model will be able to deliver the desired *fps*. All the source formats available commercially such MPEG,AVI,WMF can be recompressed to nearly 70% of the original size hence a big bandwidth reduction during streaming. Due to heterogeneous sharing of smart resource it is critical to match the device codec with a software decoder model which can handle up and down trans-coding rate at the file level. It also seamlessly enhances the indexing capability, which is needed with the trans-coded file in compressed format using Gloumb codes. Here we have discussed lossless trans-coding and a quality based rate distortion for very low bandwidth and memory requirements using trans-repairing algorithm.

Figure 4.5: PCM 8-bit to 16 bit decoding with variable volume selection [1..64].

Table 4.4: Adjusted PCM values for selected volumes (1 AND 128).

Sample Index	Volume = 128 (Max. codec values)	Volume = 1 (Min codec values)
-128	-32247	-520
-127	-31995	-516
-126	-31744	-512
-125	-31492	-507
-124	-31240	-503
-123	-30988	-499
-123	-30736	-495
-121	-30484	-491
-120	-30232	-487
-119	-29980	-483
-118	-29728	-479
-117	-29476	-475
-116	-29224	-471
-115	-28972	-467
-114	-28720	-463
-7	-1763	-28
-6	-1511	-24
-5	-1259	-20
-4	-1007	-16
-3	-755	-12
-2	-503	-8
-1	-251	-4
0	0	0
+125	32000	507
+126	32256	512
+127	32512	516

Chapter 5

Sub-problems in Quality of Sensor Logs

*B*y ranking the information extracted from sensor logs (as shown in Figure ??) and its features, data mining algorithms can efficiently predict interesting events from static and mobile streams. The content of this chapter is based on a published book chapter¹ from a Machine Learning Text on Sensor Networks. The research methodologies and analysis were part of the Yahoo! Summer School on Information Retrieval (IR), held at Indian Institute of Science, Bangalore, 2011. The study used WEKA(of Waikato, 2008) data mining toolkit introduced by Google© architect member to explores how low-cost sensor streams can help data mining algorithms when ground truth data is too small to be effective in practice. Section 5.10 examines sensor domain temporal features in the context of Forest Fire events. Sections 5.18 describes how correlated temporal features are used in ranking the training samples.

¹Intelligent Sensor Networks: Networks Signal Processing and Machine Learning. Vasanth Iyer and S.S. Iyengar and Niki Pissinou. "Using Event Log Performance and F-measure Attribute Selection", Publisher Taylor & Francis, 2013.

5.1 Supervised Classification of Skewed Sensor Logs

5.1.1 Introduction

During pre-processing event-log (as shown in Figure ??), training samples are sorted by their area of fire damage and classified into high, medium, small and accidental small fire. The pre-processing step allows studying the probability distribution and in our case the samples are highly skewed, giving an estimate that accidental small fires are more likely to be compared to large fires. Statistically we are interested in the many factors that influence accidental small fires; from the training samples, it follows a normal distribution. Graphically describing normal distribution, it takes the form of a bell-shape curve, which is also known as the Gaussian function. This is a first approximation for a real valued random variable, which tend to cluster around a single mean value. The sensor model needs to learn the expected ranges for the baseline attributes being measured, giving better density estimation with increasing samples count. The baseline discrete parameters capture only the sensor ranges, making event prediction function hard to train with a Gaussian density function, without specific temporal understanding of the datasets. The dynamic features present in a sequence of patterns are localized and used to predict events, which otherwise may not be an attributing feature to the static data mining algorithm.

We have studied the spatial features and baseline discrete sensor measurements and all the attributes available that have a high classification error, which can sometime account to 50% of the errors in the case of accidental small fire category. Further, factors that can cause such fires are investigated, wherein we include data pertaining to human specific temporal attributes such as number of visitors and traffic patterns coming into the forest area, thus filtering events with local significance. Temporal attributes is a better estimator, given the type of training samples, which

Figure 5.1: Histograms of empirical samples.

are difficult to calibrate and any approximation may induce false alarms. Relevance based ranking function is suitable to order higher bound sample chosen by domain experts as ideal estimates and still maintaining the desired low false alarm rates. The method of ranking uses the function of sensor precision and event relevance weights, which are then linearly added to represent data from fire activity logs. Section 5.2 and 5.3 provides related work and state of the art of label-less learning vs ground-truth extreme event samples. Section 5.4 defines sensor measurements and fire activity to model the data and algorithm computational complexity.

5.2 Background

(Frank & Asuncion, 2010) provides collection of supervised databases, which are used for the empirical analysis of event prediction algorithms with unsupervised datasets from distributed wireless sensor networks. Sensor networks generates huge amounts of data which needs to be validated for its relevance. Keeping only necessary data helps avoiding high computational overloads due to data redundancies.

Figure 5.2: Bayes net representation.

Calibration of sensors may not always be possible and the data aggregating algorithms need to have domain rules to detect any outliers from the datastream given that other parameters are kept constant. With a dataset of forest fire events for a region, the training algorithm can transform correlated attributes from sensor networks datastream to validate and classify the events and reject the outliers reliably.

The preliminary work models the empirical data with a ranking function without spatial information to predict the likelihood of different events. The concept of Information Retrieval (IR) such as precision and relevance are used in the context of sensor networks, which not only helps us understand the domain topics but increase reliability to the large dataset. The theme of this case study is forest fire and environmental conservation. We study the environmental factors that can be attributed to such events, for example temporal attributes such as humans, peak weather conditions, surface fuel buildups and wind spread factors. The burnt area which is the ground truth is broadly studied with respect to small and large fires. The framework needs to extend queries in topics which are spatially aware, making sensing an essential source of discovering information.

	positive	negative
positive	tp	fn
negative	fp	tn

Table 5.1: Confusion Matrix of Classifier.

5.3 State of the Art

The sensitivity of a sensor network not only depends on the accuracy of individual sensors but also on the spatial distribute collaborative sensing of neighboring sensors. Now we can define the data mining criterion, which is a standard measure to evaluate queries in a sensor networks. **Precision** How close does a features measurement match at every independent value, when measured in time.

Accuracy How redundant is features measurement, when measuring correlated spatially arranged sensors with medium to large datasets.

$$accuracy = \frac{t_p + t_n}{n}$$

$$precision = \frac{t_p}{t_p + f_p}$$

$$F_\beta = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

The first criterion, which is precision cannot be generalized in the case of **WEATHER** dataset, as weather data is not independently identically distributed (i.i.d's), but happens to be highly correlated feature. As most of the generative models like Naive Bayes assumes i.i.d strictly, one needs to find alternative models, which better approximate the observed samples. A better model than Naive Bayes is the network version of this model called Bayes Net, which specifies entities that are dependent

	Precision	Recall	Average	F-score
Algorithm-I	0.5	0.4	0.45	0.444
Algorithm-II	0.7	0.1	0.4	0.175
Algorithm-III	0.02	1.0	0.51	0.0392

Table 5.2: Mean comparison of precision and recall.

and probabilities, and are independent of the observed phenomena. A typical Bayes net is illustrated in Figure 5.2, which identifies entities such as small, large forest fires and its dependencies. The multi-feature that allow us to analyze the general pattern in **WEATHER** dataset and local events is further studied by including temporal dependencies, in the form of human traffic patterns. The enhanced model not only improves the statistical assumptions, it performs much better in error handling, which is one of the major disadvantages of large, low resource energy constrained sensor networks. In this work one of the goals is to better estimate the performance of the underlying algorithms used. To illustrate the need for such a metric we use three algorithms and measures their precision and accuracy as shown in Table 5.2. The average is calculated by combining precision and accuracy. Algorithm III has the highest average even though it has very low precision. To overcome this limitation we adapt a *harmonic mean*, which is also called the F-score (see Appendix). The calculated value of F-score is shown in the final column for all the three algorithms. By using F-score the weights do properly rank the algorithm III as the lowest due to its low precision value.

Adding sensor data to the manual forest fire event logs allows study of automated correlated real-time information, which when properly trained allows estimation and classification of future sensor outputs from the same geographical region. Event logs contain spatial information such as GPS and the area of the fire-damaged region but lacks correlated information, which could lead to better estimation of the fire event

under study. Moreover, the spatial information is collected manually and takes time to classify, while the sensor measurements are measured in real-time and can be approximated by a machine learning algorithms to classify events and notify if a fire alarm condition has been reached.

Ranking functions allows filtering unrelated data and present only relevant information to the user’s query. In the Information Retrieval (IR) domain, there are many efficient ways to rank the relevance of a document in a collection given a user query. Similarly, we can rank the order of occurrence of events given the precision and relevance of the prior fire probabilities given a geographic area hypothesis. The terms precision and relevance are inversely proportional. In problems where the recorded evidence is small and rare one can instead use a precision scale to rank the evidence and make others correlated events as relevant, balancing out the summed weights used in ranking of events. We position the precision weights higher whenever fire events occur along with higher alarm conditions. The high alarm condition is always true for precision ranking and holds for accidental small, medium and large fires equally, unlike relevance, which accounts for majority of fires in our case only accidental small fire and small fires which constitute 80% of the total fires. The precision ranking uses ground truth such as actual fire evidence, further eliminating any possibility of errors due to outliers and weak evidences. These ground truths are well-established for naturally occurring phenomenon, which occur rarely and leave significant evidence of the burnt areas in meter square of the forest. A relevance ranking weight does an exhaustive search of all the prior fire events, which makes it unique to the natural habitat of the particular geographical area, where the concept is machine learnt.

5.4 Sensor Measurement and Fire Activity

The samples of the fire events are sorted in terms of Burnt Area (BA) in hectors(ha), the frequency distribution versus BA as shown in Figure 5.1. We can model the BA in terms of a function as shown below, which allows us to study the behavior of fire activities over time and predict newer events reliably.

$$BA = f(x) \quad (5.1)$$

5.4.1 Sorting by BA in ha.

The histogram shows that the BA is skewed with large number of small fires and very few large fires, making likelihood of small fires more predictable. Fires are further classified into four categories, which enables us to select the performance of the system in terms of precision and relevance. Figure 5.1(b) shows the new classes without any other weather attribute that could cause the events.

5.4.2 Sorting by BA with Temporal Attributes

$$\begin{aligned} V_{\text{train}}(\text{FireEvent}_{\text{Day of week}}) \leftarrow \\ \hat{V}(\text{FireEvent}_{\text{Day of week}})_{\text{Temporal Variable}} + \\ \hat{V}(\text{FireEvent}_{\text{Day of week}})_{\text{Correlated measurements}} \end{aligned} \quad (5.2)$$

$$\begin{aligned} \text{Temporal Variables} = \text{Month of the year} + \\ \text{Day of the week} \end{aligned} \quad (5.3)$$

$$\begin{aligned} \text{Correlated measurement} = & \textit{temperature} + \\ & \textit{humidity} + \textit{wind} + \textit{rain} \end{aligned} \quad (5.4)$$

$$\text{Classfires} = \{\text{accidental; small, medium, large}\} \quad (5.5)$$

5.4.3 Estimating Training Values with Sample Data

Sample datasets are based on 517 rule sets from UCI forest fire repository. The equation representing the target function of burnt area from the empirical data is given in equation (5.1). In our case the hypothesis to be maximized in terms of the temporal attributes are shown in equation (5.2) for a four class classification, as shown in equation (5.5) and according to equation (5.1). The assumption here is that the training set D is an unbiased representation to learn the concept c and can estimate the inputs x_i . The previously defined dependent variable **Fire Location**, which is used to estimate the independent correlated measurements and their relation to the temporal attributes are given in equations (5.2), (5.3) and (5.4). The target concepts are present in the training samples and we like to see the influence of adding sensor measurements to accurately learn the concepts of the human induced accidental small fires versus the more natural accruing types of medium and large fires. For the sake of clarity of machine learning domain we convert the correlated sensor data to ordinal (Witten & Frank, 2005) types, as illustrated below.

$$\textit{temperature} = \{\textit{cool}; \textit{mild}; \textit{hot}\} \quad (5.6)$$

$$\textit{humidity} = \{\textit{normal}; \textit{high}\} \quad (5.7)$$

$$\textit{wind} = \{\textit{true}; \textit{false}\} \quad (5.8)$$

The model estimation of the target function with weights w_1, w_2 as shown allows us to minimize the training error, where x_1, x_2 are temporal and correlated measurements.

$$\hat{V} = w_1x_1 + w_2x_2 \quad (5.9)$$

The learning algorithm needs to define the best fit for the given hypothesis and adjust the weights to minimize the error and miss-classifications.

$$E \equiv \sum (V_{train}(FireEvent) - \hat{V}(FireEvent))^2 \quad (5.10)$$

5.4.4 Algorithm Complexity

Search space consists of all the possible patterns of the features, given our data model, $4 * 3 * 3 = 36$ possibilities for each rule when using attributes temperature, humidity and wind. As there are 517 rules from the collected dataset instances, each rule can have 36 possibilities. The complete search space will have 36^{517} different possibilities. To minimize the complexity of search space, we can further cut down on the sample instances by using spatial clustering and removing any redundancies in similar features. Given the $\langle X, Y \rangle$ positions, we can cluster the possible fires types into different groups such as accidental small fires and others, which have medium and larger burnt area as large fires. As measuring ambient phenomena are correlated we expect clustering would be best suited. Let us take five clusters to contain all the samples. Now the search space is reduced to $36^5 = 60 * 10^6$ possible rule sets. As these methods are used with pre-processing to reduce redundancies in the model these are very practical optimizations of machine learning algorithms. To judge the effectiveness of the model and the classification effectiveness we initially

rely on real-valued numeric model such as (Iyer *et al.*, 2010c) to estimate the errors. In contrast to the previous approach we use ordinal values as defined in equations (5.6-5.8) to build a tree classifier and further reduce errors.

5.5 Alarm Ranking Function for Accidental Small Fires

An event data log such as forest fires, may be incomplete, How do we infer knowledge from the datasets?. Relevance factor of fire event concept can be defined as all fire events that are tagged in the log, as we are interested in reporting fire occurrences. The inverse concept precision is seen from the histogram plot in Figure 5.1(a) whwre there are very few large fires. When reporting on major fire events the highest ranked samples are retrieved, which has a higher rank leading to a precision concept close to 1.

5.5.1 Ranking Function

To design of a good ranking function the relevance and precision of the events have to be balanced in way to express a summable numerical quantity, which signifies the importance of the new sample and how reliable the prior probabilities were. As the ranking function is evaluated for a given query, we use define the query criteria for retrieving accidental small fires and large wild fires for our collection.

$$\text{Precision} = \frac{\text{number of relevant forest fire events retrieved}}{\text{number of forest fires retrieved in query}}$$

$$\text{Relevance} = \frac{\text{number of relevant forest fire events retrieved}}{\text{number of relevant forest fires classified}}$$

Figure 5.3: Twice the precision Vs relevance. Fire activity vs Burnt Area, yields a logarithmic relation.

5.5.2 F-measure

The weighted harmonic mean of precision and relevance, the traditional F-measure or balanced F-score is because recall and precision are evenly weighted. The general formula for non-negative real β is based on van Rijsbergen's effectiveness E given by

$$\text{Alarm}_{\text{rank}} = 1 - \frac{1}{\alpha \frac{1}{p} + (1 - \alpha) \frac{1}{r}}$$

We further look into accidental small fires, as they are very probabilistic and any conceptual link to the attributes may lead to rank the idea of precision and relevance of the collection. α – the alarm weight are calculated based on the reliability of the ground truth, higher precision weightage is given due to prior assumption that large and medium fires are not accidental small fires. The precision weights factor for small fires can then be evaluated given α by $(1 - \alpha)$.

5.6 Performance of fire topics classification using temporal ranking

$$BA_{Days\% < 50 \text{ ha}} = \alpha + \beta \ln(\mathcal{NF}) \quad (5.11)$$

where $\alpha = 2.895$ and $\beta = 1.265$, which is the variance of the BA data versus fire activity, showing logarithmic $O(\lg(BA))$ complexity as shown in Figure 5.3(a). Topics in the training samples collection are classified into two categories, which are accidental small fires and large wild fires. To evaluate the performance of the two we use weighted precision versus relevance to estimate the ranking information $F@0.5$ for the above equation. The F-scores are calculated and weighted for high reliability by using $F@0.5$, which is twice the precision compared to its equivalent relevance scale. Reliability and precision are proportionally weighted, while relevance is inversely proportional. The performance scores shows that query evaluation for accidental small fires is 1.7 times higher when compared to queries for large wild fires or from the same collection. The plot in Figure 5.3(b) shows queries for accidental and small fires for the temporal attribute days of the week, where the accidental small fire F-scores have much higher values.

5.6.1 Ranking Accidental Small Fires

Accidental small wild fires are possible all through the year, making is a viable application for automated sensor measurements. The measurements such as temperature, humidity and wind gust are automated, while temporal attributes such as human traffic, day of the week are used to study the small fire events. Following the peaks of the plot suggests that, high-alarm in Figure 5.4 during weekends is followed by only one high alarm day during the normal week. The weightage of the ranking

Figure 5.4: Fire alarm days ranked using $F@0.5$.

Other Test	Yellow Region	Blue Region	Temporal	FWI
Alarm	H=Hit	H=Hit	H=311	H=69
No Alarm	M=Miss	M=Miss	M=111	M=3
False Alarm	F=False Alarm	Z=Null hypothesis	F=8	F=307

Table 5.3: Performance of ranking with false alarm rates.

suggests small fires are caused due to temporal attributes such as human traffic and vehicular routes more than any observed correlated sensor measurements.

5.7 Alarm Ranking Function for Large Fires

In the previous section we used burnt area and data from sensor networks to classify forest fires into four classes. In this section we will instead use domain knowledge to precisely predict fires by using fire weather indexes {low; moderate; high; very high} as shown in Table 5.3. FWI is calculated using ISI and BUI, where ISI represents the Initial Spread Index and BUI represent the Build Up Index. These indicate

FWI TYPES	Measure(normalized)
Low (AF)	0-8
Medium (SF)	8-13
Highg (MF)	13-32
Very High (LF)	$32 > FWI < 80$

Table 5.4: FWI classes.

fire behavior and represent rate of fire spread, fuel consumption and fire intensity respectively. All FWI indexes are significantly correlated with the number of fires and the burned area, especially when $BA > 100$ is the area burnt by large fires. The average FWI index variation during the year is shown in Figure 5.5(a) and Figure 5.5(b). It increases during the month of May and peaks in August, September and starts reducing in the month of October. The equation below shows a numerical representation of Burnt area (BA), which is mean daily burned area per month and the mean daily number of fire events per month (NF).

$$BA_{FWI} > \mathbf{50\ ha} = (BUI) + (ISI)^x \quad (5.12)$$

where Initial Spread Index (ISI), Buildup Index(BUI) are calculated from the environment for a given fuel type. The FWI index is highly correlated with the number of fires and the burnt area. The plot in Figure 5.5, shows the correlated region when $FWI > 33$ (very high) in the case of large fires.

5.8 Performance of Fire Topics Classification using FWI Ranking

Since large fires can damage more than 50ha in total amounts for majority of the burnt area(ha), it is a high priority to avoid large fire incidents and help forest conservation. As they are hard to detect and has a varying threshold it is also a cause of false alarms (Krishnamachari & Iyengar, 2004) in an automated system. Plotting all the correlated FWI components, which relate to fire activity, plot from Figure 5.5(a) shows that peak months have a gradual increase of FWI index and are also correlated with large fire incidents. The area of high correlation is shown

Figure 5.5: Lower-bound range of FWI = (blue) and higher-bound range FWI > (yellow), shows all large fires (red) fall into the yellow region.

$$\begin{pmatrix} & \text{Decision made} \\ & \text{LargeFire} & \text{SmallFire} \\ \text{LargeFire} & 0 & 1000 \\ \text{SmallFire} & 1 & 0 \end{pmatrix}$$

Table 5.5: Cost function matrix for misclassification

in yellow with lowest false alarm for a given FWI threshold. The lower-bound conditions for fire activity are shown in blue, which has higher false alarm due to valid area above the yellow region.

The temporal correlation for large fires using the FWI F-score is plotted in Figure 5.5(b). It shows that large fires are invariant to temporal changes and performance is better than small fires for a given precision and recall measure.

5.8.1 Misclassification and cost of false alarms

Testing the performance of algorithm error rates is simulated, which allows us to find the sensitiveness of the system to false positives. False positives have more significance with higher bound values such as large fires, which are very rare and hard to classify. A hit may be defined in terms of precision to avoid false alarms

Burnt Area(hectors)	AUG	MON	TEMP	HUMIDITY	WINDY	PRIOR PROB	PREDICTOR VAR
>1ha	► 0.34	► 0.14	► 0.46	► 0.17	► 0.42	► 0.47	► 57%
> 1ha <= 10ha	0.39	0.15	0.35	0.13	0.38	0.33	25.0%
> 10ha <= 50ha	0.30	0.14	0.43	0.19	0.50	0.13	17.0%
> 50ha	0.33	0.08	0.16	0.08	0.37	0.04	0.02%

Table 5.6: Posteriors probabilities for background WEATHER dataset data for the peak month August.

(Krishnamachari & Iyengar, 2004). From the cost matrix above, we show that when using FWI the alarms are very precise, while using temporal the alarms are more accurate which includes large amounts of false positives.

5.9 Machine Learning Algorithms

Probabilistic algorithms when used with density estimation and class classification yields the lowest number of error. This allows us to provide a baseline analysis of the system attributes being used.

5.9.1 Naive Bayes

One can use Naive Bayes (Duda *et al.*, 2000), which by design presumes the class densities a priori, which have been determined and are accurate. The model calculates the class conditional probabilities of the input feature vectors. To understand the underlying skewed structure of the dataset, we further create thresholds for accidental small fires compared to medium and large fires as shown in Table 5.7. So we have the four possible values for the target variable as given in equation (5.5).

5.9.2 User Query

To validate the model let us predict the fire activity outcome of a peak summer month, from the dataset (Cortez & Morais, 2010). We see that August has significant

FIRE TYPES	RECORDED
Accidental (AF)	247
Small (SF)	175
Medium (MF)	71
Large (LF)	24

Table 5.7: Target variable occurrences.

FIRE TYPE	MONTH=AUG
Accidental	0.004
Small	0.002
Medium	0.001
Large	0.00004

Table 5.8: Likelihood of fires for the month of August.

number of reported fires compared to other months. Estimating the probabilities of fire events given the attribute values for the class

$$\begin{aligned} ? = & \{Month = August; Day = Monday\} \\ & \{Temperature = Cool; Humidity = High; Wind = True\} \end{aligned}$$

5.8 shows the estimated class conditional densities for the independent variables temperature, humidity and wind conditions, which are calculated using temporal attributes **month** for the dataset. The datasets are further explored using two temporal variables, which are **month** and the **day of the week** as shown in Table 5.8 and Table 5.9. The temporal variables introduced in the dataset helps us to gain the insight of users' dependencies with fire prediction model.

$$g_i(x) = P(\omega_i | \mathbf{x}) = \frac{p(\mathbf{x} | \omega_i) P(\omega_i)}{\sum_{i=0}^{i=4} p(x | \omega_j) P(\omega_j)} \quad (5.13)$$

Substituting the corresponding highlighted values from Table 5.6 through Table 5.9 in the above equation (5.13), we get the posterior probability of accidental small

DAYS	ACCIDENTAL	SMALL	MEDIUM	LARGE
MON	► 35	► 27	► 10	► 2
TUE	28	21	11	4
WED	22	24	5	3
THU	30	21	9	1
FRI	42	31	12	0
SAT	42	24	11	7
SUN	48	27	13	7
TOTAL	► 247	► 175	► 71	► 24

Table 5.9: Posteriors probabilities for temporal feature day of the week.

fire.

$$fire_{accidental}^{\hat{}} = \frac{0.0007547}{0.003565} = 57\% \quad (5.14)$$

$$fire_{small}^{\hat{}} = \frac{0.000333}{0.003565} = 25\% \quad (5.15)$$

$$fire_{medium}^{\hat{}} = \frac{0.000223}{0.003565} = 0.17\% \quad (5.16)$$

$$fire_{large}^{\hat{}} = \frac{0.000000287}{0.003565} = 0.02\% \quad (5.17)$$

The posterior probabilities for the month of August for the data collected in Portugal (Cortez & Morais, 2010) are high as is - the likelihood of accidental small fires. By cross-validating the known fact that in summer the likelihood of wild fires is high than Bayes rule can classify the dataset for accidental and small fires with high accuracy. We use a simulation framework in the following sections to further prove our initial hypothesis from the datasets. It is shown that the training time for Naive Bayes scales linearly in both the number of instances and number of attributes.

5.10 Tree Classifier

In this section, we will focus on the domain rules, which are applicable to the learning system. Tree classifiers lend itself to use ML rules (Mitchell, 1997) when

searching the hypothesis by further branching on specific attributes. The design of such a classifier needs to sort the weights or entropies (Witten & Frank, 2005) of the attributes, which is the basis of its classification effectiveness.

ID3 is a popular tree classifier algorithm, to implement ID3 as illustrated in Figure 5.6 (a) and Table 5.10 showing with the sensor attributes. Let (S) be a collection of samples then using the tree algorithm, which uses entropy to split its levels is given by

$$Entropy(S) = \sum_{i=0}^{i=c} p(i) \log_2 p(i) \quad (5.18)$$

Let us assume a collection (S) has 517 samples (Cortez & Morais, 2010) with 248, 246, 11 and 12 of **accidental, small, medium, large** fires respectively. The total entropy calculated from equation (18) is given by

$$\begin{aligned} Entropy(S) &= \frac{248}{517} \log_2 \frac{248}{517} + \\ &\quad \frac{246}{517} \log_2 \frac{246}{517} + \\ &\quad \frac{11}{517} \log_2 \frac{11}{517} + \\ &\quad \frac{12}{517} \log_2 \frac{12}{517} = \\ &\quad \quad \quad 1.23 \end{aligned}$$

5.10.1 Attribute Selection

ID3 uses a statistical property called information gain to select the best attribute. The gain measures how well the attribute separates training targeted examples, when classifying them into fire events. The measure of purity that we will use is called information and is measured in units called bits. It represents the expected amount of information that would be needed to specify whether a new instance should be classified accidental, small, medium or large fires, given that the example

reached that node. The gain of an attribute is defined by and illustrated in Table 5.10. Using the calculated attribute for information gain we show that **temp** attribute is used before the **wind** attribute to split the tree after the tree root.

$$Gain(S, A) = Entropy(S) - \sum_{i=0}^{i=c} \frac{S_v}{|S|} Entropy(S_v)$$

$$\begin{aligned} Entropy(S_{Hot}) &= \frac{9}{36} \log_2 \frac{9}{36} + \\ &\quad \frac{23}{36} \log_2 \frac{23}{36} + \\ &\quad \frac{3}{36} \log_2 \frac{3}{36} + \\ &\quad \frac{1}{36} \log_2 \frac{1}{36} = \\ &\quad 1.282 \end{aligned}$$

$$\begin{aligned} Entropy(S_{Medium}) &= \frac{23}{96} \log_2 \frac{23}{96} + \\ &\quad \frac{65}{96} \log_2 \frac{65}{96} + \\ &\quad \frac{3}{96} \log_2 \frac{3}{96} + \\ &\quad \frac{5}{96} \log_2 \frac{5}{96} = \\ &\quad 1.175 \end{aligned}$$

$$\begin{aligned}
 Entropy(S_{Cool}) &= \frac{117}{269} \log_2 \frac{117}{269} + \\
 &\quad \frac{146}{269} \log_2 \frac{146}{269} + \\
 &\quad \frac{2}{269} \log_2 \frac{2}{269} + \\
 &\quad \frac{4}{269} \log_2 \frac{4}{269} = \\
 &\quad 1.05
 \end{aligned}$$

$$\begin{aligned}
 Entropy(temp) &= \frac{43}{517} * 1.282 + \\
 &\quad \frac{139}{517} * 1.175 + \\
 &\quad \frac{335}{517} * 1.05 = \\
 &\quad 1.08
 \end{aligned}$$

$$Gain(S, temp) = 1.23 - 1.08 = 0.192$$

$$\begin{aligned}
 Entropy(S_{HIGH}) &= \frac{162}{249} \log_2 \frac{162}{249} + \\
 &\quad \frac{72}{249} \log_2 \frac{72}{249} + \\
 &\quad \frac{8}{249} \log_2 \frac{8}{249} + \\
 &\quad \frac{7}{249} \log_2 \frac{7}{249} = \\
 &\quad 1.1952
 \end{aligned}$$

(a) Tree classifier.

(b) Bayes Network.

Figure 5.6: Weka algorithm toolkit.

$$\begin{aligned}
 Entropy(S_{LOW}) &= \frac{68}{133} \log_2 \frac{68}{133} + \\
 &\quad \frac{59}{133} \log_2 \frac{59}{133} + \\
 &\quad \frac{2}{133} \log_2 \frac{2}{133} + \\
 &\quad \frac{4}{133} \log_2 \frac{4}{133} = \\
 &\quad 1.24
 \end{aligned}$$

$$\begin{aligned}
 Entropy(wind) &= \frac{361}{517} * 1.1952 + \\
 &\quad \frac{156}{517} * 1.24 = \\
 &\quad 1.20
 \end{aligned}$$

$$Gain(S, wind) = 1.23 - 1.20 = 0.025$$

The internal tree representation for m attributes from n samples will have a complexity of $O(\lg n)$, with increasing inputs, given by parameter n , the height of the tree will not grow linearly as in the case of Naive Bayes. On the other hand

Month	Temp	Wind
Not shown	info: 1.08	info: 1.20
Not shown	gain: 1.23-1.08 = 0.192	gain: 1.23-1.08 = 0.025

Table 5.10: Gain ratio calculation for tree using entropy.

complexity of building a tree will be $O(mn \lg n)$.

5.11 Simulation

Open-source workbench called WEKA (of Waikato, 2008) is a useful tool to quantify and validate results, which can be duplicated. WEKA can handle numeric attributes well, so we use the same values for the WEATHER dataset data from the UCI (Frank & Asuncion, 2010) repository datasets. The class variable has to be a nominal one, to allow WEKA (of Waikato, 2008), we convert all fire types to "0" or "1". Where "0" is of accidental small fire and "1" is for large fires making it a two class classifier, the results are shown as confusion matrix in Table 5.20, Table 5.16 and Table 5.18. Naive Bayes correctly classifies accidental and small fires(209 out of 247) were as the J48 Tree classifier does far more, 219 out of 247 and SVM with high precision (235 out of 247).

As WEKA uses kappa (of Waikato, 2008) statistics internally for evaluating the training sets, a standard score of $> 60\%$ means training set is correlated, using J48 simulation, we get 53.56% just below and when using SVM, we get 0.68 above the correlated index. The comparison on results shows that J48 tree classifier does better than Naive Bayes by 25% and the corresponding SVM does 35% overall showing least bias of the three models. Therefore using sensor network measurements accidental and small fires can be predicted with high precision using SVM classifier.

WEKA Stats	Results	Summary
Correctly Classified Instances	267	51.64%
Incorrectly Classified Instances	250	48.35%
Kappa statistic	0.1371	
Mean absolute error	0.3022	
Root mean squared error	0.3902	
Relative absolute error	94.86%	
Root relative squared error	97.84%	
Total Number of Instances	517	

Table 5.11: Evaluation on training set for Naive Bayes.

WEKA Stats	Results	Summary
Correctly Classified Instances	373	72.14%
Incorrectly Classified Instances	144	27.85%
Kappa statistic	0.5356	
Mean absolute error	0.1938	
Root mean squared error	0.3113	
Relative absolute error	60.83%	
Root relative squared error	78.04%	
Total Number of Instances	517	

Table 5.12: Evaluation on training set for J48 tree classifier.

5.11.1 Simulation Analysis

WEKA attribute statistics and its effective correlation score. Table 5.11, Table 5.12 and Table 5.13 show kappa and other comparison statistics for Naive Bayes, J48 tree and Support Vector Machine classifiers respectively, for small fires. The experiment is repeated with FWI, which are shown for J48 Tree classifier in Tables 5.23.

5.11.2 Error Analysis

Equation (5.11) specifies the regression model error and its following confusion matrix from the simulation scores are shown in Table 5.20, Table 5.16, Table 5.18 and 5.24, upper bound of small fire(AF+SF) has over 90% precision in SVM, 80% accuracy for J48-Tree and 61% overall for Naive Bayes. The corresponding baseline

WEKA Stats	Results	Summary
Correctly Classified Instances	421	81.43%
Incorrectly Classified Instances	96	18.56%
Kappa statistic	0.6893	
Mean absolute error	0.0928	
Root mean squared error	0.3047	
Relative absolute error	29.14%	
Root relative squared error	76.40%	
Total Number of Instances	517	

Table 5.13: Evaluation on training set for SVM linear classifier.

	LF	MF	SF	AF
LF	0	1	7	16
MF	0	5	12	54
SF	0	7	53	115
AF	0	0	38	209

Table 5.14: Confusion matrix for Naive Bayes using training set.

performances including all fires categories is 82% for SVM, 72.1% for J48-Tree and Naive Bayes is 51.64%, which is due bias towards small fires and only SVM by design is the least biased.

When FWI classification (Cortez & Morais, 2010) is as given in equation (5.12) for large fires prediction, is is more precise with better precision as shown in the confusion matrix Table 5.21. The percentage of correctly classified is $> 95\%$, making it reliable with few false alarms.

LF	MF	SF	AF	
LF	7	0	7	10
MF	0	29	15	27
SF	1	7	118	49
AF	0	5	23	219

Table 5.15: Confusion matrix on training set for J48 Tree classifier.

	LF	MF	SF	AF
LF	0	2	8	14
MF	2	7	2	40
SF	3	12	71	89
AF	4	15	59	168

Table 5.16: Confusion matrix on testing set for J48 Tree classifier.

	LF	MF	SF	AF
LF	7	0	5	12
MF	0	31	15	25
SF	0	1	148	26
AF	0	0	12	235

Table 5.17: Confusion matrix on training set for SVM linear classifier.

	LF	MF	SF	AF
LF	0	0	7	17
MF	0	1	17	53
SF	0	3	42	130
AF	0	4	51	192

Table 5.18: Confusion matrix on testing set using SVM.

