UNIVERSITY OF CALIFORNIA,
IRVINE

Essence: Machine Learning Approaches to Scalable and Energy Efficient Sense-making for Internet-of-Things (IoT)

THESIS
submitted in partial satisfaction of the
requirements for the degree of
Master of Science

in

Computer Science

by

Santanu Sarma

2015

To my family

Contents

List of Figures

List of Tables

Acknowledgments

I take this opportunity to express my gratitude to Prof. Nikil Dutt and Prof. Nalini Venkatasubramanian, for their guidance and support throughout the course of this work. Their expertise and contagious enthusiasm are greatly appreciated. It gives me immense pleasure to express my thanks to Prof. Alex Nicolau, Chair, Department of Computer Science, for his encouragement, support, and valuable suggestions. I am especially thankful to my wife who had been of immense help and my support system during this thesis work. Her love and care gave me a deep reason to keep my motivations at hard times. I am greatly indebted to my parents and family members for their encouragement and courage to overcome all the odds in my absence and during my stay in Irvine. I also thankful to each and everyone who made my stay at Irvine enjoyable and memorable.

List of Acronyms

IoT	Internet of Things
IoE	Internet of Everything
MCC	Mobile Cloud Computing
LC	Local Cluster
NC	Nano Cluster
CPS	Cyber-Physical Systems
CS	Compressive Sensing
MCS	Model Based Compressive Sensing
PCS	Plain Compressive Sensing
NCS	No Compressive Sensing
HCS	Hybrid Compressive Sensing
HMLCS	Hierarchical Multi-level Compressive Sensing
WNV	West Nile Virus
ML	Machine Learning
OLS	Ordinary Least Square
GLS	Generalized Least Square
PCA	Principal Component Analysis
NN	Neural Network
FFNN	Feed Forward Neural Network
RNN	Recurrent Neural Network
DNN	Deep Neural Network

ABSTRACT OF THE THESIS

Essence: Machine Learning Approaches to Scalable and Energy Efficient Sense-making for Internet-of-Things (IoT)

by

Santanu Sarma

Master of Science in Computer Science

University of California, Irvine, 2015

Professor Nikil Dutt, Chair

This thesis presents an efficient and scalable sense-making framework using machine learning techniques for Internet-of-Things (IoTs) in order to understand users, contexts, and their environments to make meaningful decisions. The proposed sense-making IoT framework, called Essence, employs combination of participatory mobile crowdsensing along with infrastructure sensing to perform sense-making using machine learning techniques. While collaborative mobile crowdsensing enables information to be gathered and shared by users who are directly involved (participatory sensing) or integrated seamlessly as needed (opportunistic sensing) through user mobile platforms, the infrastructure sensing fabric of the Essence framework provides sense-making support for scenarios where mobile sensing platforms are inadequate. To address the scalability needs of the Essence framework, we employ dimensionality reduction techniques such as principal component analysis (PCA) based heterogeneous compressive sensing techniques for approximate gathering and processing of diverse sensor data. This requires new mechanisms for sensor data collection, tunable approximate processing and machine learning based decision making using a hierarchical networking architecture to create a compressive collaborative sense-making framework for IoTs. Essence uses our previous SenseDroid mobile sensing framework with machine learning enabled decision and actuation components for IoT paradigms. The Essence framework is build using a multi-tired hierarchical architecture for sensing spatial variations of a parameter of interest, perceive spatio-temporal fields, and enable energy efficient local mobile or infrastructure sensing with a small number of measurements. This approximate, yet tunable approach combines different sensing approaches opportunistically while trading scalability (and coverage) for data accuracy (and energy efficiency). We demonstrate the application of Essence sense-making framework in predicting the presence of West Nile Virus (WNV) in a region using both compressed sensing and Neural Networks (NN) as a case study. The thesis also discuss several challenges associated with sense-making approaches for emerging IoT applications.

Chapter 1

Introduction

1.1 Overview

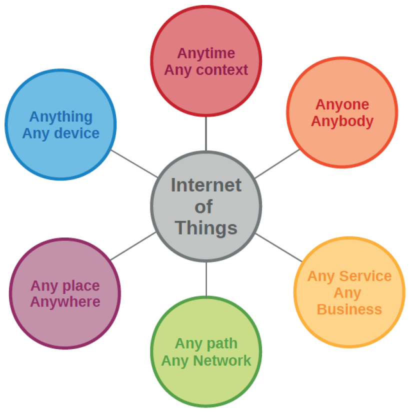
The Internet of Things (IoT) allows people and things to be connected anytime, anyplace, with anything and anyone, ideally using any path/network and any service [47]. Internet of Things has provided a promising opportunity to build powerful industrial systems and applications by leveraging the growing ubiquity of mobile, wireless, radio-frequency identification (RFID), and sensor devices. While there is a growing interest in IoTs in academia (Fig.1.1), a wide range of industrial IoT applications have been developed and deployed in recent years as illustrated in Fig. 1.2. As the Internet evolves from connection of PCs to connection of mobiles, PCs, and people (Fig. 1.3), this combination of multi-layered sensor network connecting various sensors and objects create the vision of the new frontier of the Internet : the Internet-of-Things. In IoT, this layered architecture may have additional number of sub layers as it is expected to comprises large variety of sensing capabilities, multiple networking support, multiple services to satisfy users and diverse applications as illustrated in Fig.1.3c.

According to economic analysis by Cisco Consulting Services [2], IoT will generate $8 trillion in value at stake over the next decade — $6.4 trillion in the private sector, and $1.6 trillion in the public sector. Value at Stake refers to the potential bottom-line value that can be created, or that will migrate among private and public sector organizations, based on their ability to harness the IoT over the next 10 years. This value will come from five primary drivers: innovation and revenue ($2.1 trillion), asset utilization ($2.1 trillion), supply chain and logistics ($1.9 trillion), employee productivity improvements ($1.2 trillion), and enhanced customer and citizen experience ($700 billion). While IoT will impact all private and public sector segments over the next decade, two-thirds of the estimated $8 trillion in IoT Value at Stake will be driven by three industries: manufacturing (including energy/oil and gas), public sector (particularly cities), and retail.

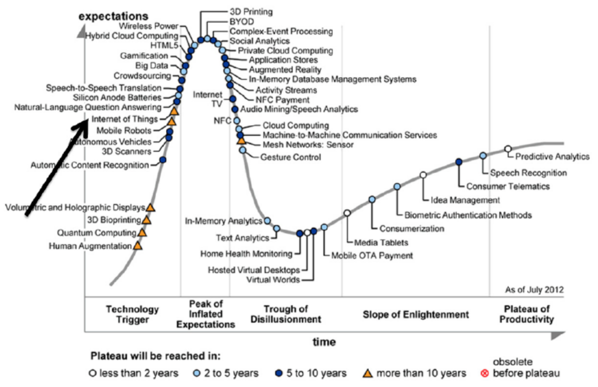
While the Internet of Things (IoT) presents private and public sector organizations with an unprecedented opportunity to drive new sources of value — including the potential to automate up to 50 percent of manual processes [2], this value comes from improving data and sense-making capabilities (integration, automation, and analysis) and overall process agility. Sense-making from the data is the key aspect of the IoT paradigm that is fundamental in deriving meaningful insights for developing innovative applications.

1.2 Sense-Making from IoT Data

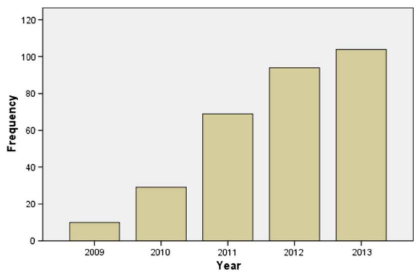
As we are moving towards the Internet of Things (IoT), the number of sensors deployed around the world is growing at a rapid pace. Market research has shown a signif-



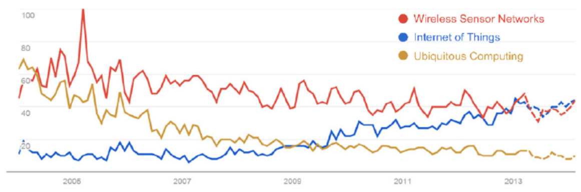
(a) [47]



(b) [16]



(c) [9]



(d) [16]

Figure 1.1: (a) Definition of the Internet of Things (b)Gartner 2012 Hype Cycle of emerging technologies. Source: Gartner Inc. [16] (c) Google search trends since 2004 for terms Internet of Things, Wireless Sensor Networks, Ubiquitous Computing. [16] (d) Number of IoT Journal articles by year in Web of Knowledge. [9]

icant growth of sensor deployments over the past decade and has predicted a significant increment of the growth rate in the future. These sensors continuously generate enormous amounts of data. However, in order to add value to raw sensor data we need to understand it. Collection, modeling, reasoning, and distribution of insights in relation to sensor data plays critical role in this challenge. To capitalize on the wide range of IoT generated data, organizations must overcome key challenge of sense-making from large volume of data.

Even though, sense-making has a wide history spanning from psychology to information theory[20, 21], we use it from an information-theoretic perspective to mean the process of understanding the connections (among people, places, environment, objects, and events) in order to anticipate their trajectories and act effectively using empirical data. Whether it is in the cloud or at the edge, IoT sense-making data must be analyzed to identify actionable insights that can be used to create better outcomes in an energy efficient way. Energy efficiency play a vital role in deploying IoT: they can reduce emission and pollution, exploit environmental conservation and surveillance, and minimize

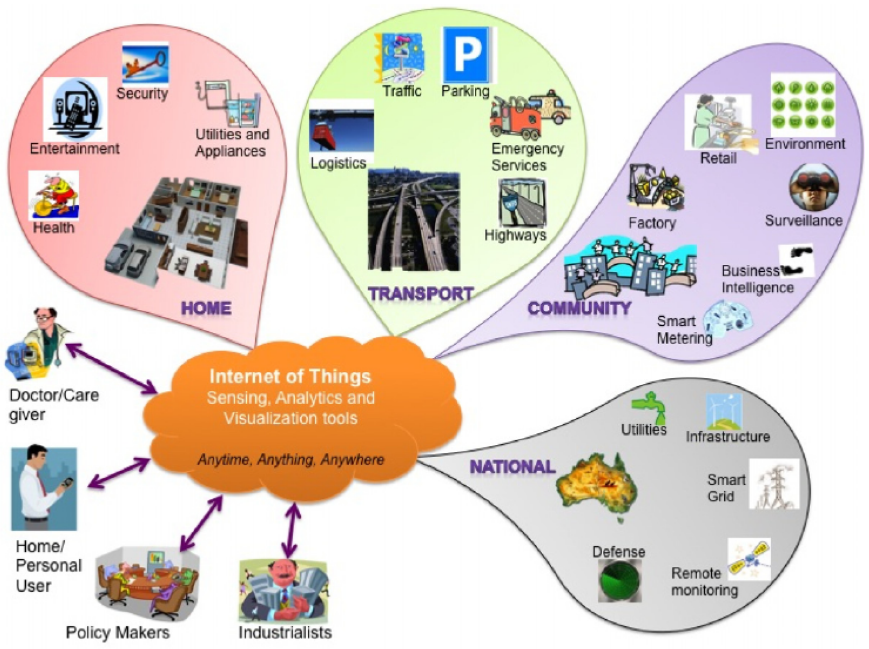
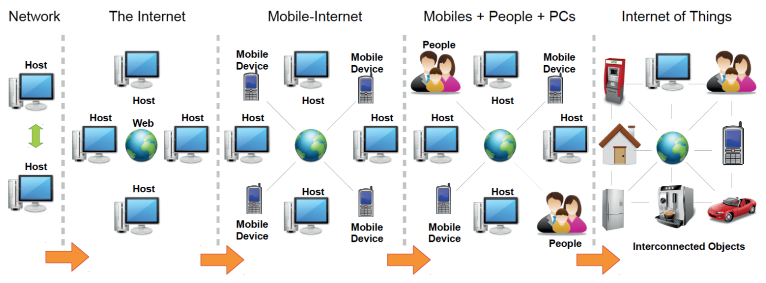


Figure 1.2: Internet of Things schematic showing the end users and application areas based on data. [16]

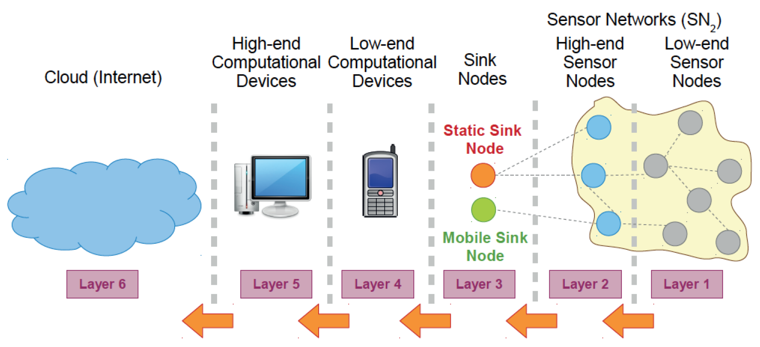
operational costs and power consumption, and increase the network operation life time and resilience. Therefore, sense-making considering both energy efficiency and scalability of the network are key challenges that need to be addressed in order to reap the immense potential of the IoT paradigm.

1.3 Thesis Contribution and Organization

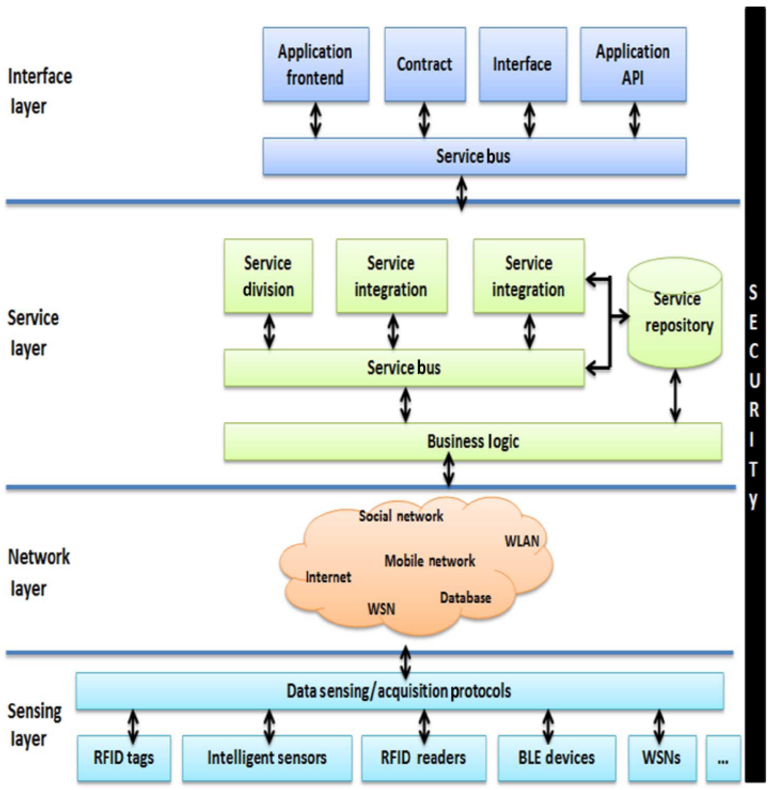
In this thesis, we present an energy efficient and scalable sense-making framework using machine learning techniques for Internet-of-Things (IoTs) in order to understand users, contexts, and their environments to make meaningful decisions. The proposed sense-making framework, called Essence, employs combination of participatory mobile crowdsensing along with infrastructure sensing along with compressive sensing techniques for sense-making. To address the scalability needs of the Essence framework, we employ machine learning techniques such as principal component analysis (PCA) based heterogeneous compressive sensing techniques for approximate gathering and processing of diverse sensor data. This requires new mechanisms for sensor data collection, tunable approximate processing and machine learning based decision making using a hierarchical networking architecture to create a compressive collaborative sense-making framework for IoTs. Essence is an extension of our previous SenseDroid mobile sensing framework with machine learning enabled decision and actuation components for IoT paradigms.



(a)



(b)



(c)

Figure 1.3: Evolution of the Internet to IoT (a) connected computers to connecting every day objects to the Internet [37] (b) Layered structure of a sensor network (b) Service oriented architecture for IoT [9].

The Essence framework is build using a multi-tired hierarchical architecture for sensing spatial variations of a parameter of interest, perceive spatio-temporal fields, and enable energy efficient local mobile or infrastructure sensing with a small number of measurements. This approximate, yet tunable approach combines different sensing approaches opportunistically while trading scalability (and coverage) for data accuracy (and energy efficiency). On the other hand, efficient implementation of neural networks (NN) and regression based predictors, enable accurate sense and decision making that can be used for several applications. We demonstrate the application of Essence sense-making framework in predicting the temperature profile and its relationship with the presence of West Nile Virus (WNV) in a locality as an example. The thesis also discuss several challenges associated with sense-making approaches for emerging IoT applications.

The thesis is organized in five chapters. Chapter 1 provives a brief introduction to the IoT and the need for efficient sense-making from vast IoT data. Chapter 2 discusses the mobile and infrastructure sensing methods, tools, and application used and developed in order to support the Essence framework. Chapter 3 discusses the use of compressive sensing and its extension for heterogenous compressive sensing for scalable and energy efficient data aggregation and sense-making. Chapter 3 also briefly discusses an example to illustrate the applicability of the framework in predicting the temperature profile and its relationship with the presence of West Nile Virus (WNV) in a region using few random measurements. The analysis of the performance, energy, and scalability of the proposed methods in Essence framework is studied analytically and through simulation in Chapter 4 followed by the conclusion and future works in Chapter 5.

Chapter 2

Mobile and Infrastructure Sensing for Internet-of-Things

2.1 Introduction

Mobile phones and smartphones have evolved to be very powerful devices that have the potential to be utilized in many application areas apart from generic communication. With each passing year, we see increasingly powerful smartphones being manufactured, which have a plethora of powerful embedded sensors like microphone, camera, digital compass, GPS, accelerometer, temperature sensors and many more [22] as shown in Fig. 2.1. They even support seamless external sensors connectivity that further equips such devices with rich and unique sensing capabilities. Moreover, the ability to easily program today's smartphones, enables us to exploit these sensors, in a wide variety of application such as personal safety, emergency and calamity response, situation awareness, remote activity monitoring, transportation and environment monitoring [19, 22].

Broadly, the emerging field of mobile phone sensing can be categorized into participatory sensing and opportunistic sensing [19]. While the user is directly involved in the participatory sensing activity, this burden is alleviated in the opportunistic sensing by delegating and automating the sensing task to the mobile phone sensing system. Different from these two mechanisms, we propose a compressive collaborative sensing approach where the users collaborate or cooperate to have better and reliable sensing information, or missing sensing information from other users in absence of specific sensor in their own phone, overcome nonavailability of specific sensor information by using infrastructure sensors, energy-efficient sensing and computation using compressive sensing and opportunistic offloading, and context-aware adaptive information exchange. Collaboration can be useful in generating more accurate and reliable information of spatial fields distributed across geographical areas and region, for example, averaging of several temperature sensor reading in a room would be more reliable than a single reading, and incident perimeter assessment and high impact region localization during disaster and emergency response operations. However, due to the lack of an existing collaborative sensing framework, the development of any mobile collaborative applications using varied sensors is a difficult and time consuming task.

In this chapter, we propose and take the initial steps toward building a mobile sensing framework, called SenseDroid for the Android platform. We propose an unified two open source sensing libraries framework to support compressive and collaborative sensing (as well as opportunistic, participatory sensing modes) along with on-demand context-aware information exchange. We implement the framework as a distributed middleware and explore its use in several emerging mobile collaborative applications.

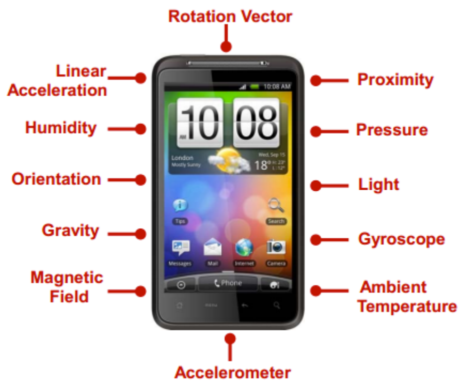


Figure 2.1: Increasing number of mobile phone sensors.

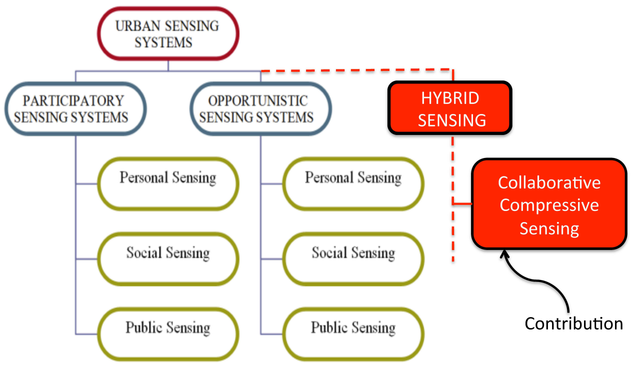
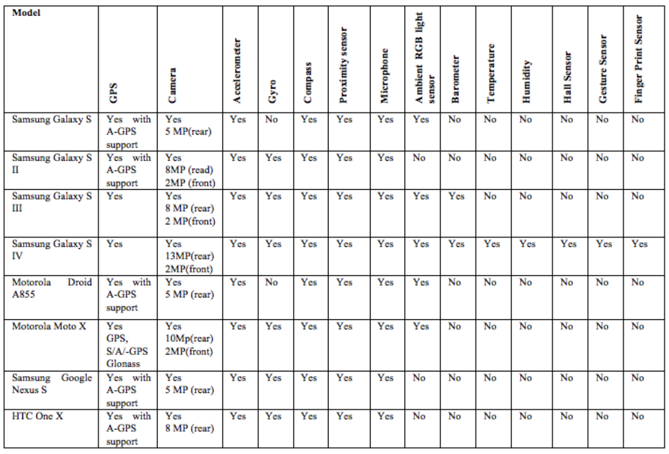


Figure 2.2: Compressive collaborative sensing as a new category under hybrid sensing of mobile phones. Compressive collaborative sensing uses combination of both participatory and opportunistic sensing and supports stochastic sampling.

Table 2.1: Mobile Phone Sensing Methods

Mobile Phone Sensing	**Participatory Sensing**	**Opportunistic Sensing**	**Collaborative Sensing**
User involvement	Very High	Very Low	Very low -to- Medium (Hybrid)
Data Collection process	User Initiated	Automatic (Passively)	Mixed (mostly automatic)
Sampling Method	Deterministic	Deterministic	Deterministic & Stochastic
Data Collection Accuracy	Low (may miss data)	High	Very High-to- Medium
Context Setting	User given	Automatic	Medium
User Burden	High	Low	Very Low -to - Medium
Context Processing	Low	High	Low- Medium (Off-loading)
Battery Consumption	Low	High	Very Low -to- Medium

Table 2.2: Sensors in Recent Android Mobile Phones



Model	GPS	Camera	Accelerometer	Gyro	Compass	Proximity sensor	Microphone	Ambient RGB light sensor	Barometer	Temperature	Humidity	Hall Sensor	Gesture Sensor	Finger Print Sensor
Samsung Galaxy S	Yes with A-GPS support	Yes 5 MP(rear)	Yes	No	Yes	Yes	Yes	Yes	No	No	No	No	No	No
Samsung Galaxy S II	Yes with A-GPS support	Yes 8MP (read) 2MP (front)	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No
Samsung Galaxy S III	Yes	Yes 8 MP (rear) 2 MP(front)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
Samsung Galaxy S IV	Yes	Yes 13MP(rear) 2MP(front)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Motorola Droid A855	Yes with A-GPS support	Yes 5 MP (rear)	Yes	No	Yes	Yes	Yes	Yes	No	No	No	No	No	No
Motorola Moto X	Yes GPS, S/A/-GPS Glonass	Yes 10Mp(rear) 2MP(front)	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No
Samsung Google Nexus S	Yes with A-GPS support	Yes 5 MP (rear)	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No
HTC One X	Yes with A-GPS support	Yes 8 MP (rear)	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No

2.1.1 Mobile Sensing Based Context as Virtual Sensors

The SenseDroid framework provides individual probes for sensing available sensors, basic preprocessing of the sensed data for calibration, and energy efficient compressive context processing to achieve individual user situation, activities, and events. In the terminology of SenseDroid framework, many of the context that are indirectly senses by computational means is referred to as virtual sensors. Unlike the basic physical sensor probes [1], SenseDroid provides several virtual sensing probes corresponding to different types of contextual informations. The basic approach adopted by SenseDroid in creating these physical and virtual sensor probes are shown in Fig. 3.3. Unlike the recent works in [32, 25, 26] in context processing, SenseDroid differs in its approach, architecture, and progressing by incorporating stochastic sampling and exploiting correlation among the contexts.

2.2 Mobile Sensing API and Prototyping Platform

To illustrate and demonstrate several features of the framework we developed an Android app which can be configured in either three mode, client, server, and client-server to cater to support the collaboration architecture. The high level system architecture of the mobile app as shown in Fig. 2.4.

In most scenarios the app operates in client-server mode where it can send and serve request to and from other nodes through the broker. The broker also operates in the

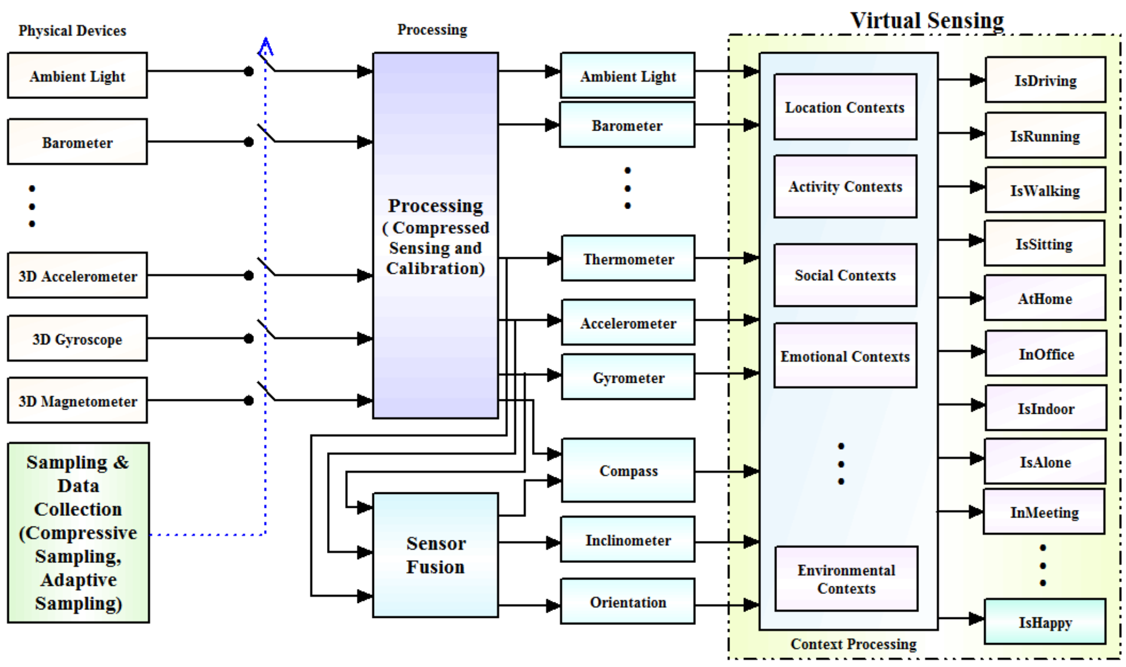


Figure 2.3: Compressive Context Processing for Personal Mobile Sensing.

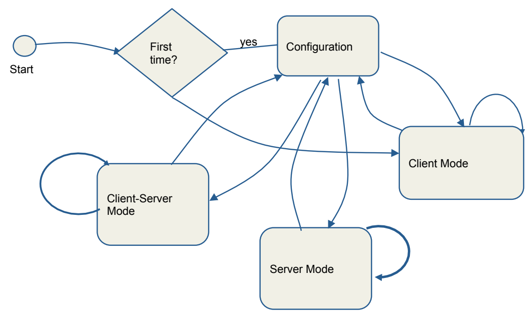
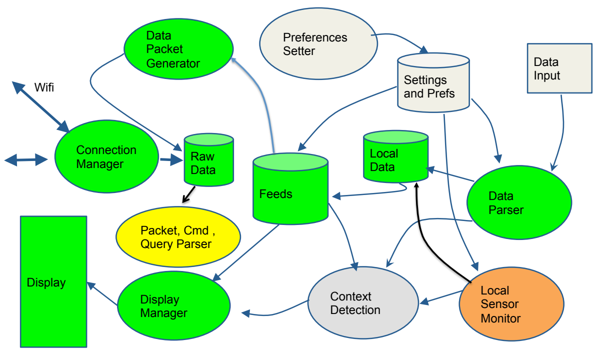
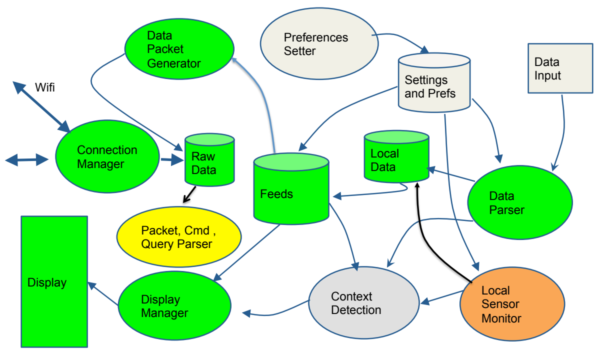


Figure 2.4: Android App system architecture for collaborative and compressive sensing.



(a) Client Mode of Operation.



(b) Server mode of application for the mobile app.

Figure 2.5: Mobile App architecture and data flow in different modes of operation.

client-server mode and have features and resources to perform processing on the collected data and disseminate the date to the interested users. The basic data flow operation of the app in the client and server mode is shown in Fig. 2.5.

The SenseDroid Android app implementation is broadly divided into two parts viz. client side and server side modules. The client side module consists of all the packages that a client node would be using. This includes several Java packages as shown in the Fig. 2.5a. The local sensor monitoring module consists of several Java packages to probe different sensor for user specified configurations. The client nodes update the middleware broker (cluster head) based on different policies (e.g., periodically, at random, etc) once commanded or receiving request from other participating nodes. The client mode of the app uses the *funf* open sensing library as shown in Fig.2.6 in order to perform mobile phone sensing and establishing communication connection to a cloud or remote source. As one of the aim of this thesis is to support different sensor sampling schemes, several function were modified and few added to support random and configurable sampling schemes, which funf can not easily support.

Fig. 2.7 shows some of the setting of the SenseDroid app that was built using two open source apps (funf and Androsens2). Most of the GUI and visualization supports are provided by Anrodsens2, where as funf provides addition probes that Androsens2 does not have. In addition to the combined feature, we include different sampling mechanism (random and configurable) , basic machine learning functions, and collaboration features in the app. Fig. 2.8 and Fig. 2.9show different sensor reading and data visualization in the SenseDroid app which are logged in a file or SQLite database in the app.

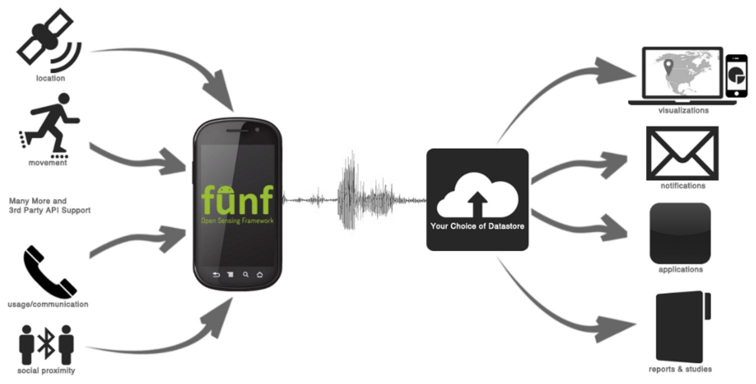
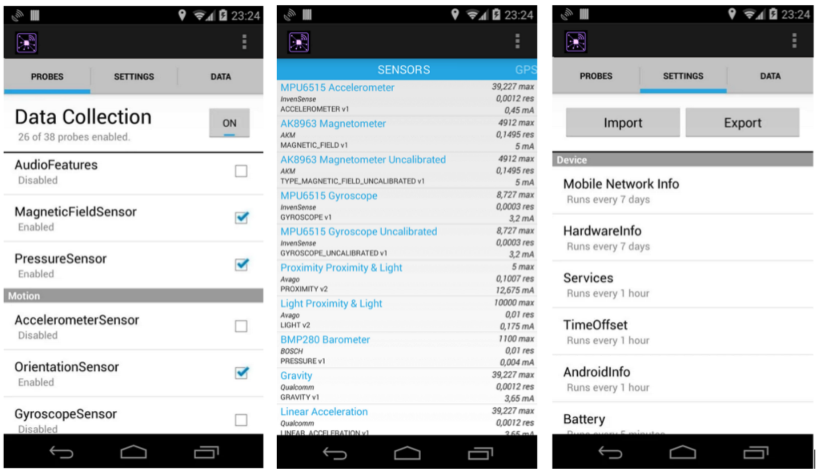


Figure 2.6: Open mobile sensing library.


23:24

PROBES | SETTINGS | DATA

Data Collection ON
26 of 38 probes enabled.

AudioFeatures
Disabled ☐

MagneticFieldSensor
Enabled ✓

PressureSensor
Enabled ✓

Motion

AccelerometerSensor
Disabled ☐

OrientationSensor
Enabled ✓

GyroscopeSensor
Disabled ☐

23:24

SENSORS | GPS

MPU6515 Accelerometer InvenSense ACCELEROMETER v1	39,227 max 0,0012 res 0,45 mA
AK8963 Magnetometer AKM MAGNETIC_FIELD v1	4912 max 0,1495 res 5 mA
AK8963 Magnetometer Uncalibrated AKM TYPE_MAGNETIC_FIELD_UNCALIBRATED v1	4912 max 0,1495 res 5 mA
MPU6515 Gyroscope InvenSense GYROSCOPE v1	8,727 max 0,0003 res 3,2 mA
MPU6515 Gyroscope Uncalibrated InvenSense GYROSCOPE_UNCALIBRATED v1	8,727 max 0,0003 res 3,2 mA
Proximity Proximity & Light Avago PROXIMITY v2	5 max 0,1007 res 12,675 mA
Light Proximity & Light Avago LIGHT v2	10000 max 0,01 res 0,175 mA
BMP280 Barometer BOSCH PRESSURE v1	1100 max 0,01 res 0,004 mA
Gravity Qualcomm GRAVITY v1	39,227 max 0,0012 res 3,65 mA
Linear Acceleration Qualcomm LINEAR_ACCELERATION v1	39,227 max 0,0012 res 3,65 mA

23:24

PROBES | SETTINGS | DATA

Import | Export

Device

Mobile Network Info
Runs every 7 days

HardwareInfo
Runs every 7 days

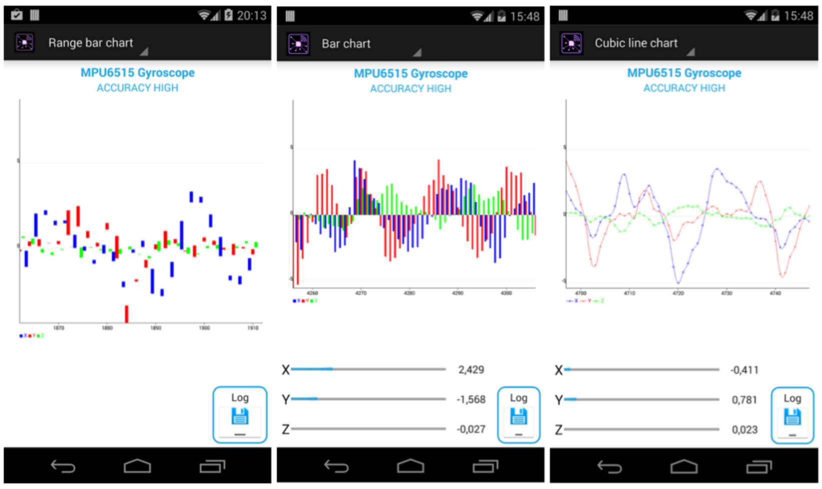
Services
Runs every 1 hour

TimeOffset
Runs every 1 hour

AndroidInfo
Runs every 1 hour

Battery

Figure 2.7: SenseDroid configuration and sensor selections.



X 2,429
Y -1,568
Z -0,027

X -0,411
Y 0,781
Z 0,023

Figure 2.8: Gyroscope Sensor reading using the SenseDroid App.

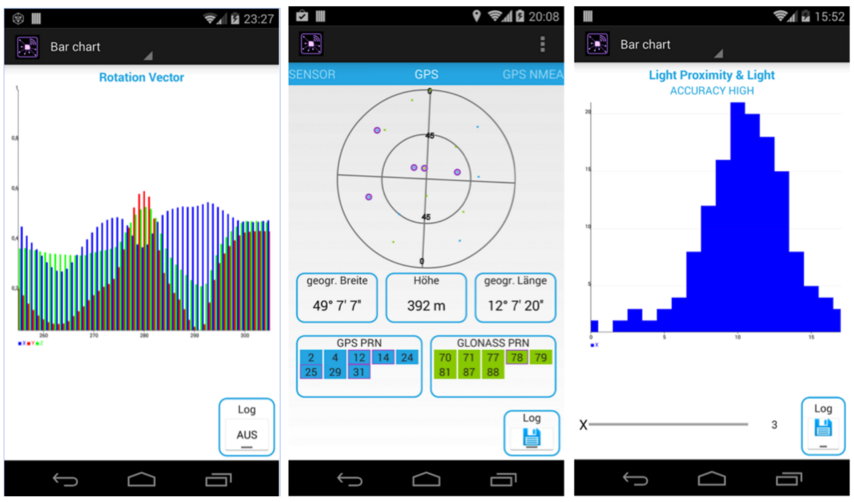


Figure 2.9: Rotation, GPS, and Proximity Sensor reading and data visualization.

2.2.1 Broker App

The broker coordinates the communications between the users, perform processing of the sensor data and collective context, and disseminate to the users through one-to-one or broadcast mode of communication. Communication sockets are used to implement the communication and collaboration layers among the users. The broker periodically sends heartbeats using UDP to maintain and keep updated network structure and topology. The client nodes respond to the specific heartbeats in the discovery process and join the network once authenticated. APIs are developed using the Android SDK and Java libraries in the Eclipse environment. We leveraged open source and freely available software to develop the application including the cloud interface. We used the LAMP stack along with Drupal open-source CMS to collect the sensor data in the cloud. The compressive sensing algorithm is implemented both in the client mobile nodes and the Broker.

2.3 Infrastructure Sensing

In order to provide support for infrastructure sensing in the IoT paradigm, we use a Xilinx Zynq FPGA based embedded SoC development kit [3] along with a Analog Devices software defined radio development kit as shown in Fig. 2.10. The Linux OS and devices drivers are developed and supported on these development board to perform infrastructure

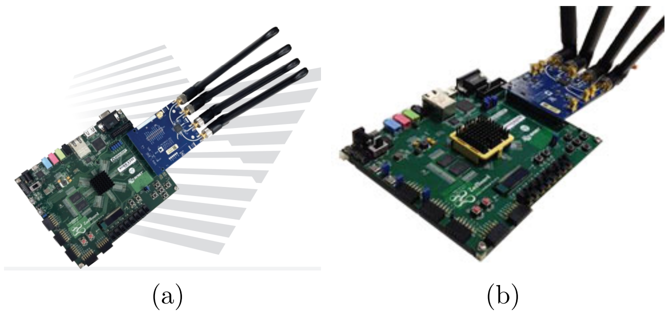


Figure 2.10: Zynq-7000 AP SoC / AD9361 SoftwareDefined Radio Evaluation Kit.

.

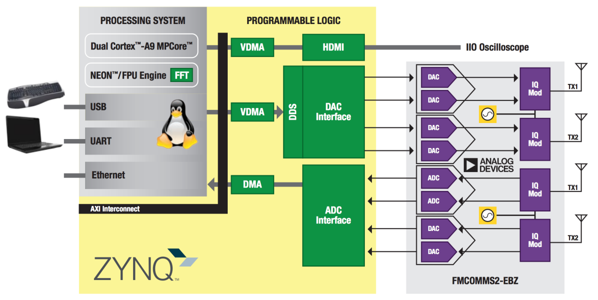
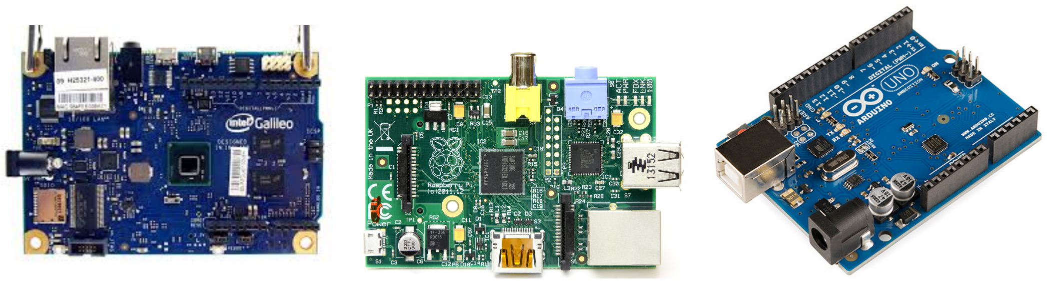


Figure 2.11: Xilinx Zynq ZC 702 based software defined radio development board.

sensing and provide communication to the Internet through the ethernet interface of the board. In addition to the Xilinx Zynq FPGA board, the infrastructure sensing setup is extended with three popular development boards (the Intel Galileo, Raspberry Pi, and Audruino) as in Fig. 2.12, which are supported by Linux with open source sensing APIs and SDKs as listed in Appendix B.

2.3.1 Linux support for the industrial IO (IIO) subsystems [11]

The Industrial I/O subsystem is intended to provide support for devices that in some sense are analog to digital or digital to analog converters (ADCs, DACs). Devices that fall into this category are: ADCs, Accelerometers, Gyros, IMUs, Capacitance to Digital Converters (CDCs), Pressure Sensors Color, Light and Proximity Sensors, Temperature Sensors, Magnetometers, DACs, DDS (Direct Digital Synthesis), PLLs (Phase Locked Loops), Variable/Programmable Gain Amplifiers (VGA, PGA).



(a) (b) (c)

Figure 2.12: Infrastructure sensing boards (a) Intel Galileo (b) Raspberry Pi 2 (c) Arduino.

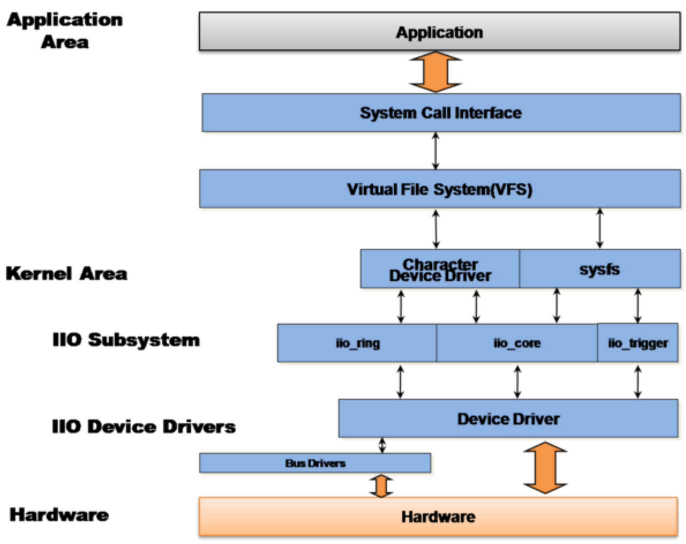


Figure 2.13: Linux subsets support for IIO.

The overall aim is to fill the gap between the somewhat similar hwmon and input subsystems. Hwmon is very much directed at low sample rate sensors used in applications such as fan speed control and temperature measurement. Input is, as its name suggests focused on human interaction input devices.: Keyboard, Mouse, Touch Screen, Joystick. In some cases there is considerable overlap between these and IIO. A typical device falling into the IIO category would be connected via SPI or I2C. However typical DMA operated devices such as ones connected to a high speed synchronous serial (McBSP, SPORT) or high speed synchronous parallel (EPI, PPI) or FPGA peripherals are also subject to this subsystem.

Functionality of IIO includes : basic device registration and handling, polled access to device channels via sysfs, event chrdevs, hardware ring buffer support, trigger and software ring buffer support, etc.

2.4 Summary

In this chapter we discussed mobile sensing framework using an open source libraries and Android SDK to collect sensor and environment data, users and device contexts and the ability to exchange among participating users. The app is built to support random and configurable sampling and compressive sensing. We have implemented and tested different sensor reading the mobile phone, logged the data, and visualized the sensed data using several plots and graphs in the app. We also developed a infrastructure sensing prototyping testbed along with Linux OS support with a light weight IP protocol to communicate with the server through internet.

Chapter 3

Compressed Sensing Approach to Sense-making

3.1 Introduction

The proliferation of device platforms and mobile applications (12 billion devices and over 3 million apps by 2017) has changed how humans interact. These new interactions enable the mobile device/user to be an active participant in the collection, sharing and dissemination of information-end platforms/users capture and process local context and communicate this information to other platforms/services using heterogeneous connectivities. We envision that the next generation mobile ecosystem will be far more sophisticated, complex and diverse than what is currently used. With each passing year, we see increasingly powerful smartphones being manufactured, which have a plethora of powerful embedded sensors like microphone, camera, digital compass, GPS, accelerometer, temperature sensors and many more [22]. Such platforms also support seamless external sensor connectivity (e.g. on body for health and wellness monitoring) that further equips such devices with rich and unique sensing capabilities. Moreover, the ability to easily program today's smartphones, enables us to exploit these sensors, in a wide variety of application such as personal safety, emergency and calamity response, situation awareness, remote activity monitoring, transportation and environment monitoring [19, 22]. Furthermore, the ability to share the sensed content with other users and applications has enabled crowds/humans (and their devices) to become information providers in a *crowdsensing* ecosystem. Effective use of the sensed data relies on effective "sense-making" that transforms the gathered data to meaningful information for improved situational awareness, decision making and control. This thesis focuses on enabling such "sense-making" from mobile users and devices.

Broadly, the emerging field of mobile phone sensing or crowdsensing can take multiple forms [19]. In *participatory sensing*, the user is directly involved in the sensing activity; this burden is alleviated in the *opportunistic sensing* paradigm by delegating and automating the sensing task to the mobile phone sensing system. In this thesis, we argue for a collaborative sensing approach where the users collaborate or cooperate to have better and reliable sensing information and obtain missing sensing information when specific sensors are not available in their own devices. Collaboration can be useful in generating more accurate and reliable information of spatial fields distributed across geographical areas and region, e.g., multiple temperature sensor readings in a space would be more reliable than a single reading. We discuss some specific applications of collaborative sensing and sense-making in following use case scenarios.

Disaster and emergency response: Mobile intelligent networks can play a key role in emergency response, surveillance and security, and battlefield operations. Consider a fire scenario where information from in-situ and mobile sensors can help in incident perimeter assessment as well as rapid localization of regions with high impact. Coordination among fire fighters is another important aspect in fire rescue operations. A collaborative mobile crowdsensing framework can be used to coordinate among the firefighters for their own safety and as well as quick evacuation. Collaborative sensing can provide situation awareness of different users in a facility during the rescue operation.

Based on the situations, rescue operations can be coordinated more effectively to reduce response time and save precious lives.

Personal health monitoring and wellness: Mobile phone sensing has the potential to continuously collect/sense data for health and wellness analysis. UbiFit Garden [8] is a mobile phone sensing system jointly developed by Intel and University of Washington, which uses small inexpensive on-body sensors and mobile phones along with machine learning techniques for activity modeling to infer people's activities throughout everyday life. In [26], stress-level of mobile user was measured using mobile phones, while [25] explored the use of smartphones in predicting the mode of the users. This can be extended to a family or a group of related people to jointly infer their moods, and exercise routines, exposures to pollutants etc. to find combined stress quotient. The same can also be used to achieve a family health indicator.

Smart spaces and their effective utilization: Smart buildings and smart spaces can use a collaborative sensing framework to monitor dynamic environmental conditions and requirements (e.g air conditioning and lighting preferences) and allows individuals to tailor lighting levels to their personal preferences and tasks to save energy footprints [33]. It can be used to understand the pattern of a facility usage (e.g. a library or a museum) and understand group behavior to improve the facility and its service.

The rapid growth in mobile sensors (in addition to sensors in our surrounding environment) and sensing data possess a serious challenge to the existing and traditional sense-making paradigm. Sense-making from large numbers of heterogeneous sensors, data, and mobile platforms is an extremely challenging task - there is a need for new architectures and softwares that can support sense-making both effectively and at scale. Another interesting aspect has to do with the accuracy of sensing/sense-making. Applications require information at different levels of accuracy and resolution - these tolerances can be leveraged to tradeoff accuracy for scale. In this thesis, we propose a hierarchical and extensible framework to support collaborative sensing at scale. In particular, we discuss two strategies: the use of hierarchical sensing and the application of compressive sensing techniques to address scalability/accuracy concerns in mobile crowdsensing. We implement the framework as a distributed middleware for mobile platforms, called Essence, and explore its use in several emerging mobile collaborative applications. The key benefits of the proposed collaborative sensing framework are as follows:

- ability to opportunistically set different sparsity levels to exploit regional fluctuations
- ability to analyze a region with more emphasis based on criticality or knowledge of events. Multi-resolution compressive thresholds i.e. number of sensing samples collected from a region based on the size and importance.
- ability to use different basis and sensing matrix by exploiting prior available data of different regions

- ability to use heterogeneous sensors with different characteristics and quality (as in different mobile phone)
- enable more energy-efficient sensing in the framework and context-aware information exchange

3.2 Related Works

Mobile phone sensing has recently attracted extensive research attention from both academia and industry due to its attractive applications. A comprehensive review of these applications can be found in a recent survey paper [22]. Several issues relating to sensing and coverage have been well studied for mobile sensor networks [53]. However, only few works have addressed collaborative and compressive sensing specifically with mobile phones. In [29], the authors presented analytical results on the rate of information reporting by uncontrolled mobile sensors needed to cover a given geographical area, and demonstrate the feasibility of using existing software and standard protocols for information reporting and retrieval to support a large system of uncontrolled mobile sensors using a test-bed. In [44], the authors proposed a protocol, Aquiba, that exploits opportunistic collaboration of pedestrians to achieve energy efficiency and reduce data redundancy. Its performance was studied via simulations. In [48], the authors introduced mechanisms for automated mapping of urban areas that provide a virtual sensor abstraction to applications. They also proposed spatial and temporal coverage metrics for measuring the quality of acquired data.

Luo et al. [27] were the first to examine the notion of compressive sensing over large scale wireless sensor networks (WSN) to reduce the number of transmission. Their data gathering compressive scheme reduced the number of transmission from $\mathcal{O}(N^2)$ transmission to $\mathcal{O}(NM)$ where the number of measurement $M \ll N$, the cluster size. However, they assume that the data field is smooth with uniform sensor characteristics, negligible sensor noise and heterogeneity, and global constant sparsity without leveraging the local or regional fluctuations of the signal field. Due to these assumptions as well as unique differences between traditional WSN from mobile phone sensing (e.g. static vs high mobility, limited computation and power resource vs considerable computational and rechargeable energy resource, and mostly broadcast mode operation vs bidirectional multi-network operation, limited number of sensors per node to varied types of sensor in a node), naive and plain implementation of their technique can introduce redundant data communications (e.g. from the leaf nodes) and reduction in overall network throughput [28]. Moreover, the assumption of uniform compression threshold across the network regardless of the data field characteristics and the inability to exploit regional fluctuations along with the sensor characteristics in more realistic situations can result in poor compression efficiency and thereby impacting the energy efficiency. We therefore propose a hierarchical distributed architecture where the local field sparsity and sensing characteristics can be effectively and jointly exploited at different levels for efficiency and scalability. Unlike WSN nodes that usually lack computation and power resources , the mobile phones being considerably resourceful can adopt compressive sensing at each nodes for example in energy efficient context processing as discussed in the subsequent sections.