	LF	MF	SF	AF
LF	0	0	5	19
MF	0	4	5	62
SF	0	4	30	141
AF	0	0	23	224

Table 5.19: Confusion matrix on training set using Bayes network.

	LF	MF	SF	AF
LF	0	0	5	19
MF	0	2	8	61
SF	0	4	23	148
AF	0	2	28	217

Table 5.20: Confusion matrix on testing set using Bayes network.

	VERY HIGH	HIGH
VERY HIGH	371	0
HIGH	17	0

Table 5.21: Confusion matrix on training set for $FWI > 32$.

Figure 5.7: Plot of F-score for each model and its performance with WEATHER dataset training data.

5.12 Correlation of Attributes

From statistical point of view if the attributes have similar values then it creates high bias creating what is called over-fitting error during learning. In our case **temp** and **humidly** may have similar values and needs to be avoided and substituted with a suitable attribute. To pre-process and analyze, we use all the available sensor measurements in the dataset and WEKA provides the attribute selection as illustrated in Table 5.10.

We use the attribute selection wizard of WEKA to find out the best match. The analysis shows that the precision is highly dependent on Month(100%), Day(10%) and Wind(0%). In a two class classification the quantitative data is biased towards to small fires and SVM does better due to better generalization. In the qualitative analysis which based on the frequency of attributes, WEKA picks Month, which is a temporal type. The F-score of small fires are higher, when using temporal attributes as shown in Figure 5.7(b), which is also true for WEKA predictions.

Number of folds (%)	No.	Attribute
10(100 %)	1	month
1(10 %)	2	day
0(0 %)	3	temp
0(0 %)	4	RH
0(0 %)	5	wind

Table 5.22: Attribute selection 10 fold cross-validation (stratified)

WEKA Stats	Results	Summary
Correctly Classified Instances	371	95.61%
Incorrectly Classified Instances	17	4.38%
Kappa statistic	463.64	
Mean absolute error	0.0838	
Root mean squared error	0.2047	
Relative absolute error	97.5159 %	
Root relative squared error	99.9935 %	
Total Number of Instances	388	

Table 5.23: Evaluation $FWI > 32$ on training set for J48 tree classifier.

5.13 Better Model for Weather Data

In the error analysis we show that the true error performance depends on the learning algorithms and how the weights are learnt. As all the data mining algorithms assumes samples to be i.i.d it cannot perform well with highly correlated weather data. In the domain of weather data where features tend to be highly correlated, careful feature selection and a better model is needed to address the shortcomings of earlier models. Bayes Network allows us to develop a better model to define classes and events which are dependent and independent of each other. This model allows to distinguish the overlapping statics using correlated and similar features having the same range and values. The basic Bayes net performance well, while cross-validating tests show 88% precision as shown in Table 5.24 and has similar accuracy of other models proving its discriminative power for the underlying data.

Experiments	Model Performance	F@0.5 Measure		Confusion Matrix	
	Overall	Small Fire	Large Fire	Small Fire	Large Fire
Ranking-Temporal	bias	0.22 $\oplus$	0.17	-	-
Ranking-FWI	bias	0.1	0.9 $\ominus$	-	-
Weka(NB)-Temporal	generative	0.6 $\oplus$	0.1	0.6	0.3
Weka(Bayes Net)-Temporal	better Model	0.88 $\oplus$	-	0.88	0.0
Weka(J48)-Temporal	decision tree	0.7 $\oplus$	0.5	0.75	0.3
Weka SVM-Temporal	generalized	0.87 $\oplus$	0.7	0.9 $\ominus$	0.3
Weka(J48)-FWI	invariant	-	0.95 $\ominus$	0.1	0.9 $\ominus$

Table 5.24: F-measure performance for all tests compared with WEKA.

5.14 Summary

We use a query log as show in Figure ?? approach of search engines and standard statistical ranking measure to do a base line analysis and use data from inexpensive sensors to validate gainst probabilistic ML algorithms. From the Table 5.24, we use the notation for best estimate for precision- $\oplus$ and accuracy- $\ominus$ as denoted, the F-measures matches the expected WEKA simulation statistics. FWI is able to boost the weak performance of the raw data from sensors, which are typically hard to calibrate. In the qualitative analysis of small fires, the performance of a generalized classifier such as SVM is preferred. Similarly the qualitative performance for large fires is done by careful attribute selection and we show that an invariant attribute selection such as ($FWI > 32$) yields high classification precision. The use of a better model like Bayes network in the case of weather data helps to increase the performance.

Chapter 6

Data Cleaning Trees

$\mathcal{T}$ rees are used to model low to high complex hypothesis from the datastreams. It is based on Occam Razor's principle (Duda *et al.*, 2000) which says that when there are two hypothesis the simpler one is better. In this Chapter we describe how to build stream ensembles that allow better prediction of the quality of noisy sensor streams by building data-cleaning trees. Section 6.3 lists the descriptive attributes that are used to learn the concepts. Section 6.6 uses ensemble approach to further increase the accuracy of the base classifiers and provides a quality score for comparison.

6.1 Introduction

In the previous chapters, we have seen how data-cleaning methods are used to pre-process missing and faulty values from sensor streams. In this chapter we will discuss further how static and mobile streams which have spatial and temporal properties can be classified using Data Cleaning Trees. The primary objective of this classification is to label noisy streams based on Quality of Data (QoD). In the previous data-cleaning algorithm we have used basic precision and accuracy of the sensor data and compared it with kappa value to provide a function to compare noisy streams with pre-processed streams by learning hidden patterns. In the same

way we extend label-less Data Cleaning Trees (DC-Tree) learning using the entropy based impurity (Duda *et al.*, 2000) function. The impurity function of the DC-Tree can be defined as

$$\alpha \cdot size + \sum_{\text{leaf nodes}} i(N), \quad (6.1)$$

$$Size = \text{Number of leafs in the tree.}, \quad (6.2)$$

$$i(N) = \text{Impurity of nodes.} \quad (6.3)$$

$$Entropy(t) = - \sum_0^{c-1} p(i|t) \log_2 p(i|t), \quad (6.4)$$

$$Gini(t) = 1 - \sum_0^{c-1} [p(i|t)]^2, \quad (6.5)$$

$$\text{Classification error}(t) = 1 - \max_i [p(i|t)], \quad (6.6)$$

$$\text{where } c \text{ is the number of classes} \quad (6.7)$$

In label-less learning as there is no class information its impurity is based on clusters. So the impurity in the leaves is based on the maximum separating distance metrics between clusters when splitting.

$$\alpha \cdot size + \sum_{\text{leaf nodes}} i(CN), \quad (6.8)$$

$$Size = \text{Number of clusters in the tree.}, \quad (6.9)$$

$$i(CN) = \text{Impurity of cluster nodes.} \quad (6.10)$$

Figure 6.1: A clustering tree of the first-order.

Figure 6.2: Data cleaning for labelless streams.

Figure 6.3: Multi-target and input combinations.

6.2 Label-less Classification Problem Definition

In this classification problem of label-less streams we concentrate on the task of predicting multiple (Kocev *et al.*, 2007),(Vens *et al.*, 2011) attributes. The input attributes x_i take the form of a sensor data matrix representing attributes in the columns where $x_i = (x_{i1}, \dots, x_{ik})$ is a vector of k input attributes and $y_i = (y_{i1}, \dots, y_{it})$ is a vector of t target attributes as labeled in the feature induction tree in Figure 6.3.

Most of the current supervised classification needs labels and predicts according to one target value. Some of the extensions of existing learning techniques such as decision trees use *random forest* which in turn creates many trees using one attribute at a time. By using multiple trees the model grows to a huge size making it hard to maintain. The DC-tree algorithm (Kocev *et al.*, 2007) and PurestTree() algorithm 6.2 uses multiple targets as labeled clusters in Figure 6.4, which needs only a single tree and are easy to model.

6.3 Concept Learning

We investigate using the standard WEATHER dataset to learn the temporal variations which are descriptive attributes. There are four basic types of learning in data mining (Tan *et al.*, 2006) applications.

- ★ Classification Learning
- ★ Clustering
- ★ Association Learning
- ★ Regression or numeric prediction

Regardless of the type of learning these constitutes concept and the output accuracy produced by a learning scheme that uses descriptive attribute for concept descrip-

tion. As we are concerned with label-less (no class label) learning the last two from the above list are more relevant.

In practical data mining applications the following types of attributes are used:

- ★ Categorical
- ★ Enumerated or discrete
- ★ Numeric or continuous
- ★ Boolean

Machine learning can use a wide variety of information about the attributes (Witten & Frank, 2005). Some of these are:

- ★ Number of dimensions
- ★ Circular ordering
- ★ Temporal context
- ★ Partial ordering

These are often called meta-data, data about data. We like to learn more on the temporal aspects, which are listed above using the **WEATHER** dataset. Detecting and retaining all features is a prime consideration in categorizing good quality streams.

From the example **WEATHER** dataset, we enumerate a few temporal attributes. The problem is to how to classify a new day for the given temporal values. Numeric prediction (Witten & Frank, 2005) is a variant of classification learning in which the outcome is a numeric value rather than a category. The data-cleaning applications of noisy streams problem is one example. Another, as shown in matrix representation in Figure 7.1, is a version of the stream data in which what is to be predicted is not the target label Y from the sensor data matrix but rather the QoD of the real-time stream quality. With numeric prediction problems, as with other machine learning situations, the predicted value for new instances is often of less interest than the hidden features which are learned, expressed in terms of what the important attributes are and how they relate to the QoD-label outcome.

Figure 6.4: Cluster tree of 4 leafs for the WEATHER dataset.

The **WEATHER** problem is a tiny dataset that we will use repeatedly to illustrate DC-Tree cleaning methods. In general, instances in a dataset are characterized by the values of features, or attributes. In this case there are four attributes: outlook, temperature, humidity, and windy. The descriptive attributes, which form the nodes of the DC-Tree are shown in Figure 6.4.

- ★ Outlook
- ★ Wind

The rest of the numeric attributes are used in clustering as shown in Figure 6.4.

- ★ Temperature
- ★ Humidity

Figure 6.5: Cluster tree with pruning enabled.

Evaluating error from 7.1 and substituting values for variables in equations 6.4, we have the generalization error (RMSE) before and after pruning.

Before pruning = 2.6751

After pruning = 6.7338

In the data-cleaning metric the classification error for the tree in Figure 6.4 is 2.6751. It has a higher QoD(Iyer *et al.*, 2013.) label than the pruned version with lesser number of nodes as shown in Figure 6.5. A Machine Learning framework, which allows combining outputs of many classifiers is known as ensemble learning. It is one of the most popular standards to improve robustness and accuracy of the base classifier. Some of the well know methods are bagging and randomforest, which uses ensemble learning. The framework also supports Stacking, Voting and Random Subspaces, which combines multiple classifier outputs. Data driven analysis shows that the empirical evidence combining classifiers improves accuracy if the classifiers are "*independent*", i.e. its learning function is different inside each classifier. In this chapter, we will look into various ensemble methods.

6.4 Related Work

Many studies have been conducted to improve sampling techniques. Similarly, there are many studies that use ensembles to improve classification accuracy. The most important assumption is sampling rule of i.i.d's independence and this also holds for independency of the classifier's accuracy. This implies that the attribute partitioning methods perform better as compared to data partitioning. Most of the current approaches emphasis data partition in terms of cross-validation and reduction in training error. The feature selection problem plays an important role in domain targeted research. It can be viewed as a multi-criterion optimization problem, which uses heuristic, search or optimization techniques. (Kohavi & John, 1997) discusses a feature relevance model and show how various feature subset selection problems are solved. The research survey includes two models: (i) wrapper model and (ii) filter model. The wrapper model uses the feature subset selection as feature evaluation and is optimal. In the second case of filter model the induction algorithm uses information content such as inter-class distance and, statistical dependence, that are dependent on particular algorithms. Some of the research work in filter model is reviewed by (Blum & Langley, 1997)

6.4.1 Justification of Ensemble Method

Due to small training sets and data values affected by faulty sensor measurements, let us assume that the base classifier has an error rate ($\epsilon = 0.35$). The ensemble classifier uses the methods stated below to combine the output of individual classifiers. Let us assume that the base classifiers use the same training set and induction algorithms, then the ensemble will also misclassify the same test sample as done by the base classifier. If we assume the base classifiers are independent, which means the errors

are uncorrelated. Then the ensemble will make an error for a test sample only if more than half of the base classifiers predict incorrectly. Now, substitute this assumption into a binomial probabilities as given in equation.

$$\binom{n}{r} p^r (1-p)^{n-r} \quad (6.11)$$

Substituting the error rates ($\epsilon = 0.35$) of the ensemble classifier in our case is

$$e_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06 \quad (6.12)$$

We take a classification approach to data-cleaning algorithm learning to reduce errors and estimation of sensor data. A general form for data-cleaning algorithm in terms of training with samples is shown in the equation, where the estimation is dependent on bias and variance of the observed data and the noise present in the measurement domain.

$$Training_{error} = Bias_{observed} + Variance_{algorithm} + Noise_{measurement} \quad (6.13)$$

6.4.2 Factors that Affect Bias

Generalization is an important aspect of the classification approach as our model was too complicated then the decision boundary would too. This would lead to a solution, which would be too premature because the rules need to perform well with observed samples which are still to be seen. Another approach is to obtain a better estimate of the parameters measured by ground sensors by getting more training samples. Due to constraints in power and the capacity of sensor lifetime it

not feasible to obtain large training samples in continuous time. A design choice would be to simply and use a less complex boundary. This could bring down the performance of the classifier but would mean better performance on novel patterns.

6.4.3 Complexity and Algorithms

Generally there are two types of algorithms supervised learning and unsupervised learning. A learning algorithm will work on any training instance which has attributes and a target label. The stronger the assumption of the attribute assumption the higher the bias. In general, the stronger the assumption of the type of decision boundary the higher the bias. For e.g. the Nearest-neighbor classifier is more sensitive to the same training set compared to a decision tree classifier.

6.4.4 Bias and Variance for Continuous Values

Sensor data is normally continuous and can consist of many reading which are high precision floating-point values. In this case we have a continuous value sensor measurement with noise, we can estimate with n samples in a set $\mathcal{D}$ generated by $\mathcal{F}(x)$. The regression function estimate $g(x; \mathcal{D})$ for a given training set $\mathcal{D}$. Due to random variation in time-series data, for some window length L the approximation will be excellent while for other data sets of the same size the approximation will be poor.

$$Var(X) = \mathcal{E}_{\mathcal{D}}[(X - \mu)^2] \quad (6.14)$$

$$Var(X) = \mathcal{E}_{\mathcal{D}}[X^2 - 2X\mathcal{E}_{\mathcal{D}}[X] + (\mathcal{E}_{\mathcal{D}}[X])^2] \quad (6.15)$$

$$= \mathcal{E}_{\mathcal{D}}[X^2] - 2\mathcal{E}_{\mathcal{D}}[X]\mathcal{E}[X] + (\mathcal{E}_{\mathcal{D}}[X])^2 \quad (6.16)$$

$$= \mathcal{E}_{\mathcal{D}}[X^2] - (\mathcal{E}_{\mathcal{D}}[X])^2 \quad (6.17)$$

Substituting in the regression estimate

$$= \underbrace{\mathcal{E}([g(x; \mathcal{E})]) - \mathcal{F}([x])^2}_{bias} + \underbrace{\mathcal{E}_{\mathcal{E}}[(g(x; \mathcal{D}) - \mathcal{E}_{\mathcal{D}}[g(x; \mathcal{D})])^2]}_{variance} \quad (6.18)$$

The equation consists of two additive terms, the first squared expression is the bias, which when minimized allows better estimate of the function $\mathcal{F}$ from $\mathcal{D}$. The second term signifies the variance term (Duda *et al.*, 2000), which can be attributed to a statistical property and if kept to a low value should affect the Root Mean Square (RMS) error in a minimal way with an existing low bias or no bias term.

It can be shown from the above equation that if we assume a large number of samples $n \rightarrow \infty$ for training the model then bias term will be reduced and only dependent upon the noise factor and variance would be reduced to zero or to a desired quality labels as described in the next sections.

6.5 Different Types of Noise in the Domain

As noise is uncorrelated with the measure sensor values (X) data-cleaning algorithms need to pre-process missing values, anomalies and faulty sensors. The other type of noise is based on supervised target labels (Y), in the context of data-cleaning it can be attributed to quality labels or if the class is known it can be assigned a class-label. In sensor networks the target label estimate can be due to the two following reasons. (i) One due to clustering and the border nodes may be hard to classify due to ties in the segmentation algorithms. (ii) In the second case, which is due to low quality temporal training data sets, which do not capture the events and otherwise has no correlation with the ambient phenomenon being tracked.

In pattern classification there are two major types of learning algorithms (i)

	Domain	Task	N	Attr	Class	Regr
E_1	WEATHER	Sensor	4	2	Small	✓
E_2	FOREST FIRES	Sensor	12	4	Medium	-
E_3	Mobility	Mobile Sensor	4	2	Large	✓

Table 6.1: Datasets for static and mobile sensor streams.

Unsupervised and (ii) Supervised learning methods. One is based on well-known prior probabilities of the estimated random variables and the other is based on a metrics which uses a similarity score. We further define a pattern classification task when situations call for label-less streams, where one needs to estimate the labels which cannot be directly observed. Such variable are classified as hidden or latent variables, which represent useful information in classifying real-time sensor streams in their respective domains. Labeling or estimating a target variable is same as hidden variables.

6.6 Ensemble of Data Cleaning Trees

Data-cleaning using ensemble model (Kocev *et al.*, 2007),(Vens *et al.*, 2011) is a novel approach to feature extraction and concept learning. From equation (5.12) it can be seen that the ensemble model outperforms the base classifier’s performance if the individual classifiers errors are lower than 0.5. To see the effectiveness of the above assumption we use the UCI (Frank & Asuncion, 2010) FOREST FIRE dataset using the DC-Tree implementing. The results are compared in the Table 6.2 below.

6.6.1 Prototype Function

The classification performance of the DC-Tree is compared with 3-datasets small, medium and large. The quality of the DC-Tree, which uses clustering can be done in

	Training			Testing		
	DCTBag	DCTRF	S-Target	DCTBag	DCTRF	S-Target
WEATHER	2.6751	2.6751	2.7839	11.782	11.782	9.8453
FOREST FIRES	0.8538	0.8591	0.996	1.130	1.1033	1.3162
FOREST FIRES(regr)	32.541	32.544	63.59	35.988	35.79	65.080
TRAJECTORIES	0.1622					

Table 6.2: Comparison of ensemble learning for multi and single target DC-trees.

two ways. We apply the principle that clustering can accomplish classification and regression in unsupervised stream learning. In the case of classification for known categories cross-validation and predictive accuracy is used. The prototype function uses a distance based function, which generalizes label-less learning over regression and classification methods in supervised learning. Samples can be members of more than one cluster or all clusters compared to the traditional k-means which always uses a simple mean criteria from the centroid. Often the function is to minimize the sum of the squared error (SSE) given by,

$$SSE(C_1, C_2, \dots, C_k) = \sum_{j=1}^k \sum_{i=1}^m w_{ij}^p \text{dist}(x_i, c_j)^2 \quad (6.19)$$

In the case of numeric attributes, regression and cross-validation is used to decrease the relative error, which is also the mean square error of prediction divided by the mean squared error of a default hypothesis always predicting the mean. The prototype function is shown in the equation below.

$$RE = \sum_{i=1}^n \frac{d(e_i, \bar{e}_i)^2}{d(e_i, p)^2} \quad (6.20)$$

Figure 6.6: Boot strapping used in bagging during model training.

6.6.2 Splitting Criteria

The quality of DC-Tree is based on how many clusters are formed from equations (5.8-5.10), hence a criteria which maximizes the intra-cluster distance is chosen such as a F-Test. Stopping criteria are based on statistical significance test, such as χ^2 test (Clark & Niblett, 1989). F-Test is very sensitive to the the underlying variance in the datasets and used as a test of quality of variances. Given a criteria to split a cluster into two, will significantly reduce the intra-cluster variance. This process can be repeated until the dataset is found to be homogenous at the leafs, this is described in the PurestTree() Algorithm 6.2,(Koccev *et al.*, 2007) line 4. A similar F-score criterion is used in the previous Chapter 5 to rank the importance of the attributes in the dataset.

$$F = \frac{SS/(n-1)}{(SS_L + SS_R)/(n-2)} \quad (6.21)$$

6.6.3 Bagging

One of the examples of ensemble is bagging (Quinlan, 1996) as shown in Figure 6.6, which uses a bootstrap replicates of the training set and using each of these

replicates to construct one classifier. The sampling uses sample with replacement, and N is sufficiently large then $1 - \frac{1}{e} \simeq 63\%$ of the original training dataset are actually used. The variance show in equation (5.8) can be high in some datasets due to over-fitting and by using only a percentage of the samples the variance discussed earlier can be reduced. Bagging does not focus on any particular instance of the training dataset and is therefore avoids over-fitting when trained with sensors data with noise. When extending DC-Tree with bagging, the base classifier uses $DCT(E)$ as given in Algorithm 1 to vote on the final test sample classification.

6.6.4 Random Forests

A powerful ensemble method is a random forest (Breiman, 2001), (Kocev *et al.*, 2007), (Vens *et al.*, 2011) and ensemble of trees. From the given set of attributes a random subset is analyzed at each node and the best feature is selected and the whole tree is constructed. If the number of attributes that are retained is given by a function f of the total number of input attributes x , then by setting $f(x) = x$, we obtain bagging procedure. It can be theoretically proven that the upper bound for generalization error of random forest converges to the error equation 6.22 if the number of trees is sufficiently large. When applying random forest, the *BestTest* is used, which selects attributes without bias as shown in Algorithm 2.

$$Generalization \leq \frac{\bar{\rho}(1 - s^2)}{s^2} \quad (6.22)$$

Where ρ is the average correlation (Tan *et al.*, 2006) among trees and s is a quality that measures the "strength" of the tree classifiers. From equation (5.9) it is intuitive that, when the trees get more correlated the generalization error increases. When applied to sensor datasets, which tend to use clustering to save energy and

hence the spatial values collected tend to be correlated. To avoid such specific domain problems one can use multi-objective decision trees, where the descriptive attributes (higher level features) are used to predict target attributes (sensor measured) accurately.

6.7 Results

Due to noisy sensor data and data aggregation functions which do not taken into account of the many requirements of the domain, many of the temporal features are lost during aggregation. In this experiment we show that ensemble learning using bagging and random forest performs well with noisy low-cost sensor samples when using DC-Trees. The formation of DC-Trees allows to preserve the temporal variations rather than averaging them over time. This suggest as discussed in (Kocev *et al.*, 2007),(Vens *et al.*, 2011) that the non-trivial relations that may exist between target attributes are preserved as in the case of correlated sensor measurements.

Algorithm 6.1 DataCleaningTree DCT(E)

```

1:  $(t^*, h^*, \mathcal{P}^*) = \text{PurestTree}(E)$ 
2: if  $t^* \neq \text{none}$  then
3:   for each  $E_k \in \mathcal{P}^*$  do
4:      $tree_k = \text{DCT}(E_k)$ 
5:   return node( $T^*, \bigcup_k tree_k$ )
6:   end for
7: else
8:   return leaf(Prototype(E))
9: end if
```

Algorithm 6.2 PurestTree(E)

```
1:  $(t^*, h^*, \mathcal{P}^*) = (none, 0, \theta)$ 
2: for each possible test  $t$  do
3:    $(\mathcal{P} = \text{partition induced by } t \text{ on } E)$ 
4:    $h = Var(E) - \sum_{E_k \in \mathcal{P}} \frac{|E_k|}{|E|} Var(E_k)$ 
5:   if  $(h > h^*) \wedge Acceptable(t, (P))$  then
6:      $(t^*, h^*, \mathcal{P}) = (t, h, \mathcal{P})$ 
7:   end if
8: return  $(t^*, h^*, \mathcal{P}^*)$ 
9: end for
```

Part III

Mobility

Chapter 7

QoD using SPOTLESS

SPOTLESS is a framework for data-cleaning and quality labeling of data streams. In this chapter we will study the meta-data, which describes data about data. The work uses Qualnet event driven simulator (QualNet, 2008) for wireless and BER protocol analysis. Most of the chapter is from a recent paper¹ presented at the workshop on Quality of Information. Section 7.3 explains the QoD evaluation process. Section 7.3 describes missing data in spatial streams in section 7.5.1 elucidates BER error for the wireless 802.15.4 MAC layer.

7.1 Introduction

Unsupervised data collected from sensors applications are transmitted in short streams, and with the advent of Data-mining and Machine Learning algorithms, *Stream Learning* has justified its need in improved preprocess and provides measurements with a context-based stream QoD metric. Stream Learning is a fast growing concept primarily due to increasing availability of location-aware functionality in mobile applications, which generate large scale machine generated data in everyday from wireless, mobile GPS and sensors. Unlike databases, these categories of infor-

¹Vasanth Iyer and S. Sitharama Iyengar and Niki Pissinou and Shaolei Ren. SPOTLESS: Similarity Patterns Of Trajectories in Label-less Sensor Streams. IEEE PERCOM IQ2S:Workshop on Information Quality and Quality of Service for Pervasive Computing, 2013.

mation in streams are real-time, constantly changing and probabilistic in nature. In a survey in 2009, Gartner published that the data will grow over 650%, which may lead to an information overload for existing computation and bandwidth standards. When estimating and predicting labels, current methods need to bridge the gap between the contrasting small amounts of static training samples, vast amounts of label-less mobile sensor data.

Due to inherently high redundancy in sensor data streams and the cost of transmission, many data stream values are averaged to a single estimate of the measure parameter. In a traditional deployment, for example, to increase the sensitivity of data-streams, typically the sampling period is increased or the density of the number of sensors deployed in a cluster is increased in the region of query interest. The important aspect of all these techniques is to increase the data accuracy and dissemination of the sensor streams to provide enough samples of the evolving patterns of interest. The current gap is to reliably deliver spatial and temporal variations of the time series data without data packets being corrupted due to multi-hop routing in wireless outages or high measurements seen due to a few bad sensors.

With the availability of smart phones with inexpensive GPS in them, samples can be accurately tagged with location information using Data MULES when collecting data from static ground sensors. The trajectory sampling used by Data MULES has more variance in the parameter's temporal range compared to when using WSN data aggregation (e.g. very few training samples to estimate). The learning is reduced by repeatedly sampling the same GPS tagged trajectories over time.

The streams' data can be organized into two categories (1) is obtained by spatial sampling periodically from static sensors. The other is obtained by sampling temporal variations by collecting data from ground sensors using a mobile Data MULES, which have applications in the area of road traffic-related probes and Mil-

itary Unmanned Autonomous Ground Vehicles (UAGV). The first category uses Independently Identically Distributed (i.i.d.) local sampling and the second category uses a time-stamp with global GPS coordinates to tag the location and position for the complete end-to-end path. In our simulation, we assume the Random Way-point (QualNet, 2008) mobility model to collect data. Here, we are not interested in the instantaneous sampling, as in the case of other mobile applications, but in studying temporal overlaps in data streams, which can be further classified by using stream QoD-based labels. When training samples are few during the learning of the overlapping ensemble of trajectory streams, we can re-sample those trajectory segments which have sparse intervals to get a finer bound of the temporal sensitivity of the sensor network deployed.

7.2 Motivation

Datacleaning algorithms consist of pre-processing raw data and filling in missing values, smooth noisy data, identifying and removing outliers, and resolving inconsistencies. We can broadly classify the data transformation issues of Datacleaning algorithms into sensor data normalization and network packet aggregations.

7.2.1 Prior and Related Work

As a technical contribution to this chapter, we are interested in generating stream QoD based labels of unsupervised data, SENSORML (OGCNET, 2008) streams and event data (Iyer *et al.*, 2013) present in logs of remote sensing and mobile applications. Most of the current techniques rely on local sampling at the sensor nodes to provide the precision and accuracy for the sensor network application. With the advent of mobile sensors and inexpensive GPS receivers, innovative sampling meth-

ods and mobile applications are available to provide locations-specific and temporal granularity in stream learning. Our normalization method uses recent advances in matrix completion (Goldberg & Zhu, 2010) to improve the precision of the spatially sampled data streams with mixed real values, and uses a classification (Iyer *et al.*, 2013),(Goldberg & Zhu, 2010) approach to learn application-level features. In addition to static datacleaning mobile wireless (Chen *et al.*, 2010) trajectory-based streams are studied with respect to fading and shadowing effects at the receiver. Our method uses data sampled from mobile trajectories which have spatial and temporal values to learn the hidden latent features. These feature set are used in context with stream QoD labels to further classify real-time streams.

7.3 Stream QoD Evaluation

The underlying SPOTLESS (Similarity Patterns Of Trajectories in Label-lEss Sensor Streams) processing assumes a full-rank (Goldberg & Zhu, 2010) matrix removing redundant columns present in square or skinny matrices, where k is the maximum number of observed features (columns) and we assume ($k=1,2,3$) to separate the observed features from the missing ones. Our cluster classification method uses a rule based on combining fully observed local attribute k weights, by which it learns the missing features by using gradient-decent methods.

7.3.1 Precision of Data Stream Using Spatial Features Learning

The SPOTLESS framework uses some of the data cleaning methods, as described below:

Definition 7.3.1 *Selecting k sensors from m possible measurements reduces the error of estimation. The brute force method uses $\binom{m}{k}$ permutations, which is computa-*

tionally exhaustive, when k and m are large. We select a minimum set k , which reduces the training error. The computation can be efficiently accomplished by matrix factorization, which takes $O(m^3)$ operations and is invariant to the number of attributes in the dataset.

Measurements taken from distributed sensor networks may contain faulty measurements. The data-cleaning algorithm needs to compute the accuracy of the estimated measurement from all the observed sensor patterns. This correlated pattern matching the training set makes it possible to predict the missing sample's estimate even when some sensor samples are erroneous or faulty.

Problem Definition

We define the problem of distributed sensor network sensor selection in the following way. The sensor measurement for a set of I placements with J attributes is needed. A non-faulty sensor can be represented as (i, j, x_{ij}) denoting the i.i.d. sample, where $i \in 1, \dots, I, j \in 1, \dots, J, x_{ij} \in \chi \in O$. We assume that the subscript ij belongs to the same set of the i.i.d. sample dataset. In a typical sample dataset, there will be a finite set of observed i.i.d. samples from non-faulty sensors, T , which are used for training. All the non-faulty observed samples are in (i, j) as O . In a typical WSN cluster $O \ll |I| \cdot |J|$, we are interested in the lower-bound of the measured range. In this data-cleaning framework, we use a Machine Learning based algorithm on sensor streams data to minimize the root mean square (RMS) error, which is defined as:

$$RMSE(T) = \sqrt{\frac{1}{|T|} \sum_{(i,j) \in T} (\hat{x}_{ij} - x_{ij})^2} \quad (7.1)$$

The sensor data values are organized as (i, j) in matrix X , as shown in Figure 7.1(a), where the missing or above range values are in $(i, j) \notin O$, as shown in Figure 7.1(b).

In distributed sensor networks, the training samples (non-faulty) and the faulty samples are part of the same dataset collected in time.

7.3.2 Classifying of Data Stream Using Temporal Relevance

Definition 7.3.2 *Select t trajectories from d possible datasets to reduce the error of label estimation. The brute force method uses $\binom{d}{t}$ permutations, which is computationally exhaustive, when t, d are large. We select a minimum set t probes heuristically, which reduces the transductive training error. The computation can be efficiently accomplished by a random tree classifier, which takes $O(t^2 \times d)$ operations.*

7.4 Numerical Modeling of Sensor Data Matrix

The observed values which are i.i.d. samples, can be represented as a system of equations, as described in Figure 7.1 (a). Its corresponding factorization matrix is derived in Figure 7.3, describing the training and the faulty sets (random mask). The missing value estimation is accomplished by using the least mean square error calculation of the observed non-faulty sensor stream. The approximating matrix steps are as follows:

The goal of matrix factorization is to represent a larger matrix, such as in Figure 7.1(a), with two approximately smaller one, which have standard forms, such as Cholesky (Goldberg & Zhu, 2010), QR (Goldberg & Zhu, 2010) and SVD (Goldberg & Zhu, 2010) (Singular Value Decomposition has higher precision). The matrix coefficients, which are real valued constants, pose serious numerical stability issues, such as bias, while estimating the coefficients. Several techniques like regularization, use a damping factor to represent larger coefficients. Here, we show a simple technique to factorize a sensor stream representation (Iyer *et al.*, 2013),(Goldberg

& Zhu, 2010),(OGCNET, 2008). Figure 7.1(a) shows a general matrix consisting of all feature values. Figure 7.1(b) has some missing spatial features denoted by a mask of the same matrix in Figure 7.1(a).

7.4.1 Matrix Completion

Definition 7.4.1 *Let $X_1, \dots, X_n \in \mathbb{R}^d$ be feature vectors associated with n measurements. Let $X = [x_1 \dots x_n]$ be a $d \times n$ feature matrix whose columns are the sensor measurements. Let there be t binary classification range query tasks, $y_1 \dots y_n \in \{-1, 1\}^t$ be the label vector, and $Y = [y_1 \dots y_n]$ be the $t \times n$ label matrix. Due to the wrong calibration of sensors, entries in X or Y can be missing at random. Let Ω_x be the index set of observed labels in Ω_y . Our main goal is to predict the missing labels $y_{i,j}$ for $(i,j) \notin \Omega_y$. This is unsupervised, and without labels we can incorporate transductive learning to find the labels. We like to decrease the training error (RMS)(Iyer et al., 2013) of each label to the observed labels and de-noise the observed features in X .*

To optimally represent the matrix X , which has many missing spatial features, we use $M \in \mathbb{R}^{I \times K}$ and $S \in \mathbb{R}^{K \times J}$, as shown in Figure 7.3, which are denoted as M -Measurement and S -Spatial of the k -known features of the original data matrix in Figure 7.1(a). Let m_{ik} denote the elements of the approximate matrix observed feature measurements of M , and s_{kj} denote the elements of matrix S , representing

the i.i.d. samples of the spatial deployed neighboring sensors.

$$\begin{aligned}
\hat{x}_{ij} &= \sum_{k=1}^K m_{ik} m_{kj} = m_i^T m_j \\
e_{ij} &= x_{ij} - \hat{x}_{ij} \quad \text{for } (i, j) \in \mathfrak{R} \\
e'_{ij} &= \frac{1}{2} e_{ij}^2 \\
\Sigma \text{Squared Error} &= \sum_{(i,j) \in \mathfrak{R}} e_{ij}^2 \\
RMSE &= \frac{SSE}{|\mathfrak{R}|} \\
(M^*, S^*) &= \arg \min RMSE
\end{aligned}$$

$\hat{x}_{ij}$ denotes how close the i^{th} feature of the j^{th} sensor stream measurement is as shown in general matrix equation, with random mask, as shown in Figure 7.1. e_{ij} denotes the training error on the (i,j)-th observed measurement, and SSE denotes the sum of the squared training errors.