3.3 Hierarchical Architecture and Middleware for Sense-Making

The key idea of our approach to achieve scalable sense-making is to exploit hierarchical architecture combined with tunable/configurable compressive sensing both in spatial and temporal dimensions at different levels. The conceptual architecture of the framework supporting multi-tiered collaborative and compressive sensing is illustrated in Fig. 3.1 where the network is hierarchically organized and spatially distributed through multiple local cluster (LCs) which in turn is formed from spatial distribution of nano cluster (NCs). The NCs consists of mobile nodes connected to a central head or a broker. The head broker in the LCs in turn communicate with other LCs and the public cloud in the next hierarchy. The NCs are formed in the region of interest and the workload of the sink nodes (i.e. broker) is distributed among multiple sink nodes in the LCs such that all the mobile nodes need not flow the information to a single node to overcome network range and scalability bottlenecks. This hierarchy allows the nodes to collaborate through the broker (performing the operation of the sink node/collector) and concatenate the results of the NCs for the local region.

The hierarchical approach is based on the observation that the number of random observations from any region should correspond to the local spatio-temporal sparsity as well as the NC size instead of the global sparsity. Intuitively, this should work better than the global scheme as the local correlation among the nodes can be exploited in the local area (i.e. LCs) than global area. Besides, local sparsity is easy to compute and often prior available data about the local regions can be exploited to improve the sensing efficiency and data transmission requirements thereby saving energy.

The Essence Middleware: We next present **Essence**, a collaborative sensing middleware platform that enables mobile phone based physical sensing and sensemaking. The following are the key features in Essence:

- **Mobile Phone & Infrastructure Sensing**: Essence enables and provides data capture from different sensors on (or attached to) mobile phones by providing configurable sensing probes. The user can configure the sensing probes and sampling techniques through a sensing API.

- **Context Determination, Analysis & Processing**: Essence enables the use of the sensed information to determine high level features such as user activities,

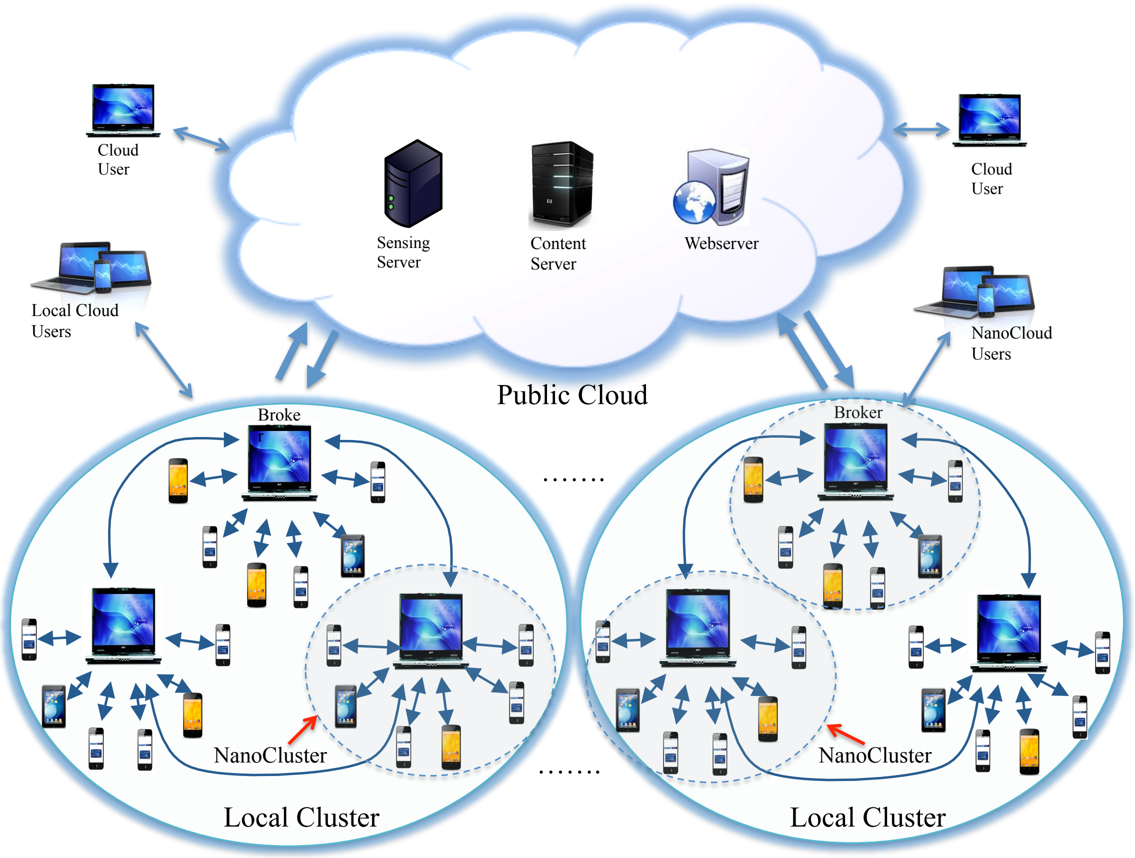


Figure 3.1: Multi-tiered hierarchical structured mobile cluster architecture for scalable collaboration and compressive sensing. The multi-tiered architecture consists of hierarchy of local cluster which in turn consists of nano cluster.

physiological parameters, events, and their correlations. The shared sensing and context are used to determine group context, behavior, and preferences.

- **Decision Making and Control:** Esssence using the sensemaking machine learning approaches to perform meaningful decision and control that can physically actuate based on distributed feedback. The Essence framework differs in this specific aspect with respect to our previous SenseDroid framework. Multimodal information is fused to obtain better inferences and predictions.

- **Communication and Collaboration**: Essence provides libraries and APIs for communication, service discovery, and collaboration among mobile phones for different network topologies (e.g. client-server and peer-to-peer).

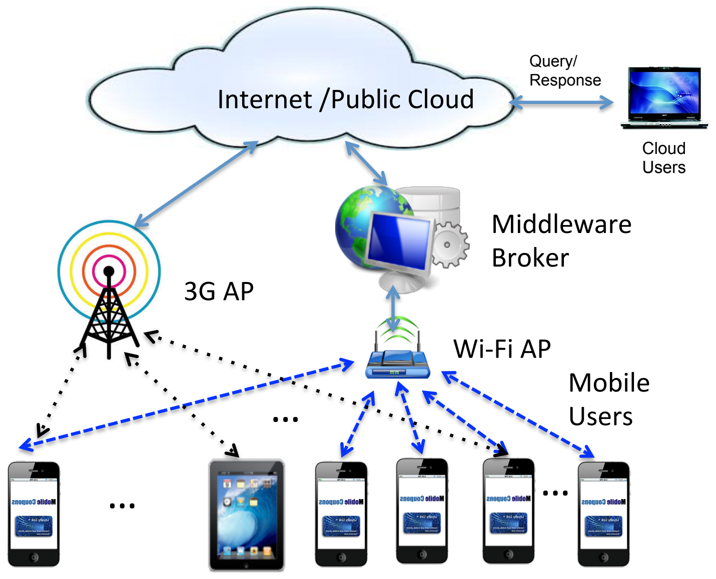
- **Data Logging and Retrieval**: Essence provides data management routines and interface to a light weight database such as SQLite for data logging and efficient sensor data processing and storing.

- **Query and Filtering**: Essence supports on-demand query and filtering functionality from different participating users. Filtering helps deliver only the relevant information to collaborating users.

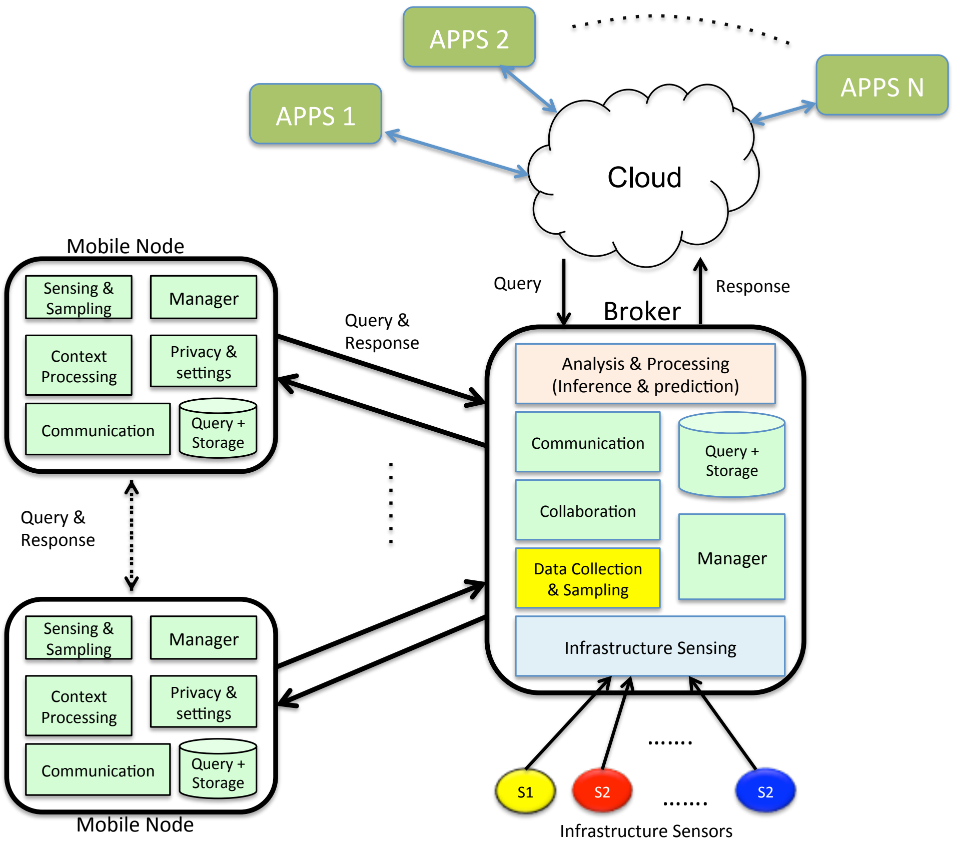
We have developed and implemented an initial prototype application for Android smartphones to test some of the above functionalities of the Essence framework. In general, the architecture and design features can be ported to other platforms. Essence is distributed across the different levels of the hierarchical architecture described above.

Fig. 3.2 shows the details of the broker functionalities and the mobile node middleware and application components necessary in an NC to supporting collaboration and compressive sensing over proposed hierarchical network. Unlike a traditional WSN, the mobile NC supports bidirectional data flow between the nodes and the broker using multiple networks like WiFi, GSM, bluetooth etc. that enables dissemination of collective information and collaboration among the mobile nodes through the broker. However, in case of the compressing sensing approach, the broker performs stochastic (random) spatial sampling in various nodes. If N mobile sensors nodes are uniformly and randomly distributed in an NC such that the compressive sample of given sparsity requires M random measurements from these N nodes, the broker initiates these measurements by commanding and telemetering the selected nodes with the sensor. The broker can also use measurement from infrastructure sensors in absence of either enough sensor in the mobile nodes or to off-load the burden of sensing cost from the mobile nodes.

The mobile phones /nodes in the NCs provide unique sensing abilities and capabilities for both physical sensors as well as computationally enabled virtual sensors [42] as shown in Fig. 3.3. The Essence framework provides individual probes for available physical sensors along with their configurable measurement parameters such as sampling rate, duration etc. and fuse these physical sensors measurements to construct more meaningful sensors (e.g. orientation, compass and inclinometer sensors in Fig. 3.3). In similar ways, indirect sensing by computational means can be used to derive computationally enabled virtual sensors such as situation specific contexts pertaining to user location, activity, environment, health, emotions, and social scenarios. Thus, in addition to basic physical sensor probes [1], Essence provides several virtual sensing probes corresponding to different types of contextual informations. Moreover, unlike some of the recent works in mobile context processing [32, 25, 26], Essence employs compressive sensing in the temporal dimension to exploit the temporal correlation in the sensor measurements to achieve energy efficient contexts determinations. As an example, we use compressive sampling instead of continuous uniform measurement of the GPS and WiFi to derive the 'IsIndoor' flag with similar accuracy while saving energy consumptions. This 'IsIndoor' flag spatial field can be used, for instance, during an earthquake to assess the potential dangers to human life. Thus, unlike WSN and the work in [27], due to the considerable computational



(a)



(b)

Figure 3.2: (a) Mobile phone sensing and collaboration architecture (b) basic components of the mobile thin client and the broker in Essence framework.

and programable resources in mobile phone, the use of configurable compressive sensing at each node enables the unique ability to jointly perform spatio-temporal compressive sensing of both physical and virtual sensors in the proposed framework. As an example, Fig. 3.4 shows the reconstruction accuracy of a accelerometer signal of 256 samples from just 30 random samples in determining the 'IsDriving' context of the mobile node. When the same is applied using the spatial compressive sensing over a region, can provide indications to the traffic situations. We explain the details of spatio-temporal compressive sensing approach and its fundamentals in the next section.

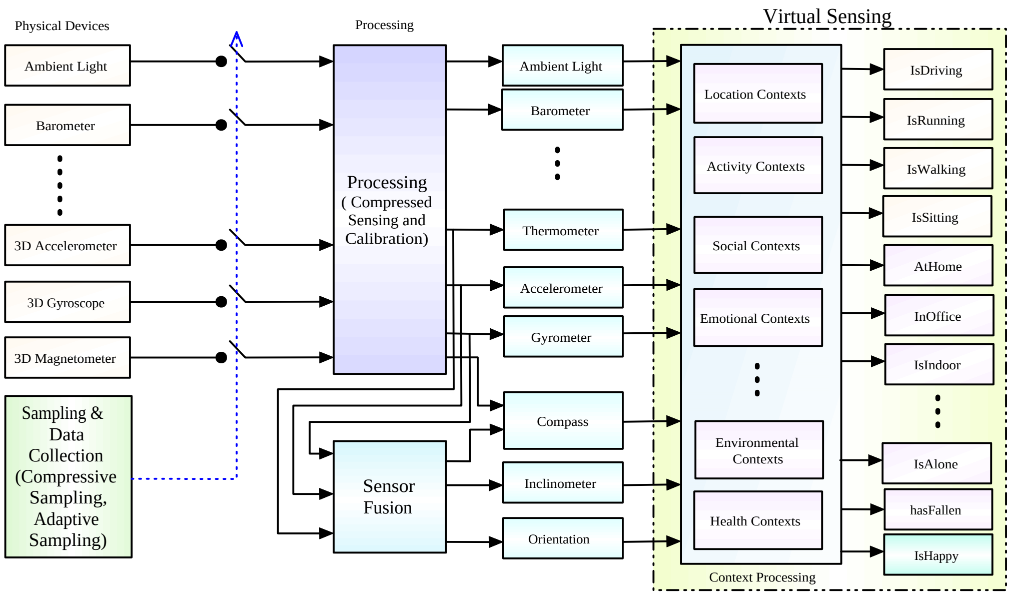


Figure 3.3: Essence provides sensing probes for several physical sensors and ability to construct virtual sensors using compressive context processing. The additional virtual sensing abilities and probes provide unique opportunities for collaborative applications.

3.4 Collaborative Compressive Sensing for Hierarchical Sense-making

Essence framework supports multi-tiered data aggregation of spatio-temporal sparse fields and its reconstruction. A sparse signal is a signal that can be represented with a small number of nonzero coefficients and hence can contain most of its salient information in a relatively small number of random projections. It follows that if a signal is compressible in some orthonormal basis, then a very accurate reconstruction can be obtained from

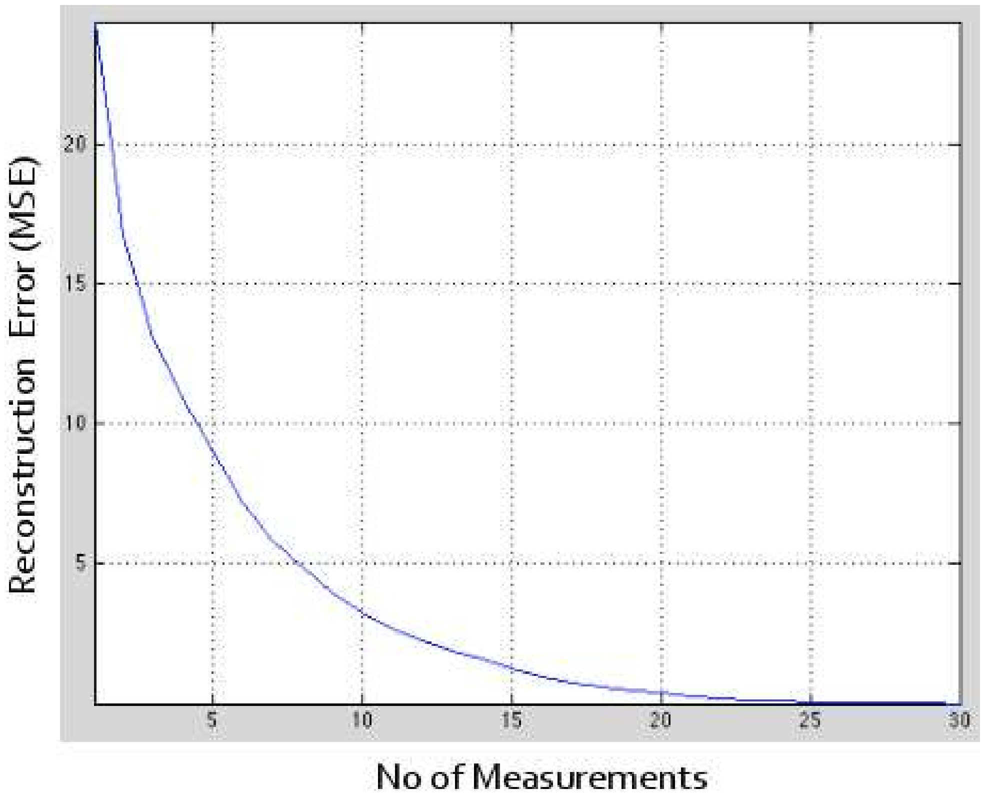


Figure 3.4: Accuracy of reconstruction as a function of number of measurements. As the number of measurements (or compression ratio) increases, the reconstruction error is reduced.

random measurement and projections. This sampling approach is used in a hierarchical architecture to show that signals and spatio-temporal sparse fields can be accurately recovered from fewer random measurements and projections contaminated with noise. In order to achieve this, the total spatial field area is subdivided into zones and each zone is covered by the mobile local cloud (LCs). The total spatial field is then the sum of the all the subfields computed and processed by the local cloud. The multi-tiered hierarchical structured architecture enables compressive processing as different levels of granularity and accuracy. Increased emphasis, attention and resources can be directed to the areas of most impact and effects.

Let $f(i, j)$ be the two-dimensional spatial field map where i and j represents the coordinate of the location within the two-dimensional spatial field of a zone as in Fig. 3.6. When the spatial field map is discretized to $f[i, j]$, the coordinates $i \in \{1, 2, .., W\}$ and $j \in \{1, 2, .., H\}$ where W and H are the width and height of the discretized spatial field map respectively. If we consider prior available data of a LC – a set of T spatial fields $\mathrm{F} = \{f_1[i, j], f_2[i, j], ..., f_T[i, j]\}$ taken at time instants $t_1, t_2, .., t_T$, these can be used

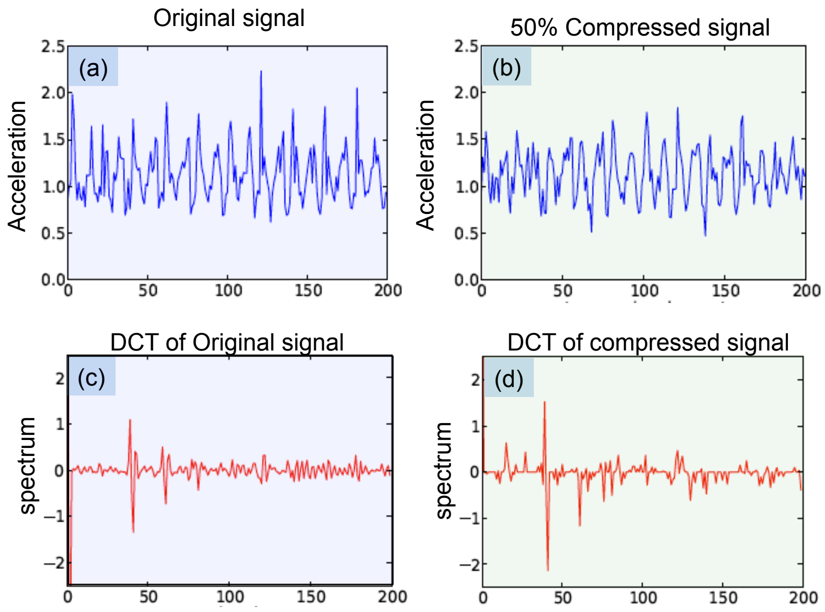


Figure 3.5: Compressive mobile context processing: sensing of the accelerometer during walking. Determination of the IsWalking context with 50% less samples can save half the sensing energy cost. The compression ratio adaptively selected by middleware according to the sparsity of the signals used in the specific context determination.

to improve sensing by exploiting local correlation during reconstruction. Specifically, to monitor a discrete spatial field map that has $N = WH$ variable parameters (and hence N-dimensional), the number of sensors needed is less than equal to number of unknown variable or parameters i.e. $\#Sensors \leq \#Parameters$. By transforming the data, a low dimensional representation is possible with a very less error. In the transformed domain, the spatial field map can be represented using few principal components/parameters such that $K \ll N$ (with the assumption of sparsity [12, 6]).

Let the two-dimension spatial field map $f[i,j]$ can be represented using a one-dimensional vector $x[k]$ where $1 \leq k \leq N$ and $N = WH$ such that

$$x[k] = f\left[k \, mod \, H, floor\left[\frac{k}{W}\right]\right]. \tag{3.1}$$

In other words, stack the columns of the two-dimensional map to transform into a vector where N is the total no of grid points and $x[k]$ represent the sensor measurement at k-th grid point. The set of one dimensional spatial field traces $\Gamma = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_T}\}$ can thus be represented as matrix $\mathbf{X}$ of size $T \times N$ with each row indicating a trace $\mathbf{x}$. In many scenarios , in spite of the sparsity of the field, the full spatial field $x[k], k = 1..N$ at all N locations is not available . Only M samples obtained from M spatial field sensors ($M \ll N$) located at $L = \{i_1, i_2, ..., i_M\}$ is available. Let $x(L)$ denotes the field measurement at these locations. The spatial field characterization problem is then the

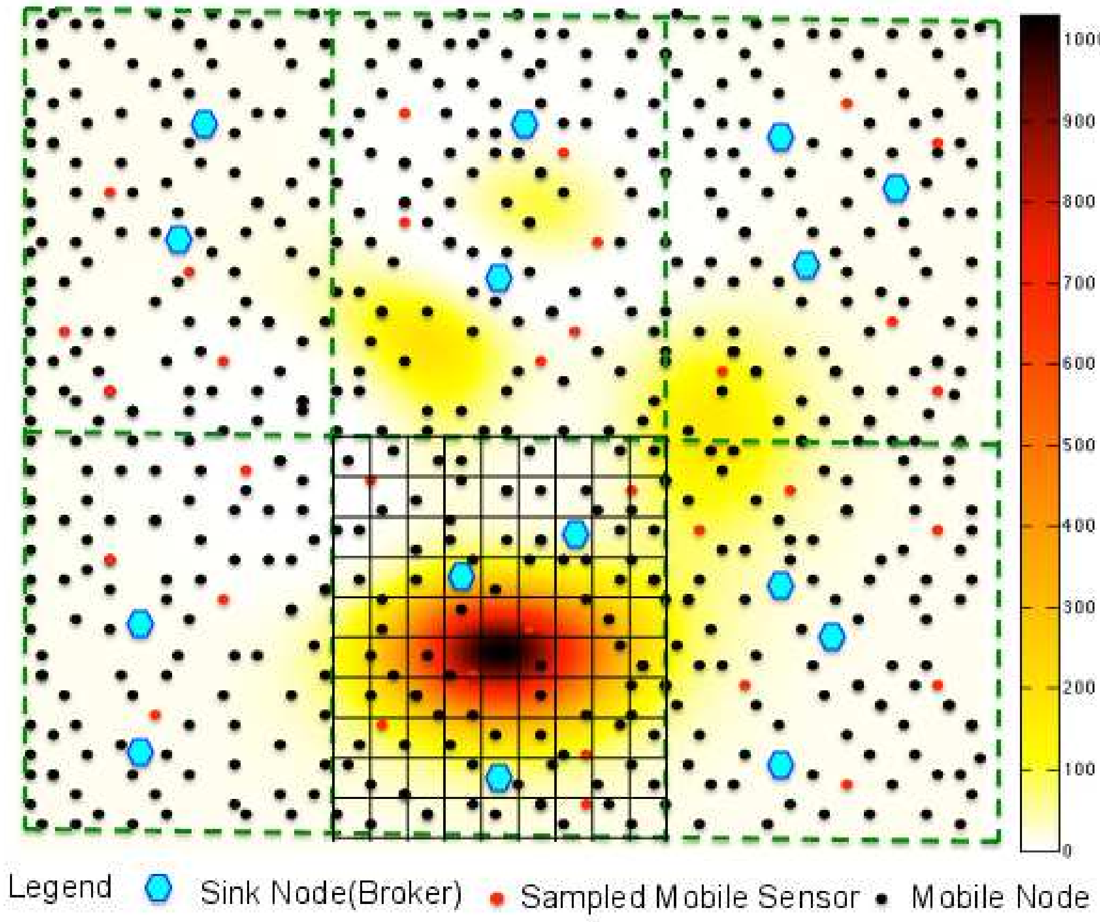


Figure 3.6: Distributed collaborative compressive mobile sensing of spatio-temporal sparse fields. Based on the type of sensing field, the signal sparsity, and accuracy requirement, the middleware broker decides the compression ratio during data aggregation in each zone.

estimate the field at each of N points given measurement of M sensors at location L in a LC.