7.5 Regression and Data Cleaning

In mobile wireless sensor networks (mWSN in matrix in Figure 7.1), the data samples are gathered using spatial clustering and mobile trajectories to sample temporal values. We will define the reconstruction of sensor streams for both the cases, as described by the dataflow chart in Figure 7.2.

7.5.1 Error 1: Missing Data

STEP 1: Use model assumptions of sensor selection from Definition 1, and the matrix factorization technique to increase precision, as described in Definition 3, when applying data cleaning Step 1, which computes matrix coefficient weights of the sensor stream (Iyer *et al.*, 2013), (Grosanic *et al.*, 2012), (OGCNET, 2008) with

$$\begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{21} & a_{2n} \\ a_{31} & a_{31} & a_{3n} \\ a_{41} & a_{41} & a_{4n} \\ a_{m1} & a_{m1} & a_{mn} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_m \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_m \end{pmatrix}$$

(a) **Attributes** **Estimate** **Observed**

$$\begin{pmatrix} a_{11} & - & a_{1n} \\ a_{21} & a_{21} & a_{2n} \\ a_{31} & a_{31} & a_{3n} \\ - & a_{41} & a_{4n} \\ a_{m1} & a_{m1} & - \end{pmatrix} \begin{pmatrix} X_{\Omega_x} \\ X_{\Omega_x} \\ X_{\Omega_x} \\ X_{\Omega_x} \\ X_{\Omega_x} \end{pmatrix} = \hat{X}$$

(b) **Missing Values**

$$\begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{21} & a_{2n} \\ a_{31} & a_{31} & a_{3n} \\ a_{41} & a_{41} & a_{4n} \\ a_{m1} & a_{m1} & a_{mn} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_m \end{pmatrix} = \begin{pmatrix} Y_{\Omega_y} \\ - \\ Y_{\Omega_y} \\ - \\ Y_{\Omega_y} \end{pmatrix}$$

(c) **Unknown Labels**

Figure 7.1: Empirical evaluation of the accuracy in the proposed hypothesis.

Figure 7.2: The prototype of mobile probe sensor dataset being validated with SensorML's static ground truth estimates.

$$\begin{aligned}
\begin{matrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_m \end{matrix} x_{lbound} &= \begin{pmatrix} 19.9 & 38.8 & 45.08 \\ - & 29.25 & - \\ 19.4 & 38.6 & 48.76 \\ 30.0 & 40.5 & 50.76 \\ 19.8 & 39.07 & - \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_m \end{pmatrix} = y \\
M &= \begin{pmatrix} 3.15612279 & 3.32055619 & 2.8046053 \\ 1.94200791 & 3.25275995 & 1.79754018 \\ 3.36247006 & 3.13477718 & 3.75306667 \\ 2.72765219 & 5.74597238 & 1.14535816 \\ 3.21062777 & 3.28008688 & 2.86255375 \end{pmatrix} \\
S &= \begin{pmatrix} 1.56913487 & 4.46313597 & 0.04110127 \\ 4.28057593 & 4.21979554 & 4.00696943 \\ 4.58024309 & 5.86156674 & 3.98526762 \end{pmatrix} \\
\hat{x}_{k=3} &\approx \begin{pmatrix} 19.88774894 & 38.76005916 & 45.096574 \\ 17.63866341 & 29.24158276 & 35.12481645 \\ 19.42136163 & 42.65985059 & 48.73261097 \\ 29.97218582 & 40.51216606 & 50.73826944 \\ 19.79503629 & 39.05479727 & 45.33994652 \end{pmatrix} \\
\hat{x}_{k=2} &\approx \begin{pmatrix} 19.89157822 & 38.76705248 & 45.08909335 \\ 15.40147753 & 29.24164018 & 34.16729664 \\ 19.4175203 & 42.75810799 & 48.73479405 \\ 29.97080536 & 40.50774752 & 50.74237403 \\ 19.79697602 & 39.05335721 & 45.32668706 \end{pmatrix} \\
\hat{x}_{k=1} &\approx \begin{pmatrix} 21.40909576 & 37.47090348 & 45.44734371 \\ 16.70096795 & 29.23058335 & 35.45290465 \\ 22.33654593 & 39.09415727 & 47.41613992 \\ 24.31880099 & 42.56356526 & 51.62408162 \\ 21.7031853 & 37.98562867 & 46.07163858 \end{pmatrix}
\end{aligned}$$

Figure 7.3: QR Factorization of a sample static measurement matrix X_{lbound} and its covariance matrices M and S is shown for $k = 3$ sensors. The estimated values of $\hat{x}$ are shown for $k=3,2,1$ with decreasing precision.

missing attributes' values. The transformed sensor data matrix has approximate coefficients for the unknown prior attributes, which gives the best approximation using the observed feature measurements. Figure 7.3 shows varying estimates of the original sensor data matrix when learning from features $k=1,2$ and 3 . The higher the number of observed features, the better will be the performance of the learning function to approximate the missing values.

Best Fit	Temp.	Humidity	Light	Kappa
Spatial Sampling	0%	0%	0%	17.78462
Error after cleaning (ω)	(1-0.95)=5%	(1-0.95)=5%	0=100%	13.0278
Real error $\frac{Kappa_{exact}}{Kappa}$	-	-	-	$(1 - 0.73) = 27.0\%$

Table 7.1: Missing features (Kappa metric)

Best Estimate	Temp.	Humidity	Light
Missing Data	16.5%	12.9%	66%
Transductive Label Error	(1-0.95%)=5%		

Table 7.2: Missing labels (unsupervised label-less learning).

7.5.2 Error 2: Network Aggregation

STEP 2: When dealing with streams from mobile trajectory issues, which are related to fading and shadowing, the data cleaning algorithm needs to account for errors induced by high BER due to multi-hop routing. The Data MULE needs to compensate for packet-level MAC errors and provide a minimum required stream QoD to help label and classify the training samples in Step 2. When simulating the various aspects of our mobility model, the percentage of probes are varied, and the communications between the ground sensors are simulated with different network topologies. Wireless simulation of mobile routing allows one to study how Waypoint mobility models and the node’s crosslayer BER losses vary in large distributed sensing mobile applications.

7.5.3 Error 3: Transductive Label Learning

STEP 3: The feature induction-based learning method does not use training labels; instead the training sample’s attributes which best fit the data distribution are induced. Unsupervised feature extraction using random attributes selection used in transductive label learning makes it possible to learn from hidden features (higher order) not obvious in the distribution. The task-based learning approach used by

Data MULES, does not use labeled training data from the spatial streams but rather uses unlabeled samples and induces features, which reduce the overall training error. Data MULE's can be used in many applications, which do not need any communication infrastructure, such as a base station. Two such examples, are roadside weather monitoring using mobile sensors to alert the driver, and post-safety measures. These stream data values sampled are close to the actual ground truth. For example the road condition seen by the driver could be a real-time update of the local observation affecting safety, such as dense fog, as well as slippery and wet road surfaces. For the stream to be assigned a safety label, the observed samples must have contextual parameter relevance from the normal ranking order, or we call the data stream plausible.

7.5.4 Datacleaning and Mobility

The main research we undertake here is the adaptive channel coding to distinguish sensor streams using relevance stream QoD labels, learned in the context of channel conditions at the mobile decoding receiver. The receiver needs to accurately predict the corrupted bits representing the current measurement during decoding. Data corruption can be due to packet collision from neighboring nodes, low data-link quality in terms of fading, and shadowing due to mobility segments used by Data MULES.

7.6 Results

We select a noisy dataset (Iyer *et al.*, 2013),(Goldberg & Zhu, 2010),(Chen *et al.*, 2010), which has measurements for temperature, humidity, and light found across a lab area measured using 100 wireless sensors over a period of several weeks. The

(a) Baseline Noise.
(No Mobility)

(b) Baseline Noise using BPSK.
(No Mobility)

(c) Random Fading & Shadowing
(20 Data MULES).

(d) Random Fading &
(20 Data MULES using BPSK).

(e) Random Fading & Optimal Shadowing
(Clustering & 20 Data MULES).

(f) Random Fading & Optimal Shadowing
(Clustering 20 Data MULES using BPSK).

(g) All Trajectories - BPSK

(h) All Trajectories - Turbo

Figure 7.4: Fading and shadowing effects, clustering and without mobility. Figure 7.4(g) is a comparison of 802.15.4 using BPSK. Figure 7.4(h) Turbo codes with block length of 256.

sensor stream dataset collected from each iteration (epoch) of the sensor network is further divided into spatial clusters. This localized selection allows us to estimate the measurements from non-faulty sensors (m) among many faulty sensors by using the best feature set (N), which minimizes the measurement error. Parameters measured locally are shown in Table 7.2; humidity has fewer missing values (12.9%) compared to temperature (16.9%) and light (66%) in our example dataset.

7.6.1 Kappa-based Stream QoD

The data cleaning, which is performed in Step 1, handles missing data values' normalization by using the kappa function (Iyer *et al.*, 2013) available in *R* (<http://www, 2005>) to label the stream quality. After the matrix completion step, the RMS error of the stream containing best estimates for the missing values is compared with the performance of the original noisy dataset with missing values. From the table in Figure 7.1, the quality of the sensor stream shows an improvement of 27%.

7.6.2 BER-based Stream QoD

When analyzing the BER model, we use a QualNet (QualNet, 2008) mobile simulator, which measures wireless fading and shadowing losses at the receiver. The decoder performance at the receiver is compared with BPSK and Turbo BPSK error correction codes with block lengths supported by the 802.15.4 protocol. Interference seen at the receiver due to collision (Iyer *et al.*, 2013),(Goldberg & Zhu, 2010) has been a major source of re-transmission and cause of energy drain. The results which support this are shown in Figures 7.4(a) and Figure 7.4(b).

When studying the aspects of fast fading and slow shadowing, we use Data Mule probes in the simulation, which account for 20% of the total static ground

sensors. Simulator uses a Random-way-point mobility model. The results of Step 2 data cleaning are shown in Figures 7.4(b) and Figure 7.4(c). In Data MULES, the mobile sensor aggregation errors are reduced by up to 6 times when using BPSK error correction coding, compared to standard uncoded modulation. This is because the BPSK decoder is not able to handle symbol interference, which we have earlier discussed in this section.

The same simulation is performed using turbo-codes (Jiang, 2013). The simulation results show that turbo codes are more robust to mobile affects of fading and shadowing. During fast-fading, turbo codes are able to use the interleaver to keep the SNR constant during the symbol period while decoding. The shadowing due to slow mobility patterns is completely eliminated with cluster formation, as shown in Figures 7.4(e) and Figure 7.4(f), when RSSI (Receiver Signal Strength Index) is optimal within a given wireless coverage area. Figures 7.4(e) and Figure 7.4(f) compare the coding gain in MATLAB (Jiang, 2013) with varying block sizes of 1000 and 512 symbols at the decoder. The results show that even though the BER do not differ significantly, fading and shadowing are better handled when ground sensors are clustered and encoding uses a longer code length.

7.6.3 Task-based Stream QoD label learning

We measure samples using the mobility framework mWSN. The sampling of sensor data is highly probabilistic. The Intel static dataset (CSAIL, 2012) used earlier is not suitable for unsupervised label learning. Therefore, we use a Monte Carlo simulation method to generate values for the inputs (temperature, humidity and light). The number of random trajectories taken by the Data MULE nodes represents the number of evolutions in the Monte Carlo method. For each iteration of the random inputs, the observed variable Y is saved, and the complete set of sensor samples

from the multi-hop path forms a unique trajectory each time. As the distribution is unknown for our dataset attributes, which are temperature, humidity and light, we cannot easily estimate the label error of Y . We use R (<http://www>, 2005) to define sampling of the observed measurement by using *sample with replacement* Figure 6.6, and the command shown below, where t is the number of times mobile sampling is performed.

$$ystar = sample(y, replace = T)$$

The transductive classifier uses random forest algorithm's (Vens *et al.*, 2011) sampling and replacement boosting to estimate finer temporal variations in the given dataset. We use the WEKA (Iyer *et al.*, 2013) attribute analyzer and set option to 20 fold cross validation. The results on the mWSN label learning from a simulated temporal dataset using temperature, humidity and light are shown in Table 7.2. The transductive label error is reduced to 5% when using induced attributes learning utilizing decision trees. The results reflect that when temperature is used as a dependent attribute, the algorithm is able to predict the labels well, even in a small training dataset, where 16.5% of the labels are missing.

7.7 Summary

Our key findings are that SPOTLESS framework parameter estimates have better precision as they use lower bound of QoD and QoI thresholds, when it comes to correcting spatial measurement errors. The computation complexity of the two categories is given in Definitions 1 and 2. Matrix completion always uses a full rank (corrects the sensor local variations) matrix. Therefore, estimates are optimal and closer to the ground truth, even when the data is sparse due to many missing fea-

tures. The SPOTLESS data cleaning is robust due to random data-link channel errors, which are caused by MAC layer corruption during network packet aggregation. When simulated using SPOTLESS service, the normalized data stream was error-free and its matrix coefficient was close to 95% accurate, compared to original raw sensor data, a 27% increase in spatial consistency. Channel states are monitored, and appropriate variable rate adaption is used at the Data MULE transmitter to minimize the packet aggregation at the network layers. The SPOTLESS data-cleaning service uses soft-phy (Chen *et al.*, 2010) channel feedback information to estimate BPSK symbol decoding rates to avoid fast fading errors. SPOTLESS service adapts with changing wireless coverage during mobility, which in turn helps reduce decoding errors by 50% and conserves the sensor node’s energy by avoiding packet retransmissions. The SPOTLESS service classifies end-to-end GPS location context streams using a task-based multiple tree boosting learning algorithm. Each task is a trajectory taken by the Data MULES from source to sink. The higher the PER for a Data MULES trajectory, the lower is its stream QoD label making it redundant. The simulation results in Figures 7.4(a-f) support the theoretical estimate of static and mobile BER results using MATLAB, shown in Figures 7.4(g-h). The SL classifier works off-line to categorize these unsupervised streams into good or redundant samples. The Monte Carlo simulation of the temporal task-based label classifier shows that 95% accuracy is achieved in processing multi-hop data streams.

Chapter 8

Managing Mobile Trajectories using *HEAPS*

$\mathcal{H}$ _{HEAPS} is a model for processing streams ensembles from mobile users. A hidden Markov model (HMM) is used as the likelihood predictor for GPS trajectories. This work was inspired by a paper written by authors at Clemson University (Yu *et al.*, 2013) on training sample sizes. In this chapter we present new results using R-statistical© package and HMM C++ tool-kit to analyze GPS trajectories from GEOLIFE datasets(Zheng *et al.*, 2012) provided by Microsoft Research©. In Section 8.2 a sequence classifier is described, which allows to model time-series data. Training and classifying mobile streams using HMMs Ensemble Aggregating Patterns from Streams (Duda *et al.*, 2000) - HEAPS model is described in Section 8.3.

8.1 Introduction

One of the main problems of target motion analysis (TMA) is to estimate the trajectory of an object from noise corrupted sensor data. The idea needs modeling of the states of the trajectories by a hidden Markov model (HMM). Typically the time-series data is sampled by Geo Position Service (GPS) receivers and reports the date, time and location information every few seconds. The user-logs (Li & Zheng, 2008), (Zheng, 2008a), (Zheng, 2008b) provide many interesting temporal and useful patterns, which can predict and classify new unlabeled GPS coordinates

from mobile users. We show in Figure 8.1(a) the classical Euler’s tour which was used to model trajectories from one side of the town to other by using each bridge Figure 8.1(c) only once. The corresponding solution was solved using a graph model 8.1(b). Using the same analogy in the current state of smart-phone users, the user can record GPS trajectories when he is driving, walking or taking a bus around the city. We show a typical heatmap (Zheng *et al.*, 2012) of the GPS user-logs of user activities in Beijing city. This can be used to model HMM as shown in Figure 8.2. The user activities, which are processed from different user streams can be used to classify future unlabeled trajectory logs from the same location. The two main reasons for combining classifiers are to improve efficiency and accuracy of noisy sensor data. As seen in Chapter 7, the ensemble methods used are: (i) bagging Quinlan (1996) and (2) random forest Breiman (2001). One uses sample with replacement and the other uses randomized learning of attributes. In a multi-stage bagging classifier, sampling with replacement may stabilize the prediction accuracy of the classifier even when used with a small sample sizes. Creating features can be accomplished by combining different classifiers using a subset of features selected randomly. Combining classifiers can be broadly accomplished in two ways. The first one uses the original input attributes of the input patterns such as a k-nearest neighborhood Duda *et al.* (2000) classifier, which the same distance function for learning in the translated space. In the second case of neural network different training strategies are used. Each such neural net produces a posterior probability can no longer be used to determine the combined posterior probability as classifiers use different measurement spaces. We will concentrate on the second method as we are interested in relationships of correlated features of sensor streams.

(a) Ancient city.

(b) Graph model

(c) Trajectory planing using bridges

(d) Heatmap of user activity from Beijing city.

Figure 8.1: Euler's tour and trajectory tour transforms.

8.2 Sequence Classifier

We accomplish the desired functionality by using an HMMs ensembles abbreviated as HEAPS, which is trained with number of hidden states, probabilities of transition matrix, and probabilities of emission matrix. The HEAPS chooses the maximum a-posterior (MAP) value for identifying the new test sample using Bayes rule.

Figure 8.2: Distribution of latitude and longitude from the trajectory dataset. The variations can be captured in 2-States.

$$p(x, y) = \prod_{t=1}^T p(y_t | y_{t-1}) p(x_t | y_t) \quad (8.1)$$

where the probabilities $p(y_t | y_{t-1})$ current probability in state y_t given we just were in the state y_{t-1} at the previous instant $(t - 1)$. Using this assumption, the model needs only to know the previous state to calculates all its prior probabilities until now. Then, the HMM model can be characterized using the tuple,

$$\lambda = (n, A, B, \pi) \quad (8.2)$$

$$(8.3)$$

A typical Markov model uses specific parameters but in our case we like to learn from a set of data samples. As in the previous chapter we have described unsupervised learning in which the stream is unlabeled. Hidden Markov models are especially known for their application in temporal pattern mining, which have higher order states not directly observed by the system. A hidden Markov model can be considered as a generalization of a mixture model where the hidden variables (or latent variables), which constitute the mixture distribution for each observation, are states of a Markov process rather than independent of each other. One such example of HMM is shown in Figure 8.2, which has two hidden states and three observed states.

The proposed model uses a classification approach to unlabeled sensor streams so as to increase reliability. Statistically INTEL sensor dataset (CSAIL, 2012) have 30% missing values and at the same time the amount of mobile sensor data generated is increasing dramatically due to GPS enabled phones. We like to extend some of the preliminary results obtained in spatio-temporal data-cleaning algorithms to mobility models by using hidden Markov models (HMM). The ensemble of such streams can be studied using ensemble of HMMs to further learn latent features from the observed datasets. For example, some of the data-cleaning algorithms work well with GPS tagged data as shown in Figure 8.4 as it has context with the underlying map so any outliers can be visually viewed to further remove outliers during clustering.

The sensors allow aggregating all the parameters online. Due to inherent unreliability of a sensor and limited amount of training set available, it is hard to measure and label stream quality on-line. Given an observation of stream data over time the stream can be categorized using HEAPS. The number of hidden states learned by HEAPS framework using the data is accomplished using a Baum-Welch (Duda *et al.*, 2000) algorithm. This helps to create a sequence classifier whose output is

Figure 8.3: Trajectory Visualization from GEOLIFE Dataset Showing GPS States.

Figure 8.4: Hidden Markov model showing different activity recognition and its corresponding states.

not probabilities, but likelihoods and are unnormalized. Bayes rule is used, which is given by,

$$\hat{y} = \operatorname{argmax}_{w_j \in \Omega} P(\omega_j | x) \quad (8.4)$$

8.3 Training the Ensemble HMMs

We use the dataset trajectories which have GPS reading from most frequently traveled routes in Beijing city. The city falls between **39 55 N 116 25 E** latitude and longitude and each trajectory record has the latitude, longitude, date and time information recorded very few seconds. Out of the 17,621 trajectories from 182 users, 73 users have labeled their trajectories such as *walk*, *car*, *bus*, etc. An ensemble of 3 HMMS, each for walk, car and bus are trained using sufficient training data from two different users. Due to exponential complexity of computation with the increase in the number of states, we like to design an efficient HMM for the given dataset. A sample trajectory from dataset (Li & Zheng, 2008), (Zheng, 2008a), (Zheng, 2008b) was taken and plotted to find the temporal variations as shown in Figure 8.4 with activity recognition. From the plot the distribution of latitude and longitude varies to a minimum of 2-states as shown in Figure 8.3 or a maximum of 4-states. The ensemble HMM when used to test new samples was able to detect the sequence 95% of the time. As the trajectories have many common areas, if sufficient number of GPS coordinates are provided during test the HEAP framework is able to categorize the test samples with high accuracy.

8.4 Summary

The ensemble model is well justified. We have discussed in quality labels using DC-Trees in chapter 5, which deals with temporal streams. In the case of sequential time-series data we further need to distinguish the factors which effect the streams. The factors, which affect HMM model are the window-length of the time-series data. In this case the length of the training data is statistically sufficient for the model and its hidden states. These factors are further analyzed to determine the QoD

labels of time-series data as they differ from feature based categories in the previous discussions.

Chapter 9

Effects of Noise Due to User Mobility

*M*obility model plays an important role in modern sensing applications. The presence of noise is an important design consideration in wireless when deployed in harsh terrains. In this chapter we present the results using OMNET++ simulator and capture the temporal variations in time-domain with a single mobile user and fixed base stations. Most of the work is from the research paper ¹. The project was supported by Korean Science Council and National Institute of Mathematical Sciences. A cognitive radio is introduced in Section 9.3 and how primary and secondary members could share a sparsely used licensed spectrum. Section 9.4 described how to model mobility and compensate the fading effects. Section 9.5 explains new standards which co-exist with sensor networks and how it can minimize interference and enhance spectrum usage. For a through treatment of different noise types in sensor networks, please refer to the ² Distributed Sensor Networks, Second Edition: Image and Sensor Signal Processing book.

¹Vasanth Iyer, S. Sitharama Iyengar, Garmiela Rama Murthy, N. Parameswaran, Dhananjay Singh, and Mandalika B. Srinivas. Effects of channel SNR in Mobile Cognitive Radios and Co-existing Deployment of Cognitive Wireless Sensor Networks. IEEE IPCCC, 2010, pp. 294-300. Albuquerque, New Mexico, USA.

²(Iyengar & R, 2012) <http://www.crcpress.com/product/isbn/9781439862827>

9.1 Introduction

Cognitive radio and sensor networks studied here are both considered static, with mobile primary licensed users. There are mobile primary users models for extensions to study specific signal estimation techniques. The cognition in their part has two common modes of interference avoidance. The first approach uses overlay to make up for the unused spectrum bandwidth and the second approach uses underlay in the form of interference control. The history of cognitive radio can be attributed to the thesis work of J. Mitola in 2000, where he coined "Cognitive Radio" for a form of radio that would change its performance by detecting its environment and changing accordingly. Using mobility frame work and cognitive radios we find the trade-offs between minimum spectrum power allocation and channel rate, when operating in overlapping frequencies with primary users.

We like to study the performance of deploying dense Wireless Sensor Networks which uses the ISM band using IEEE 802.15.4 protocol in context of Cognitive networks. Due to recent emerging standards on inter-operability none of them address the distributed nature of the spectrum. Some of the deployments have adapted to frequency reuse and orthogonal spectrum allocations to have least interference and better usage of the same spectrum. These implementations allow baseline reality and also take into consideration the non-linearity of radios in practice, which introduce errors during channel coding. In this chapter we extend radio inter-operability and its specific power-ware requirements for extending the sensor network lifetime. We model interference as unlicensed users partially overlapping with primary user as shown in Figure 9.1, using a simulator as illustrated in Figures 9.3, 9.1, which give rise to co-channel interference. The varying parameters at the radio receivers are interference due to number of overlapping channels and variation caused by mo-

Figure 9.1: Frequency and bandwidth of unlicensed user partially overlapping with primary user.

bility. The interference in mobility can be seen as the phase shift due to Doppler and phase shift leading to delay spread due to frequency. The two dimensional representation of interference varying with wireless range from the primary signal and the interfering signals is represented using co-variance matrix. The correlated channels with high signal to noise ratio (SNR) have better Qos, which leads to higher link quality and interference free reception during spectrum usage.

9.2 Related Standards

Due to the availability of Software Defined Radios (SDR) and its ability to architect sophisticated spectrum sensing radios, the FCC in 2004 formed a working group to define 802.22 standards. The new standard was particularly to rural areas by sharing the television spectrum, the standard is expected to be completed by the first quarter of 2010 and with this some of the first networks could be deployed.

There are a number of elements that were set down for the basis of the 802.22 standard. These included items such as the system topology, system capacity and the projected coverage for the system. By setting these basic system parameters in place, the other areas fell into place. The model selection (Akyildiz *et al.*, 2006) for this work studies the radio performance based on mobility and the protocol performance due to interference in dense sensor networks. The crosslayer stack parameters, which affect channel capacity, SNR, energy efficiency, BER and optimum radio modulation schemes for a given interference level to exist with other radios is described in Theorem 1 through Theorem 7.

9.3 Model Prediction

9.3.1 Interference avoidance (spectrum overlay)

Using the specification of 802.22, which rely on a central command controller, the base station can have large training samples collected from all the CPE's. The dataset can be used over time to predict the spectrum availability for cooperative future scheduling. The spectrum can be categorized into highly, medium and sparsely used. We expect the coverage in urban areas, where the network may be deployed would fall into the sparse spectrum category.

The CPE's which are deployed remotely will collect the frequency of the detected signal and its time duration in its overlapping spectrum. The spectrum availability can be calculated with the overlapping intervals. Current radio designs uses packed based count (Reddy *et al.*, 2008) and time-sampling (Reddy *et al.*, 2008) techniques. In the packed capture technique, the cognitive module becomes quite complex due to large number of packets and its demanding space requirements. In the other implementation i.e. Systematic Timer based Sampling (STT), all the channels are

sampled at an interval of 1 second and provides an accurate measure of the spectrum activity to be recorded. The above techniques takes into account the capture of packets over the entire networks and uses cognitive network techniques.

In this chapter, we discuss how to data-sets can be represented in two-dimensions which model the interference and the path-loss models for cognitive radio co-existence. These dynamic channel losses are used to de-correlate the signals of the primary and secondary users.

9.3.2 IT Model (spectrum underlay)

The concept of interference temperature (Clancy & Arbaugh, 2006) is identical to that of noise temperature. It is a measure of the power and bandwidth occupied by interference. Interference temperature T_I is specified in Kelvin (Clancy & Arbaugh, 2006) and is defined as

$$T_I(f_c, B) = \frac{P_I(f_c, B)}{kB} \quad (9.1)$$

where $P_I(f_c, B)$ is the average interference power in Watts centered at f_c , covering bandwidth B measured in Hertz. Boltzmann's (Clancy & Arbaugh, 2006) constant k is 1.38×10^{-23} .

9.3.3 Ideal Model-Known Frequencies

In the ideal interference temperature model we attempt to compute only the interference due to licensed signals. Assume our unlicensed transmitter is operating with average power P , and frequency f_c , with bandwidth B .

Assuming that the radio knows the frequencies of the base station and the allowed bandwidth, then it needs to only filter frequencies in the range $f_c - B/2$ to $f_c + B/2$ overlaps n licensed signals, with respective frequencies and bandwidth of f_i and B_i .

As shown in Figure 9.1, we need to guarantee (Clancy & Arbaugh, 2006) that

$$T_I(f_i, B_i) + \frac{M_i P}{k B_i} \leq T_L f_i, 1 \leq i \leq n \quad (9.2)$$

Note the introduction of constants M_i . This is a fractional value between 0 and 10 as shown in Table 9.1, representing a multiplicative attenuation due to fading and path loss between the unlicensed transmitter and the licensed receiver. Section II described the 802.11 standard, which allows to define the licensed user. The model co-existing needs only to know which are the most overlapping frequencies in use other than the given standard specification of frequency and bandwidth of the allocated primary user's spectrum. The second step is to measure T_I in the presence of the licensed signal. Assuming we know the signal characteristics and the wireless losses we can use correlation of the measured interference, which will help isolate the redundant signal interference. Also, if we have precise knowledge of the signal's bandwidth B and center frequency f_c , we can approximate the interference temperature (Clancy & Arbaugh, 2006),

$$T_I(f_c, B) \approx \frac{P(f_c - B/2 - \tau) + P(f_c + B/2 + \tau)}{2kB} \quad (9.3)$$

where $P(f)$ is the sensed signal power at frequency f and τ is a safety margin of a few kHz.

9.3.4 Generalized Model-Unknown Frequencies

In cases where there is no prior knowledge, which could be a new network environment, we need to apply interference temperature model to the entire frequency range of operation to detect any primary user. This is typically the case with blind

source (Duda *et al.*, 2000) separation.

$$T_I(f_c, B) + \frac{MP}{kB} \leq T_L f_c \quad (9.4)$$

Assuming each licensed signal has power P_i and otherwise the interference floor is defined by the thermal noise temperature T_N , we can transform (9.4) into the following:

$$KBT_L(f_c(B - B_i) + kBT_N \sum_{j=1}^n B_j) \leq \sum_{j=1}^n B_j P_j \forall 1 \leq i \leq n \quad (9.5)$$

In a simple case with only one licensed receiver, the inequality simplifies to,

$$\frac{KBT_L}{P_1 - kBT_N} \leq \frac{B_1}{B - B_1} \quad (9.6)$$

Latter in the result analysis we will show how to measure and de-correlate in such complex environments.

9.3.5 Crosslayer STACK Model

Sensor nodes and its communication stack have limited resources per node and replacement of nodes or batteries may not be practical. Placement and deployment of sensors needs to be accounted for optimal coverage and the needed density of sensors for reliable calibration of the measured values. We use a crosslayer model due to the nature of wireless as the sensor connectivity is not always available. The crosslayer architecture allows us to account for energy dissipated at each level during the entire lifetime, so that the power-aware protocols can adapt the radios for optimal coverage and longer useful lifetime.

9.4 Modeling Mobility in Wireless Channels

Rayleigh fading is used to describe the characteristic of the wireless channels, which are used by wireless receivers. The Rayleigh model assumes that signal has passed through such a medium and will vary randomly or fade according to Rayleigh model. The Doppler power spectral density of a fading channel describes how much spectral broadening it causes. The effect on pure signal, when it passes through such a channel.

$$S_\nu = \frac{1}{\pi f_d \sqrt{1 - \left(\frac{\nu}{f_d}\right)^2}} \quad (9.7)$$

Where ν is the frequency shift relative to the carrier frequency. The equation is valid only for values of ν between $\pm f_d$. The Doppler model is shown in equation (9.7) and the Rayleigh model is described here and in equation (9.8). With this, we can extend it to simulate mobility by summing up the sinusoidal.

Table 9.1: Peak SNR measured values of the diagonal elements from the complete data sets. (Method-1 Vs Method-2)

Values of $\rho_{i,j}$	Measured M_{Hi}	Measured M_{Low}	Ideal	Ideal	General	General
			Upper-Bound	Lower-Bound	Upper Bound	Lower Bound
$\rho_{cor}=1.3133$	3.792	0.08	0.46	0.32	1.4102	1.2292
$\rho_{cor}=1.3747$	4.362	0.022	1.8211	0.3454	2.0216	-1.2292
$\rho_{cor}=2.0741$	4.222	0.034	2.2354	0.4157	2.0216	0.8926
$\rho_{cor}=2.0868$	3.917	0.063	2.2293	0.5303	2.310	0.2139
$\rho_{cor}=2.4209$	4.073	0.047	2.4209	0.09	2.7578	-0.7437
$\rho_{uncor}=0.8169$	4.362	0.013	1.399	0.3201	1.3003	0.0308

The calculation of the co-efficient of the real and imaginary parts used by the Rayleigh model can be redefined for a scatter, which is uniformly distributed around a circle at angles α_n with k rays emerging from each scatter.

In the described model we use multiple radio receivers $R_p, R_{s1}, R_{s2} \dots R_{sm}$. The normalized autocorrelation function of a Rayleigh faded channel with motion at a

Ideal primary user model 1 -frequency and (in timestamp) for a mobile node.

General user model - Primary user with coexisting static secondary users 2-frequencies varying position in time of the primary user (in timestamp).

Tracking SNR ρ_{cor} co-efficient in PCA of the optimal correlated channels.

Separating SNR ρ_{uncor} co-efficient in ICA of uncorrelated and unknown source frequencies.