Any vector $\mathbf{x}$ can be represented using a basis Φ as,

$$\begin{aligned} x[k] &= \sum_{n=0}^{N} \Phi[k,n]\alpha[n], \\ \mathbf{x} &= \mathbf{\Phi}\alpha \end{aligned} \tag{3.2}$$

where $\alpha[n]$ are the coefficients of the expansion over the basis Φ. Once we define a basis Φ for the data, knowing the coefficients α is equivalent to knowing the spatial field map $\mathbf{x}$. The basis Φ is often selected as transformation matrix of FFT or DCT.

If the signal is k-sparse in the transformed domain, then the locations of the non-zero K coefficients in α can be denoted by $J = \{j_1, j_2, ..., j_K\}$. If M sensors are available at location $L = \{i_1, i_2, ..., i_M\}$, then the sensor samples in (3.2) can be represented as:

$$\begin{aligned} x[i_1] &= \Phi[i_1, j_1]\alpha[j_1] + \cdots + \Phi[i_1, j_K]\alpha[j_K] \\ \vdots &= \vdots \\ x[i_M] &= \Phi[i_M, j_1]\alpha[j_1] + \cdots + \Phi[i_M, j_K]\alpha[j_K] \end{aligned} \tag{3.3}$$

$$x(L) = \Phi(L, J)\alpha(J). \tag{3.4}$$

We can approximate spatial field map with $\hat{x}$ with a linear combination of K columns of Φ and K elements of α out of N as

$$\hat{\mathbf{x}} = \begin{bmatrix} \Phi[1,1] & \cdots & \Phi[1,K] \\ \vdots & \ddots & \vdots \\ \Phi[N,1] & \cdots & \Phi[N,K] \end{bmatrix} \begin{bmatrix} \alpha[1] \\ \vdots \\ \alpha[K] \end{bmatrix} = \mathbf{\Phi_K}\alpha_\mathbf{K} \tag{3.5}$$

where the subscript Kindicates the selection of K columns **.**The spatial field map is now defined by only K coefficients of α_K in the basis Φ_K given by:

$$\alpha_\mathbf{K} = \mathbf{\Phi}^\dagger_\mathbf{K}\hat{\mathbf{x}} = \left[(\mathbf{\Phi}^*_\mathbf{K}\mathbf{\Phi_K})^{-\mathbf{1}}\,\mathbf{\Phi}^*_\mathbf{K}\right]\hat{\mathbf{x}} \tag{3.6}$$

where $\Phi^\dagger_K$ is the pseudo inverse for the overdetermined system in (3.5). To solve (3.6) , we need the knowledge of the field $x[k]$ at every location $k = 1\cdots N$. For M available sensors at location $L = \{i_1, i_2, ..., i_M\}$, we can represent $x(L)$ as:

$$\mathbf{x_S} = \begin{bmatrix} \Phi[i_1,1] & \cdots & \Phi[i_1,K] \\ \vdots & \ddots & \vdots \\ \Phi[i_M,1] & \cdots & \Phi[i_M,K] \end{bmatrix} \begin{bmatrix} \alpha[1] \\ \vdots \\ \alpha[K] \end{bmatrix} = \tilde{\mathbf{\Phi}}_\mathbf{K}\alpha_\mathbf{K}, \tag{3.7}$$

where $\tilde{\Phi}_K$ is a matrix formed from the rows of $\mathbf{\Phi_K}$ corresponding to the sensor location L, $\mathbf{x_S}$ is the sensor measurement (known). The coefficients $\alpha_\mathbf{K}$ are unknown in the system in (3.7) and can be solved for two cases: (a) underdetermined case, $M < K$ (b) overdetermined case , $M \geq K$. For the underdetermined case $M < K$ solution with infinite solutions, the coefficients are assumed to be sparse and only non-zero significant coefficients are selected. The solution of $\alpha_\mathbf{K}$ can be uniquely determined by solving the following optimization [6, 12]:

$$\begin{aligned} &\underset{\alpha_\mathbf{K}}{Minimize} \quad \parallel \alpha_\mathbf{K} \parallel_0 \\ &subject\ to\ \mathbf{x_S} = \tilde{\mathbf{\Phi}}_\mathbf{K}\alpha_\mathbf{K} \end{aligned} \tag{3.8}$$

where $\parallel \bullet \parallel_0$ stands for $\mathcal{L}_0 - norm$ of a vector i.e. the number of non-zeros in the vector. Optimization in (3.8) attempts to minimize the number of non-zeros in $\alpha_\mathbf{K}$while satisfying the constraint for a unique solution that is as sparse as possible. The optimization problem (3.8) is NP-hard and hence is extremely difficult to solve [12, 6]. A more efficient technique to find a sparse solution is based on $\mathcal{L}_1 - norm$ regularization -a relaxed version of $\mathcal{L}_0 - norm$ [12, 6]:

$$\begin{array}{c} \underset{\alpha_{\mathbf{K}}}{Minimize} \quad \| \alpha_{\mathbf{K}} \|_1 \\ subject\ to\ \mathbf{x_S} = \tilde{\mathbf{\Phi}}_{\mathbf{K}} \alpha_{\mathbf{K}} \end{array} \tag{3.9}$$

where $\| \bullet \|_1$ stands for $L_1 - norm$ of a vector i.e., the summation of the absolute value of all elements in the vector. The $\mathcal{L}_1 - norm$ regularization in (3.9) can be re-formulated as an Linear Programing (LP) problem and solved efficiently. As the cost function $\| \alpha_K \|_1$is not smooth, linear programing can not be directly applied. To apply LP, slack variables are introduced $\{\theta_i : i = 1, 2, .., K\}$ such that $-\theta_i \leq \alpha_i \leq \theta_i$ and (3.9) can be re-written as [6]:

$$\begin{array}{c} \underset{\alpha_{\mathbf{K}}, \theta}{Minimize} \quad \theta_1 + \theta_2 + \cdots + \theta_K \\ subject\ to\ \mathbf{x_S} = \tilde{\mathbf{\Phi}}_{\mathbf{K}} \alpha_{\mathbf{K}} \\ -\theta_i \leq \alpha_i \leq \theta_i \end{array} \tag{3.10}$$

Intuitively, by minimizing the cost function in (3.10), all the constraints become active i.e. $| \alpha_i |= \theta_i$ and hence (3.9) and(3.10) is equivalent. Roughly speaking, if the N-dimensional vector $\alpha_{\mathbf{K}}$ contains K non-zeros and the linear equation $\mathbf{x} = \mathbf{\Phi} \alpha_{\mathbf{K}}$ is well-conditioned, the solution α_K can be almost uniquely determined (with a probability nearly equal to 1) from M sampling points, where M is in the order of $O(K * log(N))$ [45, 12, 6]. Note that M (the number of sensors or measurements) is a logarithmic function of N (the number of unknown parameters).

A convenient closed form solution of $\alpha_{\mathbf{K}}$ based on ordinary least square estimate (OLS) for well-conditioned overdetermined case [i.e. $M \geq K$ and $rank(\tilde{\Phi}_K) = K$] is given by :

$$\alpha_{\mathbf{K}} = \tilde{\mathbf{\Phi}}^{\dagger}_{\mathbf{K}} \mathbf{x_S} = \left[\left(\tilde{\mathbf{\Phi}}^{*}_{\mathbf{K}} \tilde{\mathbf{\Phi}}_{\mathbf{K}} \right)^{-\mathbf{1}} \tilde{\mathbf{\Phi}}^{*}_{\mathbf{K}} \right] \mathbf{x_S}. \tag{3.11}$$

On the other hand, a generalized least square (GLS) solution considering sensor heterogeneity and noisy measurement $\mathbf{x_S} + \mathbf{w}$ where the noise $\mathbf{w}$ is having a distribution with covariance $\mathbf{V}$ and mean $\boldsymbol{\mu}$ i.e. $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{V})$ results in:

$$\boldsymbol{\alpha_K} = \tilde{\mathbf{\Phi}}^{\dagger}_{\boldsymbol{K}} \boldsymbol{x_S} = \left[\left(\tilde{\mathbf{\Phi}}^{*}_{\boldsymbol{K}} \boldsymbol{V}^{-\mathbf{1}} \tilde{\mathbf{\Phi}}_{\boldsymbol{K}} \right)^{-\mathbf{1}} \tilde{\mathbf{\Phi}}^{*}_{\boldsymbol{K}} \boldsymbol{V}^{-\mathbf{1}} \right] \boldsymbol{x_S} \tag{3.12}$$

The same problem can be re-written for full spatial field reconstruction for M sensors located at $L = \{i_1, i_2, ..., i_M\}$ as a sparse regression problem, which can be formulated as the following optimization [46]:

$$\begin{array}{c} \underset{\alpha}{Minimize} \quad \| \mathbf{x} - \mathbf{\Phi} \alpha \|_2^2 \\ Subject\ to \quad : \| \alpha \|_0 \leq K \end{array} . \tag{3.13}$$

The optimization problem in (3.13) can be effectively solved using the orthogonal matching pursuit (OMP) algorithm [46].The solution to (3.13) is affected by the sensor noise and errors, which in turn effects reconstruction accuracy. Total error ϵ introduced in the reconstruction is sum of coefficient truncations or approximation error, ϵ_a , error due to numerical ill-conditioning, ϵ_c, and measurement noise error, ϵ_m . Note that, once we have fixed M, increasing K will in general increase the reconstruction error ϵ_c (worse conditioning) and decrease the approximation error ϵ_a (better approximation). Therefore, we should pick an optimal K such that the sum ϵ is minimal. In presence of sensor noise and errors $\mathbf{w}$, sensor measurement in (3.7) becomes

$$\mathbf{x_s} + \mathbf{w} = \tilde{\mathbf{\Phi}}_{\mathbf{K}} \alpha_{\mathbf{K}}. \tag{3.14}$$

There is no exact solution to (3.14) for α_K but a least square estimation (LSE) problem is solved such that the error w.r.t. the measured field is minimized using the algorithm in Fig. 3.7. The algorithm is primarily implemented in the brokers but is also used by the nodes for context processing.

3.5 Application of Essence Framework for Predicting the Presence of West Nile Virus

In this section we use the compressive sensing technique in Essence framework to predict temperature profile of a region from few measurements and then extend the correlation between the temperature and the presence of harmful viruses, called West Nile Virus (WNV) [17, 30]. West Nile virus is most commonly spread to humans through infected mosquitos. Around 20% of people who become infected with the virus develop symptoms ranging from a persistent fever, to serious neurological illnesses that can result in death. West Nile virus (WNV) is a leading cause of mosquito-borne disease in the United States (Fig. 3.8a). Annual seasonal outbreaks vary in size and location. Predicting where and when higher than normal WNV transmission will occur can help direct limited public health resources.

The WNV virus presence has correlation with that of the temperature and weather profile of the region. Above average annual temperature was associated with increased likelihood of higher than normal WNV disease incidence, nationally and in most regions as illustrated in Fig. 3.9a. Lower than average annual total precipitation was associated with higher disease incidence in the eastern United States, but the opposite was true in most western regions [17, 30]. Although multiple factors influence WNV transmission, these findings show that anomalies in temperature and precipitation are associated with above average WNV disease incidence.

In 2002, the first human cases of West Nile virus were reported in Chicago. By 2004 the City of Chicago and the Chicago Department of Public Health (CDPH) had established a comprehensive surveillance and control program that is still in effect today. Every week from late spring through the fall, mosquitos in traps across the city are tested for the virus. The results of these tests influence when and where the city will spray airborne pesticides to control adult mosquito populations. Given weather, location, testing, and spraying data, this competition asks you to predict when and where different species of mosquitos will test positive for West Nile virus. A more accurate method of predicting outbreaks of West Nile virus in mosquitos will help the City of Chicago and CPHD more efficiently and effectively allocate resources towards preventing transmission of this potentially deadly virus. The location of all the traps and all the sprays that were carried out through out the city is shown in Fig. 3.10.

Our objective is to make sense of the presence of the WNV virus from the various earlier measurements and weather data and predict it as accurately as possible at location through out the complete region. As accurate the prediction of the WNV virus requires collection and time consuming test for detecting the presence of WNV virus, any reduction in the number of tests required will reduce the overall time in making an early prediction for the whole region. As the presence of the WNV virus in region is inherently a sparse signal, we can directly apply random measurement results from different traps across the city to reconstruct and recover the complete signal. Thus, without waiting for more test to be carried out, the compressive sensing approach of the Essence framework can be used to predict the presence of the virus quickly.

In order to do these, the CS framework assumes that traps nodes as sensing node which can be randomly sampled to sense the presence of the WNV virus. Even though, physical test needed to be conducted to find virus presence in the mosquitos in the trap, we make the assumptions this information for any trap can be obtained seamlessly. If the total no of traps are N, then $M = KlogN$ random measurements would be sufficient to predict the presence of the WNV virus in the remaining traps. Fig. 3.11 shows the target and training input data. Fig. 3.12 shows the Matlab code for the simulation of the CS based WNV prediction.

Fig. 3.13 shows the CS based recovery and prediction of for $M = 900$ measurements to $M = 1800$ measurements out of 9000 signal points (traps). In other words, the results indicate that if we make 900 random measurements of the mosquito traps, it may not be sufficient enough to predict presence of the WNV virus in the 9000 traps. On the other hand, with 1500 random measurement, the WNV virus can be predicted in all the traps as shown in Fig. 3.13. The effect of the number of measurement on the signal recover is plotted in Fig. 3.14. A similar approach is adopt for the temperature prediction across all the location based on few measurements.

3.6 Summary

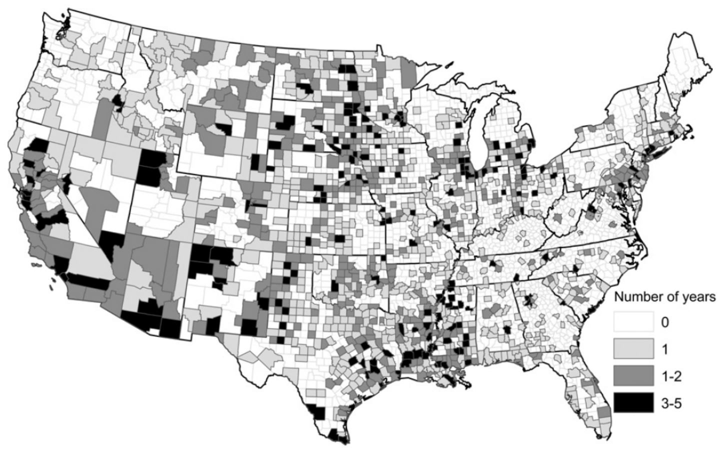
In this chapter we discussed the foundation for compressive sensing based sensor selection, data aggregation, and data recovery in the sense-making process. We showed how the compressive sensing approach can be applied in temporal as well as spatial dimension in order to improve scalability while trading accuracy or vice versa. We extended the approach to consider heterogeneity of sensors in order to develop the capability to apply to large class of applications. Lastly, we show an application of the framework in predicting West Nile Virus with few measurements.

Algorithm: Compressive Heterogeneous Sensing

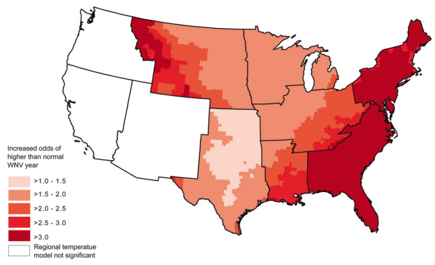
Input: measured vector $\mathbf{x_S}$, target sparsity M(no of sensor measurement), and size of full signal N
Output: Coefficient Indices J, Sensing matrix $\tilde{\Phi}_K$, reconstructed signal $\hat{x}$

```
1. Initialize coefficient indexes $J = \emptyset$, residual $\mathbf{e}_r = \mathbf{x_S}$, $\alpha_K = \emptyset$
2. Formation of the basis matrix $\Phi$
3. **While** stop criteria not met **do**
    (a) $\mathbf{e_r^{new}} = \Upsilon(\mathbf{e_r})$, where $\Upsilon$ is an interpolation function such that $\Upsilon : \mathbb{R}^K \to \mathbb{R}^N$
    (b) $\alpha_\mathbf{r} = \Phi^\dagger \mathbf{e}_r^{new}$
    (c) Choose a subset of coefficient indices $I \subset J$ from $\alpha_\mathbf{r}$ for deciding the significant coefficients
    (d) Update coefficient index set $J = J \cup I$
    (e) Find the coefficients using Ordinary Least Square (OLS) or Generalized Least Square(GLS):
        i. OLS Solution for homogenous sensors: $\alpha_\mathbf{K} = \tilde{\mathbf{\Phi}}_\mathbf{K}^\dagger \mathbf{x_S} = \left[\left(\tilde{\mathbf{\Phi}}_\mathbf{K}^* \tilde{\mathbf{\Phi}}_\mathbf{K}\right)^{-\mathbf{1}} \tilde{\mathbf{\Phi}}_\mathbf{K}^*\right] \mathbf{x_S}$
        ii. GLS Solution for heterogenous sensors: $\alpha_\mathbf{K} = \tilde{\mathbf{\Phi}}_\mathbf{K}^\dagger \mathbf{x_S} = \left[\left(\tilde{\mathbf{\Phi}}_\mathbf{K}^* \mathbf{V}^{-\mathbf{1}} \tilde{\mathbf{\Phi}}_\mathbf{K}\right)^{-\mathbf{1}} \tilde{\mathbf{\Phi}}_\mathbf{K}^* \mathbf{V}^{-\mathbf{1}}\right] \mathbf{x_S}$ where $\mathbf{V}$ is covariance matrix of sensor accuracy characteristics.
    (f) $\mathbf{e_r} = \mathbf{x_S} - \tilde{\mathbf{\Phi}}_\mathbf{K} \alpha_\mathbf{K}$
   **end while**
4. Reconstructed signal $\hat{x} = \Phi_K \alpha_K$
```

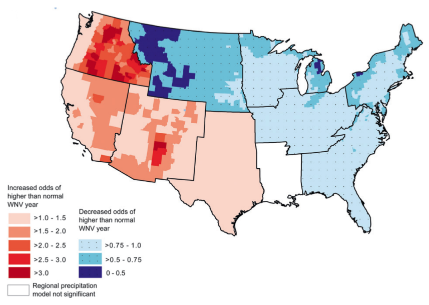
Figure 3.7: Compressive Heterogeneous Sensing for IoTs.



(a)

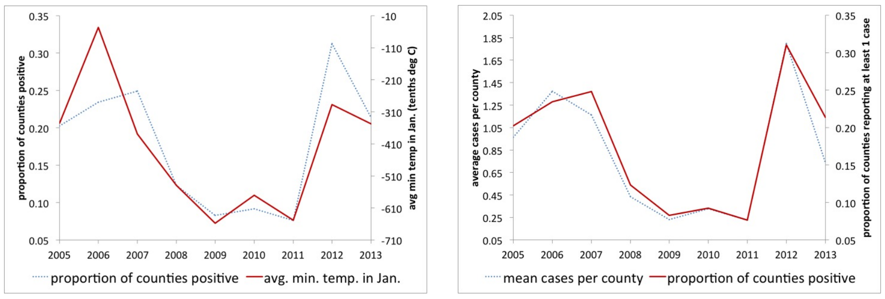

Increased odds of higher than normal WNV year
>1.0 - 1.5
>1.5 - 2.0
>2.0 - 2.5
>2.5 - 3.0
>3.0
Regional temperatue model not significant

(b)


Increased odds of higher than normal WNV year
>1.0 - 1.5
>1.5 - 2.0
>2.0 - 2.5
>2.5 - 3.0
>3.0
Decreased odds of higher than normal WNV year
>0.75 - 1.0
>0.5 - 0.75
0 - 0.5
Regional precipitation model not signifiicant

(c)

Figure 3.8: (a)Annual average incidence of WNV in each county in USA (b)Annual average temperature in each county in USA, (c)Annual precipitation in each county in USA.



(a) (b)

Figure 3.9: Correlation between presence of WNV and the temperature. (a)Minimum temperature predicts reporting (b) WNV human cases and reporting per county.

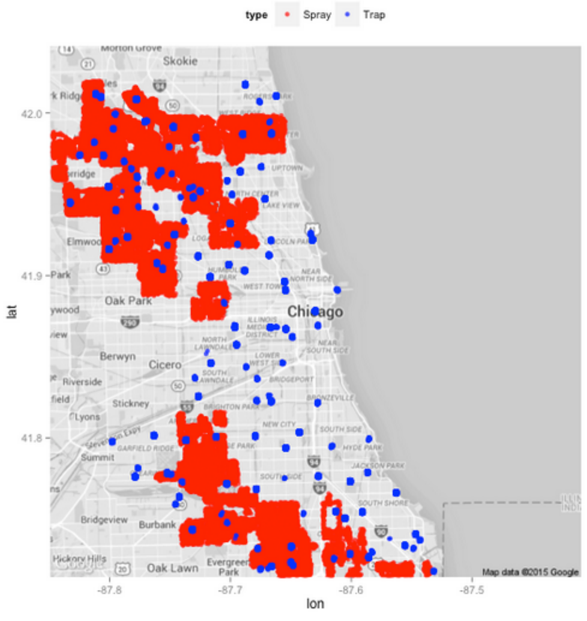
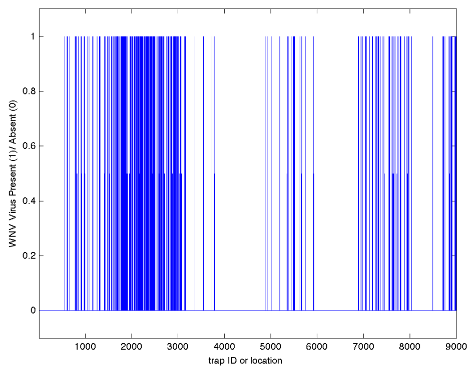
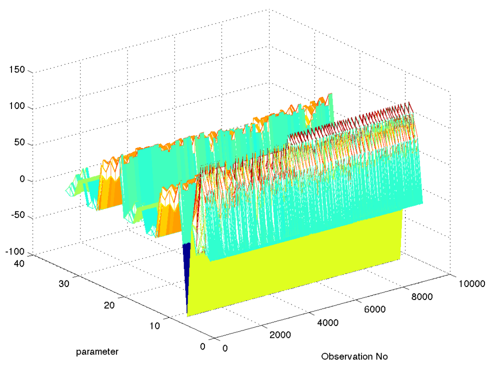


Figure 3.10: Mosquito trap locations and all sprays in 2007-2014.



(a)



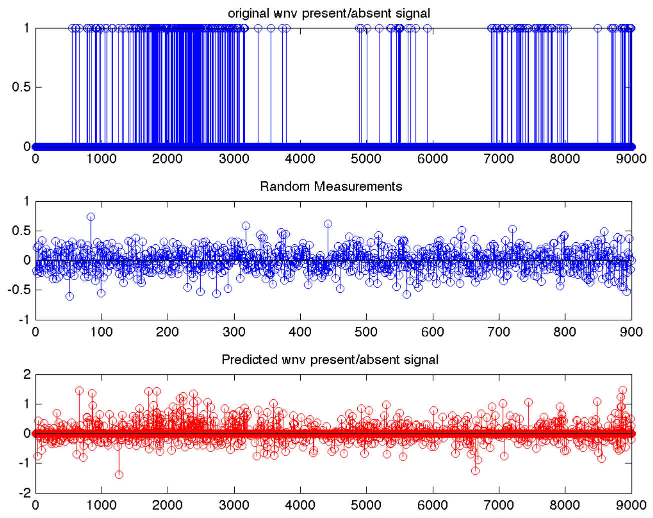
(b)

Figure 3.11: WNV target and train data. (a) WNV presence(1)/absence (0) at different trap location (b) weather and other monitored parameters (total of 40 parameters) data at the trap locations.

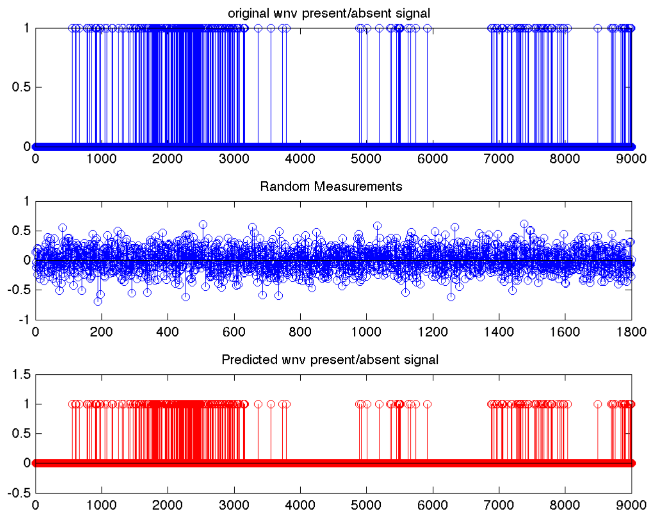
%Matlab Code for WNV Virus predictions

```
% signal length
N = length(train_y);
% number of spikes in the signal
T = nnz(train_y);
% number of observations to make
M = 0.1*N*i;
% wnv signal
x=train_y'
% measurement matrix
disp('Creating measurment matrix...');
A = randn(M,N);
A = orth(A')';
disp('Done.');
% observations
y = A*x;
% initial guess = min energy
x0 = A'*y;
% solve the LP
tic
xp = l1eq_pd(x0, A, [], y, 1e-3); %using L1-magi toolbox
toc
%plot the results
figure(2)
set (2, 'color', [1 1 1])
subplot(3,1,1)
stem(x)
title('original wnv present/absent signal')
subplot(3,1,2)
stem(y)
title('Random Measurements')
subplot(3,1,3)
stem(xp,'-r')
title('Predicted wnv present/absent signal')
print(2, '-dtiff', ['wnv_k_', num2str(i)])
```

Figure 3.12: Matlab code for the compressed sensing based WNV prediction.