Figure 9.2: Temporal variations in time-domain with a single mobile user and fixed base stations.

constant velocity is a zeroth-order Bessel function of the first kind:

$$R\tau = J_0(2\pi f_d\tau) \quad (9.8)$$

9.4.1 Level Crossing Rate

The level crossing rate is a measure of the rapidity of the fading. It quantifies how often the fading crosses some threshold, usually in the positive-going direction. For Rayleigh fading, the level crossing rate is:

$$LCR = 2\pi f_d \rho \exp^{-\rho^2} \quad (9.9)$$

where f_d is the maximum Doppler shift and ρ is the threshold level normalized to the root mean square (RMS) signal level:

$$\rho = \frac{R_{thresh}}{R_{rms}} \quad (9.10)$$

Let us design a way to detect and isolate the primary user R_p and secondary users $R_{s1}, R_{s2} \dots R_{sm}$ for channels $ch_1, ch_2, ch_3 \dots ch_{48}$. For single mobile receivers the interference noise due to path-loss has a single spike, but for multiple users the interference noise due to path-loss degrades into a Gaussian curve (Bristow, 1983). From equation 9.2, we calculate values M_i for the interference seen at multiple channels in time.

$$M_{cov} = \begin{pmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{pmatrix} \quad (9.11)$$

To maximize the correlation we need to optimally select the diagonal elements of the matrix shown in equation 9.11 and values in Table 9.1, such that equation (9.12) can

be used to select channels that can differentiate primary user from the secondary user's interference.

$$G = \max \det(M) = \rho_{11} = \rho_{22} = 1 \quad (9.12)$$

9.5 Sensor Network Life-Time Vs Interference

Sensor network standards are placed at the low power range of the spectrum and use overlapping frequencies of the unlicensed ISM sub-channels bands. Typically the range of sensor radios is limited and tends to be deployed in dense configuration for maximal sensing coverage. With the multi-hop broadcast nature of the wireless channel, the sensor network protocols uses channel access protocols to avoid collisions. We study different channel access protocols and their effect on dense deployments. The lifetime of sensor networks would be shortened due to loss of energy at the protocol level due to collisions, and retries due to pathloss errors and idle listening. If a distributed protocol is used then the depletion of energy at the node is due to over usage of its limited energy reserves, which happens when routing algorithms use distance vector metrics instead of power-aware metrics. The system suffers from energy holes, which ultimately partitions the network rendering the quality of the data from the networks inaccurate as it represents a small count of the overall connected sensors.

The lifetime of the sensor network can be represented by the equation below.

$$Lifetime = \frac{C_{batt} \times V \times 60 \times 60}{Energy_{Tx} + Energy_{Rx} + Energy_{Idle} + Energy_{Listen}} \quad (9.13)$$

It uses minimal energy when not transmitting. Most of the IEEE 802.15.4 radios use power-aware protocol, which efficiently duty-cycle using sleep modes. Due to the nature of energy saving at every level of the stack, cross-layer optimization is an

important aspect of scalable sensor network. We study the lifetime by optimizing the necessary protocols to allow longer lifetime.

9.5.1 Distributed Model with Radio abstraction

Power-aware distributed protocols works directly at the routing layer. This allows us to rotate the cluster heads periodically from a percentage of nodes in the network. MAC and radio abstraction is assumed and only TDMA based synchronization is needed to coordinate the nodes. The protocol allows us to configure the percentage of nodes to be elected as cluster head, which essentially acts like router and other nodes are configured to aggregate the data to minimize the number of bits needed for transmission. Due to distributed nature of selecting the cluster heads (transmitters) the average energy consumed is reduced compared to traditional routing protocols. As transmission uses most of the energy in the protocol, it allows us to maximize the lifetime of the sensor network, before a node fails due to energy depletion. Routing protocol over head is more than lower layer protocols as it needs to have higher level of topology for synchronization.

$$CH_{Lifetime} = T_{transmit} + T_{receive} \quad (9.14)$$

$$Node_{Lifetime} = T_{receive} \quad (9.15)$$

9.5.2 Single hop Model with best effort 802.11 Radio scheduling

The 802.11 radios have three states Transmit (Tx), Receive (Rx) and Idle. The communication protocols spend a percentage of time depending on the ambient events. The MAC protocol uses lower layers for neighborhood discovery and communication and has very low overhead. The physical layer protocol does not do well with data

aggregation as in the case of time coordinated LEACH, which does not improve the reliability of the sensor network.

$$Node_{Lifetime} = T_{transmit} + T_{receive} + T_{listen} + T_{Idle} \quad (9.16)$$

9.5.3 Multi-hop Model with Sync 802.15.4 Radio scheduling

The 802.15.4 is designed from the physical layer where the sensors are calibrated. Due to the nature of the wireless false events can get transmitted due to high floor noise. Sensor MAC protocols are specifically designed to adapt to ambient operating environment. This includes averaging the floor noise and using a link layer based approach to data aggregation. The MAC radios are designed to be receiver centric as the sensor network protocols spend more time sleeping and idling during its lifetime. The protocols extends the single hop discussed earlier to multi-hop by using a sync packet, which allows us to schedule forwarding nodes and chance of being awake the next time the transmission is probable.

$$Node_{Lifetime} = T_{transmit} + T_{receive} + T_{listen} + T_{Idle} + T_{Sleep} \quad (9.17)$$

The detection of the secondary user transmitter is ensured at the forwarding receiver nodes, which allow enough number of nodes to get awakened during the broadcast enabling synchronization. A beacon is implemented part of low-power listening (LPL) radios synchronization for the next hop routing nodes. For reliable synchronization the transmitting wakeup beacon is twice the period as compared to the LPL radio active period.

$$\text{Amortized dissipation} = \frac{Energy_{Tx} + Energy_{Rx} + Energy_{Idle}}{3} \quad (9.18)$$

9.6 Simulation

The simulation is set to analyze two fundamental aspects of cognition, one to identify the source signal of a primary user during mobility as illustrated in Figure 9.3, noise due to path loss and interference due to secondary users and the second one is to maximize the lifetime of a large network by using minimum power. Table 9.2 defines the detection algorithms for cognitive radio and Table 9.3 defines the crosslayer protocol models used by cognitive wireless sensor networks.

Figure 9.3: Mobility framework and wireless channel for calculating pathloss.

9.6.1 Mobility and cognitive radio sub-channels

With IT overlap models described in equation (9.2), we calculate different values for the variable terms $T_I(f_i, B_i)$ and $\frac{M_i P}{k B_i}$. A OMNET mobility framework simulator, using interference modeling as derived in Section IV and Table 9.3. We use two models to analyze the data collected from the mobility simulator, one uses the covariance technique (Kumar *et al.*, 2008),(Iyer *et al.*, 2008g) to optimize as shown in equation (9.12), when not too much of co-channel interference is seen. The second

Table 9.2: Cognitive Radio Channel Interference Simulation Setup

Methods	Model	Metric
Mobility Traces	OMNET++	Doppler
Propagation	Rayleigh-Jake's	Phase, freq
Radios	1 Primary user	T_I Cognitive
Channel	48 Sub-channels	M_i
Noise	1 Secondary user	Floor-noise
Covariance	R Project for Statistical Computing	ρ_{cov}
ICA	R Project for Statistical Computing	ρ_{uncov}

method uses ICA (Duda *et al.*, 2000), which de-correlates under the presence of heavy co-channel interference.

9.6.2 Two Dimensional Representation of Interference using Covariance

Figure 9.2 shows different values of M_i seen at the primary receiver with no secondary users, as it can be seen that it has unique peaks, which vary in time. It is seen that the convolution of two Gaussian functions is again a normal Gaussian function, and thus the distribution for the sum of two independent normal random variables is again normal. If they are independent, their joint distribution (Duda *et al.*, 2000) has the form of

$$p(x) = \prod_{i=1}^d p(x_i) = \prod_{i=1}^d \frac{1}{2\pi\sigma_i} e^{-1/2((x_i - \mu_i)/\sigma_i)^2} \quad (9.19)$$

For the measured interference (Clancy & Arbaugh, 2006) due to mobility and spectrum interference due to overlapping with n licensed bands over time, $d = 48$, see

simulation section and Table 9.1.

$$= \frac{1}{(2\pi)^{d/2} \prod_{i=1}^d \sigma_i} = \exp \left[-\frac{1}{2} \prod_{i=1}^d \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2 \right] \quad (9.20)$$

This can be written in a compact matrix form if we observe that for this case the co-variance matrix is diagonal, i.e.,

$$\Sigma = \begin{pmatrix} \rho_1 & 0 & .. & 0 \\ 0 & \rho_2 & .. & 0 \\ \vdots & \vdots & \searrow & \vdots \\ 0 & 0 & .. & \rho_d \end{pmatrix} \quad (9.21)$$

The measured co-efficient is shown in columns of Table 9.1 and the corresponding co-variances are calculated in calculated columns of the current table. Initially, when only the primary user is using the spectrum and has mobility with constant fading and changing wireless range, the signals seen at the receiver have a sharp spike, which is shown in Figure 9.2. Figure 9.2 shows the effects of interference at the primary receiver, the plot does not have any spatial or time varying properties, as it is uniformly distributed, which follows a Gaussian distribution. To separate the interference from the secondary users, we need to compute the lower and upper bounds of the interference floor, which is computed by the thermal noise temperature T_N . The co-efficient of the co-variance matrix of all the 5 data-sets are chosen to maximize G as shown in Table 9.1. As we use the correlation between signals, which are due to multi-path scattering, we plot the upper bound response of the attenuation of the channels. This is shown in Figure 9.2, where it is seen as completely correlated and described in equation (9.3).

9.6.3 Estimation of Interference using ICA

The above method uses correlation matrix to maximize the determinant to find the primary and secondary users. ICA uses a method to seek components, which are varying independently and thus differentiates the primary user and the rest. This method is preferred when the noise level is very high in the channels. The measured coefficients for ICA analysis using R-System package fastICA (Jones *et al.*, 2009), are tabulated in Table 9.1 (ICA_1, ICA_2 , where users=2), $\rho_{uncorr} = 1.30032658$ for number of signal equal to two, which are approximately lower bound of the last result set $\rho_{corr} = 1.313362745$. Figure 9.2 shows how ICA can perform well when the sources are unknown and blind or shadowed to measurements, when the noise level is above the given interference threshold, as shown in equation 9.6.

9.6.4 Crosslayer stack and cognitive sensor network

The previous method uses cognitive radio approach and hence is more accurate at the node level and we have shown that it can perform well in cases of node mobility and blind source separation. In our experiment we assume a large scale deployment of at least few hundred nodes and the protocol will scale up to thousand of nodes. The sensor networks protocols are broadly divided into distributed and MAC layer protocol. The use of distributed routing allows us to calculate the upper bound performance using equation (9.14) and the best effort mode, which uses the MAC allows us to calculate the lower bound as the protocol designer has more option to adapt to traffic patterns. In the case of distributed protocols the x-axis as illustrated in Figure 9.4, represents the number of cluster heads, which can be varied and the y-axis represents the average energy dissipation during its lifetime. In the case of the MAC protocol, where the radio states are part of the communication protocol it

can further adapt. Our experiment measures the energy consumption of the radio as defined in equation (9.13), the x-axis represents many channel access protocols as illustrated in Figures 9.4. The y-axis represents the lifetime in seconds for a large deployment. In our experiments, the performance of the clustering based protocols

Table 9.3: Simulation setup of 802.15.4 cognitive WSN network.

Stack Methods	MAC Model	Routing Metric
Radio abstraction	Distributed	Cluster Heads
Best effort	Single-hop	Channel control
Best effort	Multi-hop	LPL
Best effort	Multi-hop	Sync

does well when the percentage of cluster heads is less than 20% of the total network size. In the best effort mode the protocols that use multi-hop routing, does well in terms of overall reliability and network throughput.

9.6.5 Dense Deployment and Channel Interference

Figures in 9.4 shows how radio idling and fixed range radios in dense deployment further effects the overall performance in a negative way.

9.7 Summary

Cognitive radio and its role for coexisting with a spectrum has been emphasized with evolving standards. Most of our experiments were done at the cognitive radio layer, which allowed us to closely design the algorithms for signal detection and also use power-aware protocols to manage the radios. In the power-aware signal analysis we compared the ideal spectrum model with the generalized spectrum model to obtain upper and lower bounds of the thermal interference for the variable frequency

Distributed WSN Model in time (in routing rounds) .

802.11 WSN MAC in time (in lifetime seconds) .

Multi-hop WSN MAC in time (in lifetime seconds) .

Startup Radio energy characterization in time (in seconds).

Radio Range Interference - Low, Medium and High deployment density in time (in lifetime seconds) .

Figure 9.4: Co-existing of WSN and node density

range of primary user. We improvised by calculating the floor noise due to co-channel interference and detecting the primary user with least power as in the case of 802.22 standards. To find the performance of our method, when the primary user frequencies are unknown, which is the case in wireless sensor networks, we computed ICA for the entire spectrum and showed that the primary user detection is possible when coexisting with secondary users or when noise dominates the desired feature thresholds. The deployment strategies of sensor networks allowed us to compare other detection methods specially in dense overlapping of the lower end spectrum usage.

9.8 Interference vs Energy Model

Theorem 9.8.1 *★ Co-channel Interference*

$$SIR = 10 \log_{10} \left[\left(\frac{D}{R} \right)^n K_l^{-l} \right] \quad (9.22)$$

Gaussian Model

★ Adjacent Channel Interference

$$SIR = C \frac{P_{TX}}{d^n} \left(\frac{P'_{TX} \int_w G(f) df}{(d)^n} \right)^{-1} \quad (9.23)$$

Transmitter sensitive

★ Multi-path fading

$$\cos(2\Pi ft + \theta) \Rightarrow \text{Doppler Spectrum Model} \quad (9.24)$$

The effects of fading θ can be combated by using diversity to transmit the signal over multiple channels that experience independent fading and coherently combining them at the receiver. The probability of experiencing a fade in this composite channel is then proportional to the probability that all the component channels simultaneously experience a fade, a much more unlikely event.

For the IT model described in equation (9.2), we calculate different values for the variable terms $T_I(f_i, B_i)$ and $\frac{M_i P}{k B_i}$. The reliability of a digital system is measured in terms of the error rate in the transmission link. BER - Bit Error Rate, SER - Symbol Error Rate, FER - Frame Error Rate, PER - Packet Error Rate. The SER characterizes the performance of the modulator. The BER is measured at the bit-level in terms of the number of bits that are received erroneously.

9.8.1 Channel Capacity

Theorem 9.8.2 *Shannon showed that in an AWGN channel, the maximum bit-rate C that can be achieved with arbitrarily low error rate over a given transmission bandwidth WT is bounded by the expression below:*

$$E_b = \frac{E_s}{\log_2 M} \quad (9.25)$$

$$\bar{E}_b = \sum \frac{p_i E_s^i}{\log_2 M} \quad (9.26)$$

where p_i is the probability for the occurrence of the i^{th} symbol with energy.

$$E_b = \frac{\max E_s^i}{\log_2 M} \quad (9.27)$$

Theorem 9.8.3 *Where E_b is expressed in terms of the peak symbol energy in the*

signal constellation. E_s^i . where $\gamma = E_b$ and $\bar{\gamma} = \bar{E}_b$

$$\frac{C}{W_T} \leq \log_2 \left(1 + \frac{P}{N_o W_T} \right) = \log_2 \left(1 + \bar{\gamma} \frac{C}{W_T} \right) \quad (9.28)$$

Symbol Energy:

$$E_b = \frac{E}{\log_2 M} \quad (9.29)$$

where $P/N_o W_T$ is the SNR, C/W_T is the maximum achievable bandwidth efficiency in bps/Hz, and γ is the average E_o/N_o defined by equation (9.2).

9.8.2 Error Rate Bounds.

Theorem 9.8.4 *SER on the otherhand is measured at the symbol level in terms of the number of symbols that are in error. A symbol error is made when the received signal falls outside of its decision region. A symbol error leads to errors as the symbol is erroneously mapped to an incorrect bit-pattern. Let n be the number of bits per symbol. Then, SER may be bounded in terms of BER as shown below:*

$$P_b \leq P_s \leq n.P_s \quad (9.30)$$

To express PER in terms of BER

$$P_p = 1 - (1 - P_b^{L_p}) \quad (9.31)$$

9.8.3 A basic measure using BER

Theorem 9.8.5 *SER, FER, and PER all depend on BER, a basic measure for digital system is based on BER which can be expressed in terms of $\frac{E_b}{N_o}$, where E_b is the energy per bit and N_o is the equivalent noise spectral density over the signal*

bandwidth. The variations of BER with $\frac{E_b}{N_o}$ depends on the channel and the type of the demodulator. Channel Models:(see Table A.1 for BPSK-modulation)

★ AWGN

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) \quad (9.32)$$

★ Rayleigh Fading

$$P_b = \frac{1}{2} \left(1 - \frac{\sqrt{E_b/N_o}}{1 + E_b/N_o}\right) \approx \frac{1}{4E_b/N_o} \quad (9.33)$$

9.8.4 Design goals for using BER

Theorem 9.8.6 *Additive white Gaussian noise (AWGN) in general, in a channel, BER is exponentially related to E_b/N_o , while in a fading channel, BER is inversely related to E_b/N_o .*

For a given BER, a digital system with lower $\frac{E_b}{N_o}$ requires lower transmission power, which can improve battery lifetime of the communication device and the system capacity.

9.8.5 Energy Efficiency

Theorem 9.8.7 *Energy efficiency can be more accurately defined, when taking into account both energy and bandwidth, we define $f(x)$ of a system to be the amount of E_b/N_o required for a given bandwidth efficiency:*

Table 9.4: Energy Efficiency Comparison used by Radios

BPSK	9.09	1	11.0
GMSK	10.8	1.35	11.5
QPSK	9.09	2	16.5
8-PSK	19.82	3	11.8
16-PSK	55.41	4	6.8
32-PSK	171.2	5	3.6
8-QAM	13.93	3	16.8
BFSK	17.78	1	5.6

Part IV

Protocols

Chapter 10

Research Methodologies used in WSN Simulations

This Chapter¹ describes the quality of measure in terms of service needed for sensor network protocols. It uses Naive Bayes (Iyer *et al.*, 2009b) to estimate the sensor errors at the network layer and compares the MAC layer using K-nearest neighbor. It shows that the MAC error is bounded twice by the Bayes error rate.

Sections 10.1-10.11 describe various cross-layer protocols and compares their throughput and power efficiency. In Section 9.12, the same protocols are extended to accommodate renewable sensor nodes which needs to be delay tolerant.

10.1 Main contributions and organization of the work

In this related work section, we model the sensor networks as a unsupervised learning and clustering process. We classify nodes according to their static distribution to form known class densities(CCPD). These densities are chosen from specific cross-layer features, which maximizes lifetime of power-aware routing algorithms. To circumvent computational complexities of a power-ware STACK we introduce path-loss models at the nodes only for high density deployments. We study the

¹Iyer, V. and Iyengar, S. S. and Rama Murthy and Balakrishnan, N. and Phoha, V. Distributed Source Coding for Sensor Data Model and Estimation of Cluster Head Errors using Bayesian and K-Nearest Neighborhood Classifiers in Deployment of Dense Wireless Sensor Networks. Proc. Third International Conference on Sensor Technologies and Applications, SENSORCOMM 2009.

Two-dimensional sensor placement.

Flooding and dividing the network into zones (1-hop, 2-hops and 3-hops).

Tree routing from data collection nodes to the base station root(1).

Figure 10.1: Simulation of a two-dimensional sensor placement showing routing tree.

(a) Temp. and Light are correlated - Measured.

(b) Temp. and Humidity are correlated - Measured.

(c) Query logs showing how long each of the web pages was browsed by the user - Non measurable.

Figure 10.2: On-line and sensor logs showing measured and non-measurable fields.

cluster heads and formulate the data handling capacity for an expected deployment and use localized probability models to fuse the data with side information before transmission. So each cluster head has a unique P_{Max} (Iyer *et al.*, December 2008) but not all cluster heads have the same measured value. If the cluster size is n , from the cluster equation (Iyer *et al.*, 2008g) then the entropy of data aggregation is $H(S) = -\sum_{i=0}^n P(X_i) \log P(X_i)$.

In a lossless mode if there are no faults in the sensor network then we can show that the highest probability given by P_{Max} is ambiguous if its frequency is $\leq \frac{n}{2}$ otherwise it can be determined by a local function. We further show that the event detection at the cluster heads can be modeled with a pattern 2^m and m , the number of bits can be a correlated pattern of 2 bits. These local algorithms are further studied to optimize on power, fault detection and to maximize on the distributed routing algorithm used at the higher layers. From these bounds in large network, it is observed that the power dissipation is network size invariant. The performance of the routing algorithms is solely based on success of finding healthy nodes in a large distribution. It is also observed that if the network size is kept constant and the density of the nodes is kept closer then the local pathloss model affects the performance of the routing algorithms. We also obtain the maximum intensity of transmitting nodes for a given category of routing algorithms for an outage constraint, i.e., the lifetime of sensor network.

10.2 Distributed Algorithms

There exists a significant body of literature for cross-layer networks performance using distributed nodes. This is the framework for MITs μ - AMPS project (Heinzelman *et al.*, 2000), which focuses on innovative energy optimized solutions at all

levels of the system hierarchy, from the physical layer and communication protocols up to the application layer and efficient DSP design for micro sensors nodes. Sensor networks contain too much data for an end-user to process. Therefore, automated methods of combining or aggregating the data into small set of meaningful information is required (Heinzelman *et al.*, 2000). In addition to avoiding information overload, data aggregation, also known as data fusion, can combine several unreliable data measurements to produce a more accurate signal by enhancing the common signal and reducing the uncorrelated noise. Here we will categorizes some of the work done in energy-aware models with respect to reusability (cluster-head) and develop a data fusion framework, which allows us to avoid unreliable data in any arbitrary size network. Simulations show that LEACH can achieve as much as a factor of 8 reductions in energy dissipation compared with conventional routing protocols. In addition, LEACH is able to distribute energy dissipation evenly throughout the sensors, doubling the useful system lifetime for the networks we simulated. Thus, communication between the sensor nodes and the base station is expensive, and there are no high-energy nodes through which communication can proceed.

10.3 Information Processing

Rate distortion theory gives theoretical bounds for how much compression can be achieved using lossy compression methods. Many of the existing audio, speech, image, and video compression techniques have transformations, quantization, and bit-rate allocation procedures that capitalize on the general shape of rate-distortion functions. Rate distortion theory was created by Claude Shannon in his foundational work on information theory. In rate-distortion theory, the rate is usually understood

as the number of bits per data sample to be stored or transmitted. The notion of distortion is a subject of on-going discussion. In the most simple case (which is actually the most used), distortion is defined as the variance of the difference between input and output signals (i.e., the mean squared error of the difference).

10.4 Distributed source coding with side information

In characteristics with Sensor networks the data compression algorithms, which optimize on bringing down data redundancy, which is related to correlated sensor readings and using a probability model to efficiently compress data at the cluster heads.

The Slepian-Wolf rate (Slepian & Wolf, 1975) region for two arbitrarily correlated sources x and y is bound by the following inequalities. This theorem can be adapted using equation (Iyer *et al.*, 2009b),

$$R_x \geq H\left(\frac{x}{y}\right), R_y \geq H\left(\frac{y}{x}\right) \text{ and } R_x + R_y \geq H(x, y) \quad (10.1)$$

If the correlated sources differing by a few bits, the possible number of codewords can be represented as 2^m where m = no. faulty bits (Iyer *et al.*, 2008b). In our case $m=2$ as the parameters are distributed whilst collected locally at the cluster head.

In sensors networks several measured values are sensed in a distributed manner and these are aggregated according to the users query. The goal of all the encoder is analogous to the previous section where it uses cosets. Equations (Iyer *et al.*, 2009b) in compression rate illustrates the bin formation to reduce the overall bits needed for transmission. Considering the case of distributed sensing application, the encoder is further designed with a machine learnable redundancy range which is specific to each and every application. This mutually redundant measured range

is correlated with sensors which are in the same wireless range and connected to a parent. This information, also called side information is shared with the decoder. Owing to side information, even fewer numbers of bits are needed to represent the changing values coming from each cluster heads transmitting to the joint decoder. Encoder and decoder have access to the side information Y , which is correlated to X and can be represented by the equation (Slepian & Wolf, 1975). According to the Slepian-Wolf Theorem (Slepian & Wolf, 1975), established in 1971, the number of bits needed by using the theorem is fewer, than the total entropy for both the two arbitrarily correlated sources $H(x)$, $H(y)$.

10.5 Event Detection

As sensor networks have to be reactive in nature and process the underlying information as an event, the data model needs to address how it can convey the bits to a central location.

When the probabilities are all very different, then when a symbol with a low probability arrives, you feel more surprised, get more information, than when a symbol with a higher probability arrives. $I(p)$: A function which measures the amount of information-surprise, uncertainty - in the occurrence of an event of probability p .

$$I(p) = \log\left(\frac{1}{p}\right) = -\log p$$

10.6 Reliability of Sensor Networks

One of the consequences of Huffman coding is its probability model, which allows us to generate prefix code close to the entropy of the source $H(S)$. The code length is $H(S) + 1$. As in sensor networks and wireless sensor networks reliability

of transmission is needed and usage of limited power during transmission. Due to compression of data, we can reduce the number of bits transmitted and detect errors detection while decoding in noisy environment. The root of the probability tree is 1 for a normal distribution. One can further model the data gathered from a dense geometrical distribution a which for sensor network is $a < 1$. If the codebook is designed in such a way that only a single bit is changed between two sensed values then the reliability of the decoder is fairly high (tamely faulty). For varying values of $a = 0.5, a = 0.6, a = 0.7, a = 0.8, a = 0.9$ a reliable buffer can be assembled with least number of bits making an assumption the compression efficient is close to $H(S)$. This type of cost function allows us to make reliable transmission in a dense sensor network.

10.6.1 Bayesian Distributed Fault Detection Algorithm

Fault-event disambiguation problem in sensor networks. Our proposed solution, in the form of Bayesian fault recognition algorithms (Krishnamachari & Iyengar, 2004), exploits the notion that measurement errors due to faulty equipment are likely to be uncorrelated, while environmental conditions are spatially correlated. We show through theoretical and simulation results that the optimal threshold decision algorithm we present can reduce sensor measurement faults by as much as 85-95 percent for fault rates up to 10 percent.

10.7 Application Protocol Algorithms

10.7.1 Estimate of the sensed value for known densities

The simulated routing algorithms such LEACH-S (Heinzelman *et al.*, 2000), LEACH-E (Iyer *et al.*, 2008e) and CRF (Iyer *et al.*, 2008e) as described in the above table use the knowledge that the nodes which are sensing are correlated and have known densities such as cluster size and radio range. The sensed values are i.i.d distributed and their variance $\neq 0$. The underlying model uses different ways to select the cluster heads to minimize the error rate. When the sensor faults happen due to fixed energy resources at the cluster head the total energy unused at the end of its lifetime is the residual rate (Iyer *et al.*, 2008d). The routing algorithms tries to minimize this error criteria. As this model uses the network layer and the only dependant variable is the fixed lifetime model (Iyer *et al.*, December 2008). The complexity of the algorithm can be defined by using the standard implementation of the LEACH distributed algorithm and its power-aware variations.

$$\bigcirc g(x) = f(x) : 0 \leq f(x) \leq cg(x) \quad (10.2)$$

$$\Omega g(x) = f(x) : 0 \leq cg(x) \leq f(x) \quad (10.3)$$

$$\Theta g(x) = f(x) : 0 \leq c_1g(x) \leq f(x) \leq c_2g(x) \quad (10.4)$$

Complexity of the routing algorithms for LEACH is shown in equation (Iyer *et al.*, 2008a), LEACH-E equation (Iyer *et al.*, 2008a) and CRF equation (Iyer *et al.*, 2008e). In the next section we will use only the lower layer such as power-ware MAC and estimate the multi-hop routing errors. In this case the model is not dependant on the fixed energy resources and only dependant on k-neighborhood

Figure 10.3: Two dimensional model for simulation of distributed clustering

rules that it uses to find its multi-hop nodes as shown in figure 10.3(b). As the node probability is not known a priori the error rates are much higher than the persistence clustering.

10.8 Simulation

10.8.1 Results from the network layer

Simulation models large number of nodes and calculates the most likely time during the lifetime, when sensor faults are more likely to happen. The Table 12.3 shows number of cluster heads and the fault rate for distributed clustering and passive clustering (Iyer *et al.*, 2009d). Simulation results confirms the the fault rate is network size invariant and converges to the optimal values derived in Theorems 1 and 2 (Iyer *et al.*, 2009d).

Table 10.1: Summary of notations for analysis of routing fault-rate

Symbols	Definition
N	Total number of deployed nodes
n	Number of nodes in the cluster
μ	Density of the class
P_{MAX}	Bayesian class rule
R_x, R_y	Entropy of correlated sources
R, r	Radio Range
P	K-neighborhood fault probability
P^*	Bayesian probability
ω	Bayesian classes
S	Data source node
D	Destination node
θ	Nodes residual energy
CH	Cluster head
$P(\omega_i \mathbf{x})$	Conditional probability
$P(\mathbf{x} \omega_i)$	Class conditional probability

Figure 10.4: Show energy dissipation node loading(with link layer abstraction).

10.8.2 Power Aware Routing Protocols

Results from the Network layer

MAC Abstraction is assumed and the convergence of the nearest neighbor for distributed clustering and passive clustering are derived, the distributed clustering case

is

$$P^*(e) = \int P(e|x)p(x)dx \quad (10.5)$$

$$P = P^* \quad (10.6)$$

Passive clustering is given by

$$P = 2P^* \quad (10.7)$$

As shown in simulations (Iyer *et al.*, 2008d) where lower bound for LEACH-S when it becomes faulty and the remaining residual energy using the cross-layer simulator is $P(e)=0.27\%$ which is the fault rate. In the case of passive clustering when node density $p = 0.1$ or using the k-neighborhood rule (Iyer *et al.*, 2009d) the node densities are unknown in this case due to high likelihood of faults. The protocol simulation shows (Iyer *et al.*, 2009d) that the upper bound has error rate of $P(e)=0.41\%$ which converges to the proof derived in Theorem 3 and Theorem 4 and the upper bound.

Figure 10.5: Show energy dissipation node loading using multi-hop and hardware energy optimizations.

Figure 10.6: MAC performance comparison.

This work uses the existing simulation models that are network-based, and specified the Qos framework for data reliability needed for sensor networks. The fusion of energy harvesting applications with power-aware MAC was studied in terms of deployment of low-level protocols for 802.11, CSMA, and B-MAC. Furthermore, if the data is routed using multi-hop algorithms and is MAC-centric, then the distributed sleep scheduling (Iyer *et al.*, 2009c) is observed to reduce the percentage of energy lost during overhearing and collision. The performance of routing algorithms with MAC losses has a long tail which is similarly observed in Power Law. Lastly, the asymptotic lower and upper bounds for data-link reliability has been theoretically predicted using Bayes probability. Simulation results show that the probability of data-link reliability is greater for clustering algorithms due to conditioning at the cluster head in CSMA, which helps sensor network protocols and is more energy efficient when using B-MAC. FARMS which uses a low-duty cycling MAC and built-in

Figure 10.7: Data aggregation using SPEED/Directed Diffusion/LEACH

renewable energy harvesting part of the routing protocol performs well in a dense sensor network configuration and has the added advantage of longer lifetime with Qos close to regular sensor network algorithms.

10.8.3 MAC Aggregation Protocols

10.8.4 Results from the MAC layer using a propagation model

In the previous case the MAC abstraction was used, which does not take into account the propagation losses and protocol retries at the MAC level. To simulate the wireless channel we use GlomoSIM(UCLA, 2000) bit error rate(BER) simulator and implement the routing algorithms for multi-hop cases. The routing algorithm implemented is SPEED, which is a geographic routing algorithm that uses two-dimensional coordinate space to calculate the path from the node coordinates. Many runs into the protocol simulation suggest that the radio characterization for CSMA as shown in Figure 10.11(a) and B-MAC are comparable, Figure 10.11(b) when the node densities are known.

The radio characterization for CSMA (Iyer *et al.*, 2008a) is prone to faults when

compared to B-MAC, when using in multi-hop modes where the node densities are unknown. The protocol performance results show that the data packets received during useful lifetime is $3\times$ times better Figure 10.11(b) in B-MAC when compared to CSMA and error rates are $P \geq 2P^*$ higher than the theoretical Bayesian limit (Iyer *et al.*, 2009b) of $P = 2P^*$ as derived in Theorems 3 and 4.

10.9 Bit Error Ratio (BER)

The BER rate is dependent on the packet size and the frequency of the radio. As BER increases with distance and multi-hop routing algorithms with no Qos support, we try to address this in terms of the actual payload which are typically sensed values. As the BER rate is constant for a given channel we try to reduce the data error rate by further reducing numbers of bits needed to transmit by finding values which differ by only 1-bit. This is accomplished by using a $\tilde{H}(p)$, pmf with a geometrical cluster for a fixed radio range. This technique further sorts on lower values to minimize any potential error which can trigger a false high alarm.

10.9.1 Addressing high Bit Error Ratio (BER) and noisy channels

Frequency dependency

For short distances,

$$d = \frac{\text{Distance between the communicating Sensor Nodes}}{\text{Speed of Light}} \quad (10.8)$$

$$P_r = \frac{(P_t \times G_t \times G_r \times \lambda^2)}{(4\pi)^2 \times d^2 \times L^2} \quad (10.9)$$

★ P_t is the transmitted signal power

- ★ P_r is the received signal power
- ★ G_t, G_r are the antenna gains of the transmitter and the receiver respectively.
- ★ L is the system loss, and λ is the wavelength.

For long distance use, two-ray ground reflection model

$$P_r = \frac{(P_t \times G_t \times G_r \times \lambda^2)}{(4\pi)^2 \times d^2 \times L^2} \quad (10.10)$$

h_t and h_r are heights of transmit and receive antennas respectively and d is the distance. The above equation shows a faster power loss than in the Free Space Model as distance increases. The other observation is it is independent of frequency for longer distances, so a higher such as 2.4MHz compared to 900MHz will be able to reduce the BER rate. Thus, a distribution has a min-entropy of at least b bits if no possible state has a probability greater than 2^{-b} . Here we use a 3-bit prefix code.

Resilient error correction by code design

An application for this decaying exponential variant was given in (Shannon, 1948). In this application, a communication channel has a window of opportunity with a total duration (in bits) distributed geometrically with parameter a . The probability of successful transmission is given probability mass function p and $a > 1$, find a code minimizing

$$L_a(p, l) \triangleq \log_q \sum p(i) a^{l(i)} \quad (10.11)$$

$$P_{Success} = a^{L_a(p, l)} \quad (10.12)$$

Figure 10.8: Show wireless channel simulation setup for minimizing transmission error due to transmission interference.

For $K \leq 1$ which is the kraft inequality, 2^{-L} if $L = 1$ if $k = 0.5$ The probability of successful transmission is for a given probability mass function p and $0 < a < 1$, find a code minimizing $\log \sum_{i \in \chi} p(i) a^{l(i)}$ and $P_{success} = a^{L_a(p,l)}$. If $a=0.5$ in equation (5.5) and the cost between 1.0 and 2.0 the success probability is $= 0.5$ and considering the case for $a=0.9$ in equation (5.5) and $L=1.1$, the success probability is $= 0.71$.

A few observations can be used to find a series of improved lower and upper bounds on optimum maximum pointwise redundancy based on

- ★ In a Huffman-like tree for a maximum pointwise redundancy code, the weight of the root w_{root} determines the maximum pointwise redundancy, $R^*(l,p) = \log w_{root}$.
- ★ The total probability of any subtree is no greater than the weight of the subtree. This can be inductively observed.
- ★ In the Huffman-like coding, items are merged by nondecreasing weight. This can be observed by noting that any new merged items have weight greater than either of its merged items. In fact, any new merged item has weight at least twice as great as either of the merged items, due to (Gallager, 1978)

A fourth observation is in the form of the following lemma:

Lemma 1: Given a probability mass function p for $n = |\chi|$ items, if $p(1) \leq 2p(n-1)$, then a minimum maximum pointwise redundancy code can be represented by a complete tree, that is, a tree such that $\sum 2^{-l(i)} = 1$ and $|l(i) - l(j)| \leq 1$ for all $i, j \in \chi$.

Proof: A code with minimum maximum pointwise redundancy is always obtained when using a Huffman style algorithm combining the items with the smallest weights, w' and w'' , yielding a new item of weight $w = 2\max(w', w'')$, and this process being repeated on the new set of weights, the tree thus being constructed up from the leaves to the root. Since a tree always satisfies that $\sum 2^{-l(i)} = 1$. Consider the tree formed by the application of this algorithm. Since the first (and thus least weighted) combined item is of weight $2p(n-1)$, clearly no combined item need be merged with another item until the point at which item 1 is merged or thereafter. The algorithm can, in this case, be seen as pairing off items in the order of a queue sorted from least weighted to most weighted and placing the paired-off items in the rear of the queue. Because items are processed with increasing weight, this processing occurs in queue order, and thus, at any given point, every item is processed about the same number of times as any other; the difference can only be one. This is true when the algorithm terminates and codeword length is equal to the number of times an item is (by itself or as part of a combined item) processed. Thus $|l(i) - l(j)| \leq 1$ for all $i, j \in \chi$, and the complete code tree is optimal. We can now present the improved redundancy bounds.

Figure 10.9: Using Huffman trees to compute min-entropy with $|l(i) - l(j)| \leq 1$

Figure 10.10: Fault redundancy Huffman codes using codes with 2^{-b} codebook

10.9.2 Cross-Layer Lifetime analysis - MAC Integration

Lifetime Modeling

To calculate node duty cycle and lifetime, we develop an arrangement of virtual clusters, which streams data to the base station. Table 12.3 lists the primitive operations performed by MAC states, which are extended for power-aware for normal traffic. The virtual clustering protocol implement a minimal version of LEACH, which allows select cluster heads periodically and rotates them in a distributed way. Data aggregation is done from all the nodes part of the virtual clusters during the current time slot. We simulate a low data rate application, which samples every 10 seconds and passes the data to a base station. The MAC integration uses CSMA/BMAC/TDMA for testing the sample application with periodic traffic. As CSMA does not have any power saving scheme it does not perform well. On the other hand B-MAC has low power listening(LPL), which allows to cut down on idle listening. Here we introduce a basic protocol, which is specific to TDMA based frame.

A node will keep listening and potentially transmitting as long as it is in active period. An active period ends when no activation event has occurred for a time TA. An activation event is:

- ★ The firing of a periodic timer
- ★ End-of transmission of a node's own data packet or acknowledgement

A node will sleep if it is not in an active period. Consequently, TA determines the minimal amount of idle listening per frame. Due to better synchronization of the TDMA frame based MAC, the idle listening is cut down when TA period is reached. Results from simulation of 100 nodes with virtual clustering show that TDMA-TA

(a) CSMA, B-MAC with LPL and CSMA-TDMA Frame (b) CSMA-TA Frame, B-MAC-TDMA-TA Frame.

Figure 10.11: MAC performance in lifetime seconds

based power optimization using B-MAC at the lower layer does 5x times better to enhance the lifetime of the sensor network.

Table 10.2: Summary of notations for analysis of MAC performance.

Symbols	Definition
C_{Sleep}	Current Sleep schedule period of the node
Tx	Transmit energy consumed during its lifetime
Rx	Receive energy consumed during its lifetime
Idle	Idle energy consumed during its lifetime
$TDMA_{no-duty-cycle}$	When using virtual clustering of nodes n, the node is awake for the complete time-slot.
$TDMA_{TA}$	When using virtual clustering of nodes n, the node is put to sleep immediately after the protocol completes for the current time-slot.
C_{batt}	Capacity of battery
V	Voltage
$L_{preamble}$	Preamble Length (bytes)
L_{packet}	Packet Length (bytes)
t_i	Radio Sampling Interval (s)
n	Nodes residual energy
CH	Neighborhood Size (node $\leq 20\%$)
r	Sample Rate (packet/s)
t_l	Expected Lifetime (s)

Chapter 11

Renewable Energy

11.1 Renewable Energy: Qos Simulation of Wireless Sensor Networks

In the related work Chapter¹ we show results from protocols which are delay tolerant it even so more important in renewable energy routing used by sensors. The adaptive sensors use non-conventional energy such as vibration, solar and other sources which help renew its battery in a periodic manner. Due to this some nodes may not be available in the routing process as it may be between charging and the protocol needs to account for such delays and overheads.

11.2 Introduction

Tuning the cross-layer parameters is very essential when working with constrained wireless sensor network (WSN) protocols. As the unique nature of deployment includes the needs of harsh environments for sensor networks, the integration of data Qos and the network throughput Qos is an essential part of a well-designed

¹Iyer, V. and S.S. Iyengar, and Murthy, R. and Srinivas, M. B., N. Balakrishnan, Vir Phoha and Hochet. Multi-hop Scheduling and Local Data Link Aggregation Dependant Qos in Modeling and Simulation of Power-aware Wireless Sensor Networks. Proc. Wireless Communications and Mobile Computing Conference, IWCMC09, June 21-24, 2009, Leipzig, Germany

sensor network. As data is sensed at the lower layers that are MAC-dependent, we need to analyze the reliability of the data gathering protocol as well as its functions in a dense deployment. The sensor network STACK will need to have a tight-link layer acknowledgement to achieve the desirable Qos. The lower-level protocol works in a very small tiny window that is typically defined as the time to live (TTL) of the underlying protocol, and hence, it is very efficient and requires enough correlated sensors to measure the parameters, such as light, temperature and humidity, and bestow a time-stamp to them. Once the data payload is defined with the updated information about time and value, it has to be efficiently routed. A design should be constructed in such a way that it allows to multi-hop the payloads using least energy, and at the same time, efficiently schedule the node for local housekeeping, clustering, and ambient activities. Hence, a software infrastructure needs to be developed, which can allow queuing of messages, selection of best routes, and guarantee delivery mechanism to the available sink node that uses average Qos based on the available active nodes and residual energy.

Power-aware sensor hardware specifications have evolved and have specifically given deterministic metrics for sensing and routing tasks in terms of μA and mA , indicating that data transmission is very much more power draining than normal sensing tasks. For a constant battery model delivering $mA - h$ power for the underlying radio model specified by the manufacturer, the minimal transmission is observed to be the preferred mode of operation for prolonged lifetime of the sensor network. This typical lifetime model shows that to operate in such a mode, the burden on Rx and Idle tasks, which when combined, take equivalently enough power when compared with the data transmission, suggesting that the design is for normal wireless network and not for energy-aware WSN. In this chapter, we will use practical models that employ duty-cycling at the lower layers and measure the impact

of Tx, Rx, and Idle over a long period of time, and use an amortized power-drain cost to accurately predict the lifetime of the simulated sensor network.

11.3 Motivation

Qos for sensor network can be based on the reliability of data and the power consumption during its entire useful lifetime. Currently, most of the metrics are defined for individual layers, namely network and MAC layers, as this writing application needs to select the right routing algorithms and its corresponding hardware that provides sufficient bandwidth using the MAC, to provide custom solutions. This is a daunting task, and as most of the application developers do not know the details of the MAC, it makes it even harder to deploy and scale a large network. To simplify the many dependencies, we have used the Qos of the sensor service, which is independent of any specific algorithm, protocol, or battery model. This service-based measurable index is solely based on its dependence on the data link layer MAC. The lower-layer MAC provides adaptability for reliable data and saves power to sense the new data from the sensors, as well as offers constant tension to update new data and compute reliable paths to the sinks.

The power savings calculated is solely based on the sleep states in motes, duty-cycling, harvesting periodicity, and reliable fusion of the sensor data, which makes it independent of the energy model. To simulate the cross-layer dependencies, the setup is broken down into three steps: the first step uses a non-network simulator that only implements the distributed algorithms that are dependent on the random ad hoc placement of nodes per meter square, resource allocation, cluster head rotation, and variable RF control. The second step uses network simulator that has loadable MAC modules and a corresponding battery model for Tx, Rx, and Idle

states with their respective mA used by the radio. This allows accurate measurement of energy drain for Tx, Rx, and Idle for a real-time clock, for aggregating the sensor data. The third step combines the first and second steps, coding into real sensor node using the network simulator that can deploy large number of nodes for a given RF radio range. This allows accurate measurement of the data periodicity for a CBR source that routes the data payload over the sensor network.

The simulated results calculated are prioritized for parameters that are not dependent on large constant energy resources but are based on Qos, which include data reliability, MAC latency, and energy-savings owing to sleep scheduling. These MAC-dependent parameters are adapted for ultra-low-duty cycles for renewable lifetime model of energy-harvesting application that rely on ambient energy (FARMS) (Iyer *et al.*, 2009c). This simulation approach allows the integration of specific needs for emulating a real mote, as well as the study of the adaptation of MAC to Qos, which is independent of the lifetime model.

11.4 Renewable Energy Model and Energy Harvesting

For a multi-hop sensor network using a lifetime renewable energy model (Iyer *et al.*, 2009c), an active node that is ready to transmit at a given instance will take a time period t to reach its neighbors, and the response time for receiving the message back will be $2 \times t$ or a preset TTL value. To efficiently multi-hop, we need to at least have one neighbor that can respond to the active node, but in a large WSN network, the traffic is directly proportional to the number of neighbors. The model has to take into consideration the example deployment that has sufficient number of receiving nodes to respond to the on-demand traffic generated by active nodes. If the receiving nodes do not have sufficient energy to aggregate the new data and

re-transmit them, then it needs to have an active queuing by which it can avoid the complete loss of the new data it received, but relaying it when it has enough charge resulting in a latency $f(x) = \sqrt{n} = 1$, n number of nodes or best-effort Qos. To study the MAC characteristics for constrained devices, which are uniquely dependent on varying node densities as well as limited transmission ranges and power, the model needs to have a scheduling periodicity and must select new available active nodes in the multi-hop path between nodes to forward the data. The routing algorithm that is dependent on the density of the network also needs to find a leader node to multi-hop from the clusters to the sink nodes.

11.5 Lifetime Modeling-Syetsm Performance Using MAC Dependence

11.5.1 Multi-hop MAC's

Limited research has been carried out on integrating different network layers into one layer or to investigate the benefit of cross-layer interactions between routing and MAC layers for sensor networks. Most of the existing protocols can use RTS/CTS extensions to achieve collision-free broadcast. Hence, to have an effective cross-layer scheduling, one of the suggested design is the combination of MAC/physical-layer integration and Routing/MAC/physical-layer integration. A variable length TDMA scheme has been proposed, in which the slot length is assigned according to some criteria for optimum-energy consumption in the network. Among these criteria, the most crucial ones are the information about the traffic generated by each node and the distance between each node pair. Based on these values, a linear programming (LP) problem has been formulated, in which the decision variables are normalized

time-slot lengths between the nodes. The LP program is solved using an LP solver that returns the optimum number of time slots for each node pair as well as the related routing decisions for the system.

The proposed solution could be beneficial in situations where the required data must be prepared. However, it is generally difficult to obtain the node-distance information and the traffic generated by the nodes. Besides, the LP solver can only run on a powerful node. The dynamic behavior of sensor networks will require online decisions that are very costly to calculate and hard to adapt to an existing system. Multi-hop Infrastructure Network Architecture (MINA) is another method for integrating MAC and routing protocols. This proposed design uses a layered multi-hop network architecture, in which the network nodes with the same hop count to the base station are grouped into the same layer. Channel access is a TDMA-based MAC protocol combined with CDMA or FDMA. The super-frame is composed of a control packet, a beacon frame, and a data transmission frame.

The beacons and data frames are time-slotted. In the clustered network architecture, all the members of a cluster submit their transmission requests in beacon slots. Accordingly, the cluster head announces the schedule of the data frame. The routing protocol is a simple multi-hop protocol where each node has a forwarder node at one nearer layer to the base station. The forwarding node is chosen from the candidates based on the residual energies. Moreover, the transmission range of the sensor nodes is a decision variable, as it affects the layering of the network (the hop-counts change). The simulations were run to determine a good range of values for a specific scenario.

11.6 Performance Analysis of WSN Data Aggregation Algorithms

Our performance evaluation of data aggregation algorithms uses a single-hop neighbor discovery and a distributed method to select a cluster head. After the completion of the protocol process, the metrics measured are the control protocol overhead, the data payload received without errors at the cluster head, and the running average of the performance of the data aggregation algorithms, i.e., if all the neighbors respond successfully, then it is considered as 100% aggregation, otherwise a loss of data aggregation is reported owing to insufficient data. In this study, we assumed a population of M sensor nodes, sharing the same noiseless radio channel, without any hidden or exposed terminals. In addition, we also considered a single-hop fully connected cluster of nodes, i.e., all the sensors can hear transmission from any sensor in the cluster (Iyer *et al.*, 2009c). Furthermore, no central control existed and each sensor node had an equal probability of generating a packet for transmission to any of the rest of the $M - 1$ nodes in the cluster. Every packet consisted of a header and a payload part. The header was used for synchronization of the receiver and carried control information, such as the address of the receiver node and the total length of the packet.

The payload was the useful sensor information transmitted between the nodes. Owing to the high density deployment of the sensor network, the MAC protocol must be designed with features, such as collision avoidance when multiple sensors access the same channel, filtration of sensed payload data from interference noise, and decreased battery consumption owing to idle listening when the protocol is idle, which consumes as much energy as it does when receiving, as in the case of IEEE 802.11. The simulation of the variation of MAC with node density is shown in

Figure 11.1.

11.6.1 Link Protocols

Data link protocols can be categorized into two main IEEE wireless standards.

Sampled

Communication is unsynchronized, data transfer wakes up the receiver. Some examples are B-MAC, Aloha with Preamble Sampling, Mica1 LPL, CC2500, Reactive Radio.

Slotted

Communication is synchronized, data transfers occur in slots. Some examples are S-MAC, T-MAC, TRAMA, 802.15.4. In sensor networks, multicast is an important type of communication pattern. In protocols that include clustering, cluster heads communicate with their members and thus, all the intended receivers may not be the neighbors of the cluster head, but may just be a subset of the neighbors. We used this type of data aggregation at the cluster heads using clustering for various link protocols. We used GlomoSIM, with 100 nodes deployed in a 140 » 140 m with a radio range of 50 m. As this is a close deployment of a dense network, we expected a lot of collision and data loss.

11.6.2 Carrier Sense Multiple Access - CSMA-CA

Carrier Sense Multiple Access (CSMA) and its variants appear in several major MAC protocols designed for WSNs (Polastre *et al.*, 2004), such as S-MAC, T-MAC, Shift, and IEEE 802.15.4. The CSMA-based protocols have the benefits of low complexity,

scalability, and ability to adjust to population changes. On the other hand, they suffer from energy-wasting problems, such as packet collisions, overhearing, and idle listening.

Assumptions

Constant length packets, No errors, except those caused by collisions. No capture effect, each host can sense the transmissions of all the other hosts.

The propagation delay is small when compared with the transmission time, once a node receives a packet that needs to be sent, it broadcasts a jam signal onto the network to make sure that the channel is clear, as well as to inform the other devices not to broadcast. CSMA-CA acts to prevent transmission collisions before they occur, unlike CSMA-CD (Detect). To evaluate CSMA with the GlomoSIM, we used two categories of WSN algorithm: one using clustering, which has lots of time-synchronization overheads, and the other using multi-hop, which is highly distributed owing to its independent synchronization. As the definition of CSMA is collision avoidance, it performs well in a dense network, outperforming B-MAC and 802.11 in total successful data aggregation at the cluster heads. The simulation results are shown in Figure 11.1, 11.2.

11.6.3 Design for collision avoidance

$$CSMA = \text{Message Transmission} = 2 \times \textit{Propagation}$$

Clear Channel Assessment and Packet Backoff for channel arbitration. Link-Layer Acknowledgment for reliability, especially when picking leader nodes owing to path selection. Low Power Listening for low power communication, only a link protocol

(a small core of media access functionality). Contains a set of interfaces that allow services to tune its operation.

To eliminate idle listening, it is absolutely crucial for the MAC protocol to support the duty-cycling mechanism. To evaluate B-MAC with the GlomoSIM, we used two categories of WSN algorithm as described earlier. As the design of B-MAC is receiver-centric, it is always capable of receiving packets to the next hop with much less drop rate in multi-hop routing, when compared with CSMA and 802.11. The simulation results are shown in Figures 11.1, 11.2.

Design for Collision Avoidance

B-MAC = Preamble Length = Sleep Schedule = 100 ms

11.7 Network Model

Figure 11.1: CSMA MAC performance with constant radio range

Figure 11.2: B-MAC performance with constant radio range

Figure 11.3: FARMS performance with constant radio range

Figure 11.4: Bayesian reliability bounds for Theorem 1, 2

It is important to design and test the behavior of MAC protocols based on the kind of power used in scheduling states of the MAC. We identify two main techniques for dense wireless Sensor MAC protocols.

11.7.1 Local Power Dissipation Due to RF Interference

When MAC losses are considered as in figure 11.1,11.2, using power-aware MAC like CSMA and BMAC, the test bed provides many unique characteristics of cross-layer STACK analysis. If one takes into account the pathloss model which is the additional parameter of interest, it can be show that the interference due to high density MAC transmission during data forwarding, clustering and normal channel management has a long depleting tail and also follows Power Law (Lowen & Teich, 1970) dissipation $\left\| \frac{1}{(1+r^2)} \right\|$, where r is the Transmitter range in meters. Figure 11.1, 11.2 shows the lifetime plot for various routing algorithms with MAC in varying densities (75,140, and 440 meter squares). BMAC performance better when compared to CSMA and delivers reliability on events at $P \geq 4.0$ and at the same time some of the event and query based algorithms have losses close to theoretical lower (Lowen & Teich, 1970) bounds of $P = 3.0$.

11.7.2 Data Link Probability for Sensor Data Aggregation

The problem of Data Link probability for sensor data aggregation can be addressed in two different models which are adaptive to traditional sensor network protocols and the more flexible low-level MAC protocols services.

Random Sampled Data Events

When two variables are statistically dependent, knowing the value of one of them let us get a better estimate of the value of the other one. Consider a simple illustration of a two-variable MAC state where both states x and y are either charged(=1) or charging(=0). Suppose that a large number n of pairs of xy nodes are randomly

produced. We assume the number of neighbors can be represented in terms of $\sqrt{(N)}$, where N is the order of the clusters.

Theorem 11.7.1 *Let $n_{i,j}$ be the number of pairs in which we find $x = i$ and $y = j$, that is, we see the $(0,0)$ pair n_{00} times, the $(0,1)$ pair n_{01} times, and so on, where $n_{00} + n_{01} + n_{10} + n_{11} = n$. Suppose we schedule the node pairs where $y = 1$ —that is, the nodes which are available to receive and have sufficient energy $(0,1)$ and the $(1,1)$ pairs. Clearly, the fraction of those cases in which x is also charged 1 is*

$$\frac{n_{11}}{n_{01} + n_{11}} = \frac{n_{11}/n}{(n_{01} + n_{11})/n}$$

Intuitively, for a large sensor network deployment to know the probability of a successful transmission pairs $P(x|y)$ when a forwarding neighbor is available, when $y = 1$ and n is large. And indeed, this is what we get, because $\frac{n_{11}}{n}$ is approximately $P(y)$ for large n .

Distributed Data Events Slotted at Scheduled Intervals

If an even such as node clustering C occurs in m different ways $C_1, C_2, C_3 \dots C_m$ and if these m cluster head subevents are mutually exclusive —that is, cannot occur at the same time —then the probability of distributed clustering occurring is the sum of the probabilities of the subevents C_i . In particular the random variable y which can be random or node's remaining residual power can assume the value y in m different ways —with $x = v_1, x = v_2, \dots, x = v_m, \dots$. Because these subevents are mutually exclusive, from the Law of Total probability that $P(y)$ is the sum of the joint probability $P(x,y)$ over all possible values for x . We assume the number of neighbors can be represented in terms of $\sqrt{(N)}$, where N is the order of the clusters.

In this clustering case it is $\sqrt{(N)}$. Formally we have,

$$P(y) = \sum_{x \in \chi} P(x, y)$$

But from the definition of Theorem 1 $P(y|x)$ we have

$$P(x|y) = \frac{P(y|x)P(x)}{\sum_{x \in \chi} P(y|x)P(x)}$$

Theorem 11.7.2 *The above Equation is called the Bayes rule. Note that the denominator, which is just $P(y)$, is obtained by summing the numerator over all x values. By writing the denominator in this form we emphasize the fact that everything on the right-hand side of the equation is conditioned on x . This is the other way of saying that the data-link quality is conditioned for a particular power-aware Qos chosen by the clustering algorithm. Hence the provability of error at the data-link layer is superior than the previous case, Theorem 1 is conditionally dependant and has the probability of error greater than the Bayes rule.*

Lemma 11.7.3 *Sequence of length n from the source. In sensor each element in the sequence is independent and identically distributed (i.i.d.), then we can represent entropy in bit length.*

$$H(S) = - \sum P(X_1) \log P(X_1)$$

Comparison to Bayes Probability

The theorems allows to predict the asymptotic bounds of the data-link interference and allows modeling of the sensor network in terms of complexity defined for clustering $\mathcal{O}(\sqrt{(N)} \log N)$ for the two cases. The error bounds are shown in Figure 11.4 in terms of Bayesian probability. The data capacity as defined in Lemma 1 and is dealt

in detail in a related work in (Iyer *et al.*, December 2008),(Iyer *et al.*, 2008d),(Iyer *et al.*, 2009b).

11.8 MAC Performance Using Ultra-low Duty Cycle

From the traditional energy model, a dynamic renewable model was adapted, as shown in Figure 11.3, which makes optimization at the lower layer, an essential part of the energy harvesting model. Sensor network harvesting provides mechanisms for network protocols to operate efficiently, and the three key elements of the renewable energy protocol design are, data reception, data transmission and neighbor Management. This phase of the enhancement of the simulator allows the improvement on the scheduling of the nodes that participate during routing using application control, as described in Theorem 1, more detailed of the protocol implementations are provided in the FARMS paper (Iyer *et al.*, 2009c).

11.9 Summary

This work used the existing simulation models that are network-based, and specified the Qos framework for data reliability needed for sensor networks. The fusion of energy harvesting applications with power-aware MAC was studied in terms of deployment of low-level protocols for 802.11, CSMA, and B-MAC. Furthermore, if the data is routed using multi-hop algorithms and is MAC-centric, then the distributed sleep scheduling (Iyer *et al.*, 2009c) is observed to reduce the percentage of energy lost during overhearing and collision. The performance of routing algorithms with MAC losses has a long tail which is similarly observed in Power Law. Simulation results show that the probability of data-link reliability is greater for clustering algo-

algorithms due to conditioning at the cluster head in CSMA which helps sensor network protocols and improves energy efficiency when using B-MAC. FARMS which uses an ultra low-duty cycling MAC due to varying charging times performs well in a dense sensor network configuration and has significant application role.

Chapter 12

Experimental Setup

In this Chapter¹ we introduce the system model and derive some required results for the cross-layer power analysis. It is shown that the lifetime of a large sensor network follows a Power Law (Lowen & Teich, 1970) distribution.

12.1 System Model and Notation

Figure 12.1: (a) Cross Layer resource allocation and optimizer (b) WSN simulator

The design of interoperable sensors for the new wireless standards is a task for VLSI hardware and software (firmware) domains. Resource constrained prototypes are hard to deploy and debug so we use a cross-layer simulator due to its unique

¹Iyer, V. and Murthy, R. and Srinivas, M. B. and Hochet, B. C-ERROR Simulator for Development for Sensor and Location Aware Sensing Applications. In Proc. 3rd International Conference on Sensing Technology ICST.

deployment needs. These include unreliable wireless channels, remote sensing and non-replenishable energy resource and wastage of energy due to idleness and collisions in the underlying protocol. The sensor application goal is to use large number of sensors and use collaborative processing for local tasks. This type of distributed processing not only saves precious sensor's energy but makes it more reliable than using a few nodes and also creates a secure network. Simulator gives a controlled environment to deploy large amounts of nodes and measure faults due to resource bottlenecks at each cross functional layer of the application.

Simulation is performed at each abstraction layer such as energy management, network discovery or point to point communication. The same concepts can be extended to an emulator with common programming methods which once tested can be deployed into hardware running co-resident real-time OS with real drivers and IO's connected. Common goal is to find a micro-OS which interfaces with RS-232, Bluetooth, USB, GPRS or SDIO. This allows transparency to communicate with the external world and allows an easy testing platform with networked PC's. The communication to the PC or a laptop is through a central coordinator node, which typically has enough resources and helps manage the sensor network and works as a gateway node between sensing and IP based networks. The coordinator stores all the global parameters, which it needs to maintain the network and at the same time manages a data stream with a given communication rate. In contrast to other simulators that give a lot of standard statistics this method allows debugging of the resource constraints for custom sensing applications. It easily ports into real mote hardware and tests their reliability in one common framework.

The architecture combines the upper network layers which deal solely with distributed optimization with the frequent connectivity needs required by the lower layers to achieve balanced performance in terms of energy savings for sensing and

routing activity. Since the hardware layers are non-programmable, it is best to use the specification for low cost, low memory and low data rate requirements. The NetworkEmbedded Test-Bed infrastructure for management and operation of the Test-Bed gives information of the overall running tasks and also supplies management data of the state of the sensors such as expected events, drifting of the clocks, and low resource state indicators. The research presents implementation of different power-aware algorithms and their effects on routing and complexity of implementation.

It uses a JENNET/IEEE802.15.4 ZigBee development platform to validate the memory requirements and target needs used during abstracted simulation. The flexibility in setup allows the target platform to choose according to the needs such as network management, routing or real-time sensing functionality of the OS. This research summary discusses work in progress regarding opportunities and challenges related to cross-level simulation and optimization based on realistic scalable reliability. The cross level approach is shown in Figure 12.1(a), and the corresponding baseline architecture is shown in Figure 12.1(b). The respective timing, resource, computation complexity and energy model are shown in Figure 12.1(a).

What are the reliabilities that are built-in-to the system for such a test-bed? A real-time system responds in a (timely) predictable way to all individual unpredictable external stimuli arrivals. It is important to note that the average performance is not the issue.

12.2 Power-aware Complexity

The lifetime of sensor networks is typically factored into the resources it is deployed with, as by design it is unattended (i.e. no replacement of batteries) and it coexists

for many months to some years. The numbers of sensor nodes typically runs into hundreds to thousands in a large environmental monitoring application. As the number of nodes in such applications is enormous when compared to typical networks it uses clustering algorithms in which typically 20%-30% (Iyer *et al.*, 2008e) of the nodes aggregate the data of the remaining 70%-80% (Iyer *et al.*, 2008e) of the connected nodes. These cluster heads are data concentrators, which can be modeled as a device CODEC, compressor/decompressor. The sensors which are attached to the nodes typically sense temperature, humidity and light. It is true, however, that the sensor measurements in the operation region are spatially correlated (since many environmental phenomena are) they tend to be very similar. In a CODEC a probability model is used which gives the highest probability to the most frequently occurring values reported by the sensors within the same cluster. This allows transmitting peak values with least amount of bits as the underlying compression algorithm assigns least number of bit for frequently occurring values. This probability distribution is sent with the data values to the central coordinator. So each cluster head has a unique P_{Max} (Iyer *et al.*, December 2008) but not all cluster heads have the same measured value. As recent developments of VLSI(Spr, n.d.) and MEMS technologies have made it possible to package self-powered sensors and wireless radio components which together is capable of collecting and processing new sensor data for a period of many months to few years without replacing the internal batteries. The miniaturized sensors are sensitive to the available effective range to the energy consumed per bit. The **instantaneous drain** on the internal batteries is evident and the study shows that

$$\begin{aligned}
\text{Energy consumed per bit to transmit} &= \frac{100pj}{\text{bit}} m^2 \\
\text{Energy consumed to receive a bit} &= \frac{50nj}{\text{bit}} \\
\text{Transmit Energy} &= E_{amp} + d_{i,j}^2 \quad (12.1)
\end{aligned}$$

Where $\mathbf{d}$ is the distance to transmit between sensors $\mathbf{i}$ to sensor $\mathbf{j}$. From this we get the Power rule based on the distance d of nearest sensor to the farthest away sensor, substituting in the above equation (12.1) and summing up the total energy required for all transmissions within one meter, two meters, three meters, four meters and extending up to $(d-1)$ meters to a progressive sequence in equation (12.1) (as shown in Figure 12.2(a)).

Figure 12.2: (a) Shows fixed energy overhead with distance in transmission, (b) Plot of the theoretical expected lifetime using Power Law, being used to demonstrate ranking of popularity. To the right is the long tail, to the left are the few that dominate (also known as the 80-20 rule).

$$PowerLaw = 1^2 + 2^2 + 3^2 + 4^2 + \dots + (d-1)^2 + d^2 \quad (12.2)$$

To sum up the total energy consumption we can write it in the form of Power Law equation. (12.2)

$$PowerLaw = f(x) = ax^2 + o(x)^2 \quad (12.3)$$

Substituting d-distance for x and k number of bits transmitted, we equate it as in equation (12.3).

$$PowerLaw = f(d) = kd^2 + o(d)^2 \quad (12.4)$$

Taking Log both sides of equation (12.4),

$$\log(f(d)) = 2\log d + \log k \quad (12.5)$$

Notice that the expression in equation (12.5) has the form of a linear relationship with slope k, and scaling the argument induces a linear shift of the function, and leaves both the form and slope k unchanged. Plotting to the log scale as shown in Figure 12.2(b) we get a long tail showing a few nodes dominate the transmission power compared to the majority, similar to the Wikipedia reference 80-20 rule of Power Law (Iyer *et al.*, December 2008).

12.3 Scale Invariance Property

As novel sensor applications are deployed to provide reliable data over the life-time (Iyer *et al.*, December 2008) of the sensor network, with current routing algorithms (Iyer *et al.*, 2008e) which are dependent to communicate with a central coordinator there is an instantaneous drain on the sensors. A typical 9V battery communication for an RF sensor to transmit over 10 meters range will drain out as per the capacity table in reference (Lowen & Teich, 1970). As shown in the previous equation in logarithmic scale for point to point transmission, we can extend this by clustering c

nodes in the same range as shown in equation (12.6).

$$f(d) = kd^2 + o(d^2) \quad (12.6)$$

$$f(cd) = k(cd^2) = c^k f(d) \alpha f(d) \quad (12.7)$$

From equation (12.7) we can infer that the property is scale invariant even with clustering c nodes in a given radius k . This is validated from the simulation results later in the chapters.

12.4 Fault Rate

Large deployment of sensor networks that use an efficient distributed algorithm to select cluster heads which allows us to extend the lifetime (Iyer *et al.*, December 2008) to function without faults. The fault rate of such an algorithm can be defined as the residual percentage of good sensors when the network incurs faults due to resource drain. This is typically referred to as the sensor networks residual energy; if the fault rate is higher the cluster head selection algorithm is less optimal. The two-dimensional simulation model is expressed for distributed and passive cluster based routing. In the simulation setup the fault rate is measured for both the cases for algorithm complexity, multi-hop dependency, MAC layer (Ye *et al.*, 2006), (Polastre *et al.*, 2004) losses and Bit error rates.

We look for the optimal distributed threshold to select cluster heads c , out of all possible routing algorithms, that maximizes the probability of c , Cluster head selection with least percentage of good remaining sensors, number of faults given the original measurement M . The measurement estimate $\hat{M}$ is given by:

$$P_{max} = \frac{\text{Clustering Algorithm}}{\text{Residual energy of the network at the Lifetime calculation}} \quad (12.8)$$

Figure 12.3: Measurement metrics for node failures

By Bayes theorem this is equivalent to

$$P_{max} = \frac{\text{Residual energy of the network at Sim1, Sim2, Sim3 calculation}}{\text{Clustering Algorithm}} \times P_{\text{of a node to be a CH}} \quad (12.9)$$

12.5 Bayesian Classifier

Sensor networks are deployed in a dense configuration with its limited radio range and fixed non renewable energy resources due to computational/networking characteristics of sensor networks. To collaboratively use the limited resources distributed algorithms, select a single node which transmits serially using its UART

pre-processed sensed data information using many local resources. As the cost of radio transmission is much more than local sensing, the sensor network uses two different topologies to address the energy cost at the cross-layer stack. The network layers uses the upper layers assuming MAC layer abstraction to optimally pick cluster heads by using a fixed probability density function (pdf) of a network resource at the node, such as, remaining battery energy. This type of pdf is power-aware as it uses a collaborative function to minimize over use of network resources thus avoiding pre-mature node failures.