(a)



(b)

Figure 3.13: CS based recover and prediction of WNV for measurements (a) $M = 900$ (unable to recover) (b) $M = 1800$ (fully recovered).

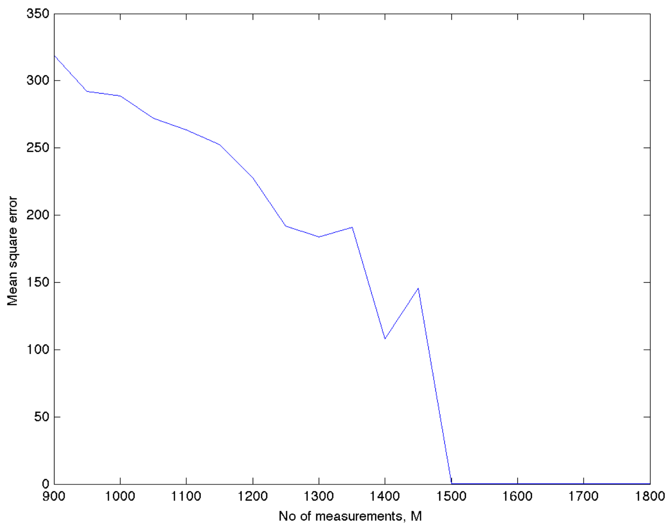


Figure 3.14: Effect of number of measurement M on the accuracy of prediction.

Chapter 4

Performance, Energy, and Scalability Analysis

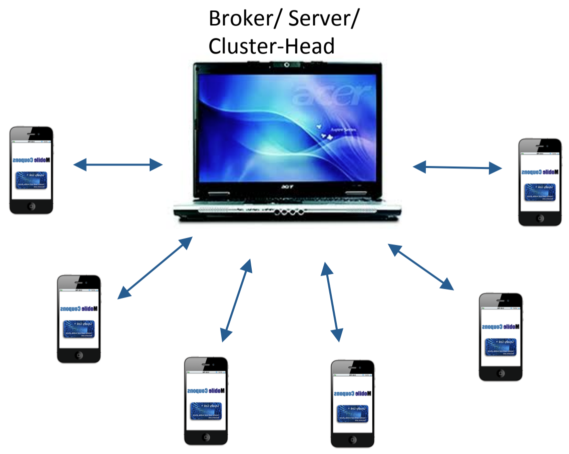
4.1 Introduction

The architecture of the Essence framework supporting collaborative sensing and multi-tiered compressive sensing using the network architectures as illustrated in Fig. 4.1 and Fig. 3.1 where the mobile nodes are connected to the cluster head (middleware broker) and configuration mechanism exists to support both. The end user in the Essence framework can be an independent user, a participating node, or combination of both depending on the application and usage. A participating node can receive, supply, or both receive and supply simultaneously with other nodes and provision to enable duplex mode of communication among the nodes receive on-demand data is considered. Processing happens at the appropriate level and node, with the aim to perform the processing at the right place and right time. In this analysis, we consider that the data is captured at the edge of the network by the leaf nodes, and the "centre" of processing are the cluster head (middleware broker) and the subsequent hierarchical levels. We interchangeable use the term middleware broker with cluster head due to its capability to perform additional processing (CS encoding, recovery and reconstruction). As our heterogenous hierarchical network architecture consists of different mobile phone as sensor nodes with appreciable amount of storage and processing capability along with infrastructure sensing nodes, some amount of processing at the leaf nodes are also possible. This provides the capability of compressing sensing of the several sensor time-series data at the leaf nodes, before the data is being sent to the cluster head for further processing.

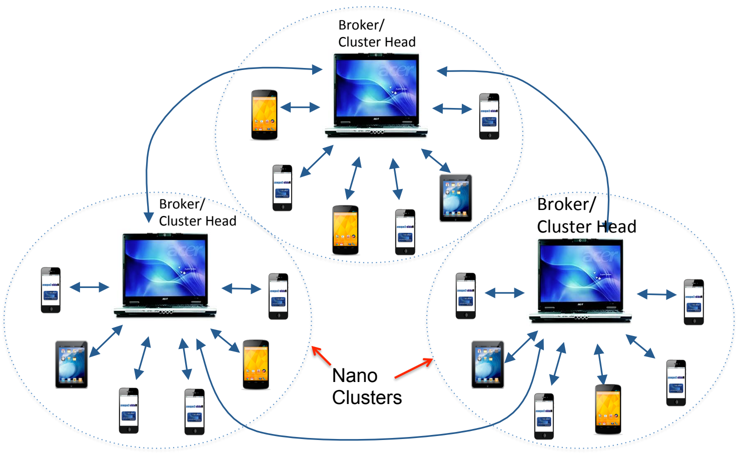
The Essence framework can be qualitatively assessed using several performance parameters to maintain a quality-of-service (QoS) for collaborative sensing. To evaluate the performance of the Essence sense-making framework, we consider the execution time and energy consumed as performance matrices to compare various sense-making methods. Note that sense-making methods involve sensor selection/encoding, data aggregation, and data processing and reconstruction. In order to analyze and perform quantitative comparisons of the sense-making scheme with the existing state of the art, we first define a few parameters as listed in Table 4.1.We first consider the collaboration sense-making scheme using client-server and mobile agent based models as discussed in [38] as our baseline in order to compare collaborative sense-making using compressive sensing for one level of hierarchy as shown in Fig. 4.1. The thesis then extend the same for multiple hierarchy and advance sense-making compressive sensing approaches as discussed in Section 4.3.

4.2 Collaborative Compressive Sense-making

In collaborative processing, the most commonly used computing model is client/server based, where individual sensors (the clients) send raw data or preprocessed data to a processing center (the server) and data integration is carried out at the center. The client/server-based computing generally requires many round trips over the network in order to complete one transaction. Each transaction consumes network bandwidth and



(a) NanoCluster (NC) in client/server mode



(b) Cluster of non-overlapping NC forming a Local Cluster (LC)

Figure 4.1: Network Architecture for the Hierarchical Sense-Making Framework (a) NanoCluster client-server architecture (b) Local Cluster hierarchical structured architecture.

communication energy. The network connection needs to be alive and healthy through the entire time of the transaction, otherwise the transaction has to restart if it can at all. In order to better illustrate how this computing model perform integration, Fig. 4.2 presents a temporal and spatial comparison of the life cycle of their migration units. In the client/server-based model Fig. 4.2, the clients (S_j and S_k) first read the data files into memory using t_{oh} overhead time, then transfer them to the server using transfer time t_{trans}. Here, we assume S_j and S_k are identical nodes and can start data transfer at the same time and thus these data files might arrive at the server simultaneously but can only be processed serially. After receiving all the incoming data files, the server can start processing, which would take t_{proc} amount of time.

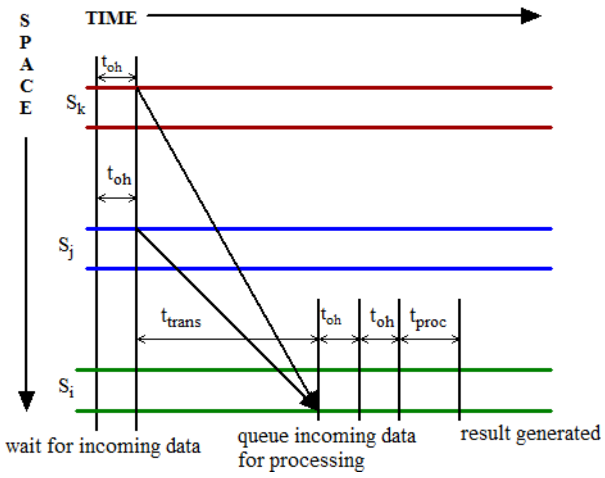


Figure 4.2: Life cycle of Data in Client/Server Architecture [38].

A simplified version of the above scenario is the case when two nodes want to collaborate between them, exchange information / data with each other on-demand. In such a case, the detailed sequence of steps involved in collaboration between two mobile sensing nodes through the cluster head/broker is shown in Fig. 4.3. If the certain category of information requested by a node from another node is not available and can be approximated (e.g, by the data from a near by node or using compressive sensing), the server (cluster head) process the request to generate an approximate result, and pass it to the requesting node. We call this mode of collaboration as collaborative compressive sensing.

We analyze the performance (the total execution time) and the total energy consumption for different aggregation method in the following subsections for the single level hierarchy and then extend the results to multi-level hierarchy in the subsequent sections.

4.2.1 Execution Time

The execution time is the time spent to finish processing a task. In the proposed system, it starts from the time a query is generated to the time the node returns with results. For simplicity of discussion, we assume three sensors nodes S_i , S_j and S_k and assign S_i as the processing center (i.e. S_i as server, S_j , S_k as clients in the case of client/server-based model). In the client/server-based model, it is from the time the clients send out data to the time the data processing is finished and results are generated at the server. The execution time consists of three components, t_{trans}, t_{oh}, and t_{proc}, as illustrated in Fig. 4.2, where t_{trans} is the time in transferring the sensor data from one node to the other, represent the overhead time in accessing the data /file system, and represents the processing time. A few factors that can affect the execution time include

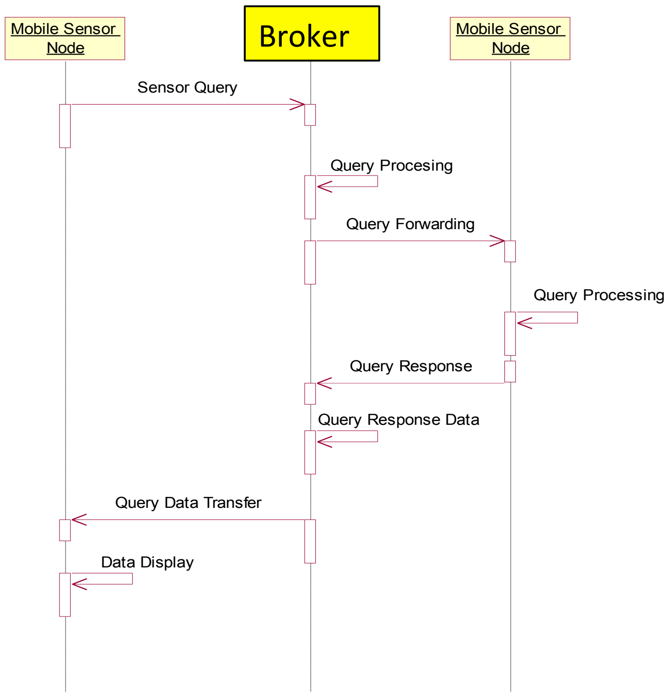


Figure 4.3: On-demand information exchange between two mobile nodes in cluster.

the network transfer rate v_n , the data processing v_d , the data file size s_f (the size of raw data each node collects), the overhead of file access o_f (the time used to read and write a data file), the number of sensor nodes N .

Thus, for client/server-based computing, the data transfer time is

$$t_{trans} = N \times s_f / v_n \tag{4.1}$$

and the overhead time is

$$t_{oh} = 2N \times o_f \tag{4.2}$$

(assuming the time used to read and write the data file is the same); and the data processing time is

$$t_{proc} = N \times s_f / v_d \tag{4.3}$$

Therefore, the total execution time using the client/server-based model is

$$t_{Baseline} = t_{trans} + t_{oh} + t_{proc} = \frac{N \times s_f}{v_n} + 2 \times N \times o_f + \frac{N \times s_f}{v_d}. \tag{4.4}$$

In the analytical model described above, the component that is most difficult to measure is the data transfer time, where retransmission and error control are not considered. Unfortunately, these factors occur quite often in sensor networks because of the use of wireless link, where simulation models are developed for more accurate estimation of the data transfer time t_{trans}.

In case of compressed sensing based measurement and aggregation, where the cluster head makes M random measurements out of the N nodes, we define the signal compression ratio γ as

$$\gamma = M/N. \tag{4.5}$$

If the reconstruction of the complete signal from M random measurements incurs $t_{reconst}$, the total execution time using the compressed sensing is

$$t_{CS} = \frac{M \times s_f}{v_n} + 2 \times M \times o_f + \frac{M \times s_f}{v_d} + t_{reconst}. \tag{4.6}$$

We consider the reconstruction time to be proportional to the time complexity of the reconstruction algorithm used. In CS data aggregation, each cluster head involves two key data processing tasks: CS random measurements encoding process and CS recovery process. For a general CS encoding process, the multiplication of an $M \times N$ matrix and an $N \times 1$ vector requires $(M \times N + N + M)$ working storage and MN multiplications and $(N-1)M$ additions for computation operations. Therefore, the data encoding cost is bounded by $O(NM)$. If CoSaMP [34] is adopted to recover CS random measurements, it has been proved that CoSaMP recovers N samples of data from M random measurements using $O(N)$ working storage and $O(NlogN)$ operations for each iteration [34]. Therefore, the reconstruction time can be modeled as $t_{reconst} \leq a_r * (N \times M + Nlog(N)) + b_r$, where a_r, b_r are constants. The time complexity of the model based compressive reconstruction and encoding algorithm are shown in Table 4.1. and is found to be $O(Nlog(N))$ [4]. In case of MCS, if a model based compressive sensing and reconstruction algorithm is used [18], $t_{reconst} \leq a_r * (2 * Nlog(N)) + b_r$. Thus the execution times for the conventional plain compressive sensing (PCS) and model based compressive sensing (MCS) are given by:

$$t_{CS} = \frac{M \times s_f}{v_n} + 2 \times M \times o_f + \frac{M \times s_f}{v_d} + a_r * (N \times M + Nlog(N)) + b_r, \tag{4.7}$$

$$t_{MCS} = \frac{M \times s_f}{v_n} + 2 \times M \times o_f + \frac{M \times s_f}{v_d} + a_r * (2 * Nlog(N)) + b_r, \tag{4.8}$$

4.2.2 Energy Consumption

Sensor nodes are normally composed of four basic units: a sensing unit, a processing unit, a communication unit, and a power unit. Among these units, communication and sensing consume most of the energy.

Table 4.1: Comparison of recovery times for compressive sensing algorithms.

Reference	Measurement Bound	Recovery Time	Matrix multiplication time
MCS1[5]	$O(K)$	$O(NK)$	$O(NK)$
MCS2[4]	$O(K\frac{log(N)}{loglog(N)})$	$O(NK)$	$O(Nlog(N))$
MCS [18]	$O(K)$	$O(Nlog(N))$	$O(Nlog(N))$

Similar to the formulation of the execution time, the energy consumption for the two computing models depends on three components: energy consumed in data transfer E_{trans} , overhead processing E_{oh}, and data processing E_{proc}.

According to [14], the energy consumed in data transfer can be modeled using a linear equation ,

$$E_{trans} = c \times size + d \tag{4.9}$$

where d is a fixed component associated with device state changes and channel acquisition overhead, *size* is the size of data being transferred, and c is a coefficient indicating the amount of energy consumed by transferring 1 Byte of data. The values for c and d are different between data transmission and data receiving. Therefore, we separate into two parts,

$$E_{tx} = c_{tx} \times size + d_{tx} \tag{4.10}$$

for transmission and for receiving:

$$E_{rx} = c_{rx} \times size + d_{rx} \tag{4.11}$$

We use a similar linear equation to model the energy consumption in overhead processing,

$$E_{oh} = c_{proc} \times size \tag{4.12}$$

where c_{proc} is the coefficient indicating the amount of energy consumed in processing per byte of data. Since we only have knowledge of time spent for overhead processing and the amount of data that can be processed in 1 sec, we can derive the size of data, the so-called equivalent data size s_e , that takes o_f amount of overhead time to process, that is, $s_e o_f$ for the client/server-based model. Since s_e varies when the number of clients changes we choose an average value as listed in Table 4.2 based on [38, 14].

Since no additional data processing takes place for the client/server model, we choose to neglect E_{proc} . Similar to (4), the energy consumption model we use for client/server-based model is given by [38]:

$$E_{Baseline} = N \times \left[(c_{tx} \times s_f + d_{tx}) + (c_{rx} \times s_f + d_{rx}) + 2\,(c_{proc} \times s_e \times o_f)\right]. \tag{4.13}$$

On the other hand, as addition processing is performed in CS based data aggregation, E_{proc} need to be modeled to consider the effect of CS encoding and reconstruction algorithms. In order to model the energy consumption of the reconstruction algorithm, we make few assumption to simplify the derivation. If we consider $c_{reconst}$ is the average power consumed during the reconstruction process, the energy consumed $E_{reconst}$ can be modeled as $E_{reconst} = c_{reconst} \times t_{reconst}$.

Thus the total energy consumed when M random measurements in compressed sensing mode are made is given by :

$$E_{CS} = M \times \left[(c_{tx} \times s_f + d_{tx}) + (c_{rx} \times s_f + d_{rx}) + 2\,(c_{proc} \times s_e \times o_f) + E^{CS}_{reconst} \right], \quad (4.14)$$

$$E_{MCS} = M \times \left[(c_{tx} \times s_f + d_{tx}) + (c_{rx} \times s_f + d_{rx}) + 2\,(c_{proc} \times s_e \times o_f) + E^{CS}_{reconst} \right], \quad (4.15)$$

where the reconstruction energy for CoSaMP and the MCS method is modeled as :

$$E^{CS}_{reconst} = c_{reconst} \times \{a_r * (N \times M + Nlog(N)) + b_r\} \quad (4.16)$$

$$E^{MCS}_{reconst} = c_{reconst} \times \{2 * a_r * Nlog(N) + b_r\}. \quad (4.17)$$

.

4.2.2.1 Effect of network size on execution time and energy

As the execution time and energy consumption of different collaborative sensing schemes depends on the network size , we perform simulation study for different network size for two levels of hierarchy. Fig. 4.4a shows the trend on the execution time as the network size grows on different collaborative and compressive sensing methods. We observed that as the network size increases, the execution time for client/server and mobile agent based approach[38] increases linearly. The mobile agent based collaborative sensing approach as discussed [38] shows reduced execution time compared to client/server based model. With compressive sensing approach produces relatively lesser execution time compared to client/server based model. The effect on execution time is much higher as compression ratio γ reduces. Among the two CS methods, the execution time with MCS method [18] is relatively less than that of the CoSaMP [34]. While the MCS method [18] scales almost linearly, the execution time for CoSaMP [34] always exceeds almost linear scaling MCS. As the compression ratio γ approaches 1, the advances of the compressive sensing methods are reduced compared to client/server method as expected.

A similar effect is seen on the energy consumption as shown in Fig. 4.4bfor different methods as the network size increases. Since the energy consumed in the CS reconstruction stage of modeled based compressive sensing (MCS) method is relatively less than

Table 4.2: Simulation Parameters

Sl No	Variable Name	Description	Remarks
	S_i	Mobile Sensor node S_i	
	S_j	Mobile Sensor node S_j	
	S_k	Mobile Sensor node S_j	
	t_{trans}	Transfer time form one node to other	
	t_{oh}	Overhead time in accessing files/data	
	t_{proc}	Processing time	
	$t_{reconst}$	processing time for reconstruction	
	$t_{Baseline}$	Total execution time in client/server model	
	t_{CS}	Total execution time in CS model	
	v_n	Network transfer rate	2 Mbps
	v_d	Data processing rate	100 Mbps
	s_f	Size of the data file each node collects	10 KB
	o_f	Overhead time of file access (read/write access)	0.0125 sec
	o_a	Overhead time of broker agent	
	N	Total no of sensing nodes	2- 1000
	M	No of measurement in a Local Cloud	
	a_r, b_r	Constant for CS reconstruction time	
	E_{trans}	Energy consumed in data transfer	
	E_{oh}	Energy consumed in overhead processing	
	E_{proc}	Energy consumed in data processing	
	$E_{reconst}$	Energy consumed in CS reconstruction	
	$E_{Baseline}$	Total energy consumed in client/server model	
	E_{CS}	Total energy consumed in CS model	
	c	Coefficient, amount of energy consumed per byte	
	c_{tx}	Energy consumed per byte in transmission	1.9
	c_{rx}	Energy consumed per byte in receiving	1.425
	d	Fixed coefficient of energy	
	d_{tx}	Coefficient of energy in transmission	454
	d_{rx}	Coefficient of energy in receiving	356
	c_{proc}	Energy consumed in processing per byte of data	0.7125
	$c_{reconst}$	power consumed in reconstruction per unit time	
	λ	fractional constant	0–1
	$size$	Size of data being transferred	
	s_e	Equivalent data size that take o_f sec time	470KB
	A_i	Area covered by the cluster in level i	m^2
	T	No of hierarchical levels in data aggregation	
	n	No of sub-regions, degree of the hierarchical tree	
	s	Speed of each mobile sensor node (m/s)	m/s

that of the CoSaMP method, the overall energy consumption for the MCS method is relatively less than that of the CoSaMP [34] as the network size increases. Besides, as the compressing ratio γ reduces, the energy consumption of both MCS and CoSaMP reduces correspondingy. As compressing ratio γ approaches 1, the energy consumption of the CS methods approaches that of the client/server based method. Thus, data compression provides us with a specific knob (i.e., amount of compression) that allows us to save communication energy between all tiers at the cost of some extra computation (required for the compression process)

4.2.2.2 Effect of network size and data file size collected by each nodes on execution time and energy

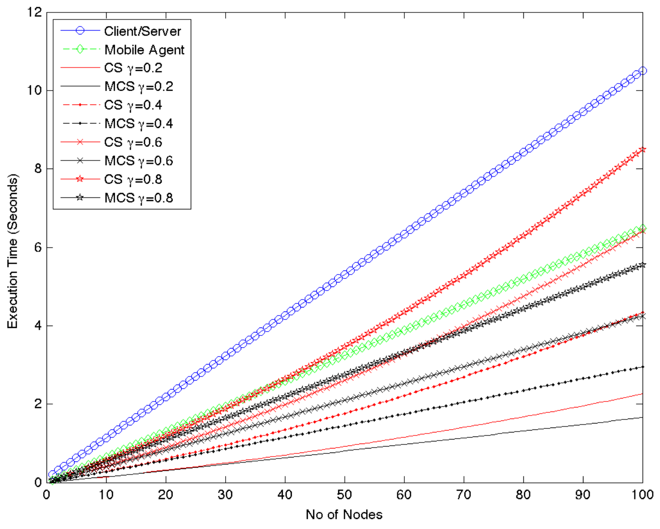
In order to study the effect of the size of the data file each node collects on the execution time and the energy efficiency of the complete network, we summate the complete network for given compression ratio γ and varying file size s_f . Fig. 4.5a and Fig. 4.5b shows the effect of the data file size on the execution time and energy consumed as the network size increase. It can be observed from Fig. 4.5a that as the file size s_f increase, the execution time of sense-making increases for both CoSaMP and MCS method. The file size s_f has similar scaling effect as the compression ratio γ.

The effect on the energy consumed shows a similar in trends. As observed in Fig. 4.5b, the relative energy saving between the CoSaMP and MCS method are marginal, however if the data collected by each node can be reduced, the energy saving can be substantially scaled for the same compression ratio γ.

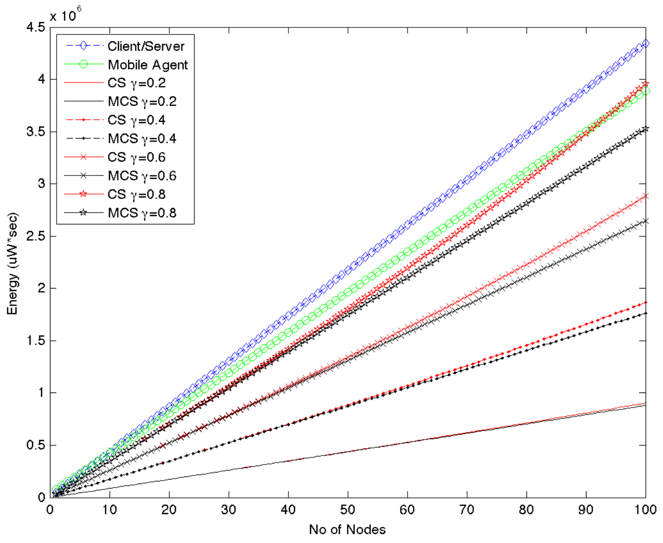
4.3 Scalable Multi-Level Hierarchical Aggregation Based Sense-making

The hierarchical aggregation of the data can be performed at a particular level in several ways as shown in Fig. 4.6.