The MAC layer uses a k-neighborhood distance algorithm to find other nodes within its own limited range and uses a multi-hop schedule to the specific data transmitting node. This scheduling allows multi-hop nodes to use sleep cycles and lower their energy consumption while idling. These multi-hop algorithms use low-power listening and use a preamble to wake up nodes, sleep cycles when the transmitter is completely off and traffic based preamble to synchronize nodes to receive the data payload.

If $\theta_1, \theta_2, \theta_3$ are the metrics for power-aware simulation such as residual energy(*rei*), and if actual data is simulated then these are observed values by the sensors. As large scale sensor deployment is a dense deployment as the readings are correlated only an average θ_1 needs to be transmitted. As the clustering is based on the network layer which optimizes on radio range and not the sensing region is always approximated and corrected using some training samples using less number of bits to be transmitted. This is the fundamental design based on power-aware data model.

Figure 12.4: Estimation of CH selection error & MAC layer routing using Bayesian P^*

Table 12.1: Load simulation for lifetime routing when $N=100$ and $F=1$ (nano sec.)

Routing Algorithm	Tx	Idle	Rx
LEACH	10^9 seconds	10^9 seconds	10^9 seconds

12.6 Lifetime Calculation

To understand the layer III MAC duty cycle we use a micro kernel realtime clock, which allows us to measure different states of the MAC. It can be conveniently divided into Tx, Rx, Idle times for a given node's lifetime. Duty-cycling can be defined as

$$Lifetime = \frac{C_{batt} \times V \times 60 \times 60}{Energy_{Tx+Rx+Idle+Lis}} \times \text{duty cycle} \quad (12.10)$$

$$\text{Amortized dissipation} = \frac{Energy_{Tx+Rx+Idle}}{3} \quad (12.11)$$

Table 12.2: Energy consumption in duty cycle when $N=100$ and $F=1$

Routing Algorithm	Tx	Idle	Rx
LEACH	measured value	measured value	measured value

Substituting this amortized dissipation value for a standard battery of 2000 mAh into the lifetime equation with 0.01% duty cycle a 100 milli-second preamble MAC.

$$Lifetime = \frac{2000 \text{ mAh}}{\text{Amortized dissipation mAh}} \times 0.01\% \quad (12.12)$$

$$Lifetime = \text{Useful time in days (predicted for a AA battery)} \quad (12.13)$$

12.7 Compression Rate

It is well known that the Huffman algorithm (Huffman, 1952) finds a code minimizing average redundancy; this is so well known that the problem itself is often referred to as the "Huffman problem." The Huffman algorithm is a greedy algorithm built on the observation that the two least likely items will have the same length and can thus be considered siblings in the coding tree. A reduction can thus be made in which the two items of weights $w(i)$ and $w(j)$ can be considered as one with combined weight $w(i) + w(j)$, and the codeword of the combined item determines all but the last bit of each of the items combined, which are differentiated by this last bit. This reduction continues until there is one item left, and, assigning this item the null string, a code is defined for all input items. In the corresponding optimal code tree, the i^{th} leaf corresponds to the codeword of the i^{th} input item, and thus has weight $w(i)$, whereas the weight of parent nodes are determined by the combined weight of the corresponding merged item. Van Leeuwen gave an implementation of the Huffman algorithm that can be accomplished in linear time given sorted probabilities (Huffman, 1952). Shannon (Shannon, 1948) had previously shown that an optimal l_{opt} must satisfy.

$$H(p) = \sum p(i) l^{opt} l(i) < H(p) + 1 \quad (12.14)$$

or equivalently,

$$0 \leq \tilde{R}(l^{opt}, p) < 1 \quad (12.15)$$

Less well known is that simple changes to the Huffman algorithm solve several related coding problems which optimize for different objectives, shown to satisfy redundancy bounds of the form

$$\tilde{R}(p) \leq \tilde{L}(p, l^{opt}) < \tilde{H}(p) + 1 \quad (12.16)$$

$$0 \leq \tilde{R}(l^{opt}, p) < 1 \quad (12.17)$$

for some entropy measure $\tilde{H}$, cost measure $\tilde{L}$, and redundancy measure $\tilde{R}$. These bounds are the first of their kind for nontraditional Huffman codes, bounds that are functions of both entropy and $p(1)$, as in the traditional case (Gallager, 1978), (Prisco & Santis, 1994). However, they are not the first improved bounds for such codes; more sophisticated bounds on the optimal solution for one of these problems were given by Slepian & Wolf (Slepian & Wolf, 1975) for correlated sources.

Table 12.3: Summary of Notations for Analysis of Huffman Coding.

Symbols	Definition
$H(p)$	Normal entropy
$\tilde{H}(p)$	Number of nodes in the cluster
$\tilde{R}$	Correlated redundancy measure
$\tilde{L}$	Optimal Huffman codelength
$P(1)$	Huffman tree using pdf distribution

12.7.1 Slepian & Wolf Theorem

The Slepian-Wolf rate (Slepian & Wolf, 1975) region for two arbitrarily correlated sources x and y is bounded by the following inequalities. This theorem can be

adapted using equation (12.18)

$$R_x \geq H\left(\frac{x}{y}\right), R_y \geq H\left(\frac{y}{x}\right) \text{ and } R_x + R_y \geq H(x, y) \quad (12.18)$$

If the correlated sources are differing by a few bits, the possible number of codewords can be represented as 2^m where m = no. of faulty bits (Iyer *et al.*, 2008b). In our case $m=2$ as the parameters are distributed whilst collected locally at the cluster head.

12.7.2 Distributed Source Coding with Side Information

In sensors networks several measured values are sensed in a distributed manner and these are aggregated according to the users query. The goal of all the encoders is analogous to the previous section where it uses cosets. Equations (9.26-9.28) illustrate the bin formation to reduce the overall bits needed for transmission. Considering the case of distributed sensing application, the encoder is further designed with a machine learnable redundancy range, which is specific to each and every application. This mutually redundant measured range is correlated with sensors, which are in the same wireless range and connected to a parent. This information, also called side information is shared with the decoder. Owing to side information, even fewer number of bits are needed to represent the changing values coming from each cluster heads transmitting to the joint decoder. Encoder and decoder have access to the side information Y , which is correlated to X and can be represented by the equation (12.18). According to the Slepian-Wolf Theorem (Slepian & Wolf, 1975), established in 1971, the number of bits needed by using the theorem is lesser,

than the total entropy for both the two arbitrarily correlated sources $H(x)$, $H(y)$.

$$\begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{vmatrix} = 00 \quad (12.19)$$

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{vmatrix} = 10 \quad (12.20)$$

$$\begin{vmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{vmatrix} = 01 \quad (12.21)$$

$$\begin{vmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{vmatrix} = 11 \quad (12.22)$$

12.8 Probability Mass Function

In probability theory, a probability mass function (abbreviated *pmf*) is a function that gives the probability that a discrete random variable is exactly equal to some value (which occurs in a given static region of high density or a two dimensional model such as planar graphs). A *pmf* differs from a probability density function (abbreviated pdf) in that the values of a pdf, defined only for continuous random variables, are not probabilities as such. Instead, the integral of a pdf over a range of possible values $(a, b]$ gives the probability of the random variable falling within that range. As an example, if every node has its residual value as *rei* index then if a given node needs to optimally do a communication task it needs to find only the total energy it needs from its neighbors *rei* index's, which adds up to a total cost of 1. This is typically a local cumulative algorithm as it terminates in $O(1)$ operation.

12.9 Probability Density Function

In probability theory, a probability density function (abbreviated *pdf*), or density, of a random variable is a function which describes the density of probability at each point in the sample space (mostly drawn with an overall global distribution). The probability of a random variable falling within a given set is given by the integral of its density over the set. As an example, if every node has its residual value as *rei* index then if a given algorithm needs to find active cluster heads it looks for highest or better *rei* index from all the available active nodes to assign cluster heads. This is typically a distributed global cumulative scale algorithm as it terminates in $O(n)$ operations.

Part V

Summary

Chapter 13

Summary

This thesis deals with modeling data such as climate and extreme events and the performance of the model when tested with real observed low-cost sensor data. The two goals are hard to satisfy due to the small size of training sets available, the noise factor and assumptions used for a priori estimates. Due to these reasons it is hard to pick an ideal model and this research considers practically combining both data attributes and measurement errors by using an ensemble model. The ensemble model, when applied to multi-target regression of label-less streams, as the traditional sensor streams is expensive to manually label. The proposed model uses ensemble to divide and conquer the space and computation required by sensor network. Static streams use parallel sensor measurements to extract events of interest and mobile streams use a probability model to classify the sequential data. The chapters are organized into five parts. There are Introduction, Stream Processing, Mobility, Protocols and Summary.

Chapter 2 overviews the classification approach to sensor data quality and the types of cross-validation methods such as stacking, voting and ensembles used to increase the precision and accuracy of the base classifier. Our theoretical assumption show the ensemble classifier is a binomial distribution and if all the base classifiers have an error rate of < 0.5 the overall ensemble model can always have a better accuracy. It has been generally shown that in the literature the ensemble model

works well. In this work we further extend this assumption to multi-target label-less and multi-target regression learning.

As the datasets normally have their attributes as numeric types and some can have ordinal types it is challenging to select the good combination of attributes for the underlying learning algorithms. The algorithm is not biased for the specific dataset by doing so. This technique of sampling in sensor network specifies spatial or temporal features selections. Chapter 3 explains spatial sensor errors and fault tolerance called Brook-Iyengar algorithm, which uses weighted average (Brook & Iyengar, 1996). The same principle is applied to compress sensing (Baron *et al.*, 2005) using Wavelets decomposition and explains how loss of important spatial features can be avoided during high-level aggregation. Chapter 4 explains how a better model should be developed by using domain knowledge in understanding the rules using temporal features. Two datasets are explored: one in multi-media and the other in forest fire event logs collected over four years in Portugal. The first approach uses a compression technique to adapt to the number of bits represented, while the other method uses a classification approach and F-measure statistics to rank the training set to better explain the events. The work further extends the base classifiers by introducing Bayes Net, which does not rely on any specific prior assumption such as Naive Bayes making it a better model for new test samples.

Only clustering is considered an unsupervised learning algorithm compared to Bayes, Support Vector Machines and Decision trees. Chapter 5 introduces clustering using trees, which are called Data-cleaning trees. The DC-tree allows us to classify label-less datasets using number attributes to form clusters at the leaf nodes. The intermediate nodes of the DC-tree are nominal attributes, which learn a concept and help lower the training error. Any two datasets generated from the same process can be easily compared and given a quality flag and hence its rank. The splitting

of the nodes uses variance by calculating the f-test (Clark & Niblett, 1989). Due to f-test being a very sensitive test, further splitting of the nodes will help drastically reduce variance. When combined with techniques such as bagging and random forest for random attribute learning the label-less learning using a distance function. The overall DC-Tree ensemble reduces errors as the features used in DC-Tree clustering have correlated attributes. The more the number of pure clusters the higher the reliability of the stream, which has many observed temporal features closer to the ground truth event, which are being monitored.

Chapter 6 introduces a pre-processing step called SPOTLESS (Iyer *et al.*, 2013.) to account for missing and faulty sensor data. Unlike classification it uses local feature extraction (Krishnamachari & Iyengar, 2004) from reliable sensors and recreates the missing or faulty values. A matrix factorization algorithm with weight **learning** is used to recalculate the missing sensor row values. One of the assumptions that the sensor matrix has high redundancy and is sparse making the factorization technique less computationally expensive. The main challenge in the thesis is to come up with a metric for stream quality and SPOTLESS uses a normalization factors (**kappa**) from all the meta-data to allow easy comparison of local static streams and its quality measure. Chapter 7 gives an account of sequential data, which are generated from GPS based mobile sensors. Typically in sensor networks an event is detected by a combination of spatially placed sensor which observes with a probability. This gives rise to the concept of hidden states (Duda *et al.*, 2000) or virtual sensors. A hidden Markov model is used which allows to learn from the data its model parameters, if enough samples are available. As in Chapter 6 an ensemble model is used, but it is an ensemble of HMMs called HEAPS. The HEAPS model allows us to learn from mobile trajectories of different activities. The specific unnormalized probability likelihoods are calculated. These likelihoods are still comparable for

various activities being monitored such as walking, car driving and bus rides. The learning algorithm uses a labeled dataset and once trained for a given geographical coördinate system the HEAPS model can predict new test coördinates using MAP (Duda *et al.*, 2000) and categories it to the closest activity already learned with high- level of accuracy.

The domain of sensor networks involves more than data-cleaning such as wireless radios and collaborative computation processing. Some of the key concepts of mobility using wireless are covered in Chapter 8 using different mobility models and how fading can be minimized. Chapter 9 takes an applied research approach when compared to the remaining part of this thesis. Many simulation models are built to perform routing at the network layer, MAC layer and renewable sensor cluster heads. The simulation allows us to understand the scalability of large sensor networks with a given density of nodes, transmission range of the radio and wireless fading to provide and keep up a minimum level of Qos.

13.1 Putting it all together

The quality issues mentioned between Chapter 2 and Chapter 5 define the sub-problems in stream processing with respect of spatial and temporal features extraction for label-less learning. Chapters 6 and Chapter 8 take an ensemble classification approach to static and mobile generated streams using SPOTLESS, DC-Tree, and HEAPS ensemble Frameworks. Figure 13.1 summarizes the major components in label-less learning with quality metric and the matching chapters of each system are shown. The datasets used are from Intel labs INTEL, Microsoft Research GEOLIFEWEATHER, UCI Machine learning repository FOREST FIRES and WEATHER. As the model of ensemble came together the SPOTLESS emphasized on row-wise

Figure 13.1: The sensor data-matrix of the overall Data-Cleaning System, with relevant chapters for Static and Mobile Streams as indicated.

feature extraction and DC-Tree uses column-wise concept learning. Both the frameworks use ensembles to minimize the RMSE error and lower the RMSE when the quality of the stream is higher. QR Factorization is used to learn the missing values on the sensor data matrix from a given set of observed sensor measurements. Using the sensor data provided by INTEL dataset we are able to restore close to 95% of the missing values by learning from observed features. This amounts to 27% more information quality due to spatial consistence compared to the noisy stream. The information gained here can be compared to the precision of the sensor measurement, which not only helps to provide a better estimate but also avoids faulty sensor reading to be reported. The false alarm rates can be reduced by using the reliable sensors whose feature measurements can correct the faulty readings. The disadvantage of this approach is that we do not know how it will perform with future samples. Accuracy of such streams may not be useful in quality labeling until features with more complexity are used.

The WEATHER dataset is used to further learn complicated features useful to classify higher quality streams. Using descriptive column-wise attributes such as outlook and windy the numeric attributes such as temperature and, humidity are estimated. For a small dataset the multi-attribute ensemble methods does not provide a good feature extractor as it actually does perform lower than single target classifiers. When used with larger training samples the multi-attribute ensemble model always outperformed the single target classifier feature extractor. This is attributed to the ensemble model, which preserves the correlated relationships within the attributes, inducing higher order features. The information gained by using higher order features allows better performance on unseen samples, and improving model confidence. Table 5.24 shows that the accuracy is close to 95% of FWI features used for large-fires. This emphasizes that complex features allow higher ac-

curacy when used even with simple classifiers. Using the ensemble model the large fires classification accuracy is increased by using multi-attribute learning from 0.859 to 0.9963 as shown in Table 6.2 using single target learning.

The crossover point between model training and testing helps to understand the model confidence. The model confidence is based on unseen data and care should be taken not to overfit the model. Chapter 6 on ensembles introduces several pre-processing steps in data-cleaning to avoid overfitting.

So far the datasets did not have time-stamp or any sort of sequential features to it. When sequential data is studied it is necessary to learn the hidden or latent states. Chapter 8 introduces how to learn hidden states from observed data using hidden Markov Models. This ensemble model extends the previous approach to data-cleaning for static streams and learns the state transition probability matrix. Given the current state the new state is predicted from its transition probability. If the state belongs to one of the known HMMs then it is labelled with the known activity. The dataset uses continuous variables, which are GPS coordinates. The GEOLIFE dataset can be learned by a HMM with two states. New GPS samples can be classified with high accuracy using the HMM ensemble (HEAPS) with up to 95% accuracy if number of training samples are large enough and the training window is configurable. The GEOLIFE dataset has atleast 30 minutes of GPS recordings.

Data travels in a multi-hop fashion in sensor networks from source to destination. Due to wireless interference and channel noise the data payload is susceptible to high BER errors. This can cause many retries during decoding and drain the sensor network. So the framework introduced in Chapter 4 initially and 7 in detail 4 uses forward error correction codes. The forward correction codes have built in redundancy to recover from 1 to 2 bit errors. As sensor networks payloads are measured values these can be processed as the values may not vary more than a

few bits. The results from Figures 7.4 (b,d,f) show that the error are significantly reduced for mobile and static receivers affected by fading and shadowing. The use of reversible compressed streams further help to minimize the packet length and provide re-synchronization during burst errors. The Huffman code is reserved by combining suffix code by EXOR as shiwn in Figures 4.1,4.2 to the stream symbol. This allows a simple way to correct block errors in sensor networks. The research methodologies chapters introduce the system model and simulation bench marks for the experiments. Figure 12.2 show the lifetime of a large deployed sensor networks follows a Power Law distribution or 80%-20% rule. The cross-layer error rates that determining the total useful lifetime for known and unknown node densities are estimated to show that best-effort MAC level implementation is bounded by top-down design by twice its error rate as shown in Figure 12.4. The appendix has useful codes references for computational algorithmic complexity and various low level network protocols.

13.2 Toward the Future

Data-cleaning and forward error correction are well studied in the context of wireless sensor networks. However, there is still room for real-time stream processing and events. Intelligent sensor provides data-driven algorithms in Machine Learning to predict process variables which are hard to measure. There are several practical estimation issues when dealing with meta-data such as in wireless uses soft-phy and inherent unreliability of the measured data from low cost sensors. In the design we presented: (i) data-cleaning method, (ii) feature selection, (iii) non-linearity of the generation process (iv) real-time input versus network training. We presented a framework to calculate Fire Weather Index (FWI) from sensor readings, which will

enable members to predict and better understand the problem of natural fires. The initial tests show that the FWI indexes calculated by using this real-time framework were able to track and predict category-I fires with finer resolution and accuracy. We also showed that the data generated from many such sensors can produce streams, which even though smaller and easier to compute but needs to be computed more frequently due to uncertain fire conditions. Real-time calculations addresses two dynamic ranges in the FWI indexes: (i) the actual diurnal moisture is measured as an on-line quantity (ii) the wetting rate has a resolution in log scale, which represents the true estimate of Fine Fuel Moisture Code (FFMC) present in the ground (iii) the drying rate has a resolution in log scale, which represents a closer day-to-day estimate of FFMC after the wet state from rain. These finer trends are close to the ever changing conditions seen in fires over 24-hour period. Large in a vast unprotected forest will eventually produce vast amounts of data, which further needs processing to accurately estimate a non-linear measurement to a point estimate in the day-to-day trends calculation of FWI-indexes. The framework uses Storm or SQLStream incorporating hourly FWI-algorithms such as Monte Alegre and Monte Alegre Chant.

Similarly collecting data from cheap cell phones and mobile sensors can help us understand how activity recognition can be learned. This has a direct knowledge of how common diseases spread by far in a community.

Bibliography

- Akyildiz, Ian F., Lee, Won-Yeol, & Vuran, Mehmet C. 2006. NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey.
- Balakrishnan, Hari, Madden, Samuel, & Bychkovsky, Vladimir. 2006. A Measurement Study of Vehicular Internet Access Using In Situ WiFi Networks. *MobiCom'06*. @csail.mit.edu <http://cartel.csail.mit.edu>.
- Baron, Dror, Duarte, Marco F., B.Wakin, Michael, Sarvotham, Shriram, & Baraniuk, Richard G. 2005. Distributed Compressive Sensing. Proc, Rice University, Texas: Pre-print. USA, 2005.
- Bartlett, Peter L., & Wegkamp, Marten H. 2008. Classification with a Reject Option Using a Hinge Loss. *Journal of Machine Learning*.
- Bell, Timothy C., Cleary, John G., Witten, Ian H., & Compression, Text. 2002. Prentice Hall Advanced Reference Series.
- Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer 2006.
- Blum, A., & Langley, P. 1997. Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, Dec., 245–271. *AI Intelligence*, 97(,):245–271.
- Breiman, L. 2001. Random Forests. *Machine Learning*, 5–32.
- Bristow, Q. 1983. Airborne g-ray spectrometry in uranium exploration principles and current practice. *Int. J. Appl. Radiat. Isot.* 1, 199229.
- Brook, Richard, & Iyengar, S. S. Sitharama. 1996. Robust Distributed Computing and Sensing Algorithm. *ACM*. 1996.
- Calibration, NE Portugal:, & to Fire Management, Application. 2002. Using the Canadian Fire Weather Index (FWI) in the Natural Park of Montesinho. }.
- Chen, Binbin, Zhou, Ziling, Zhao, Yuda, & Yu, Haifeng. 2010. *Efficient Error Estimating Coding: Feasibility and Applications*. SIGCOMM.

- Clancy, T. Charles, & Arbaugh, William A. 2006. *Measuring Interference Temperature*. Internal Report. Department of Computer Science, University of Maryland, College Park. Laboratory for Telecommunication Sciences, Department of Defense.
- Clark, P., & Niblett, T. 1989. *The CN2 Algorithm*. Machine Learning. 3(4):261–284.
- Clouqueur, Thomas, Ramanathan, Parameswaran, Saluja, Kewal K., & ching Wang, Kuang. 2001. *Value-fusion versus decision-fusion for fault-tolerance in collaborative target detection in sensor networks*. Page pp. of: In Proceedings of Fourth International Conference on Information Fusion.
- Cortes, C., & Vapnik, V. 1995. *Support vector Network*. Machine Learning, 273–297. 20, 1995.
- Cortez, Paulo, & Morais, Anibal. 2010. *A Data Mining Approach to Predict Forest Fires Using Meteorological Data*. Internal Report. University of Minho, Portugal.
- CSAIL. 2012. Sensor Model Language (SensorML). [//db.csail.mit.edu/labdata/labdata.html](http://db.csail.mit.edu/labdata/labdata.html). [Online; accessed May 15th, 2012].
- Demsar, D., Dzeroski, S., Larsen, T., Struyf, J., Axelsen, J., Pedersen, M., & Krogh, P. 2006. *Using Multi-objective Classification to Model Communities of Soil Microarthropods*. Ecological Modelling 191(1) (2006) 131. 143.
- Duda, Richard O., Hart, Peter E., & Stork, David G. 2000. *Pattern Classification* (2nd Edition). Wiley-Interscience.
- Frank, A., & Asuncion, A. 2010. UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science. <http://archive.ics.uci.edu/ml>. [Online; accessed Dec 12/1/2012].
- Gallager, R. G. 1978. *Variations on a Theme by Huffman*. IEEE Trans. on Information Theory, 24(6), 668–674.
- Girod, Bernd. 1999. *Bidirectionally Decodable Streams of Prefix Code-Words*. IEEE Communications Letters, VOL. 3, NO. 8, August 1999. "Related work on Huffman coding."
- Goldberg, Andrew B., & Zhu, Xiaojin. 2010. *Transduction with Matrix Completion: Three Birds with One Stone*. NIPS, 757–765. pp.
- Grosanic, S., Dinkel, A., , & Piszczek, S. 2012. *Optimized Traffic Control with Benchmarked Road Weather Data*. SIRWEC 2012, 23–25. Helsinki, May 2012.

- Group, IETF Working. 2005. *Multiparty Multimedia Session Control specification (MMUSIC)*.
- Harb, H., , & Chen, L. 2001. *A Query by Example Music Retrieval Algorithm*. Maths-Info department. Ecole Centrale de Lyon. France, 2001.
- Hawking, David, , & Robertson, Stephen. 2003. *On Collection Size and Retrieval Effectiveness*. CSIRO mathematical and Information Sciences. Canberra, Australia.
- Heinzelman, Wendi Rabiner, Chandrakasan, Anantha, & Balakrishnan, Hari. 2000. *Energy-Efficient Communication Protocol for Wireless Microsensor Networks*. Page 8020 of: HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8. Washington, DC, USA: IEEE Computer Society.
- [http://www. 2005. R-project. org/](http://www.R-project.org/). [Accessed May 20th, 2012].
- Huffman, D. A. 1952. *A Method for the Construction of Minimum-Redundancy Codes*. Proceedings of the IRE, **40**(9), 1098–1101.
- IETF. 2008. Audio/Video Transport (AVT) specification Working Group. <http://datatracker.ietf.org/wg/avt/charter/>. [Online; May 15th, 2008].
- Information, Geographic. 2003. *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys)*. November 5-7. Redwood, CA.
- Iyengar, S. S., Parameshwaran, Nandan, Phoha, Vir V., Balakrishnan, N., , & Okoye, Chuka D. 2010. *Fundamentals of Sensor Network Programming: Applications and Technology*. ISBN: 978-0-470-87614-5 Hardcover 350 pages December 2010. Wiley-IEEE Press.
- Iyengar, S. Sitharama, & R, Richard. 2012. *Second Edition: Distributed Sensor Networks*. Image and Sensor Signal Processing Series, Chapman & Hall/CRC Computer & Information Science Serie. ; s Published:September 24, 2012 by Chapman and Hall/CRC - 764 Pages.
- Iyer, V., Murthy, Rama, & Srinivas., M.B. *Performance Analysis of Power-aware Algorithms and Pattern Classification in Large Sensor Networks*. In: SENNET 2007, VIT, University of Applied Sciences, Karlsruhe, Germany and Indian Nuclear Society, Vellore, India.
- Iyer, V., Murthy, Rama, & Srinivas., M.B. 2006a. *Information Processing of Wireless Sensor Networked Motes by Harvesting Residual Power*. In: Wireless for Next Generation Networks (WONGEN).

- Iyer, V., Murthy, Rama, & Srinivas., M.B. 2006b. *Simulation based optimization for Managing Lifecycle of Sensor Networks*. In: ICSCN-2006, Hyderabad, India.
- Iyer, V., Murthy, Rama, & Srinivas., M.B. 2007. *Distributed Wireless Sensor Network Architecture: Fuzzy Logic based Sensor Fusion*. In: Proceedings of the 5th EUSFLAT conference, Ostrava, Czech Republic.
- Iyer, V., Murthy, R., Srinivas, M. B., & Hochet, B. 2008a (Nov. 30 2008–Dec. 3). *C-ERROR simulator for development for sensor and location aware sensing applications*. Pages 192–199 of: Proc. 3rd International Conference on Sensing Technology ICST.
- Iyer, V., Garimella, R. M., & Srinivas, M. B. 2008b (21–22 Oct.). *Entropy Based Variable Rate Compression for Low-bandwidth Multi-media Streams*. Pages 139–145 of: Proc. First International Conference on Distributed Framework and Applications DFmA.
- Iyer, V., Murthy, Rama, & Srinivas., M.B. 2008c. *Entropy based Variable Rate Compression for Low-bandwidth Multi-Media Streams*. In: The International Conference on Distributed Frameworks and Applications Oct, 2008, Penang Maylasia (DFmA).
- Iyer, V., Garimella, R. M., & Srinivas, M. B. 2008d (25–31 Aug.). *Min Loading Max Reusability Fusion Classifiers for Sensor Data Model*. Pages 480–485 of: Proc. Second International Conference on Sensor Technologies and Applications SENSORCOMM.
- Iyer, V., Iyengar, S.S., Murthy, R., & Srinivas, M. B. 2009a. *Computational Aspects of Sensor Network Protocols (Distributed Sensor Network Simulator)*. Pages 69–91 of: Special Issue on Modern Sensing Technologies Journal (IFSA). *International Frequency Sensor Association (IFSA)*., vol. 6. *Sensors and Transducers Journal*.
- Iyer, V., Iyengar, S. S., Murthy, Rama, Balakrishnan, N., & Phoha, V. 2009b (17-21 June.). *Distributed Source Coding for Sensor Data Model and Estimation of Cluster Head Errors using Bayesian and K-Near Neighborhood Classifiers in Deployment of Dense Wireless Sensor Networks*. In: Proc. Third International Conference on Sensor Technologies and Applications SENSORCOMM.
- Iyer, V., Iyengar, S. S., Balakrishnan, N., Phoha, V., & Srinivas, M. B. 2009c (17–19 Feb.). *FARMS: Fusionable ambient renewable MACS*. Pages 169–174 of: Proc. IEEE Sensors Applications Symposium SAS.
- Iyer, V., Iyengar, S.S., Murthy, R., Srinivas, M. B., N. Balakrishnan Vir Phoha, & Hochet. 2009d. *Multi-hop Scheduling and Local Data Link Aggregation Dependant Qos in Modeling and Simulation of Power-aware Wireless Sensor Networks*. In:

Proc. Wireless Communications and Mobile Computing Conference, IWCMC09 June 21-24, Leipzig, Germany.

Iyer, Vasanth, & Iyengar, S. Sitharama. 2011. *MODELING UNRELIABLE DATA AND SENSORS: Using F-measure Attribute Performance with Test Samples from Low-cost Sensors ICDMW 2011*. In: IEEE International Conference on Data Mining-Workshops (ICDMW. Š11), 2011 Vancouver, CA).

Iyer, Vasanth, & Singh, Dhananjay. 2010. *Sensor Fusion and its Applications, Chapter: "Distributed Compressed Sensing of Sensor Data."* Edited by Ciza Thomas. Hard cover. 488 pages, Publisher: Sciyo, Chapters published August 16, 2010 under CC BY-NC-SA 3.0 license DOI: 10.5772/3302.

Iyer, Vasanth, Murthy, G.Rama, & Srinivas., M.B. 2008e. *Environmental measurement OS for a tiny CRF-STACK used in Wireless Network*. Pages 72–86 of: Special Issue: Modern Sensing Technologies. ISSN 1726-5479 ©2006 by IFSA.

Iyer, Vasanth, Murthy, G. Rama, , & Srinivas., M. B. 2008f. *Min Loading Max Reusability Fusion. Classifiers for Sensor Data Model*. In *Proc: Second international Conference on Sensor Technologies and Applications, Volume 00 (August 25 - 31, SENSORCOMM 2008)*.

Iyer, Vasanth, Murthy, Rama, & Srinivas., M.B. 2008g. *Training Data Compression Algorithms and Reliability in Large Wireless*. *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008)*.

Iyer, Vasanth, Iyengar, S. S., Balakrishnan, N., Phoha, Vir., & Srinivas., M. B. 2009e. *FARMS: Fusionable. 9781–4244. Ambient Renewable MACS*, In: *SAS-2009, IEEE-2787, 17th-19th Feb, New Orleans, USA*.

Iyer, Vasanth, Iyengar, S. S., Murthy, Rama, Balakrishnan, N., & Phoha., V. 2009f. *Source coding for sensor data model and estimation of cluster head errors using bayesian and k-near neighborhood classifiers in deployment of dense wireless sensor networks*. In *Proc: Third International Conference on Sensor Technologies and Applications SENSORCOMM* June. 2009, 17–21.

Iyer, Vasanth, Iyengar, S. Sitharama, Murthy, Garmiela Rama, Parameswaran, N., Singh, Dhananjay, , & Srinivas, Mandalika B. 2010a. *Effects of Channel SNR in Mobile Cognitive Radios and Coexisting Deployment of Cognitive Wireless Sensor Networks*. IEEE IPCCC, 294–300. Albuquerque, New Mexico, USA.