In this section we present the CS based multi-level hierarchical data aggregation scheme that is based on a hierarchy of clusters and compressive sensing as discussed in [50, 27, 49]. The aggregation method presented in this thesis is similar to the HDACS approach in [50]with the difference of using new model based compressive sensing technique and inclusion of several overheads in the aggregation process that was ignored in [50]. We first outline the clustering framework, where clusters represent non-overlapping

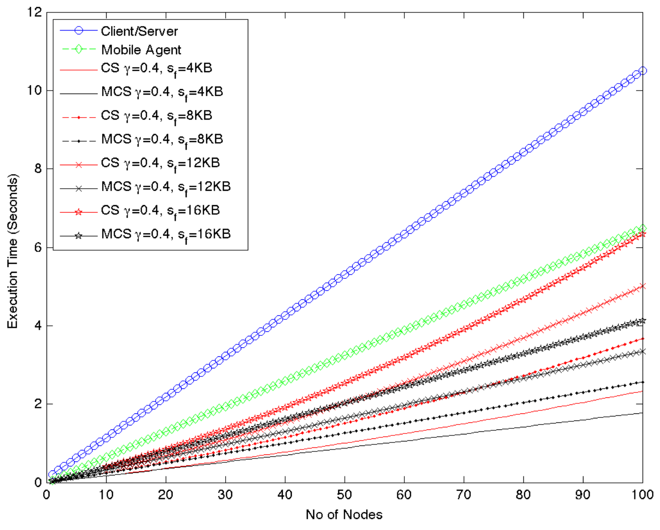


(a)

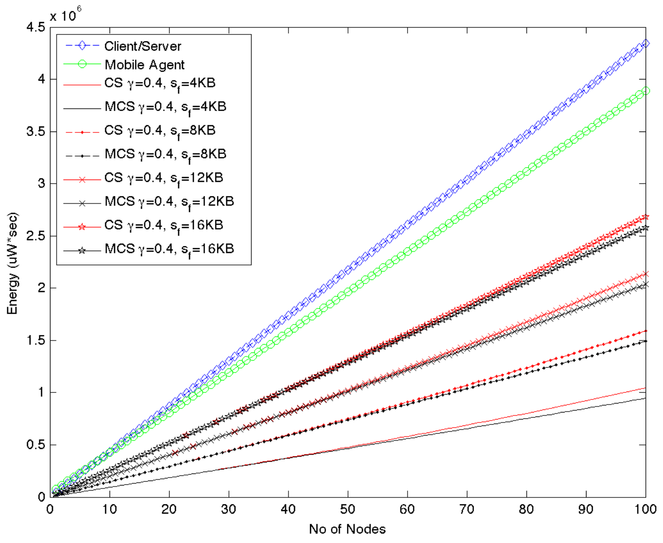


(b)

Figure 4.4: Effect of network size (number of mobile sensing node N) on (a) the execution time (b) Energy for different methods and compression ratios.



(a)



(b)

Figure 4.5: Effect of file size s_f on (a) execution time and (b) energy.

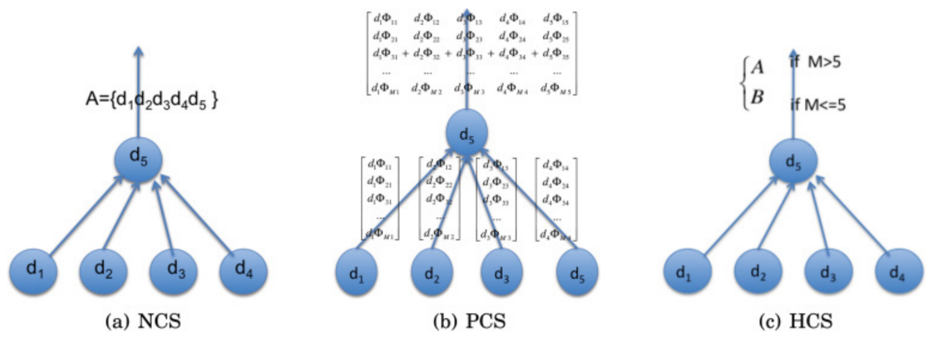


Figure 4.6: Examples of Data Aggregation (a) non-compressive sensing (NCS) , same as client-server model (b) plain-CS (PCS) (c) Hybrid-CS (HCS).

geographical regions. At level one, all the clusters are defined by identical regions each with area A. Let's assume that the hierarchy consists of T levels, number of clusters is n^{T-i}, and for cluster (l) at level i represents an area of size $A_i^{(l)}$, which further combines n subregions from level $i-1$, such that $A_i^{(l)} = n * A_{i-1}^{(l)} = n^{i-1} * A$. Following this procedure, we get the relation between the entire monitored area A_{total} and the initial cluster area A: that is, $A_{total} = n^{T-1}A$. Let $|C_i|$ represent the number of clusters at level i and we get the relation between the number of clusters and the whole area: $A_i^{(l)}|C_i| = n^{i-1}A|C_i| = A_{total}$. So, the number of clusters at level i is $|C_i| = \frac{A_{total}}{A}n^{1-i} = n^{T-1}n^{1-i} = n^{T-i}$. For each cluster l at level i , we define the following local parameters :

- $c_i^{(l)}$: cluster head node
- $v_i^{(l)}$: the list of logical children cluster head nodes
- $A_i^{(l)}$: cluster area
- $N_i^{(l)}$: the cluster size including all the nodes in the cluster
- $M_i^{(l)}$: the number of CS random measurements or the amount of data transmitted
- $d_i^{(j)}$: the distances between cluster head $c_i^{(l)}$ and its children node j, where $j \in v_i^{(l)}$
- $\gamma_i^{(l)}$: the compression ratio
- $E_i^{(l)}$: total transmission energy for cluster l

The global parameters at level i are defined as follows:

- C_i: Collection of cluster heads, $C_i = \{c_i^1, c_i^2, ..., c_i^{|C_i|}\}$
- $|C_i|$:the number of clusters, $|C_i| = n^{T-i}$

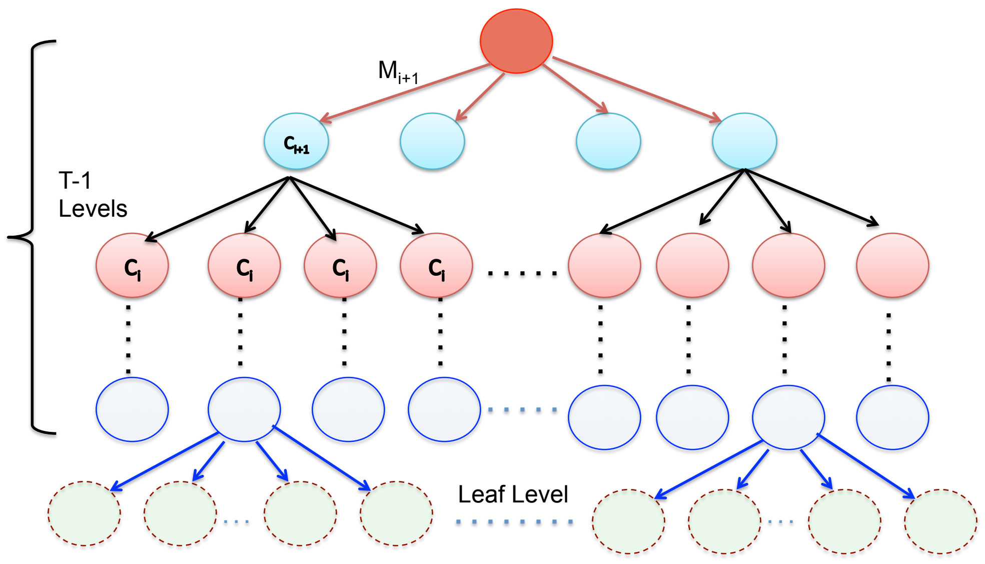


Figure 4.7: Multi-level Hierarchical Aggregation Architecture

- M_i: total amount of of data units for transmission $M_i = \sum_{l=1}^{|C_i|} M_i^{(l)}$
- E_i: the transmission energy cost for all the clusters, where $E_i = \sum_{l=1}^{|C_i|} E_i^{(l)}$

The logical relationship of clusters at different levels is shown in Fig. 4.7. The hierarchy consists of n nodes at level i for $i \geq 2$, or, in another words, the degree of the hierarchical tree is n. We also assume the number of leaf nodes $N_1^{(l)} \geq n$ for any cluster l at level one. In order to meet these clustering requirements, we randomly deployed the sensors in a 2D network topology such that at least there are n nodes in each cluster at level 1. For n^{T-1} clusters, $n * n^{T-1} = n^T$ sensors are associated. The remaining $N - n^T$ sensors are placed randomly following the uniform distribution within the entire region. The main advantage of such a network deployment is that it is easily realizable in practical spatial distribution of sensors. It also addresses issues when the condition of $N = n^T$ is not satisfied. Besides, since the number of cluster heads are at most n^{T-1}, the leaf nodes are $N - n^{T-1} \geq n^T - n^{T-1} = (n-1)n^{T-1}$. If $n > 2$, $N - n^{T-1} > n^{T-1}$, which implies that only a small number of nodes will be involved with multiple level data processing and aggregation and the only job of leaf nodes, indicated by dotted circles in Fig. 3.1, is to send their data directly to their cluster heads. In other words, the number of leaf nodes is much more than the number of cluster heads. The rules also ensure that the number of nodes in one cluster at level i satisfies $N_i^{(l)} \in \{n^i, n^i + 1, ..., n^{i+1} - 1\}$.

4.3.1 Hierarchical Data Aggregation Based Multi-Level Compressive Sensing (HMLCS)

Multi-level hierarchical aggregation enables scalable sense-making in large networks. In order to study the impact of the multilevel hierarchy, we perform CS data aggregation with several salient steps. In first step, at level 1 of the cluster hierarchy, leaf nodes send their sensed data to their respective cluster heads directly; The cluster heads receive them and take $M_1 = K * logN_1^{(l)}$ CS random measurements based on their local cluster size $N_1^{(l)}$ where the superscript (l) is used to indicate the cluster l in level 1. In the second step, at level $i(i \geq 2)$, cluster heads receive CS random measurements and they perform CS recovery algorithm to extract original data. After accumulating all the data from their children nodes, the cluster heads take $M_i^{(l)} = KlogN_i^{(l)}$ CS random measurements from all the recovered data and send them to the parent cluster heads at level $i + 1$. In the final step, second step is repeated until the cluster head at the top level, T gathers the data from all the sensors.

Compared to the methods in [28, 49] where a global threshold $M = KlogN$ is set up for the entire network, in multi-level hierarchical scheme uses multiple thresholds $M_i^{(l)} = KlogN_i^{(l)}$ which are determined by the cluster sizes at different levels with upper bound $O(KlogN)$ at the highest level T. When the level i is low, the corresponding M_i is also a relatively small number. This property hierarchy helps to reduce the data size for communication and saves the energy spent on transmission.

4.3.1.1 ***Total Amount of Data Transmitted***

Assume $n^{i+1} \geq N_i^{(l)} \geq n^i$, (if $N_1^{(l)} \geq n^2$, the depth of the logical tree is $T + 1$, which contradicts with the previous assumption that $n^{T+1} > N = n^T + N' \geq n^T$. The same requirement applies to the cluster size at other levels and the amount of data transmitted $M_i^{(l)} \in [iKlog(n), (i + 1)Klog(n)]$. Therefore, the total number of measurements M_{total} for transmission is:

$$M_{total} = \sum_{l=1}^{|C_1|}(N_1^{(l)} - 1) + \sum_{i=2}^{T}\sum_{l=1}^{|C_i|}(n - 1)M_{i-1}^{(l)} \tag{4.18}$$

Let $D_1 = \sum_{i=1}^{T-1} \frac{i}{n^i}$ and and $D_2 = \sum_{i=1}^{T-1} \frac{1}{n^i}$ we get the closed form of D_1:

$$D_1 = \frac{1/n(1{-}1/n^{T-1})}{(1{-}1/n)^2} - \frac{T{-}1}{n^T(1 - 1/n)} \tag{4.19}$$

and

$$D_2 = \frac{1/n{*}(1{-}1/n^{T-1})}{1{-}1/n} \tag{4.20}$$

Therefore, the lower bound of M_{total} is:

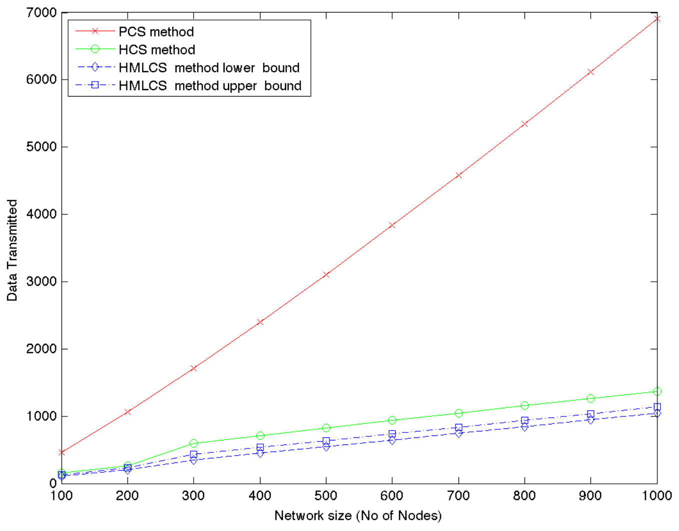


Figure 4.8: Comparison of trends in total amount of data transmission for different network size.

$$\Omega(M_{total}) = N{-}n^{T-1} + K(n{-}1)n^{T-1}lognD_1 \qquad (4.21)$$

and upper bound is

$$O(M_{total}) = N{-}n^{T-1} + K(n{-}1)n^{T-1}logn(D_1 + D_2). \qquad (4.22)$$

In order to perform a fair comparison with the existing CS based schemes, the PCS, and HCS methods are formulated in the hierarchical framework as well. The PCS method implementing data aggregation task in our hierarchical framework would transmit $M_{PCS} = NKlogN$ data [27]. Similarly, for the HCS method [28, 49] the number of transmissions are

$$M_{HCS} = N - n^{T-1} + K(n-1)n^{T-1}logND_2 + O(NlogN) \qquad (4.23)$$

Fig. 4.8 compares these schemes assuming different network sizes assuming logical cluster size at each level to be 4. These results show that compared to HMLCS scheme the total amount of data transmitted in the PCS method increases at a much faster rate with the increase in the network size. In terms of the amount of data transmitted, the relative advantage of the HMLCS method over HCS is less significant because of the fact that a bulk of sensors serve as leaf nodes at the first level, and transmit their raw data directly to the cluster heads at level 1.

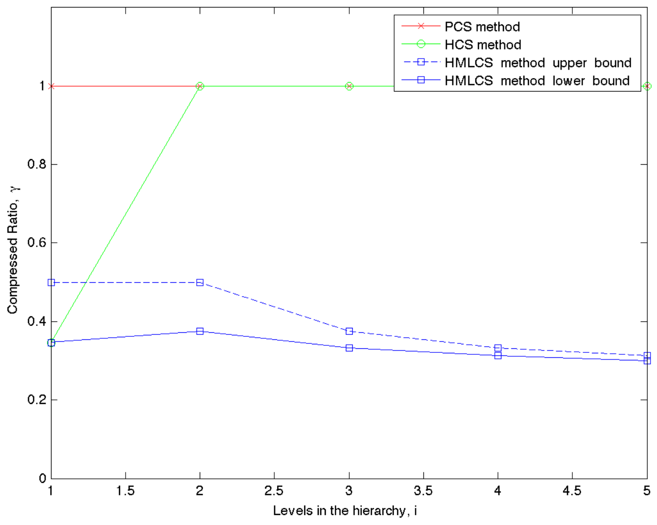


Figure 4.9: Comparison of compression ration at different levels of hierarchy.

4.3.1.2 *Data Compression Ratio*

The data compression ratio $\gamma_i^{(l)}$ is defined as the ratio of the amount of data transmitted to the amount of data available and the expression is given by [50]:

$$\gamma_i^{(l)} = \begin{cases} \frac{M_1^{(l)}}{N_1^{(l)}} = \frac{K log N_1^{(l)}}{N_1^{(l)}} & if \quad i = 1 \\ \frac{M_i^{(l)}}{\sum_{j(l)\in v_i^{(l)}} M_{i-1}^{(j(l))}} & if \quad i \geq 2 \end{cases} , \tag{4.24}$$

where $\gamma_i^{(l)} \in [\frac{1}{n}(1+\frac{1}{i-1}), \frac{1}{n}(1+\frac{1}{i})]$ if $i \geq 2$. For the PCS method, $\gamma_{PCS} = 1$, and $\gamma_{i,HCS} = \frac{M_i^{(l)}}{N_i^{(l)}}$ when $N_i^{(l)} > K log N$ satisfied for the first time; otherwise, $\gamma_{i,HCS} = 1$.

The tendency in data compression ratio at each level for different methods is shown in Fig. 4.9. It's observed that the PCS method provides no compression at all; the HCS method yields compression only for the level where the size of data reaches to the global threshold; however, the HMLCS method achieves a good compression ratio for each level which is well below 0.5.

4.3.1.3 *Energy Consumption for Transmission*

The communication among sensors is the most power-consuming task and its energy consumption analysis is a major concern. Transmission energy cost $E_i^{(l)}$ is normally a function of transmission distance $d_i^{(l)}$ and data size $M_i^{(l)}$. Therefore, $E_i^{(l)}$is modeled as $E_i^{(l)} = c_s + \sum_{j\in v_i^{(l)}} c(d_i^{(j)})^\alpha M_i^{(l)}$ [7, 15], where c_s is a distance-independent term, and c is a constant transmission cost per unit data per unit distance. The settings of c and c_s depend on the hardware and algorithms for various application tasks.

Assume $d_i^{(l)} = \sum_{t=1,(x_t,y_t)\in c_i}^{n} [(x_t - x_{c_i})^2 + (y_t - y_{c_i})^2]^{1/2}$, where (x_{c_i}, y_{c_i}) and (x_t, y_t) are the spatial coordinates of $c_i^{(l)}$ and its children nodes respectively. In a large dense uniform randomly distributed sensor network, if $i > 1$ $d_i^{(l)} = 4\frac{(n-1)}{s_i}\int_0^{b_i}\int_0^{b_i}(x^2 + y^2)^{1/2}dxdy = \frac{1}{12}\pi n^{\frac{i-1}{2}} A^{1/2}(n-1)$, where $b_i = \frac{1}{2}A_i^{1/2}$. And for $i = 1$, $d_1^{(l)} = \frac{1}{12}\pi A^{1/2}(N_1^{(l)} - 1)$. For level $i > 1$, we get $E_i^{(l)} = c_s + \frac{cM_i^{(l)}}{1+\alpha}2^{-\alpha}(n-1)\pi n^{(i-1)(\alpha-1)/2}A^{(\alpha-1)/2}$. For notation simplicity, we denote $B_1 = \frac{c}{1+\alpha}2^{-\alpha}\pi A^{(\alpha-1)/2}$. The transmission energy in level $i = 1$ within cluster l is: $E_1^{(l)} = c_s + B_1(N_1^{(l)} - 1)$. Therefore, the total transmission energy cost is:

$$E_{total} = n^{T-1}c_s + (N - n^{T-1})B_1 + n^{T-1}c_s D_2 + B_1(n-1)\sum_{i=2}^{T}\sum_{l=1}^{|C_i|} K log N_{i-1}^{(l)} n^{(i-1)(\alpha-1)/2} \tag{4.25}$$

Let $D_3 = \sum_{i=1}^{T-1} i n^{i(\alpha-3)/2}, D_4 = \sum_{i=1}^{T-1} n^{i(\alpha-3)/2}$ and $D_5 = \sum_{i=1}^{T-1}(i+1)n^{i(\alpha-3)/2}$. If $\alpha \neq 3$, we get the closed form expression of D_3 as

$$D_3 = \frac{n^{(\alpha-3)/2}(1-n^{(\alpha-3)(T-1)/2})}{(1-n^{(\alpha-3)/2})^2} - \frac{(T-1)n^{(\alpha-3)T/2}}{1-n^{(\alpha-3)/2}} \tag{4.26}$$

and the closed-form expression of D_4:

$$D_4 = \frac{n^{(\alpha-3)/2}(1 - n^{(\alpha-3)(T-1)/2})}{1-n^{(\alpha-3)/2}}. \tag{4.27}$$

Let $D_5 = D_3 + D_4$. If $\alpha = 3$, $D_3 = \frac{T*(T-1)}{2}$, $D_4 = T - 1$ and $D_5 = \frac{(T+2)*(T-1)}{2}$. Therefore, the lower bound of E_{total} is

$$\Omega(E_{total}) = n^{T-1}c_s(1 + D_2) + B_1(N - n^{T-1}) + B_1 K(n-1)n^{T-1}D_3 log n \tag{4.28}$$

and upper bound of E_{total} is

$$O(E_{total}) = n^{T-1}c_s(1 + D_2) + B_1(N - n^{T-1}) + B_1 K(n-1)n^{T-1}D_5 log n \tag{4.29}$$

Following the same derivation, we get transmission cost in the PCS aggregation

$$E_{PCS} = n^{T-1}c_s(1 + D_2) + B_1(N-n^{T-1})log N + B_1 K(n-1)n^{T-1}D_4 log N \tag{4.30}$$

Similarly, transmission cost in the HCS aggregation is:

$$E_{HCS} = n^{T-1}c_s(1 + D_2) + B_1(N-n^{T-1}) + B_1 K(n-1)n^{T-1}log N D_4 + O(N log N) \tag{4.31}$$

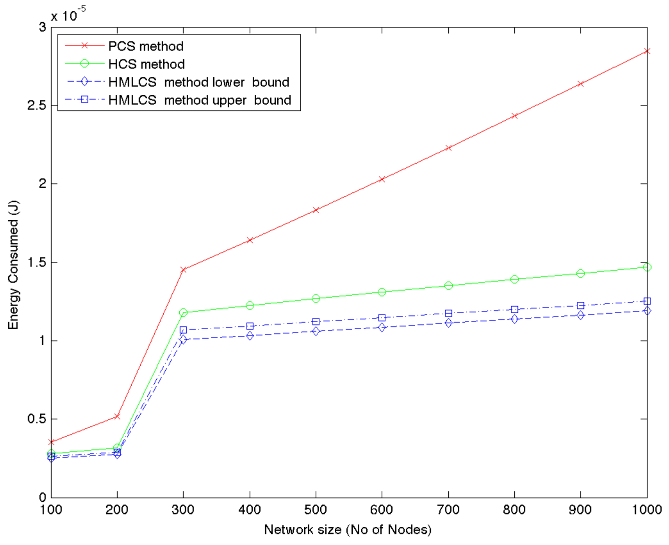


Figure 4.10: Comparison of trends in transmission energy consumption for different network size.

Fig. 4.10 shows the trends in energy consumption for transmission with diverse network scales under the assumption $\alpha = 2$ [7, 15] Similar figures are obtained when α is set as 3, 4, 5, 6, where the constant parameters in the energy model are assigned as : $c = 10^{-10}J$, $c_s = 10^{-7}Jm - \alpha/bit$ [7, 15], and $K = 1$. The result show the energy for transmission required by the PCS method is far more than the HCS method and the HMLCS method. Compared to other two methods, the HMLCS method consumes the least transmission energy and achieves the best efficiency for multilevel hierarchy.

4.4 Summary

In this chapter, we derived the performance and energy consumption for several data aggregation methods and compared them for various network size and compressive sensing methods. The closed form derivation of the performance and energy consumption of several schemes were derived by taking into account the reconstruction algorithm's time complexity and energy overhead in the formulations. The effect of network scaling, different levels of hierarchy, and compression ratio on the performance and energy for different methods are studied. The impact of file size that each sensor node collects has also been studied and compared for different scheme. Unlike the existing sate-of-the-arts, this thesis introduced a model based compressive sensing (MCS) method in the reconstruction and recovery stage of the processing and derived closed for performance and energy consumption models. Compared to the CoSaMP based recovery methods, the MCS methods provides improved performance and energy efficient as the network size scales.

Chapter 5

Neural Network Based Prediction for Sense-Making in IoT Devices

Table 5.1: Notations for symbols

Notations	Scalar		Vector		Matrix	
	e.g.	Remark	e.g.	Remark	e.g.	Remark
Variable	x	italics	$\boldsymbol{x}$, X	bold/caps italics	$\boldsymbol{X}$	caps bold italics
Constant	w	Standard	**w**, W	bold/caps standard	**W**	caps bold standard
Estimation of Variable	$\hat{y}$	hat italics	$\hat{\boldsymbol{y}}$, $\hat{Y}$	hat bold/caps italics	$\hat{\boldsymbol{Y}}$	hat caps bold italics

5.1 Introduction

This chapter presents a prediction model for sense-making based on Artificial Neural Network specifically targeting IoT devices. Implementing large-scale neural networks with high computational complexity on low-cost IoT devices may inevitably be constrained by limited computation resource, make the devices hard to respond in real-time, and consume considerable computational energy and power making sense-making in different scenarios difficult. As a result, it is crucial to understand the predictive models not only from the perspective of performance accuracy but also considering their resource usage and computation energy requirement as deemed necessary in most IoT applications. In this chapter we study these predictive model's basic concept, specification, architecture, classification and training methodology in order to improve the performance in predicting the presence of the WNV virus as discussed earlier.

5.2 Neural Network Based Predictions for Sense-Making

5.2.1 Basic Notations, Terms, and Definitions

- **Neuron** $\aleph$: An *artificial* neuron $\aleph$ is a mathematical function $\aleph : X \to \hat{y}$ or $\hat{y} = \aleph(X)$, conceived as a model of biological neurons. Artificial neurons are the constitutive units in an artificial neural network. The artificial neuron receives one or more inputs (representing dendrites) and sums them to produce an output (representing a neuron's axon). Usually the sums of each node are weighted, and the sum is passed through a non-linear function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions. They are also often monotonically increasing, continuous, differentiable and bounded.

- **Artificial Neural Network** NN: A neural network is a connection of several artificial neurons. An NN is is a mathematical function $NN : X \to \hat{Y}$ or $\hat{Y} = NN(X)$.

- **Network Architecture** NA: A data flow representation showing the connection, computations, and flow of information from inputs to output through the layers.

- **ANN Inputs** X : The artificial neuron $\aleph$ receives one or more inputs (representing dendrites). The inputs can be scalar x for single-input neuron, a vector X for a multi-input neuron, or a matrix $\boldsymbol{X}$ for a multi-dimensional-input neuron.
- **ANN Outputs** $\hat{Y}$: The artificial neuron $\aleph$ produce an output (representing a neuron's axon). The output can be scalar $\hat{y}$ or a vector $\hat{Y}$ (which approximates the target observations Y for supervised learning).
- **Target Output** Y : The ideal output observed for a given inputs X, which is approximated by the NN output $\hat{Y}$. The target $Y$is available as training data.
- **Layers** L: A design parameter for NN that indicates several layers of neurons (in a multi-layer network).
- **Training Observation Sample** s: An observation s is a pair (x, y) such that $x \in X$ and $y \in Y$ using which the function $NN : X \rightarrow \hat{Y}$ is an approximated.
- N_t: Total Number of Training Observations
- **Training Set** $\boldsymbol{TS}$: The training set is a set of observations (usually a matrix) that is the combination of the inputs $\boldsymbol{X}$ and the target outputs $\boldsymbol{Y}$, such that $\boldsymbol{TS} = [\boldsymbol{X}\boldsymbol{Y}]$ is used to train the neural network NN.
- **Validation/Test Set** $\boldsymbol{VS}$: The validation/test set is a matrix that is the combination of the inputs X_v and the target outputs Y_v, such that $\boldsymbol{VS} = [X_v\, Y_v]$ is used to validate/test the neural network NN.
- **Validation Observation Sample** v: An observation v is a pair (x_v, y_v) such that $x_v \in X_v;\ x_v \notin X$ and $y_v \in Y_v;\ y_v \notin Y$ using which the function $NN : X \rightarrow \hat{Y}$ is validated / tested.
- N_v: Total Number of Validation Observations
- **Weights** W: The gains a neuron use to approximate the
- **Learning**: An optimization process that uses a learning algorithm to optimize a cost function C. Mathematically, Given a specific task to solve, and a class of functions F, learning means using a set of observations to find $f^* \in F$which solves the task in some optimal sense. This entails defining a cost function $C : F \rightarrow \mathbb{R}$ such that, for the optimal solution $f^*, C(f^*) \leq C(f) \forall f \in F$ – i.e., no solution has a cost less than the cost of the optimal solution.
- **Learning Algorithm** LA: Same as Learning, when formally represented as a algorithm

- **Training** Tr : Training a neural network model essentially means selecting one model from the set of allowed models that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation. The process of finding the weights W of the neural network NN given the training set TS.
- **Activation function** f: A mathematical function that mimics the activation behavior of a biological neuron. Mathematically, $f : u \to y$ or $y = f(u)$ where u is the activation input. The activation function for a i^{th} neuron is $y_i = f_i(u_i)$.
- **Transfer Function** : same as activation function above. See Fig. 5.3.
- **Feature(s)** : An individual measurable property of a phenomenon being observed. Choosing informative, discriminating and independent features is a crucial step for effective algorithms in pattern recognition, classification and regression. Features are usually numeric, but structural features such as strings and graphs are used in syntactic pattern recognition. The concept of "feature" is related to that of explanatory variable used in statistical techniques such as linear regression. A set of numeric features can be conveniently described by a feature vector.
- **Performance Measure**: Its a mathematical representation to to quantify the performance (in terms of accuracy or error) of NN often measured in mean square error (MSE), mean absolute error (MAE), mean absolute deviation (MAD) etc.
- **Gradient**: First derivative of the cost function w.r.t weights.
- **Epoch** E: An epoch is one complete presentation of the data set to be learned to a learning machine. An epoch is a single step in training a neural network; in other words when a neural network is trained on every training samples only in one pass we say that one epoch is finished. So training process may consist more than one epochs. Training a NN on each item of the set once is an epoch. So, if you want to teach your network to recognize the letters of the alphabet, 100 epochs would mean you have 2600 individual training trials.
- **Training Time** T_{tr}: The time need to train a NN.
- **Computation Complexity** O: The measure of time complexity and resource (space) complexity of a neural network as a function of its specifications (number of inputs, layers etc.).

5.2.2 Neural Network Architecture

A neuron with a single N-element input vector X of dimension $(N \times 1)$ with elements $(x_1, x_2, x_3, ..., x_N)$ is shown below. W is a weight matrix of dimension $(1 \times N)$ with elements

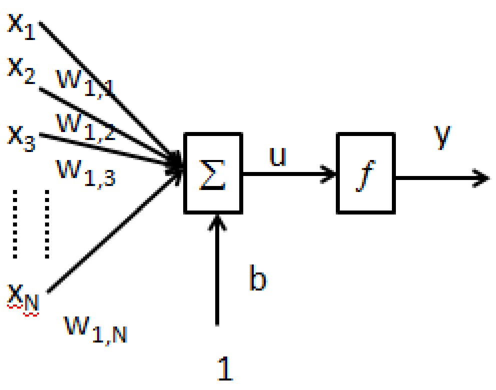


Figure 5.1: Model of Artificial Neuron.

$(w_{1,1}, w_{1,2}, \ldots w_{1,N})$. The weighted value is the dot product of the matrix $W_{(1\times N)}$ and the input vector $X_{(N\times 1)}$.The neuron has a bias b, a constant input of 1, which is summed with the weighted inputs to form the net input u. The net input$u$ is the argument of the transfer function f.

where, $u = W_{(1\times N)}.X_{(N\times 1)}+b$,where $W = [w_{1,1}w_{1,2}w_{1,3}...w_{1,N}]_{1\times N}$ and $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}_{N\times 1}$.

Thus the input to the transfer function is given by :

$$u = [w_{1,1}w_{1,2}w_{1,3}...w_{1,N}]_{1\times N}. \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}_{N\times 1} + b \tag{5.1}$$

Output is expressed as $y = f(u)= f(W.X + b)$. The transfer function used in the neural network is shown in Fig. 5.3.

In the above case output y is a scalar. Note that if there were more than one neuron, the network output y would be a vector.

5.2.2.1 Single-layer Feed-forward Network

A one-layer network with N input elements and M neurons follows

For single layer, the number of inputs elements are N, number of nodes or neurons in layer $= M$,$dim(\mathbf{u}) = dim(\mathbf{b}) = dim(\mathbf{y}) = (M \times 1)$, the dimensions of $dim(X) = (N \times 1)$, the dimensions of $\mathbf{W}$, $dim(\mathbf{W}) = (M \times N)$, the Weight matrix:

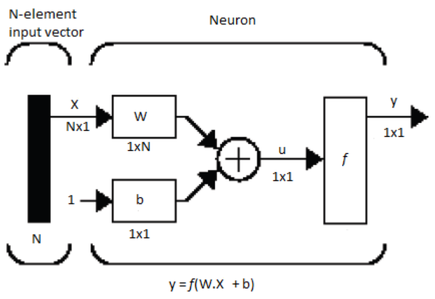
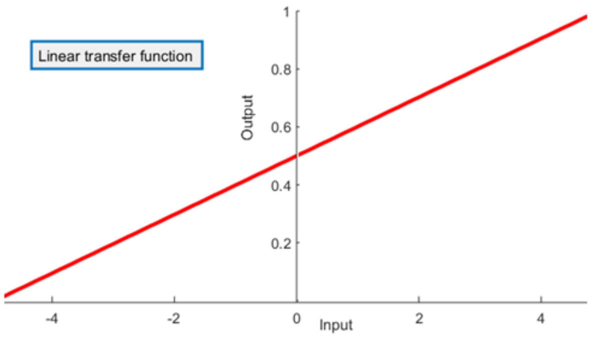
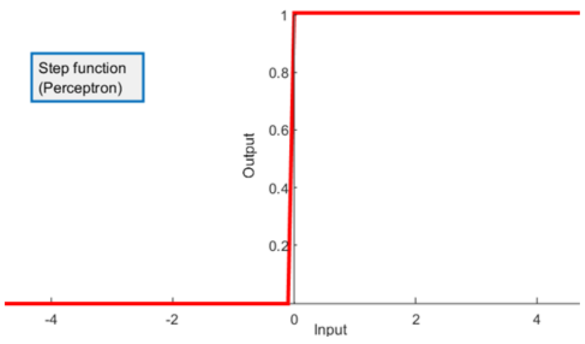


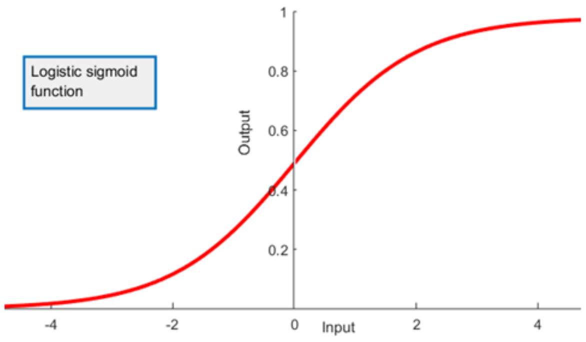
Figure 5.2: Data flow representation of a Neural Network.



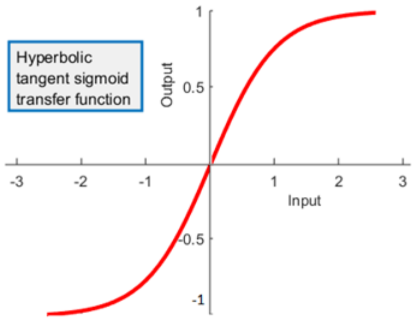
(a) Linear transfer function



(b) Step function (Perceptron)



(c) Logistic sigmoid function



(d) Hyperbolic tangent sigmoid transfer function

Figure 5.3: Transfer function used in Neural Networks.

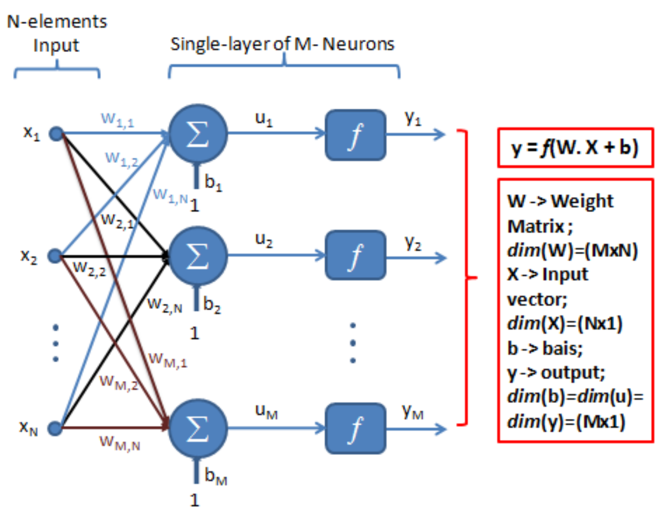

N-elements Input
Single-layer of M- Neurons
x1
x2
xN
w1,1
w1,2
w1,N
w2,1
w2,2
w2,N
wM,1
wM,2
wM,N
Σ
b1
b2
bM
1
u1
u2
uM
f
y1
y2
yM
y = f(W. X + b)
W -> Weight Matrix ; dim(W)=(MxN)
X -> Input vector; dim(X)=(Nx1)
b -> bais;
y -> output; dim(b)=dim(u)= dim(y)=(Mx1)

Figure 5.4: Single-layer Neural Network

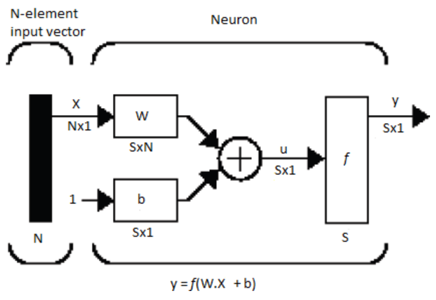


Figure 5.5

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,N} \\ w_{2,1} & w_{2,2} & \dots & w_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ w_{M,1} & w_{M,2} & \dots & w_{M,N} \end{bmatrix}$$

In this network, N and M are number of elements in input vector and number of neurons in layer respectively.Each element of the input vector $X_{(N\times1)}$ is fed to each neuron input through the weight matrix $\mathbf{W}$. The ith neuron has a summer that gathers its weighted inputs and bias to form its own scalar output u_i, where $i = 1, 2, 3, ..., M$. The various u_i taken together form an M-element net input vector $\mathbf{u}_{(M\times1)}$. Finally, the neuron layer outputs form a column vector $\mathbf{y}_{(M\times1)}$. The expression for y is shown at the bottom of the figure.

Note that N is not necessarily equal to M.

Here X is an N-length input vector, $\mathbf{W}$ is an $M \times N$ matrix, $\mathbf{y}$ and $\mathbf{b}$ are M-length vectors.

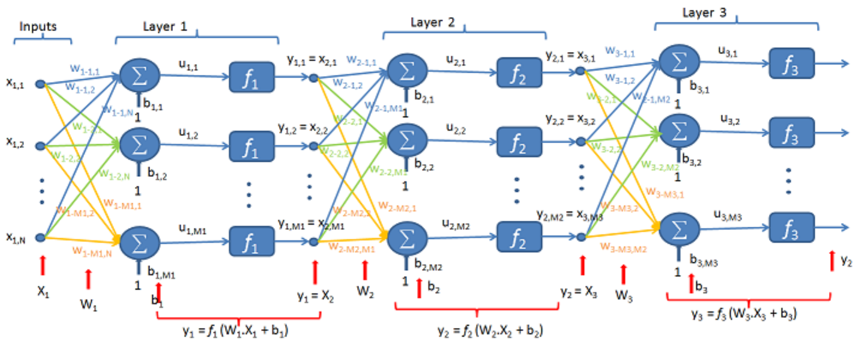


Figure 5.6: Multilayer feed forward network architecture.

$$
\begin{aligned}
&u_{(M\times 1)} = \mathbf{W}_{(M\times N)}.X_{(N\times 1)} + \mathbf{b}_{(M\times 1)}\\
&y_1 = f(u_1)\\
&y_2 = f(u_2)\\
&\vdots\\
&y_M = f(u_M)\\
&\mathbf{y}_{(M\times 1)} = f(\mathbf{W}_{(M\times N)}.X_{(N\times 1)} + \mathbf{b}_{(M\times 1)})
\end{aligned}
$$

5.2.2.2 Multilayer Feed Forward Network/Multilayer Perception

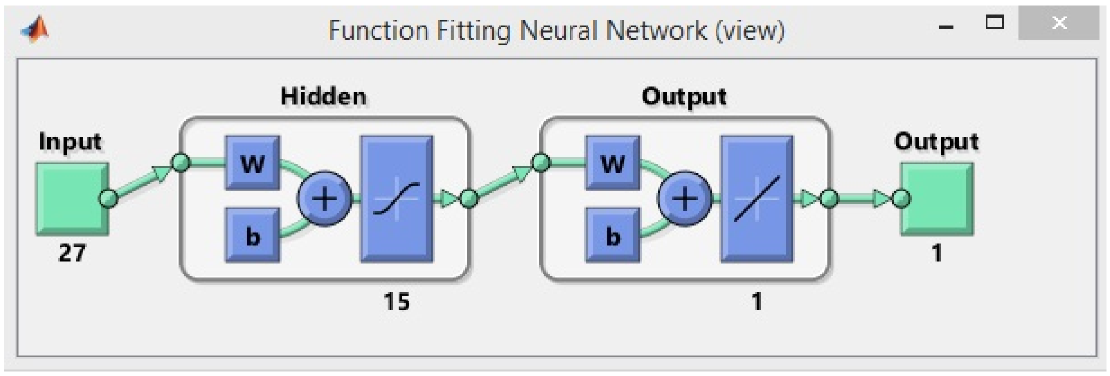
To describe networks having multiple layers, it needs to make a distinction between weight matrices that are connected to inputs and weight matrices that are connected between layers. It also needs to identify the source and destination for the weight matrices. We will call weight matrices connected to inputs input weights; we will call weight matrices connected to layer outputs layer weights. Further, superscripts are used to identify the source (second index) and the destination (first index) for the various weights and other elements of the network. To illustrate, the one-layer multiple input network shown earlier is redrawn in abbreviated form here.

5.3 Neural Network Toolbox

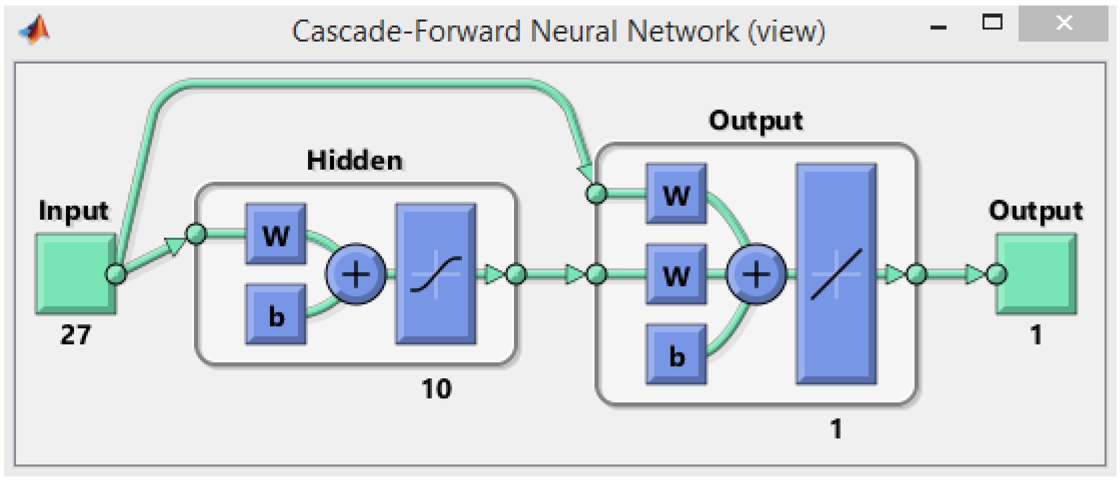
Neural Network Toolbox™ provides functions and apps for modeling complex nonlinear systems that are not easily modeled with a closed-form equation. Neural Network Toolbox supports supervised learning with feedforward, radial basis, and dynamic networks. It also supports unsupervised learning with self-organizing maps and competitive layers. With the toolbox one can design, train, visualize, and simulate neural networks. The Neural Network Toolbox can be used for applications such as data fitting, pattern recognition, clustering, time-series prediction, and dynamic system modeling and control. In this Appendix, we use the too box to train different class of neural network (NN) to predict the West Nile Viruses presence and the number of mosquitos that are collected in different traps as discussed in Chapter 3.

5.4 West Nile Virus Prediction Model Using Neural Networks

In order to design and compare various neural network architectures to predict the presence of the West Nile Virus, we first define the train and the test data setting as discussed in the Appendix A. We consider 33 parameters as input to predict the output target the WNV presence, a multi-input-single-output (MISO) NN as in Fig. 5.7 . We also develop another class of neural network of multi-input-multi-output (MIMO) class, where we predict multiple outputs viz., the WNV presence and the number of mosquitos in various traps. We develop three class of neural networks , the feedforward, cascade-forward, and the fitnet that are supported by the Matlab Neural network toolbox. We compare performance of MISO and MIMO NN and tabulate their results in Table5.2 through Table 5.4 for training and validation. We observe that the MISO NN performs poorly compared to the MIMO NNs for all the tree categories. In addition, with the increase in number of neurons in hidden layer from 5 to 30 for layer ranging from 1 to 5, the performance does not improve drastically, rather at times the performance reduces with number of layers. This behavior is observed with different training algorithms including the back propagation. A judicial choice of the number of hidden layers will be subject of future study, however as we focus in implementing these neural network in resource constraint IoT devices, we observe that there is interesting trade-off between the number of layers and neuron in each layer with that of the perfromance accuracy improvement in prediuction as well as execution time. The limited design space exploration of the NN prametric space for the particular example of WNV presence along with mosquitos present in a trap is tabulated in Table. 5.5 It is observed that, for this particular exaple, a single layer NN alost provides similar accuracy when compared to a multi-layer NN. In other words, for a small improvement in perfromance (5-10%), a large computation time or resource is required (almost 1.8×) in teh IoT device. Therefore, a judicious point where the the resoureces in the IoT devices can best be ustilized for the maximum performance need to be considered. In qaddition to the NN input and architectural parameter spaee, it is also important to consider the implementation resources for an efficient implementation. We discuss few implementation strategies to achieve that in the next section.



(a) Feed forward architecture.



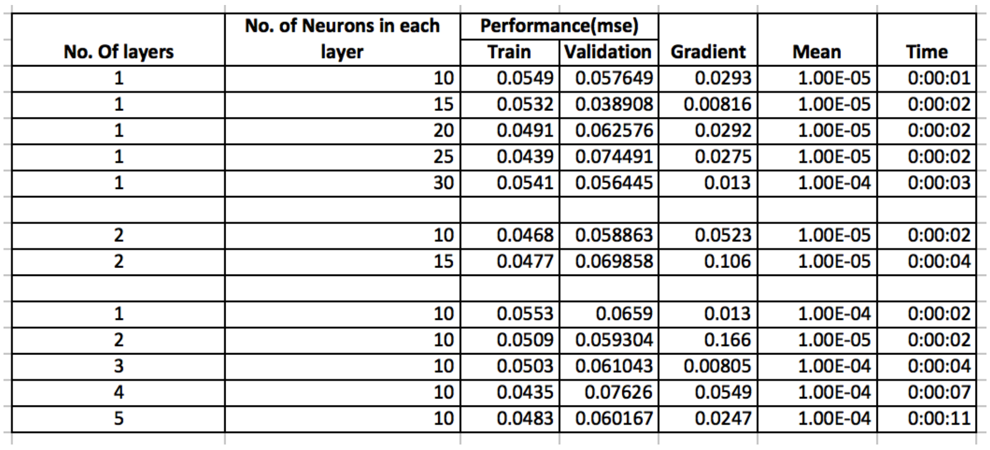
(b) Cascade forward architecture.

Figure 5.7: Architecture of a feedforward NN displayed using the Matlab Neural Network Toolbox.

Table 5.2: Feedforward NN for WNV prediction.

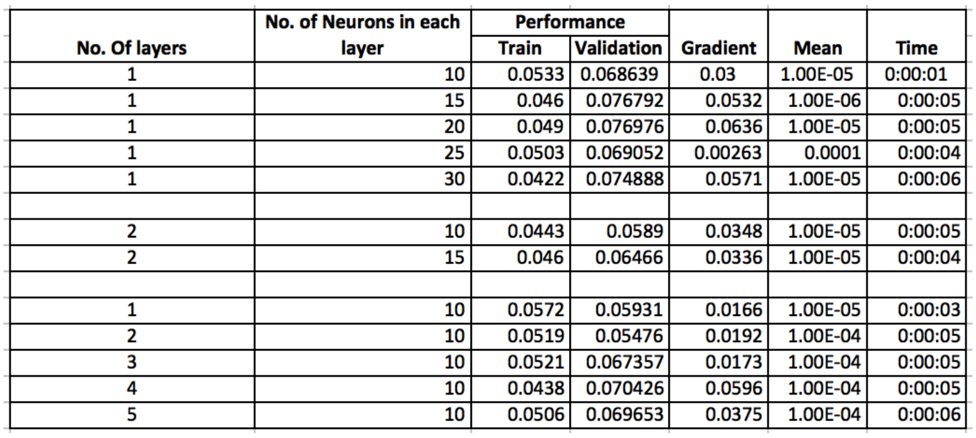
	No. of Neurons in each	Performance				
No. Of layers	layer	Train	Validation	Gradient	Mean	Time
1	10	0.055	0.05372	0.0588	1.00E-05	0:00:01
1	15	0.0512	0.075793	0.0161	1.00E-05	0:00:01
1	20	0.055	0.056438	0.0512	1.00E-05	0:00:02
1	25	0.0497	0.050183	0.092	1.00E-05	0:00:02
1	30	0.0463	0.063143	0.07	1.00E-05	0:00:03
2	10	0.0542	0.059558	0.0456	1.00E-05	0:00:02
2	15	0.0525	0.056645	0.0291	0.0001	0:00:02
1	10	0.0554	0.059228	0.0384	1.00E-05	0:00:01
2	10	0.0447	0.067447	0.0324	1.00E-05	0:00:02
3	10	0.0503	0.056536	0.0713	0.0001	0:00:02
4	10	0.0459	0.070416	0.0276	0.0001	0:00:03
5	10	0.0491	0.049552	0.0534	0.001	0:00:03

Table 5.3: Cascade-forward NN for WNV prediction.



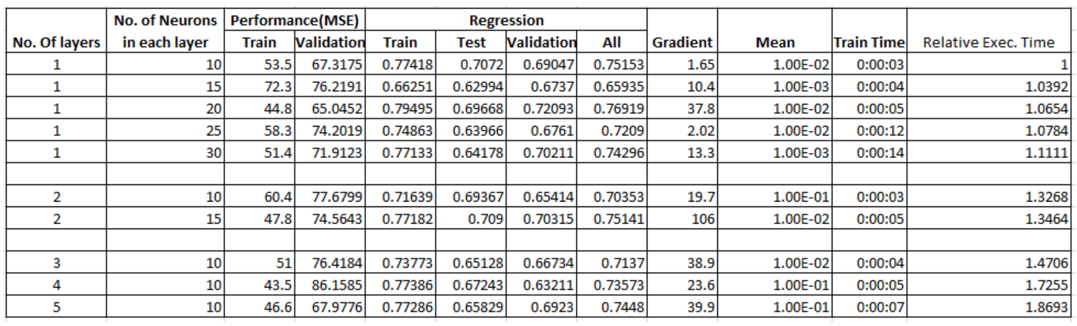
No. Of layers	No. of Neurons in each layer	Performance(mse)		Gradient	Mean	Time
		Train	Validation			
1	10	0.0549	0.057649	0.0293	1.00E-05	0:00:01
1	15	0.0532	0.038908	0.00816	1.00E-05	0:00:02
1	20	0.0491	0.062576	0.0292	1.00E-05	0:00:02
1	25	0.0439	0.074491	0.0275	1.00E-05	0:00:02
1	30	0.0541	0.056445	0.013	1.00E-04	0:00:03
2	10	0.0468	0.058863	0.0523	1.00E-05	0:00:02
2	15	0.0477	0.069858	0.106	1.00E-05	0:00:04
1	10	0.0553	0.0659	0.013	1.00E-04	0:00:02
2	10	0.0509	0.059304	0.166	1.00E-05	0:00:02
3	10	0.0503	0.061043	0.00805	1.00E-04	0:00:04
4	10	0.0435	0.07626	0.0549	1.00E-04	0:00:07
5	10	0.0483	0.060167	0.0247	1.00E-04	0:00:11

Table 5.4: fitnet for WNV prediction.