Iyer, Vasanth, Iyengar, S. Sitharama, Murthy, Garmiela Rama, Parameswaran, N., Singh, Dhananjay, , & Srinivas, Mandalika B. 2010b. *Effects of Channel SNR in Mobile Cognitive Radios and Coexisting Deployment of Cognitive Wireless Sensor Networks*. Pages 294–300 of: Performance Computing and Communications

- Conference (IPCCC). *2010 IEEE 29th International, Albuquerque, New Mexico, USA.*
- Iyer, Vasanth, Iyengar, S.S., Murthy, G. Rama, , & Srinivas, M.B. 2010c. *INSPIRE-DB: Intelligent Networks Sensor Processing of Information Using Resilient Encoded-Hash DataBase.* SENSORCOMM 2010, 363–368. *The Fourth International Conference on Sensor Technologies and Applications, Venice.*
- Iyer, Vasanth, Iyengar, S.S., Paramesh, N., Murthy, G. Rama, , & Srinivas, M.B. 2011. *Machine Learning and Data Mining Algorithms for Predicting Accidental Small Forest Fires.* SENSORCOMM 2011, 21–27. *August, 2011 - French Riviera, France.*
- Iyer, Vasanth, Iyengar, S. Sitharama, Pissinou, Niki, & Ren, Shaolei. 2013.. Pages 487–492 of: SPOTLESS: Similarity Patterns Of Trajectories in Label-IEss Sensor Streams. *IEEE PERCOM IQ2S: Workshop on Information Quality and Quality of Service for Pervasive Computing.*
- Iyer, Vasanth, Iyengar, S.S., & Pissinou, Niki. 2013. *Intelligent Sensor Networks: Networks Signal Processing and Machine Learning. Using Event Log Performance and F-measure Attribute Selection.* *Publisher Taylor and Francis.*
- Iyer, Vasanth, Murthy, G.Rama, & Srinivas., M.B. December 2008. *Training Data Compression Algorithms and Reliability in Large Wireless Sensor Networks.* Page 480–485 of: *International Journal on Smart Sensing and Intelligent Systems, VOL. 1, NO. 4.*
- Iyer, Vasanth, Iyengar, S.S, Murthy, G. Rama, & Srinivas, M.B. December 2009. *SenseSIM: Sensor Network Simulator.* *Ph.D. thesis, Louisiana State University, Baton Rouge, US.*
- Jensen, Arne, & la Cour-Harbo, Anders. 2001. *Ripples in Mathematics.* Springer Verlag 2001. 246 pp. Softcover.
- Jiang, Yuan. 2013. *A Practical Guide to Error-Control Coding Using MATLAB.*
- Jones, Owen, Maillardet, Robert, , & Robinson, Andrew. 2009. *Introduction to Scientific Programming and Simulation Using R.* Chapman and Hall/CRC, 0972–6. Boca Raton, FL, 20.
- Kocev, Dragi, Vens, Celine, Struyf, Jan, & Džeroski, Sašo. 2007. *Ensembles of Multi-Objective Decision Trees.* **4701**, 624–631.
- Kohavi, R., & John, G. 1997. *Wrappers for Feature Selection.* *Artificial Intelligence, Dec.,* 273–324. *AI Intelligence, 97(,):273–324.*

- Krishnamachari, Bhaskar, & Iyengar, Sitharama. 2004. *Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks*. Page VOL. 53 NO. 3 of: *IEEE Transactions on Computers*. *IEEE Trans. on Computers*.
- Kumar, P. T. Krishna, Iyengar, S.S, & Phoha, Vir. 2008. *Classification of radio elements using mutual information: A tool for geological mapping*. 1016/j.jag.2007.12.001.
- Kumar, Shailesh, Ghosh, Joydeep, & Crawford, Melba M. 2000. *Best-Bases Feature Extraction Algorithms for Classification of Hyperspectral Data*. *IEEE Transactions on geoscience and remote sensing*, VOL. 39, NO. 7, JULY 2000.
- Kumar, Vasanth, Kumar, Krishna, & Mohamad, Shafilluah. 1985. *Video Printer for 8-bit Microprocesor Kit*. Computer Society of India. "Student paper CSI85 - camera ready version submitted".
- Li, Q., & Zheng, Y. 2008. *Et Al*. Pages 298–307 of: *Mining User Similarity Based on Location History*. In *Proc. of GIS. Š08 (Santa Ana, CA, Nov. 2008)*. *ACM Press*.
- Lowen, S., & Teich, M. 1970. *Power-law shot noise*. vol. 36.
- Lowen, S., & Teich., M. 1970. *Power-Law Shot Noise*, *IEEE Trans Inform Volume 36*, Pages. 1302–1318. 1970.
- Miano, John. 1999. *Compressed Image File Formats: JPEG*. *ACM Press/Addison Wesley Longman*. *GIF, XBM, BMP*. *ACM Press/Addison Wesley Longman*.
- Microsoft. 2008. The Microsoft Active stream format website. www.microsoft.com/asf. [Online; accessed Dec 12/1/2005].
- Mitchell, Tom M. 1997. *Machine Learning*. MaGRAW-Hill Publications 1997.
- Moffat, Alistair. 2003. *Compression and Coding Algorithms*. The Springer International Series in Engineering and Computer Science.
- Murthy, Garimella Rama, & Iyer, Vasanth. 2007. *Distributed Wirelss Sensor Network Architecture: Fuzzy Logic based Sensor Fusion*. Pages 71–78 VOL II of: *EUSFLAT. Proceedings of the 5th EUSFLAT Conference, Ostrava, Czech Republic*.
- of Waikato, U. 2008. WEKA Machine learning software. <http://www.cs.waikato.ac.nz/~ml/weka>. [Online; May 15th, 2011].
- OGCNET. 2008. Sensor Model Language (SensorML). <http://www.ogcnetwork.net/SensorML>. [Online; accessed 12/1/2012].

- Polastre, Joseph, Hill, Jason, & Culler, David. 2004. *Versatile low power media access for wireless sensor networks*. Pages 95–107 of: *SenSys: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM.
- Prisco, Roberto De, & Santis, Alfredo De. 1994. *On the redundancy achieved by Huffman codes*. *Information Sciences*, **88**, 131–148.
- QualNet, CA. 2008. WEKA Machine learning software. <http://web.scalable-networks.com/>. [Online; accessed Dec 12/1/2012].
- Quinlan, J.R. 1996. *Bagging boosting. and C4.5*, in: *Proceedings 13th National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR.
- Rao, K. R. 1996a. *Techniques and Standards for Image. Video. and Audio Coding by Prentice Hall PTR*, 1st edition (July 8, 1996).
- Rao, K. R. 1996b. *Techniques and Standards for Image, Video, and Audio Coding*. Prentice Hall.
- Reddy, T. Bheemarjuna, Manoj, B.S., , & Rao, Ramesh. 2008 (Nov.). *Time-Based Sampling Strategies for Multi-channel Wireless Traffic Characterization in Tactical Cognitive Networks*. In: *Proc. of IEEE MILCOM 2008*.
- RFC-2326. 1998. *RTSP-Real Time Streaming Protocol Specification Document*.
- Sayood, Khalid. 2012. *Introduction to Data Compression*. *Kaufmann Series in Multimedia Information and Systems*.
- Schulzrinne, H. 1998. *Columbia U. A. Rao. Netscape and R. Lanphier, Real-Networks. Internet Engineering Task Force MMUSIC WG INTERNET-DRAFT specification*.
- Shannon, Claude Elwood. 1948. *A mathematical theory of communication*. The Bell System Technical Journal, **27**.
- Singh, Dhananjay, & Kim, Daeyeoul. 2010. *MMSP: Design a Novel Micro Mobility Sensor Protocol for Ubiquitous Communication*. *UBICOMM2010. Florence, Italy, October 25 30, 2010*. Dhananjay Singh and Daeyeoul Kim, *Performance Analysis of Gateway Discovery Techniques: IPv6 based Wireless Sensor Networks, INTERNET 2010, Valencia, Spain, September 20 25*.
- Singh, Dhananjay, & Lee, Hoon-Jae. 2010a. *Design and Performance Evaluation of a Proactive Micro Mobility Protocol for Mobile Networks*. In: *Book: Hand-held Computing for Mobile Commerce: Applications, Concepts and Technologies*. edited by DrWen-Chen Hu and Dr Yanjun Zuo.

- Singh, Dhananjay, & Lee, Hoon-Jae. 2010b. *Global Healthcare Monitoring Applications: IPv6 Based Wireless Sensor Networks*. In: Book: E-Healthcare Systems And Wireless Communications: Current And Future Challenges.
- Slepian, D., & Wolf, J. K. 1975. *Noiseless coding of correlated information sources*. Pages 226–228 of: IEEE Trans. Inform. Theory. *IEEE Trans. Inform. Theory*.
- Smucker, Mark D., Allan, James, , & Carterette, Ben. 2007. *A Comparison of Statistical Significance Tests for Information Retrieval Evaluation*. Department of Computer Science. *University of Massachusetts, Amherst*.
- Tan, Pang-Ning, Steinbach, Michael, & Kumar, Vipin. 2006. *Introduction to Data Mining*. Addison-Wesley. ISBN-13: 9780321321367 l2006 T Addison-Wesley T Cloth, 769 pp, Published 05/02/2005.
- Tanenbaum, Andrew. 2011. *Computer Networks*. Prentice Hall Fourth Edition - Section 7.4, Multimedia. *Fourth Edition - Section 7.4, Multimedia*.
- UCLA. 2000. GlomoSim, wireless sensor simulator. <http://pcl.cs.ucla.edu/projects/glomosim/>. [Online; accessed June 2007.].
- van Rijsbergen, C.J. 1979. *Information Retrieval*. Butterworths. London, second edition, 1979.
- Vasanth Iyer, Lakshmi Rajamani. 2005. *Coding Models for Data Streams*. Computer Society of India Conference, Hyderabad, India. *Student Paper, Osmania University, College of Engineering*.
- Vens, Celine, , & Costa, Fabrizio. 2011. *Random Forest Based Feature Induction*. In: Proceedings- IEEE, ICDM.
- Wan, Raymond, , & Moffat, Alistair. 2001. *Interactive Phrase Browsing Within Compressed Text*. Computer Science and Software Engineering. *University of Melbourne*. 2001.
- Witten, I. H., Moffat, A., & Bell, T. C. 1999.. *Managing Gigabytes: Compressing and Indexing Documents and Images*. San Francisco, CA, USA: Morgan Kaufmann, San Francisco, second edition, 1999.
- Witten, Ian H., & Frank, Eibe. 2005. *Data Mining, Practical machine learning*. Elsevier 2005.
- Witten, Ian H., Moffat, Alistair, & Bell, Timothy C. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*. *Morgan Kaufmann Publishing(1999 2nd Edition)*.

- Wu, X., Yu, P. S., Piatetsky-Shapiro, G., Cercone, N., Lin, T. Y., Kotagiri, R., & Wah, B. W. 2003. : *Data Mining: How Research Meets Practical Development?* Knowledge and Information Systems. 5(2), 248-261.
- Ye, Wei, Silva, Fabio, & Heidemann, John. 2006. *Ultra-low duty cycle MAC with scheduled channel polling*. New York, NY, USA: SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems.
- Yu, Lu, Schwier, J.M., Craven, R.M., Brooks, R.R., & Griffin, C. 2013. *Inferring Statistically Significant Hidden Markov Models*. Knowledge and Data Engineering, IEEE Transactions on, **25**(7), 1548–1558.
- Zadeh, L. 1965. *Fuzzy Sets*. Information Control 8, 338–353. 1965.
- Zadeh, Lotfi A. 1996. *Fuzzy sets, fuzzy logic, and fuzzy systems*. River Edge, NJ, USA: World Scientific Publishing Co., Inc.
- Zheng, Y. 2008a. *Et Al*. Pages 247–256 of: *Learning Transportation Modes from Raw GPS Data for Geographic Applications on the Web*. In Proceedings of WWW 2008. April 2008), ACM Press:.
- Zheng, Y. 2008b. *Managing and understanding your past life over maps*. Pages 211–212 of: *GeoLife: Managing and Understanding Your Past Life Over Maps*. In Proceedings of MDM. Š09, (Beijing China, April 2008), IEEE Press:.
- Zheng, Yu, Zhang, Lizhu, Xie, Xing, & Ma, Wei-Ying. 2012. *GeoLife GPS Trajectories*, MSR Asia. <http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/>. [Online; accessed Dec 12/1/2012].

Appendix A

Appendices

Algorithms

Algorithm A.1 K-means

Algorithm 1 (K-means clustering)
begin initialize $n, c, \mu_1, \mu_2, \dots, \mu_c$
do classify n samples according to nearest μ_i
recompute μ_i
until no change in μ_i
return $\mu_1, \mu_2, \dots, \mu_c$
end

Algorithm A.2 Online-K-Means On-line Clustering

Make initial guesses for the means $m_1, m_2, \dots, m_k$ Set the counts $n_1, n_2, \dots, n_k$ to zero
Until interrupted
Acquire the next example, x
If m_i is closest to x
Increment n_i
Replace m_i by $m_i + (1/n_i) * (x - m_i)$
end-if
end-until

Algorithm A.3 CONSISTENT HASHING <VALUE>: Sensor Measured values in the network

- 1: The sensed data values measured can occur in a range $0 - R$ in a distributed way.
 - 2: **if** $\Upsilon = \text{Local Maxima}$ **then**
 - 3: **create a hash KEY similar to the previous algorithm.**
 - 4: **else**
 - 5: return
 - 6: **end if**
 - 7: Each node $1 - N$ will now have a unique $(64 - 128)$ -bit hash key K which is arranged in a circle, i.e if it goes over the maximum value it will wrap to the first node. As shown in algorithm:
-

Algorithm A.4 HUFFMAN ALGORITHM: Modified priority Q for merging probability

- 1: **while** !*EndofPriorityQ* **do**
 - 2: **Select new item from Q**
 - 3: **if** $p(n) < 2p(n - 1)$ **then**
 - 4: **Merge** $Q = P(n) + P(n - 1)$
 - 5: **else**
 - 6: **Skip(n) Q**
 - 7: **end if**
 - 8: **end while**
-

Algorithm A.5 NORMAL SQL: node 1 to 100 will provide 100 network messages

- 1: SELECT X from sensornetwork region
 - 2: WHERE X GT. limit1 LT limit2
-

Algorithm A.6 DISTRIBUTED SQL: node 1 to 100 will aggregate 1 network messages and many local aggregation with hop $< \lg 100$

- 1: SELECT Υ from sensornetwork region
 - 2: WHERE Υ GT. limit1 LT limit2
-

CSMA Algorithm

```
/*
 * GloMoSim is COPYRIGHTED software. Release 2.02 of GloMoSim is available
 * at no cost to educational users only.
 *
 * Commercial use of this software requires a separate license. No cost,
 * evaluation licenses are available for such purposes; please contact
 * info@scalable-networks.com
 *
 * By obtaining copies of this and any other files that comprise GloMoSim2.02,
 * you, the Licensee, agree to abide by the following conditions and
 * understandings with respect to the copyrighted software:
 *
 * 1. Permission to use, copy, and modify this software and its documentation
 *    for education and non-commercial research purposes only is hereby granted
 *    to Licensee, provided that the copyright notice, the original author's
 *    names and unit identification, and this permission notice appear on all
 *    such copies, and that no charge be made for such copies. Any entity
 *    desiring permission to use this software for any commercial or
 *    non-educational research purposes should contact:
 *
 *    Professor Rajive Bagrodia
 *    University of California, Los Angeles
 *    Department of Computer Science
 *    Box 951596
 *    3532 Boelter Hall
 *    Los Angeles, CA 90095-1596
 *    rajive@cs.ucla.edu
 *
 * 2. NO REPRESENTATIONS ARE MADE ABOUT THE SUITABILITY OF THE SOFTWARE FOR ANY
 *    PURPOSE. IT IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY.
 *
 * 3. Neither the software developers, the Parallel Computing Lab, UCLA, or any
 *    affiliate of the UC system shall be liable for any damages suffered by
 *    Licensee from the use of this software.
 *
 * Ported to Microsoft Visual C++ August, 2008, Vasanth Iyer
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <math.h>

#include "api.h"
#include "structmsg.h"
#include "fileio.h"
#include "message.h"
#include "mac.h"
#include "csma.h"
#include "network.h"
#include "..\radio\radio_acnoise.h"
#include "..\network\nwip.h"

#define noDEBUG
#define noQDEBUG

static /*inline*/
RadioStatusType RadioStatus(GlomoNode* node, GlomoMacCsma* csma)
{
    return GLOMO_RadioGetStatus(node, csma->myGlomoMac->interfaceIndex);
}

/*
 * NAME:          MacCsmaHandlePromiscuousMode.
 *
 * PURPOSE:       Supports promiscuous mode sending remote packets to
 *                upper layers.
 *
 * PARAMETERS:    node, node using promiscuous mode.
 *                frame, packet to send to upper layers.
 *
 * RETURN:        None.
 *
 * ASSUMPTION:    node != NULL.
 */
static
void MacCsmaHandlePromiscuousMode(GlomoNode *node, Message *frame)
{
    GLOMO_MsgRemoveHeader(node, frame, sizeof(CsmaHeader));
    NetworkIpSneakPeekAtMacPacket(node, frame);
}
```

```

    GLOMO_MsgAddHeader(node, frame, sizeof(CsmaHeader));
}

/*
 * NAME:          MacCsmaDataXmit.
 * PURPOSE:       Sending data frames to destination.
 * PARAMETERS:    node, node sending the data frame.
 * RETURN:        None.
 * ASSUMPTION:    node != NULL.
 */

static
void MacCsmaXmit(GlomoNode *node, GlomoMacCsma *csma)
{
    char clockStr[GLOMO_MAX_STRING_LENGTH];
    Message *msg;
    NODE_ADDR nextHopAddress;
    NetworkQueueingPriorityType priority;

    CsmaHeader *hdr;

    assert(csma->status == CSMA_STATUS_XMIT);

    /*
     * Dequeue packet which was received from the
     * network layer.
     */

    NetworkIpOutputQueueDequeuePacket(
        node, csma->myGlomoMac->interfaceIndex,
        &msg, &nextHopAddress, &priority);

    if (msg == NULL)
    {
#ifdef QDEBUG
        printf("CSMA: Queue should not be empty...\n");
#endif
        return;
    }

    if (msg->error == TRUE) //--vi for debug
        return;

    csma->status = CSMA_STATUS_IN_XMITING;
    csma->timer.flag = CSMA_TIMER_OFF | CSMA_TIMER_UNDEFINED;

    /*
     * Assign other fields to packet to be sent
     * to radio layer.
     */

    GLOMO_MsgAddHeader(node, msg, sizeof(CsmaHeader));
    hdr = (CsmaHeader *) msg->packet;
    hdr->destAddr = nextHopAddress;
    hdr->sourceAddr = node->nodeAddr;
    hdr->priority = priority;

    GLOMO_RadioStartTransmittingPacket(
        node, csma->myGlomoMac->interfaceIndex,
        msg, hdr->destAddr, FALSE, 0);

    if (hdr->destAddr == ANY_DEST) {
        csma->pktsSentBroadcast++;
    }
    else {
        csma->pktsSentUnicast++;
    }
}

/*
 * NAME:          MacCsmaSetTimer.
 * PURPOSE:       Set a timer for node to expire at time timerValue.
 * PARAMETERS:    node, node setting the timer.

```



```

*          timerType, what type of timer is being set.
*          delay, when timer is to expire.
*
* RETURN:    None.
* ASSUMPTION: node != NULL.
*/

static
void MacCdmaSetTimer(
    GlomoNode *node, GlomoMacCdma* csma, int timerType, clocktype delay)
{
    Message      *newMsg;
    int          *timerSeq;
    int          seq_num;

    csma->timer.flag = CSMA_TIMER_ON | timerType;
    csma->timer.seq++;

    assert((timerType == CSMA_TIMER_BACKOFF) ||
           (timerType == CSMA_TIMER_YIELD));

    newMsg = GLOMO_MsgAlloc(node, GLOMO_MACLAYER, 0,
                           MSG_MAC_TimerExpired);

    GLOMO_MsgSetInstanceId(newMsg, csma->myGlomoMac->interfaceIndex);

    GLOMO_MsgInfoAlloc(node, newMsg, sizeof(csma->timer.seq));
    timerSeq = (int *) newMsg->info;

    seq_num = *((int *) newMsg->info);
    printf("SEQ_NUM OF MESS : %d\n", seq_num);

    *timerSeq = csma->timer.seq;

    GLOMO_MsgSend(node, newMsg, delay);
}

/*
* NAME:      MacCdmaYield.
*
* PURPOSE:   Yield so neighboring nodes can transmit or receive.
*
* PARAMETERS: node, node that is yielding.
*             holding, how int to yield for.
*
* RETURN:    None.
* ASSUMPTION: node != NULL.
*/

static
void MacCdmaYield(GlomoNode *node, GlomoMacCdma *csma, clocktype holding)
{
    assert(csma->status == CSMA_STATUS_YIELD);

    MacCdmaSetTimer(node, csma, CSMA_TIMER_YIELD, holding);
}

/*
* NAME:      MacCdmaBackoff.
*
* PURPOSE:   Backing off sending data at a later time.
*
* PARAMETERS: node, node that is backing off.
*
* RETURN:    None.
* ASSUMPTION: node != NULL.
*/

static
void MacCdmaBackoff(GlomoNode *node, GlomoMacCdma *csma)
{
    clocktype randTime;
    assert(csma->status == CSMA_STATUS_BACKOFF);

    randTime = (pc_nrand(node->seed) % csma->BOmin) + 1;
    csma->BOmin = csma->BOmin * 2;

    if (csma->BOmin > csma->BOmax) {
        csma->BOmin = csma->BOmax;
    }

    csma->BOTimes++;
}

```

```

    MacCsmaSetTimer(node, csma, CSMA_TIMER_BACKOFF, 10*MICRO_SECOND*randTime);
}

static //inline//
void CheckRadioStatusAndSendOrBackoff(GlomoNode* node, GlomoMacCsma* csma) {
    /* Carrier sense response from radio. */

    char TimeStringInSecond[GLOMO_MAX_STRING_LENGTH]; //JEFF

    printf("INITIAL: %d\n", csma->initialBackoff);

    if ((RadioStatus(node, csma) == RADIO_IDLE) &&
        (csma->status != CSMA_STATUS_IN_XMITING))
    {

        if (csma->initialBackoff != FALSE) //JEFF (INITIAL BO)
        {
            printf("Node: %d: INITIAL BACKOFF!\n", node->nodeAddr);
            csma->initialBackoff = FALSE;
            csma->status = CSMA_STATUS_BACKOFF;

            GLOMO_PrintClockInSecond(simclock(), TimeStringInSecond);

            printf("TIME: %15s\n >> ", TimeStringInSecond); //SHOW TIME

            MacCsmaBackoff(node, csma);

        }
        else
        {
            printf("Node: %d: AFTER INITIAL BACKOFF!\n", node->nodeAddr);
            //////////////////////////////////////////////////JEFF//////////////////////////////////////
            //After checking for the RadioStatus and CSMA status, sampling//
            //the channel to determine whether the channel is free        //
            //Calls ClearChannelAssessment function                        //
            //////////////////////////////////////////////////
            if (ClearChannelAssessment(node, csma->myGlomoMac->interfaceIndex) == TRUE)
            {
                printf("CHANNEL CLEAR!\n");
                csma->status = CSMA_STATUS_XMIT;
                csma->initialBackoff = TRUE;
                MacCsmaXmit(node, csma);

            }

            else
            {
                printf("CHANNEL IS NOT CLEAR!\n");
                MacCsmaBackoff(node, csma);

            }

        } // (INITIAL BO)

    }
    else {
        if (!NetworkIpOutputQueueIsEmpty(
            node, csma->myGlomoMac->interfaceIndex))
        {
            csma->status = CSMA_STATUS_BACKOFF;
            MacCsmaBackoff(node, csma);

        }
    }
}

/*
 * NAME:          MacCsmaNetworkLayerHasPacketToSend.
 *
 * PURPOSE:       In passive mode, start process to send data; else return;
 *
 * RETURN:        None.
 *
 */

void MacCsmaNetworkLayerHasPacketToSend(GlomoNode *node, GlomoMacCsma *csma)
{
    if (csma->status == CSMA_STATUS_PASSIVE) {

        //csma->status = CSMA_STATUS_INITIAL_BACKOFF; //SET INITIAL BACKOFF
        CheckRadioStatusAndSendOrBackoff(node, csma);
    } //if//
}

```

```

/*
 * NAME:          MacCsmPassive.
 *
 * PURPOSE:       In passive mode, check whether there is a local packet.
 *                If YES, send data; else return;
 *
 * PARAMETERS:    node, node that is in passive state.
 *
 * RETURN:        None.
 *
 * ASSUMPTION:    node != NULL.
 */

static
void MacCsmPassive(GlomoNode *node, GlomoMacCsm *csma)
{
    Message *newMsg;

    if ((csma->status == CSMA_STATUS_PASSIVE) &&
        (!NetworkIpOutputQueueIsEmpty(node, csma->myGlomoMac->interfaceIndex)))
    {
        MacCsmNetworkLayerHasPacketToSend(node, csma);
    }
}

/*
 * NAME:          MacCsmPrintStats
 *
 * PURPOSE:       Print MAC layer statistics.
 *
 * PARAMETERS:    node.
 *
 * RETURN:        None.
 *
 * ASSUMPTION:    node != NULL.
 */

static
void MacCsmPrintStats(GlomoNode *node, GlomoMacCsm* csma)
{
    char buf[GLOMO_MAX_STRING_LENGTH];

    sprintf(buf, "Number of packets from network: %ld",
            csma->pktsToSend);
    GLOMO_PrintStat(node, "MacCSMA", buf);

    sprintf(buf, "Number of packets lost due to buffer overflow: %ld",
            csma->pktsLostOverflow);
    GLOMO_PrintStat(node, "MacCSMA", buf);

    sprintf(buf, "Number of UNICAST packets output to the channel: %ld",
            csma->pktsSentUnicast);
    GLOMO_PrintStat(node, "MacCSMA", buf);

    sprintf(buf, "Number of BROADCAST packets output to the channel: "
            "%ld", csma->pktsSentBroadcast);
    GLOMO_PrintStat(node, "MacCSMA", buf);

    sprintf(buf, "Number of UNICAST packets received clearly: %ld",
            csma->pktsGotUnicast);
    GLOMO_PrintStat(node, "MacCSMA", buf);

    sprintf(buf, "Number of BROADCAST packets received clearly: %ld",
            csma->pktsGotBroadcast);
    GLOMO_PrintStat(node, "MacCSMA", buf);
}

/*
 * FUNCTION       MacCsmInit
 * PURPOSE        Initialization function for CSMA protocol of MAC layer.
 *
 * Parameters:
 *   node:         node being initialized.
 *   nodeInput:    structure containing contents of input file
 */

void MacCsmInit(
    GlomoNode *node, int interfaceIndex, const GlomoNodeInput *nodeInput)
{
    BOOL      retVal;
    GlomoMacCsm *csma = (GlomoMacCsm *) pc_malloc(sizeof(GlomoMacCsm));
    char      buf[GLOMO_MAX_STRING_LENGTH];

```

```

    assert(csma != NULL);

    memset(csma, 0, sizeof(GlomoMacCdma));
    csma->myGlomoMac = node->macData[interfaceIndex];
    csma->myGlomoMac->macVar = (void *)csma;

    csma->timer.flag = CSMA_TIMER_OFF | CSMA_TIMER_UNDEFINED;
    csma->timer.seq = 0;

    csma->status = CSMA_STATUS_PASSIVE;
    csma->BOmin = CSMA_BO_MIN;
    csma->BOmax = CSMA_BO_MAX;
    csma->BOtimes = 0;

    csma->pktsToSend = 0;
    csma->pktsLostOverflow = 0;

    csma->pktsSentUnicast = 0;
    csma->pktsSentBroadcast = 0;

    csma->pktsGotUnicast = 0;
    csma->pktsGotBroadcast = 0;

    csma->initialBackoff = TRUE;
}

void MacCdmaReceivePacketFromRadio(
    GlomoNode* node, GlomoMacCdma* csma, Message* msg)
{
    GlomoNetworkIp* ipLayer = (GlomoNetworkIp*)node->networkData.networkVar;

    if (csma->status == CSMA_STATUS_IN_XMITTING) {
        GLOMO_MsgFree(node, msg);
        return;
    } //if//

    switch (csma->status) {
    case CSMA_STATUS_PASSIVE:
    case CSMA_STATUS_CARRIER_SENSE:
    case CSMA_STATUS_BACKOFF:
    case CSMA_STATUS_YIELD: {
        int interfaceIndex = csma->myGlomoMac->interfaceIndex;
        CdmaHeader *hdr = (CdmaHeader *) msg->packet;
        if (hdr->destAddr == node->nodeAddr) {
            csma->pktsGotUnicast++;
        }
        else if (hdr->destAddr == ANY_DEST) {
            csma->pktsGotBroadcast++;
        }

        if ((hdr->destAddr == node->nodeAddr) ||
            (hdr->destAddr == ANY_DEST))
        {
            NODE_ADDR lastHopAddress = hdr->sourceAddr;

            GLOMO_MsgRemoveHeader(node, msg, sizeof(CdmaHeader));

            if (ipLayer->routingProtocolChoice == ROUTING_PROTOCOL_DIFFUSION)
                { //Non-IP protocols like directed diffusion
                    NetworkReceivePacketFromMacLayer(node, msg, lastHopAddress);
                }
            else
                { //IP protocol --vi
                    NetworkIpReceivePacketFromMacLayer(node, msg, lastHopAddress);
                }
        }
        else {
            if (node->macData[interfaceIndex]->promiscuousMode) {
                MacCdmaHandlePromiscuousMode(node, msg);
            }
            GLOMO_MsgFree(node, msg);
        }
        break;
    }
    default:
        GLOMO_MsgFree(node, msg);
        printf("MAC.CSMA: Error with node %u, status %ld.\n",
            node->nodeAddr, csma->status);
        //assert(FALSE); abort(); --vi disable foe debug
    } //switch//
}

void MacCdmaReceiveRadioStatusChangeNotification(
    GlomoNode* node,

```

```

    GlomoMacCdma* csma,
    RadioStatusType oldRadioStatus,
    RadioStatusType newRadioStatus)
{
    if (oldRadioStatus == RADIO_TRANSMITTING) {
        assert(newRadioStatus != RADIO_TRANSMITTING);
        assert(csma->status == CSMA_STATUS_IN_XMITING);

        csma->BMin = CSMA_BO_MIN;
        csma->BMax = CSMA_BO_MAX;
        csma->BTimes = 0;
        csma->status = CSMA_STATUS_YIELD;
        MacCdmaYield(node, csma, (clocktype)CSMA_TX_DATA_YIELD_TIME);
    } // if //
}

/*
 * FUNCTION      MacCdmaLayer
 * PURPOSE       Models the behaviour of the MAC layer with the CSMA protocol
 *               on receiving the message enclosed in msg.
 *
 * Parameters:
 *   node:        node which received the message
 *   msg:          message received by the layer
 */

void MacCdmaLayer(GlomoNode *node, int interfaceIndex, Message *msg)
{
    /*
     * Retrieve the pointer to the data portion which relates
     * to the CSMA protocol.
     */

    GlomoMacCdma *csma = (GlomoMacCdma *)node->macData[interfaceIndex]->macVar;
    int seq-num;

    printf("Node %d - MacCdmaLayer\n", node->nodeAddr);

    assert(msg->eventType == MSG_MAC_TimerExpired);

    seq-num = *((int *) msg->info);

    printf("SEQ = %d\n", seq-num);

    GLOMO_MsgFree(node, msg);

    if ((seq-num < csma->timer.seq) ||
        ((csma->timer.flag & CSMA_TIMER_SWITCH) == CSMA_TIMER_OFF)) {
        return;
    }

    if (seq-num > csma->timer.seq) {
        assert(FALSE);
    }

    assert((((csma->timer.flag & CSMA_TIMER_TYPE) ==
               CSMA_TIMER_BACKOFF) ||
            ((csma->timer.flag & CSMA_TIMER_TYPE) == CSMA_TIMER_YIELD)));

    switch(csma->timer.flag & CSMA_TIMER_TYPE) {
    case CSMA_TIMER_BACKOFF:
    {
        Message *newMsg;

        csma->timer.flag = CSMA_TIMER_OFF | CSMA_TIMER_UNDEFINED;

        CheckRadioStatusAndSendOrBackoff(node, csma);

        break;
    }

    case CSMA_TIMER_YIELD:
        csma->timer.flag = CSMA_TIMER_OFF | CSMA_TIMER_UNDEFINED;
        csma->status = CSMA_STATUS_PASSIVE;
        MacCdmaPassive(node, csma);
        break;

    default:
        assert(FALSE); abort();
        break;
    } // switch //
}

```

```

/*
 * FUNCTION      GLOMO_MacFinalize
 * PURPOSE       Called at the end of simulation to collect the results of
 *               the simulation of the CSMA protocol of MAC Layer.
 *
 * Parameter:
 *   node:        node for which results are to be collected.
 */

void MacCsmaFinalize(GlomoNode *node, int interfaceIndex)
{
    GlomoMacCsma* csma = (GlomoMacCsma *)node->macData[interfaceIndex]->macVar;

    if (node->macData[interfaceIndex]->macStats == TRUE) {
        MacCsmaPrintStats(node, csma);
    }
}

/** JEFF
 * FUNCTION      ClearChannelAssessment
 * PURPOSE       Called at the end of simulation to collect the results of
 *               the simulation of the CSMA protocol of MAC Layer.
 *
 * Parameter:
 *   node:        node for which results are to be collected.
 */

BOOL ClearChannelAssessment(GlomoNode *node, int interfaceIndex)
{
    BOOL IsChannelClear;

    IsChannelClear = CheckChannelStatus(node, interfaceIndex);
    return IsChannelClear;
}

```

B-MAC Algorithm

```

/*
 * Glomosim is COPYRIGHTED software. It is freely available without fee for
 * education, or research, or to non-profit agencies. No cost evaluation
 * licenses are available for commercial users. By obtaining copies of this
 * and other files that comprise GloMoSim, you, the Licensee, agree to abide
 * by the following conditions and understandings with respect to the
 * copyrighted software:
 *
 * 1. Permission to use, copy, and modify this software and its documentation
 * for education, research, and non-profit purposes is hereby granted to
 * Licensee, provided that the copyright notice, the original author's names
 * and unit identification, and this permission notice appear on all such
 * copies, and that no charge be made for such copies. Any entity desiring
 * permission to incorporate this software into commercial products or to use
 * it for commercial purposes should contact:
 *
 * Professor Rajive Bagrodia
 * University of California, Los Angeles
 * Department of Computer Science
 * Box 951596
 * 3532 Boelter Hall
 * Los Angeles, CA 90095-1596
 * rajive@cs.ucla.edu
 *
 * 2. NO REPRESENTATIONS ARE MADE ABOUT THE SUITABILITY OF THE SOFTWARE FOR ANY
 * PURPOSE. IT IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY.
 *
 * 3. Neither the software developers, the Parallel Computing Lab, UCLA, or any
 * affiliate of the UC system shall be liable for any damages suffered by
 * Licensee from the use of this software.
 *
 * Ported to Microsoft Visual C++ August, 2008, Vasanth Iyer
 */

// Use the latest Parsec version (if this gets a compiler error).
//-----
//
// File: user_mac.pc

```

```

// By: Jay Martin (jmartin@cs.ucla.edu)
// Objectives: Provide simple skeleton routines so that users can
//             more easily hook their protocols into GloMoSim without
//             modifying the changing "mac.pc" file itself.
//             This file should not change in the GloMoSim distribution
//             and should be replaced by the user modified file after each
//             GloMoSim update.
//
//-----

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

#include "Glomo.h"
#include "timer.h"
#include "resource.h"

#include "api.h"
#include "mac.h"
#include "bmac.h"

#include "structmsg.h"
#include "fileio.h"
#include "message.h"
#include "network.h"
#include "verifypointer.h"
#include "../network/nwip.h"

// The GLOMOSIM protocol number is a "unsigned short" that must be
// unique within the layers. So pick a random one to get near zero
// probability of conflict.

#define USER_DEFINED_MAC_PROTOCOLNUMBER 123

void MacInitUserMacProtocol(
    GlomoNode* node,
    const GlomoNodeInput* nodeInput,
    const char* macProtocolName,
    int interfaceIndex)
{
    if (strcmp(macProtocolName, "USER-DEFINED-PROTOCOL") == 0) {
        node->macData[interfaceIndex]->macProtocol =
            USER_DEFINED_MAC_PROTOCOLNUMBER;

        // Put function to create your datastructures and
        // initialize your protocol here. For Example,

        // YourMacProtocolInit(node, nodeInput, interfaceIndex,
        // (YourMacDataType**)node->macData[interfaceIndex]->macVar);
    }
    else if (strcmp(macProtocolName, "BMAC") == 0)
        {
            node->macData[interfaceIndex]->macProtocol = USER_DEFINED_MAC_PROTOCOLNUMBER;
            MacBmacInit(node, interfaceIndex, nodeInput);
        }
    else {
        fprintf(stderr, "CONFIG.IN Error: Unknown Mac Protocol Type "
            "%s.\n", macProtocolName);
        assert(FALSE); abort();
    }
}

void MacHandleUserMacProtocolEvent(
    GlomoNode* node, int interfaceIndex, Message* msg)
{
    // switch (msg->protocolType) {
    // case USER_DEFINED_MAC_PROTOCOLNUMBER: { Disable IP based routing for user defined MAC's

        MacBmacLayer(node, interfaceIndex, msg);

        // Put function to handle GloMoSim Events for your protocol here.
        // For Example,

        // YourRoutingProtocolEventHandler(
        //     node, (YourMacDataType*)node->macData[interfaceIndex]->macVar, msg);
    // }
    // default: --vi disable for user mac
    // printf("GloMoSim: Mac Protocol %d Not Handled\n",
    //     msg->protocolType);
    // assert(FALSE); abort();
    // }/*switch*/
}