No. Of layers	No. of Neurons in each layer	Performance		Gradient	Mean	Time
		Train	Validation			
1	10	0.0533	0.068639	0.03	1.00E-05	0:00:01
1	15	0.046	0.076792	0.0532	1.00E-06	0:00:05
1	20	0.049	0.076976	0.0636	1.00E-05	0:00:05
1	25	0.0503	0.069052	0.00263	0.0001	0:00:04
1	30	0.0422	0.074888	0.0571	1.00E-05	0:00:06
2	10	0.0443	0.0589	0.0348	1.00E-05	0:00:05
2	15	0.046	0.06466	0.0336	1.00E-05	0:00:04
1	10	0.0572	0.05931	0.0166	1.00E-05	0:00:03
2	10	0.0519	0.05476	0.0192	1.00E-04	0:00:05
3	10	0.0521	0.067357	0.0173	1.00E-04	0:00:05
4	10	0.0438	0.070426	0.0596	1.00E-04	0:00:05
5	10	0.0506	0.069653	0.0375	1.00E-04	0:00:06

Table 5.5: Feedforward NN for WNV and number of mosquito predictions.



No. Of layers	No. of Neurons in each layer	Performance(MSE)		Regression				Gradient	Mean	Train Time	Relative Exec. Time
		Train	Validation	Train	Test	Validation	All				
1	10	53.5	67.3175	0.77418	0.7072	0.69047	0.75153	1.65	1.00E-02	0:00:03	1
1	15	72.3	76.2191	0.66251	0.62994	0.6737	0.65935	10.4	1.00E-03	0:00:04	1.0392
1	20	44.8	65.0452	0.79495	0.69668	0.72093	0.76919	37.8	1.00E-02	0:00:05	1.0654
1	25	58.3	74.2019	0.74863	0.63966	0.6761	0.7209	2.02	1.00E-02	0:00:12	1.0784
1	30	51.4	71.9123	0.77133	0.64178	0.70211	0.74296	13.3	1.00E-03	0:00:14	1.1111
2	10	60.4	77.6799	0.71639	0.69367	0.65414	0.70353	19.7	1.00E-01	0:00:03	1.3268
2	15	47.8	74.5643	0.77182	0.709	0.70315	0.75141	106	1.00E-02	0:00:05	1.3464
3	10	51	76.4184	0.73773	0.65128	0.66734	0.7137	38.9	1.00E-02	0:00:04	1.4706
4	10	43.5	86.1585	0.77386	0.67243	0.63211	0.73573	23.6	1.00E-01	0:00:05	1.7255
5	10	46.6	67.9776	0.77286	0.65829	0.6923	0.7448	39.9	1.00E-01	0:00:07	1.8693

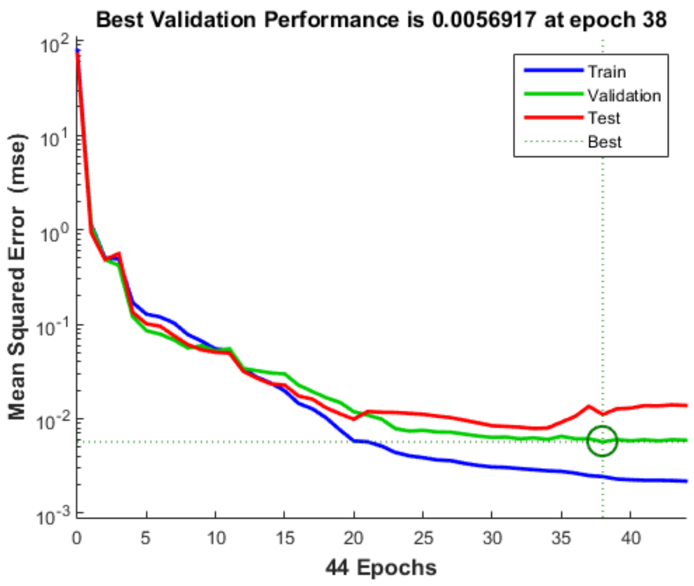


Figure 5.8: Validation of the performance of a feedforward neural network.

5.5 Strategies for Efficient Neural Network Implementation in IoT devices

In order to efficiently implement the proposed Neural network in the resource constraint IoT devices, we adopt the following strategies to improve performance and reduced resource consumption.

- **Floating point to fixed point conversion:** we convert the floating point implementation of the NN to a fixed point implementation by using Matlab fixed point tool box. This conversion to fixed point implementation reduces the memory, energy consumption, as well as computation time required to perform the NN based predictions.

- **Judicious Bit allocation :** by discriminating the number of bit in the fixed point representation of the NN, e.g., using 16 -bits instead of 32-bits for some layers, the NN resource utilization can be reduced.

- **Approximation :** by tuning the approximation of synapse functions as well as auto tuning the bits used in computation, the NN's resource utilization can be improved.

- **Computation sharing:** in case the NN is implemented in hardware, instead of using dedicated multiplier and adders for each layer, a pipelined architecture sharing the computation resources can be considered.

5.6 Summary

In this chapter we used neural network to build predictive models for sense-making in IoT devices. We used the prediction models to predict the presence of a West Nile Virus and number of mosquitoes as use case to showcase the use of the predictive models for sense-making application in IoT devices. We also briefly describe four strategies to implement the Neural network for resource constraint IoT devices by using fixed point representation of the computations, judicious bit allocation across layers, approximation of operations and transfer functions, and computation sharing e.g. use of lookup tables (LUTs).

Chapter 6

Summary and Future Works

In this thesis we proposed a scalable and energy efficient sense-making framework using a hierarchical compressive sensing for efficient collective understanding of users, contexts, and their environments. To the best of our knowledge, a scalable spatio-temporal compressive and collaborative mobile sensing framework for multi-tiered hierarchical network configurations specifically supporting physical and virtual sensing has not been considered. The proposed sense-making IoT framework, called Essence, employs combination of participatory mobile crowdsensing along with infrastructure sensing to perform sense-making using machine learning techniques such as PCA, compressive sensing, and heterogeneous model based compressive sensing techniques for approximate gathering and processing of diverse sensor data. We derived the performance and energy consumption for several data aggregation methods and compared them for various network size and compressive sensing methods. The closed form derivation of the performance and energy consumption of several schemes were derived by taking into account the reconstruction algorithm's time complexity and energy overhead in the formulations. The effect of network scaling, different levels of hierarchy, compression ratio on the performance and energy for different methods are studied. The impact of file size that each sensor node collects has also been studied and compared for different scheme. Unlike the existing state-of-the-arts, this thesis introduced and used a model based compressive sensing (MCS) method in the reconstruction and recovery stage of the processing and derived closed for performance and energy consumption models. Compared to the CoSaMP based recovery methods, the MCS methods provides improved performance and energy efficient as the network size scales. We demonstrate the application of Essence sense-making framework in predicting the presence of West Nile Virus (WNV) in a region as a case study.

This thesis presents an efficient and scalable sense-making framework using machine learning techniques for Internet-of-Things (IoTs) in order to understand users, contexts, and their environments to make meaningful decisions. The proposed sense-making IoT framework, called Essence, employs combination of participatory mobile crowdsensing along with infrastructure sensing to perform sense-making using machine learning techniques. While collaborative mobile crowdsensing enables information to be gathered and shared by users who are directly involved (participatory sensing) or integrated seamlessly as needed (opportunistic sensing) through user mobile platforms, the infrastructure sensing fabric of the Essence framework provides sense-making support for scenarios where mobile sensing platforms are inadequate. To address the scalability needs of the Essence framework, we employ machine learning techniques such as principal component analysis (PCA) based heterogeneous compressive sensing techniques for approximate gathering and processing of diverse sensor data. This requires new mechanisms for sensor data collection, tunable approximate processing and machine learning based decision making using a hierarchical networking architecture to create a compressive collaborative sense-making framework for IoTs. Essence is an extension of our previous SenseDroid mobile sensing framework with machine learning enabled decision and actuation components for IoT paradigms. The Essence framework is build using a multi-tiered hierarchical architecture for sensing spatial variations of a parameter of interest, perceive spatio-temporal fields, and enable energy efficient local mobile or infrastructure sensing with a small number of measurements. This approximate, yet tunable approach combines different sensing approaches opportunistically while trading scalability (and coverage) for data accuracy (and energy efficiency). We demonstrate the application of Essence sense-making framework in predicting the presence of West Nile Virus (WNV) in a region as a case study. The thesis also discuss several challenges associated with sense-making approaches for emerging IoT applications.

6.1 Challenges and Future Work

There are several challenges that need to be addressed and we pursue them as our future research directions. Some of these research direction are briefly explained below:

Energy Efficiency : Power consumption and energy efficiency are serious issues in mobile sensing, as continuous monitoring can largely drain the battery in a short period of time. Energy-efficiency issues have been studied in the context of mobile phone sensing recently in [43, 52, 39, 36, 31]. In [43], the authors showed that collaborative sensing can achieve over 80% power savings compared to traditional sensing without collaborations. Research in the direction of sensor scheduling, adaptive sampling, and compressive sampling and their novel combinations within the framework provide new research opportunities for energy-efficiency.

Privacy Regulation: Privacy is a major concern in collaborative sensing application. To address this we have adopted transparency, full user control, and encryption of the data that is shared. User can fully set or control their preferences, enable or disable features, control of the type of sensors and parameter that can be shared in the application to avoid any violations. In the worst case, the user can opt-out of any such applications. Research addressing privacy in mobile phone sensing is in its infancy with initial research work proposing privacy preserving incentive mechanisms [24, 51].

Incentive Mechanisms: Incentive mechanism to motivate participation and collaboration is an important aspect that need to researched to bring desirable economic properties and appropriate utility in the collaboration framework. Few recent studies in this direction looked into selecting well-suited participants for sensing services within recruitment frameworks [40], sealed-bid second-price auction to motivate user participation [10], and reverse auction based dynamic price incentive mechanisms [23]. A comparative study of different incentive mechanisms for a client to motivate the collaboration of smartphone users on both data acquisition and distributed computing applications is evaluated in [13].

Heterogeneity in Mobile Cloud: Heterogeneity manifests in several aspects (e.g. networks, sensor types, sensor accuracy) of the collaborative sensing. Our present development in primarily focused on using Wi-Fi based mobile ad-hoc networks for localize environments specifically for the local cloud. However, support for more power efficient networks like Bluetooth can considered to support the nanocloud architecture. Handling the heterogeneity in network architectures, mobile devices and sensors, as well as services is challenging area of research and need to be further studied in the context of compressive sensing [41].

Actions and Control: As a complement to sensing, the IoT offers us a way to control the physical world through displays, actuators, and switches. Many modern systems benefit from remote control because it simplifies physical interaction design and extends capabilities. In the Physical Web paradigm, anything with a display, actuator, or switch can be controlled from a browser or through a Web service, thus making it easy to integrate related information into control decisions. For example, emerging Web-connected irrigation systems might provide an interface for specifying the plants in your garden and use Web services to determine expert watering recommendations. Privacy and security One of the primary challenges for the future will be avoiding the darker consequences of a world with globally connected devices. The Physical Web could enable hackers to control our devices unless precautions are taken. The conventional Web already has security measures in place that can be applied to the Physical Web, but it is unclear if these will be suitable or even adequate for all IoT applications. In addition to hacking, social threats can result when knowledge is leaked in unexpected ways. For example, knowledge that a house is in an energy-saving mode could be a good indication that nobody is home and thus invite a burglar. These challenges will become more pressing as use of the Physical Web continues to grow.

Bibliography

[1] Nadav Aharony, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. Social fMRI: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659, 2011.

[2] A.Noronha et al. Attaining IoT Value: How To Move from Connecting Things to Capturing Insights, 2014. http://www.cisco.com/c/dam/en/us/products/collateral/cloud-systems-management/data-analytics/iot-whitepaper.pdf.

[3] Avnet. Zynq-7000 All Programmable SoC / AD9361 Software-Defined Radio Evaluation Kit, 2013. http://www.em.avnet.com/en-us/design/drc/Pages/Zynq-7000-AP-SoC-AD9361-Software-Defined-Radio-Evaluation-Kit.aspx.

[4] Bubacarr Bah, Luca Baldassarre, and Volkan Cevher. Model-based sketching and recovery with expanders. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1529–1543. SIAM, 2014.

[5] Richard G Baraniuk, Volkan Cevher, Marco F Duarte, and Chinmay Hegde. Model-based compressive sensing. *Information Theory, IEEE Transactions on*, 56(4):1982–2001, 2010.

[6] E.J. Candes and M.B. Wakin. An Introduction To Compressive Sampling. *Signal Processing Magazine, IEEE*, 25(2):21–30, 2008.

[7] Delia Ciullo, Guner D Celik, and Eytan Modiano. Minimizing transmission energy in sensor networks via trajectory control. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*, pages 132–141. IEEE, 2010.

[8] Sunny Consolvo et al. Activity sensing in the wild: a field trial of ubifit garden. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1797–1806. ACM, 2008.

[9] Li Da Xu, Wu He, and Shancang Li. Internet of Things in industries: A survey. *Industrial Informatics, IEEE Transactions on*, 10(4):2233–2243, 2014.

[10] George Danezis, Stephen Lewis, and Ross Anderson. How Much is Location Privacy Worth? In *Online Proceedings of the Workshop on the Economics of Information Security Series (WEIS 2005.* Citeseer, 2005.

[11] Analog Devices. Linux Industrial I/O Subsystem, 2012. https://wiki.analog.com/software/linux/docs/iio/iio.

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

[13] Lingjie Duan et al. Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing. In *INFOCOM, 2012 Proceedings IEEE*, pages 1701–1709, 2012.

[14] Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1548–1557. IEEE, 2001.

[15] David Kiyoshi Goldenberg, Jie Lin, A Stephen Morse, Brad E Rosen, and Y Richard Yang. Towards mobility as a network control primitive. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 163–174. ACM, 2004.

[16] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[17] Micah B Hahn, Andrew J Monaghan, Mary H Hayden, et al. Meteorological Conditions Associated with Increased Incidence of West Nile Virus Disease in the United States, 2004–2012. *The American journal of tropical medicine and hygiene*, 92(5):1013–1022, 2015.

[18] Chinmay Hegde, Piotr Indyk, and Ludwig Schmidt. Nearly linear-time model-based compressive sensing. In *Automata, Languages, and Programming*, pages 588–599. Springer, 2014.

[19] W.Z. Khan, Yang Xiang, M.Y. Aalsalem, and Q. Arshad. Mobile Phone Sensing Systems: A Survey. *Communications Surveys Tutorials, IEEE*, 15(1):402–427, 2013.

[20] Gary Klein, Brian Moon, and Robert R Hoffman. Making sense of sensemaking 1: Alternative perspectives. *IEEE intelligent systems*, (4):70–73, 2006.

[21] Gary Klein, Brian Moon, and Robert R Hoffman. Making sense of sensemaking 2: A macrocognitive model. *Intelligent Systems, IEEE*, 21(5):88–92, 2006.

[22] N.D. Lane, E. Miluzzo, Hong Lu, et al. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[23] Juong-Sik Lee and Baik Hoh. Sell your experiences: a market mechanism based incentive for participatory sensing. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pages 60–68, 2010.

[24] Qinghua Li and Guohong Cao. Providing privacy-aware incentives for mobile sensing. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, volume 18, page 22, 2013.

[25] Robert LiKamWa et al. MoodScope: building a mood sensor from smartphone usage patterns. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, MobiSys '13, pages 389–402, New York, NY, USA, 2013. ACM.

[26] Hong Lu et al. StressSense: detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 351–360, New York, NY, USA, 2012. ACM.

[27] Chong Luo, Feng Wu, Jun Sun, and Chang Wen Chen. Compressive data gathering for large-scale wireless sensor networks. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 145–156. ACM, 2009.

[28] Jun Luo, Liu Xiang, and Catherine Rosenberg. Does compressed sensing improve the throughput of wireless sensor networks? In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[29] S. Madhani, M. Tauil, and Tao Zhang. Collaborative sensing using uncontrolled mobile devices. In *Collaborative Computing: Networking, Applications and Worksharing, 2005 International Conference on*, pages 8 pp.–, 2005.

[30] Carrie A Manore, Justin K Davis, Rebecca C Christofferson, et al. Towards an early warning system for forecasting human West Nile virus incidence. *PLoS currents*, 6.

[31] Mirco Musolesi et al. Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones. In *Pervasive Computing*, pages 355–372. Springer, 2010.

[32] S. Nath. ACE: Exploiting Correlation for Energy-Efficient and Continuous Context Sensing. *Mobile Computing, IEEE Transactions on*, 12(8):1472–1486, 2013.

[33] NBA. NEEA Study On Luminarie Level Lighting Controls (LLLC) -Enlighted Technical Proof Of Concept Study. Technical report, Northwest Energy Efficiency Alliance, July, 2013.

[34] Deanna Needell and Joel A Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[35] Chicago Department of Public Health. Predict West Nile virus in mosquitos across the city of Chicago, 2015. https://www.kaggle.com/c/predict-west-nile-virus.

[36] Jeongyeup Paek, Joongheon Kim, and Ramesh Govindan. Energy-efficient rate-adaptive GPS-based positioning for smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 299–314, New York, NY, USA, 2010. ACM.

[37] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454, 2014.

[38] Hairong Qi, Yingyue Xu, and Xiaoling Wang. Mobile-agent-based collaborative signal and information processing in sensor networks. *Proceedings of the IEEE*, 91(8):1172–1183, 2003.

[39] M. Rahman, J. Gao, and Wei-Tek Tsai. Energy Saving in Mobile Cloud Computing*. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 285–291, 2013.

[40] Sasank Reddy, Deborah Estrin, and Mani Srivastava. Recruitment framework for participatory sensing data collections. In *Proceedings of the 8th international conference on Pervasive Computing*, Pervasive'10, pages 138–155, Berlin, Heidelberg, 2010. Springer-Verlag.

[41] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya. Heterogeneity in mobile cloud computing: Taxonomy and open challenges, 2013.

[42] Santanu Sarma, Nikil Dutt, and Nalini Venkatasubramanian. Cross-layer Virtual Observers for Embedded Multiprocessor System-on-chip (MPSoC). In *Proceedings of the 11th International Workshop on Adaptive and Reflective Middleware*, ARM '12, pages 4:1–4:7, New York, NY, USA, 2012. ACM.

[43] Xiang Sheng, Jian Tang, and Weiyi Zhang. Energy-efficient collaborative sensing with mobile phones. In *INFOCOM, 2012 Proceedings IEEE*, pages 1916–1924. IEEE, 2012.

[44] Niwat Thepvilojanapong et al. Opportunistic collaboration in participatory sensing environments. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, MobiArch '10, pages 39–44, New York, NY, USA, 2010. ACM.

[45] Robert Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[46] J.A. Tropp et al. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. *Information Theory, IEEE Trans. on*, 53(12):4655–4666, 2007.

[47] Ovidiu Vermesan, Peter Friess, Patrick Guillemin, et al. Internet of things strategic research roadmap. 2011.

[48] H. Weinschrott, F. Durr, and K. Rothermel. StreamShaper: Coordination algorithms for participatory mobile urban sensing. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 195–204, 2010.

[49] Liu Xiang, Jun Luo, and Athanasios Vasilakos. Compressed data aggregation for energy efficient wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on*, pages 46–54. IEEE, 2011.

[50] Xi Xu, Rashid Ansari, and Ashfaq Khokhar. Power-efficient hierarchical data aggregation using compressive sensing in WSNs. In *Communications (ICC), 2013 IEEE International Conference on*, pages 1769–1773. IEEE, 2013.

[51] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 173–184. ACM, 2012.

[52] Weiwen Zhang, Yonggang Wen, and D.O. Wu. Energy-efficient scheduling policy for collaborative execution in mobile cloud computing. In *INFOCOM, 2013 Proceedings IEEE*, pages 190–194, 2013.

[53] Shu Zhou, Min-You Wu, and Wei Shu. Finding optimal placements for mobile sensors: wireless sensor network topology adjustment. In *Emerging Technologies: Frontiers of Mobile and Wireless Communication, 2004. Proceedings of the IEEE 6th Circuits and Systems Symposium on*, volume 2, pages 529–532 Vol.2, 2004.

Appendix A

West Nile Virus Data Set

In 2002, the first human cases of West Nile virus were reported in Chicago. By 2004 the City of Chicago and the Chicago Department of Public Health (CDPH) had established a comprehensive surveillance and control program that is still in effect today. Every week from late spring through the fall, mosquitos in traps across the city are tested for the virus. The results of these tests influence when and where the city will spray airborne pesticides to control adult mosquito populations. Given weather, location, testing, and spraying data, this competition asks to predict when and where different species of mosquitos will test positive for West Nile virus. A more accurate method of predicting outbreaks of West Nile virus in mosquitos will help the City of Chicago and CPHD more efficiently and effectively allocate resources towards preventing transmission of this potentially deadly virus.

Several modeling, laboratory and field studies have shown that WNV transmission is associated with environmental factors such as temperature, precipitation, drought, and land use, as well as biological factors such as bird community structure and anthropological variables, including urbanization and human density 8,45. However, most of the studies focused exclusively on particular states, counties, or regions based on the data available to the researchers. There is a need for a concise, relatively simple model that can predict early in the year when risk of human WNV may be elevated in order to inform public health decisions, resource allocation, and public education.

A.1 Data Description for West Nile Virus Prediction

This data set is based on the data set provided in the Kaggle competion for West Nile Virus prediction [35]. In this competition, weather data and GIS data provided to analyzing and predicting whether or not West Nile virus is present, for a given time, location, and species.

Every year from late-May to early-October, public health workers in Chicago setup mosquito traps scattered across the city. Every week from Monday through Wednesday, these traps collect mosquitos, and the mosquitos are tested for the presence of West Nile virus before the end of the week. The test results include the number of mosquitos, the mosquitos species, and whether or not West Nile virus is present in the cohort.

A.1.1 Main dataset

These test results are organized in such a way that when the number of mosquitos exceed 50, they are split into another record (another row in the dataset), such that the number of mosquitos are capped at 50.

The location of the traps are described by the block number and street name. For convenience, these attributes are mapped into Longitude and Latitude in the dataset. Please note that these are derived locations. For example, Block=79, and Street= "W FOSTER AVE" gives us an approximate address of "7900 W FOSTER AVE, Chicago, IL", which translates to (41.974089,-87.824812) on the map.

Some traps are "satellite traps". These are traps that are set up near (usually within 6 blocks) an established trap to enhance surveillance efforts. Satellite traps are postfixed with letters. For example, T220A is a satellite trap to T220.

Please note that not all the locations are tested at all times. Also, records exist only when a particular species of mosquitos is found at a certain trap at a certain time. In the test set, all combinations/permutations of possible predictions are asked and are only scoring the observed ones.

A.1.2 Spray Data

The City of Chicago also does spraying to kill mosquitos. GIS data is given for their spray efforts in 2011 and 2013. Spraying can reduce the number of mosquitos in the area, and therefore might eliminate the appearance of West Nile virus.

A.1.3 Weather Data

It is believed that hot and dry conditions are more favorable for West Nile virus than cold and wet. The dataset is provided from NOAA of the weather conditions of 2007 to 2014, during the months of the tests.

Station 1: CHICAGO O'HARE INTERNATIONAL AIRPORT Lat: 41.995 Lon: -87.933 Elev: 662 ft. above sea level Station 2: CHICAGO MIDWAY INTL ARPT Lat: 41.786 Lon: -87.752 Elev: 612 ft. above sea level

A.1.4 Map Data

The map files mapdata_copyright_openstreetmap_contributors.rds and mapdata_copyright_ope are from Open Streetmap and are primarily provided for use in visualizations. Open Streetmap are allowed to use in the models.

Here's an example using mapdata_copyright_openstreetmap_contributors.rds, and here's one using mapdata_copyright_openstreetmap_contributors.txt.

A.1.5 File descriptions

train.csv, test.csv - the training and test set of the main dataset. The training set consists of data from 2007, 2009, 2011, and 2013, while in the test set you are requested to predict the test results for 2008, 2010, 2012, and 2014.

- Id: the id of the record
- Date: date that the WNV test is performed

- Address: approximate address of the location of trap. This is used to send to the GeoCoder.
- Species: the species of mosquitos
- Block: block number of address
- Street: street name
- Trap: Id of the trap
- AddressNumberAndStreet: approximate address returned from GeoCoder
- Latitude, Longitude: Latitude and Longitude returned from GeoCoder
- AddressAccuracy: accuracy returned from GeoCoder
- NumMosquitos: number of mosquitoes caught in this trap
- WnvPresent: whether West Nile Virus was present in these mosquitos. 1 means WNV is present, and 0 means not present.

spray.csv - GIS data of spraying efforts in 2011 and 2013

- Date, Time: the date and time of the spray
- Latitude, Longitude: the Latitude and Longitude of the spray

weather.csv - weather data from 2007 to 2014. Column descriptions in noaa_weather_qclcd_document
sampleSubmission.csv - a sample submission file in the correct format

Appendix B

IoT Platforms and SDKs

Table B.1: IoT Platforms and SDKs

SL No	Platform Name	Web Addess
1	Galilio, Intel	https://software.intel.com/en-us/iot/downloads
2	Edision, Intel	https://software.intel.com/en-us/iot/downloads
3	Qualcom	https://developer.qualcomm.com/hardware/iot-cellular-dev
4	Alljoyn , Qualcomm	https://developer.qualcomm.com/software/alljoyn
5	Parse (Facebook)	https://parse.com/products/iot
6	Contiki	http://www.contiki-os.org/
7	RIOT	http://www.riot-os.org/
7	mbed, ARM	https://www.arm.com/products/internet-of-things-solutions
8	Spark Core, Particle	https://www.particle.io
9	ThingsSpeak	https://thingspeak.com/
10	Brillo, Google	https://developers.google.com/brillo/?hl=en
11	Libelium	http://www.libelium.com/
12	IBM IoT kit	https://www-03.ibm.com/press/us/en/pressrelease/42227.wss
13	Broadcom	http://www.broadcom.com/application/internet_of_things.php
14	Stewart, the thing system	http://thethingsystem.com/index.html