```

```

// B-MAC -----START

/*
 * NAME:          MacBmacHandlePromiscuousMode.
 *
 * PURPOSE:       Supports promiscuous mode sending remote packets to
 *                upper layers.
 *
 * PARAMETERS:    node, node using promiscuous mode.
 *                frame, packet to send to upper layers.
 *
 * RETURN:        None.
 *
 * ASSUMPTION:    node != NULL.
 */

void MacBmacHandlePromiscuousMode(GlomoNode *node, Message *frame)
{
    GLOMO_MsgRemoveHeader(node, frame, sizeof(BmacHeader));
    NetworkIpSneakPeekAtMacPacket(node, frame);
    GLOMO_MsgAddHeader(node, frame, sizeof(BmacHeader));
}

/*
 * NAME:          MacBmacDataXmit.
 *
 * PURPOSE:       Sending data frames to destination.
 *
 * PARAMETERS:    node, node sending the data frame.
 *
 * RETURN:        None.
 *
 * ASSUMPTION:    node != NULL.
 */

void MacBmacXmit(GlomoNode *node, GlomoMacBmac *bmac)
{
    char clockStr[GLOMO_MAX_STRING_LENGTH];
    Message *msg;
    NODE_ADDR nextHopAddress;
    NetworkQueueingPriorityType priority;

    BmacHeader *hdr;

    assert(bmac->status == BMAC_STATUS_XMIT);

    /*
     * Dequeue packet which was received from the
     * network layer.
     */

    NetworkIpOutputQueueDequeuePacket(
        node, bmac->myGlomoMac->interfaceIndex,
        &msg, &nextHopAddress, &priority);

    VERIFY_ISREADPOINTER(msg->packet);

    if (msg == NULL)
    {
        /*--vi Debug fix
        #ifdef QDEBUG
            printf("BMAC: Queue should not be empty...\n");
        #endif
        */
        return;
    }

    if (msg->packetSize == 0)
    {
        return;
    }

    if (msg->error == TRUE)
    {
        return;
    }

    bmac->status = BMAC_STATUS_IN_XMITING;
    bmac->timer.flag = BMAC_TIMER_OFF | BMAC_TIMER_UNDEFINED;

    /*
     * Assign other fields to packet to be sent

```



```

    * to radio layer.
    */

    GLOMO_MsgAddHeader(node, msg, sizeof(BmacHeader));
    hdr = (BmacHeader *) msg->packet;
    hdr->destAddr = nextHopAddress;
    hdr->sourceAddr = node->nodeAddr;
    hdr->priority = priority;

    GLOMO_RadioStartTransmittingPacket(
        node, bmac->myGlomoMac->interfaceIndex,
        msg, hdr->destAddr, FALSE, 0);

    if (hdr->destAddr == ANY_DEST) {
        bmac->pktsSentBroadcast++;
    }
    else {
        bmac->pktsSentUnicast++;
    }
}

/*
 * NAME:          MacBmacSetTimer.
 *
 * PURPOSE:       Set a timer for node to expire at time timerValue.
 *
 * PARAMETERS:    node, node setting the timer.
 *                timerType, what type of timer is being set.
 *                delay, when timer is to expire.
 *
 * RETURN:        None.
 *
 * ASSUMPTION:    node != NULL.
 */

void MacBmacSetTimer(
    GlomoNode *node, GlomoMacBmac* bmac, int timerType, clocktype delay)
{
    Message      *newMsg;
    int          *timerSeq;
    int seq_num;

    bmac->timer.flag = BMAC_TIMER_ON | timerType;
    bmac->timer.seq++;

    assert((timerType == BMAC_TIMER_BACKOFF) ||
           (timerType == BMAC_TIMER_YIELD));

    newMsg = GLOMO_MsgAlloc(node, GLOMO_MAC_LAYER, 0,
        MSG_MAC_TimerExpired);

    GLOMO_MsgSetInstanceId(newMsg, bmac->myGlomoMac->interfaceIndex);

    GLOMO_MsgInfoAlloc(node, newMsg, sizeof(bmac->timer.seq));
    timerSeq = (int *) newMsg->info;

    // Jeff
    seq_num = *((int *) newMsg->info);
    printf("SEQ_NUM OF MESS : %d\n", seq_num);

    // Jeff

    *timerSeq = bmac->timer.seq;

    //GLOMO_MsgSend(node, newMsg, delay);
    // Protocol timings of PC simulation --vi
    GLOMO_MsgSend(node, newMsg, delay*10*MICRO_SECOND);
}

/*
 * NAME:          MacBmacYield.
 *
 * PURPOSE:       Yield so neighboring nodes can transmit or receive.
 *
 * PARAMETERS:    node, node that is yielding.
 *                holding, how int to yield for.
 *
 * RETURN:        None.
 */

```

```

* ASSUMPTION:  node != NULL.
*/

void MacBmacYield(GlomoNode *node, GlomoMacBmac *bmac, clocktype holding)
{
    assert(bmac->status == BMAC.STATUS_YIELD);

    MacBmacSetTimer(node, bmac, BMAC.TIMER_YIELD, holding);
}

/*
* NAME:          MacBmacBackoff.
* PURPOSE:       Backing off sending data at a later time.
* PARAMETERS:    node, node that is backing off.
* RETURN:        None.
* ASSUMPTION:    node != NULL.
*/

void MacBmacBackoff(GlomoNode *node, GlomoMacBmac *bmac)
{
    clocktype randTime;
    assert(bmac->status == BMAC.STATUS_BACKOFF);

    randTime = (pc_nrand(node->seed) % bmac->BOmin) + 1;
    bmac->BOmin = bmac->BOmin * 2;

    if (bmac->BOmin > bmac->BOmax) {
        bmac->BOmin = bmac->BOmax;
    }

    bmac->BOTimes++;

    MacBmacSetTimer(node, bmac, BMAC.TIMER_BACKOFF, randTime);
}

void CheckRadioStatusAndSendOrBackoff(GlomoNode* node, GlomoMacBmac* bmac) {
    /* Carrier sense response from radio. */

    char TimeStringInSecond[GLOMO_MAX_STRING_LENGTH]; //JEFF

    printf("INITIAL: %d\n", bmac->initialBackoff);

    if ((RadioStatus(node, bmac) == RADIO_IDLE) &&
        (bmac->status != BMAC.STATUS_IN_XMITING))
    {
        if (bmac->initialBackoff != FALSE) //JEFF (INITIAL BO)
        {
            printf("Node: %d: INITIAL BACKOFF!\n", node->nodeAddr);
            bmac->initialBackoff = FALSE;
            bmac->status = BMAC.STATUS_BACKOFF;

            GLOMO_PrintClockInSecond(simclock(), TimeStringInSecond);

            printf("TIME: %15s\n >> ", TimeStringInSecond); //SHOW TIME

            MacBmacBackoff(node, bmac); //Jeff
        }
        else
        {
            printf("Node: %d: AFTER INITIAL BACKOFF!\n", node->nodeAddr);
            ////////////////////////////////////////////////////////////////////JEFF//////////////////////////////////////////////////////////////////
            //After checking for the RadioStatus and BMAC status, sampling//
            //the channel to determine whether the channel is free        //
            //Calls ClearChannelAssessment function                        //
            ////////////////////////////////////////////////////////////////////
            if (ClearChannelAssessment(node, bmac->myGlomoMac->interfaceIndex) == TRUE)
            {
                printf("CHANNEL CLEAR!\n"); //Jeff

                bmac->status = BMAC.STATUS_XMIT;
                bmac->initialBackoff = TRUE;
                MacBmacXmit(node, bmac);
            }
        }
    }
}

```

```

                                else
                                {
                                    printf("CHANNEL IS NOT CLEAR!\n");
                                    MacBmacBackoff(node, bmac);
                                }
                                // Jeff // Jeff

                                }//JEFF(INITIAL BO)

                                }
                                else {
                                    if (!NetworkIpOutputQueueIsEmpty(
                                        node, bmac->myGlomoMac->interfaceIndex))
                                    {
                                        bmac->status = BMAC.STATUS_BACKOFF;
                                        MacBmacBackoff(node, bmac);
                                    }
                                }
                                }

/*
 * NAME:          MacBmacNetworkLayerHasPacketToSend.
 * PURPOSE:       In passive mode, start process to send data; else return;
 * RETURN:        None.
 */
void MacBmacNetworkLayerHasPacketToSend(GlomoNode *node, GlomoMacBmac *bmac)
{
    if (bmac->status == BMAC.STATUS_PASSIVE) {
        //bmac->status = BMAC.STATUS_INITIAL_BACKOFF;//SET INITIAL BACKOFF
        CheckRadioStatusAndSendOrBackoff(node, bmac);
    }//if//
}

/*
 * NAME:          MacBmacPassive.
 * PURPOSE:       In passive mode, check whether there is a local packet.
 *                If YES, send data; else return;
 * PARAMETERS:    node, node that is in passive state.
 * RETURN:        None.
 * ASSUMPTION:    node != NULL.
 */
void MacBmacPassive(GlomoNode *node, GlomoMacBmac *bmac)
{
    Message *newMsg;

    if ((bmac->status == BMAC.STATUS_PASSIVE) &&
        (!NetworkIpOutputQueueIsEmpty(node, bmac->myGlomoMac->interfaceIndex)))
    {
        MacBmacNetworkLayerHasPacketToSend(node, bmac);
    }
}

/*
 * NAME:          MacBmacPrintStats
 * PURPOSE:       Print MAC layer statistics.
 * PARAMETERS:    node.
 * RETURN:        None.
 * ASSUMPTION:    node != NULL.
 */
void MacBmacPrintStats(GlomoNode *node, GlomoMacBmac* bmac)
{
    char buf[GLOMO_MAX_STRING_LENGTH];

    sprintf(buf, "Number of packets from network: %ld",
        bmac->pktsToSend);
    GLOMO_PrintStat(node, "MacBMAC", buf);
}

```

```

        sprintf(buf, "Number of packets lost due to buffer overflow: %ld",
            bmac->pktsLostOverflow);
        GLOMO_PrintStat(node, "MacBMAC", buf);

        sprintf(buf, "Number of UNICAST packets output to the channel: %ld",
            bmac->pktsSentUnicast);
        GLOMO_PrintStat(node, "MacBMAC", buf);

        sprintf(buf, "Number of BROADCAST packets output to the channel: "
            "%ld", bmac->pktsSentBroadcast);
        GLOMO_PrintStat(node, "MacBMAC", buf);

        sprintf(buf, "Number of UNICAST packets received clearly: %ld",
            bmac->pktsGotUnicast);
        GLOMO_PrintStat(node, "MacBMAC", buf);

        sprintf(buf, "Number of BROADCAST packets received clearly: %ld",
            bmac->pktsGotBroadcast);
        GLOMO_PrintStat(node, "MacBMAC", buf);
    }

/*
 * FUNCTION      MacBmacInit
 * PURPOSE       Initialization function for BMAC protocol of MAC layer.
 *
 * Parameters:
 *     node:      node being initialized.
 *     nodeInput: structure containing contents of input file
 */

void MacBmacInit(
    GlomoNode *node, int interfaceIndex, const GlomoNodeInput *nodeInput)
{
    BOOL        retVal;
    GlomoMacBmac *bmac = (GlomoMacBmac *) pc_malloc(sizeof(GlomoMacBmac));
    char        buf[GLOMO_MAX_STRING_LENGTH];

    assert(bmac != NULL);

    memset(bmac, 0, sizeof(GlomoMacBmac));
    bmac->myGlomoMac = node->macData[interfaceIndex];
    bmac->myGlomoMac->macVar = (void *)bmac;

    bmac->timer.flag = BMAC_TIMER_OFF | BMAC_TIMER_UNDEFINED;
    bmac->timer.seq = 0;

    bmac->status = BMAC_STATUS_PASSIVE;
    bmac->BOmin = BMAC_BO_MIN;
    bmac->BOmax = BMAC_BO_MAX;
    bmac->BOtimes = 0;

    bmac->pktsToSend = 0;
    bmac->pktsLostOverflow = 0;

    bmac->pktsSentUnicast = 0;
    bmac->pktsSentBroadcast = 0;

    bmac->pktsGotUnicast = 0;
    bmac->pktsGotBroadcast = 0;

    bmac->initialBackoff = TRUE;
}

void MacBmacReceivePacketFromRadio(
    GlomoNode* node, GlomoMacBmac* bmac, Message* msg)
{
    GlomoNetworkIp* ipLayer = (GlomoNetworkIp*)node->networkData.networkVar;

    if (bmac->status == BMAC_STATUS_IN_XMITING) {
        GLOMO_MsgFree(node, msg);
        return;
    } // if //

    switch (bmac->status) {
        case BMAC_STATUS_PASSIVE:
        case BMAC_STATUS_CARRIER_SENSE:
        case BMAC_STATUS_BACKOFF:
        case BMAC_STATUS_YIELD: {
            int interfaceIndex = bmac->myGlomoMac->interfaceIndex;
            BmacHeader *hdr = (BmacHeader *) msg->packet;
            if (hdr->destAddr == node->nodeAddr) {
                bmac->pktsGotUnicast++;
            }
            else if (hdr->destAddr == ANY_DEST) {

```

```

        bmac->pktsGotBroadcast++;
    }

    if ((hdr->destAddr == node->nodeAddr) ||
        (hdr->destAddr == ANY_DEST))
    {
        NODEADDR lastHopAddress = hdr->sourceAddr;

        GLOMO_MsgRemoveHeader(node, msg, sizeof(BmacHeader));

        if (ipLayer->routingProtocolChoice == ROUTING.PROTOCOL_DIFFUSION ||
            ipLayer->routingProtocolChoice == ROUTING.PROTOCOL_LEACH)
            { // Handle it by diffusion
                NetworkReceivePacketFromMacLayer(node, msg, lastHopAddress);
            }
        else
            NetworkIpReceivePacketFromMacLayer(
                node, msg, lastHopAddress);
    }
    else {
        if (node->macData[interfaceIndex]->promiscuousMode) {
            MacBmacHandlePromiscuousMode(node, msg);
        }
        GLOMO_MsgFree(node, msg);
    }
    break;
}
default:
    GLOMO_MsgFree(node, msg);
    printf("MAC.BMAC: Error with node %u, status %ld.\n",
        node->nodeAddr, bmac->status);
    //assert(FALSE); abort(); error override --vi
} //switch//
}

void MacBmacReceiveRadioStatusChangeNotification(
    GlomoNode* node,
    GlomoMacBmac* bmac,
    RadioStatusType oldRadioStatus,
    RadioStatusType newRadioStatus)
{
    if (oldRadioStatus == RADIO_TRANSMITTING) {
        assert(newRadioStatus != RADIO_TRANSMITTING);
        assert(bmac->status == BMAC.STATUS_IN_XMITING);

        bmac->BOmin = BMAC.BO_MIN;
        bmac->BOmax = BMAC.BO_MAX;
        bmac->BOtimes = 0;
        bmac->status = BMAC.STATUS_YIELD;
        MacBmacYield(node, bmac, (clocktype)BMAC.TX_DATA_YIELD_TIME);
    } //if//
}

/*
 * FUNCTION      MacBmacLayer
 * PURPOSE       Models the behaviour of the MAC layer with the BMAC protocol
 *               on receiving the message enclosed in msg.
 *
 * Parameters:
 *   node:        node which received the message
 *   msg:         message received by the layer
 */

void MacBmacLayer(GlomoNode *node, int interfaceIndex, Message *msg)
{
    /*
     * Retrieve the pointer to the data portion which relates
     * to the BMAC protocol.
     */

    GlomoMacBmac *bmac = (GlomoMacBmac *)node->macData[interfaceIndex]->macVar;
    long seq_num;

    //printf("Node %d - MacBmacLayer\n", node->nodeAddr);

    assert(msg->eventType == MSG_MAC_TimerExpired); //--vi

    seq_num = *((int *) msg->info);

    printf("SEQ = %d\n", seq_num);

    GLOMO_MsgFree(node, msg);
}

```

```

        if ((seq_num < bmac->timer.seq) ||
            ((bmac->timer.flag & BMAC.TIMER_SWITCH) == BMAC.TIMER_OFF)) {
            return;
        }

        if (seq_num > bmac->timer.seq) {
            assert(FALSE);
        }
    }
    #if 0 //check re-transmit
    if (timerSequenceNumber == M802->timerSequenceNumber)
    {
        Mac802_11HandleTimeout(node, M802);
    }
    #endif
    assert(((bmac->timer.flag & BMAC.TIMER_TYPE) ==
        BMAC.TIMER_BACKOFF) ||
        ((bmac->timer.flag & BMAC.TIMER_TYPE) == BMAC.TIMER_YIELD));

    switch(bmac->timer.flag & BMAC.TIMER_TYPE) {
    case BMAC.TIMER_BACKOFF:
    {
        Message *newMsg;

        bmac->timer.flag = BMAC.TIMER_OFF | BMAC.TIMER_UNDEFINED;

        CheckRadioStatusAndSendOrBackoff(node, bmac);

        break;
    }

    case BMAC.TIMER_YIELD:
        bmac->timer.flag = BMAC.TIMER_OFF | BMAC.TIMER_UNDEFINED;
        bmac->status = BMAC.STATUS_PASSIVE;
        MacBmacPassive(node, bmac);
        break;

    default:
        assert(FALSE); abort();
        break;
    } /*switch*/
}

// B-MAC -----
void MacFinalizeUserMacProtocol(GlomoNode* node, int interfaceIndex)
{
    switch (node->macData[interfaceIndex]->macProtocol) {
    case USER_DEFINED_MAC.PROTOCOLNUMBER: {

        // Put function call to handle routing protocol finalization here:
        // For Example,

        // YourFinalizationRoutine(
        //     node, (YourMacDataType*)node->macData[interfaceIndex]->macVar);
    }
    default:
        printf("Glomosim Mac Protocol %d Not Handled\n",
            node->macData[interfaceIndex]->macProtocol);
        // assert(FALSE); abort(); --vi needs to be added
    } /*switch*/
}

void MacBmacFinalize(GlomoNode *node, int interfaceIndex)
{
    GlomoMacBmac* bmac = (GlomoMacBmac *)node->macData[interfaceIndex]->macVar;

    if (node->macData[interfaceIndex]->macStats == TRUE) {
        MacBmacPrintStats(node, bmac);
    }
}

/*
 * FUNCTION      ClearChannelAssessment
 * PURPOSE       Called at the end of simulation to collect the results of
 *               the simulation of the B-MAC protocol of MAC Layer.
 *
 * Parameter:
 *   node:        node for which results are to be collected.
 */

BOOL ClearChannelAssessment(GlomoNode *node, int interfaceIndex)
{
    BOOL IsChannelClear;

```

```

        IsChannelClear = CheckChannelStatus(node, interfaceIndex);

        return IsChannelClear;
    }
}
RadioStatusType RadioStatus(GlomoNode* node, GlomoMacBmac* bmac)
{
    return GLOMO_RadioGetStatus(node, bmac->myGlomoMac->interfaceIndex);
}

```

Linear Algebra Techniques

Techniques such as Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and QR Factorization use linear algebra for dimensionality reduction with processing large streams. This will be explained in more detail in Chapter 7.1.

Statistics

A.0.1 F-score

In statistics, the F score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct results divided by the number of all returned results and r is the number of correct results divided by the number of results that should have been returned. The F score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The traditional F-measure or balanced F-score (F score) is the harmonic mean of precision and recall:

$$F_{Score} = 2 \times \frac{precision \times recall}{precision + recall}$$

A.0.2 F-test

An F-test is any statistical test in which the test statistic has an F-distribution under the null hypothesis. It is most often used when comparing statistical models that have been fitted to a data set, in order to identify the model that best fits the population from which the data were sampled. Exact F-tests mainly arise when the models have been fitted to the data using least squares.

$$F = \frac{\text{explained variance}}{\text{unexplained variance}}$$

A.0.3 Dirichlet Distribution

In probability and statistics, the Dirichlet distribution (after Peter Gustav Lejeune Dirichlet), often denoted $Dir(\alpha)$, is a family of continuous multivariate probability

distributions parametrized by a vector α of positive reals. It is the multivariate generalization of the beta distribution. Dirichlet distributions are very often used as prior distributions in Bayesian statistics, and in fact the Dirichlet distribution is the conjugate prior of the categorical distribution and multinomial distribution. That is, its probability density function returns the belief that the probabilities of K rival events are x_i given that each event has been observed α_{i-1} times.

Interference Vs Energy Model

Theorem A.0.1 $\star$ *Co-channel Interference*

$$SIR = 10 \log_{10} \left[\left(\frac{D}{R} \right)^n K_l^{-l} \right] \quad (\text{A.1})$$

Gaussian Model

$\star$ *Adjacent Channel Interference*

$$SIR = C \frac{P_{TX}}{d^n} \left(\frac{P'_{TX} \int_w G(f) df}{(d)^n} \right)^{-1} \quad (\text{A.2})$$

Transmitter sensitive

$\star$ *Multi-path fading*

$$\cos(2\pi f t + \theta) \Rightarrow \text{Doppler Spectrum Model} \quad (\text{A.3})$$

The effects of fading θ can be combated by using diversity to transmit the signal over multiple channels that experience independent fading and coherently combining them at the receiver. The probability of experiencing a fade in this composite channel is then proportional to the probability that all the component channels simultaneously experience a fade, a much more unlikely event.

For the IT model described in equation (9.2), we calculate different values for the variable terms $T_I(f_i, B_i)$ and $\frac{M_i P}{k B_i}$. The reliability of a digital system is measured in terms of the error rate in the transmission link. BER - Bit Error Rate, SER - Symbol Error Rate, FER - Frame Error Rate, PER - Packet Error Rate. The SER characterizes the performance of the modulator. The BER is measured at the bit-level in terms of the number of bits that are received erroneously.

A.0.4 Channel Capacity

Theorem A.0.2 *Shannon showed that in an AWGN channel, the maximum bit-rate C that can be achieved with arbitrarily low error rate over a given transmission*

bandwidth WT is bounded by the expression below:

$$E_b = \frac{E_s}{\log_2 M} \quad (\text{A.4})$$

$$\bar{E}_b = \sum \frac{p_i E_s^i}{\log_2 M} \quad (\text{A.5})$$

where p_i is the probability for the occurrence of the i^{th} symbol with energy.

$$E_b = \frac{\max E_s^i}{\log_2 M} \quad (\text{A.6})$$

Theorem A.0.3 Where E_b is expressed in terms of the peak symbol energy in the signal constellation. E_s^i . where $\gamma = E_b$ and $\bar{\gamma} = \bar{E}_b$

$$\frac{C}{W_T} \leq \log_2 \left(1 + \frac{P}{N_o W_T} \right) = \log_2 \left(1 + \bar{\gamma} \frac{C}{W_T} \right) \quad (\text{A.7})$$

Symbol Energy:

$$E_b = \frac{E}{\log_2 M} \quad (\text{A.8})$$

where $P/N_o W_r$ is the SNR, C/W_T is the maximum achievable bandwidth efficiency in bps/Hz, and γ is the average E_o/N_o defined by equation (9.2).

A.0.5 Error Rate Bounds.

Theorem A.0.4 SER on the otherhand is measured at the symbol level in terms of the number of symbols that are in error. A symbol error is made when the received signal falls outside of its decision region. A symbol error leads to errors as the symbol is erroneously mapped to an incorrect bit-pattern. Let n be the number of bits per symbol. Then, SER may be bounded in terms of BER as shown below:

$$P_b \leq P_s \leq n \cdot P_b \quad (\text{A.9})$$

To express PER in terms of BER

$$P_p = 1 - (1 - P_b^{L_p}) \quad (\text{A.10})$$

A.0.6 A basic measure using BER

Theorem A.0.5 SER, FER, and PER all depend on BER, a basic measure for digital system is based on BER which can be expressed in terms of $\frac{E_b}{N_o}$, where E_b is the energy per bit and N_o is the equivalent noise spectral density over the signal bandwidth. The variations of BER with $\frac{E_b}{N_o}$ depends on the channel and the type of

the demodulator. Channel Models:(see Table A.1 for BPSK-modulation)

★ AWGN

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) \quad (\text{A.11})$$

★ Rayleigh Fading

$$P_b = \frac{1}{2} \left(1 - \frac{\sqrt{E_b/N_o}}{1 + E_b/N_o}\right) \approx \frac{1}{4E_b/N_o} \quad (\text{A.12})$$

A.0.7 Design goals for using BER

Theorem A.0.6 Additive white Gaussian noise (AWGN) in general, in a channel, BER is exponentially related to E_b/N_o , while in a fading channel, BER is inversely related to E_b/N_o .

For a given BER, a digital system with lower $\frac{E_b}{N_o}$ requires lower transmission power, which can improve battery lifetime of the communication device and the system capacity.

A.0.8 Energy Efficiency

Theorem A.0.7 Energy efficiency can be more accurately defined, when taking into account both energy and bandwidth, we define $f(x)$ of a system to be the amount of E_b/N_o required for a given bandwidth efficiency:

Table A.1: Energy Efficiency Comparison used by Radios

BPSK	9.09	1	11.0
GMSK	10.8	1.35	11.5
QPSK	9.09	2	16.5
8-PSK	19.82	3	11.8
16-PSK	55.41	4	6.8
32-PSK	171.2	5	3.6
8-QAM	13.93	3	16.8
BFSK	17.78	1	5.6

Index

algorithms

Trees, 2, 19, 21, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 201, 210
data-cleaning, 7, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 201, 210
k-nearest neighbor, 2, 19, 21, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 201, 210
Naive Bayes, 2, 19, 21, 25, 37, 55, 85, 102, 120, 127, 151, 158, 171, 185, 201, 210
Trees, 158

protocol

QoD, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 201, 210
QoS, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 201, 210

protocols

Bluetooth, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 186, 201, 210
GPRS, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 186, 201, 210
power-aware
802.11, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 180, 185, 201, 210
B-MAC, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 160, 171, 185, 201, 210
BER, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 163, 171, 185, 201, 210

cross-layer, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 201, 210
CSMA, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 161, 171, 185, 201, 210
lifetime, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 173, 185, 201, 210
MAC, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 158, 160, 171, 185, 201, 210
no-network, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 173, 185, 201, 210
TDMA, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 168, 171, 185, 201, 210
TDMA-TA, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 169, 171, 185, 201, 210
renewable
FARMS, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 161, 171, 185, 201, 210
RS-232, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 186, 201, 210
simulation
results, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 179, 180, 185, 201, 210
USB, 2, 19, 25, 37, 55, 85, 102, 120, 127, 151, 171, 185, 186, 201, 210

sensor

measurement, 19, 25, 37, 55, 85, 102,
120, 127, 151, 171, 185, 201, 210
model, 5, 19, 25, 37, 55, 85, 102, 120,
127, 151, 171, 185, 201, 210
simulators
power-aware
GlomoSIM, 2, 19, 25, 37, 55, 85,
102, 120, 127, 151, 163, 171, 185,
201, 210

Vasanth Iyer

May 29, 1964
Sep. 26th, 2011

Born, Tamil Nadu, India.
Oath of US Citizenship, San Francisco.

PROFESSIONAL PREPARATION

2013

Ph.D. Candidate, Computer & Information Sciences
Florida International University
Miami, Florida
Ensemble Stream Model for Data-Cleaning in Sensor Networks
GPA: 3.67 (Graduation date: Dec-2013).

1987

M.S., Computer & Information Sciences
University of New Haven
West Haven, Connecticut
Implemented Intel Bitbus protocol to enhance connectivity in noisy factory floors.

1985

B.E., Electronics & Communications
Osmania University[†]
Hyderabad, India
^{3rd} Student[†] Best Paper runnerup at CSI-1985.
^{2nd} Student[†] Best Paper runnerup at CSI-2005 .

1974–1979

R.K.M Boys' High School, Chennai, Tamil Nadu, India.
E.S.V. Memorial Medal for Best Student Allrounder Award.

PROFESSIONAL EXPERIENCE

1989–2002*

Software Design Engineer

2003-2012

Research Intern

Recent

(2004) DirectX Microsoft Team - SmartPhones
(2008) Institute of Infocomm Research, Singapore
(2009) Louisiana State University /Raytheon
(2011) Yahoo! Labs Global, Summer School
(2013) University of Edinburgh, Scotland
(1996) Canon Information Systems at Stanford, CA.
(1988) Unilever Research, Essex, CT.

Past

Teaching 2009-2012

My teaching experience has been one in which I was part of writing a programming book: co-edited "Fundamentals of Sensor Network Programming/Applications and Technology" with Dr. S.S. Iyengar, December 2010, Wiley-IEEE Press, created slides for teaching Sensor programming for professional students during Spring of 2009, 2013 at Texas Raytheon, and held followup collaborative workshop - I will gradually become good in teaching and welcome domain experience to the classroom.

Professional Memberships

- (2013) Reviewer for IEEE/ACM Transactions Journal on Networking
- (2013) IEEE International Conference on Image Information Processing (IIIT)
- (2011) Editorial Advisory Board, International Journal of Advances in Networks and Services (Netware/IARIA).

Conference Publications and Presentations

Vasanth Iyer, S. Sitharama Iyengar, Niki Pissinou, Shaolei Ren (2013): SPOTLESS: Similarity patterns of Trajectories in Label-IEss Sensor Streams. PerCom Workshops : 487-492

Vasanth Iyer, S. Sitharama Iyengar (2011): Modeling Unreliable Data and Sensors: Using F-measure Attribute Performance with Test Samples from Low-Cost Sensors. ICDM Workshops: 15-22

Vasanth Iyer, S. Sitharama Iyengar, Garimella Rama Murthy, Nandan Parameswaran, Dhananjay Singh, Mandalika B. Srinivas (2010): Effects of channel SNR in mobile cognitive radios and coexisting deployment of cognitive wireless sensor networks. IPCCC : 294-301

Vasanth Iyer, S. Sitharama Iyengar, Garimella Rama Murthy, Bertrand Hochet, Vir V. Phoha, M. B. Srinivas (2009): Multi-hop scheduling and local data link aggregation dependant Qos in modeling and simulation of power-aware wireless sensor networks. IWCMC: 844-848

Vasanth Iyer, Rammurthy Garimella, M. B. Srinivas (2008): Training Data Compression Algorithms and Reliability in Large Wireless Sensor Networks. SUTC: 480-485

Garimella Rama Murthy, Vasanth Iyer (2007): Distributed Wireless Sensor Network Architecture: Fuzzy Logic Based Sensor Fusion. EUSFLAT Conf. (2) 2007: 71-78

Journal and Book Chapters

Vasanth Iyer, S.S. Iyengar, Niki Pissinou. Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning. Published by Taylor & Francis, 2012. pp 32-52.

S. Sitharama Iyengar, Nandan Parameswaran, Vir V. Phoha, N. Balakrishnan, Chuka D. Okoye. Fundamentals of Sensor Network Programming: Applications and Technology. As book chapters 12,13, & 14 in December 2010, Wiley-IEEE Press.

Vasanth Iyer, S.S. Iyengar, G. Rama Murthy, Kannan Srinathan, Govindarajulu, and M.B. Srinivas. STACK: Sparse Timing of Algorithms using Computational Knowledge. As a book chapter in New Developments and Applications in Sensing Technology, Springer publications 2010.

Vasanth Iyer, S.S. Iyengar, R. Murthy, and M. B. Srinivas. Computational aspects of sensor network protocols. In Special Issue on Modern Sensing Technologies Journal (IFSA), volume 6., pages 69-91. Sensors & Transducers Journal, July 2009.

Vasanth Iyer, S.S. Iyengar, G. Rama Murthy and M.B. Srinivas. Distributed Source Coding for Sensor Data Model. International Journal of Simulation: Systems, Science & Technology (IJSSST), December 2009. A publication of the United Kingdom Simulation Society, EDAS London, 2009. I.J of Simulation, Vol. 10, ISSN: 1473-804x

Vasanth Iyer, G.Rama Murthy, and M.B. Srinivas. Training data compression algorithms and reliability in large wireless sensor networks - Extended version. In International Journal on Smart Sensing and Intelligent Systems, Vol. 1, NO. 4, page 480-485, December 2008

Conference Poster Presentations

Vasanth Iyer, Shailesh Kumar. WwwwhooshML (WIKIPEDIA Dataset).International Institute of Information Technology, IR Poster Session, 2012.

Vasanth Iyer, Sitharama Iyengar, G. Rama Murthy, M.B. Srinivas. STACK: Sparse Timing of Algorithms using Computational Knowledge. ICDCN 2011 PhD Symposium. 2nd-5th Jan 2011 Bangalore, India.

Vasanth Iyer, Sitharama Iyengar, G. Rama Murthy, M.B. Srinivas and Srinathan Kannan. SenseSIM: Sensor network simulator. PhD Symposium, ISSNIP 7-10th December, 2010 Melbourne, Australia.

Advisors

S.S. Iyengar and Niki Pissinou (Ph.D.,Florida International University).

Howard K. Okrent and Mathi Packiam (M.S.,University of New Haven).