Reliable Distributed Management in Uncertain Environments

by

Vinaya Chakati

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2020 by the
Graduate Supervisory Committee:

Sandeep K.S. Gupta, Chair
Partha Dasgupta
Ayan Banerjee
Anamitra Pal
Karthik Kumar

ARIZONA STATE UNIVERSITY
May 2021

ABSTRACT

Increase in the usage of Internet of Things(IoT) devices across physical systems has provided a platform for continuous data collection, real-time monitoring, and extracting useful insights. Limited computing power and constrained resources on the IoT devices has driven the physical systems to rely on external resources such as cloud computing for handling compute-intensive and data-intensive processing. Recently, physical environments have began to explore the usage of edge devices for handling complex processing. However, these environments may face many challenges such as uncertainty of device availability, uncertainty of data relevance, and large set of geographically dispersed devices. This research proposes the design of a reliable distributed management system that focuses on the following objectives: 1. improving the success rate of task completion in uncertain environments. 2. enhancing the reliability of the applications and 3. support latency sensitive applications. Main modules of the proposed system include:

1. A novel proactive user recruitment approach to improve the success rate of the task completion. 2.Contextual data acquisition and integration of false data detection for enhancing the reliability of the applications.3. Novel distributed management of compute resources for achieving real-time monitoring and to support highly responsive applications. User recruitment approaches select the devices for offloading computation. Proposed proactive user recruitment module selects an optimized set of devices that match the resource requirements of the application. Contextual data acquisition module banks on the contextual requirements for identifying the data sources that are more useful to the application. Proposed reliable distributed management system can be used as a framework for offloading the latency sensitive applications across the volunteer computing edge devices.

ACKNOWLEDGMENTS

I would like to express deepest gratitude to my advisor Dr. Sandeep K.S. Gupta for his continuous guidance and motivation throughout my Ph.D. His insights and technical discussions helped me improve my research papers, presentations. His patience and encouragement were key motivations for completing my Ph.D.

I am greatly thankful to Dr.Ayan Banerjee who has been my mentor throughout my doctoral studies. His regular technical discussions helped me in improving my work.

I would like to thank my committee members Dr. Partha Dasgupta, Dr. Anamitra Pal and Dr.Karthik Kumar for their effort and time to help me fulfill the degree requirements. I would also like to thank my colleagues of IMPACT Lab Priyanka, Madhurima, Jhung, Prajwal, Koosha, Imane, Javad, Bernard, Azamat and Apurupa for making my Ph.D experience a memorable one.

I would like to thank my friends Akhil, Sowmya, Venky, Mounika, Chinna, Deepu, Susmita, Naren for their constant support throughout my Ph.D journey. I would like to thank my friends in India Sandhya, Srinu and Dasharath for motivating and believing me all the times.

Finally, I would like to thank my parents, brother, sisters and other family members for supporting me to pursue my Ph.D.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

The proliferation of Internet of Things (IoT) has enabled large number of physical systems to be connected and monitored in real-time. Different application domains such as health-care [Doukas and Maglogiannis (2012); Catarinucci *et al.* (2015); Gope and Hwang (2016); Elhoseny *et al.* (2018)], smart grids[Li *et al.* (2011); Al-Ali and Aburukba (2015); Tonyali *et al.* (2018); Hussain *et al.* (2018)], smart transportation [Guerrero-Ibanez *et al.* (2015); Al-Dweik *et al.* (2017); Ganti *et al.* (2011)], disaster management [Kamruzzaman *et al.* (2017); Ray *et al.* (2017)], smart cities [Zanella *et al.* (2014); Sanchez *et al.* (2014); Centenaro *et al.* (2016)] have adopted IoT for designing innovative and real-time applications. Continuous data collection by the IoT devices may be useful for obtaining meaningful insights and making intelligent decisions. However, limited computing power and constrained resources on the IoT devices make it necessary for the physical systems to depend largely on external resources for executing compute and data intensive tasks. Cloud computing has been the most widely used solution for executing complex application tasks but it may not always be the feasible solution specially for the applications with low latency constraints. IoT environments have began to use Edge computing for such applications because of its ability to provide low response time services. Figure 1.1 shows the usage of edge devices and cloud servers for computation offloading in IoT environments.

Many IoT applications are real-time in nature. However, the term real-time may have different temporal granularity for different applications. For example, missing child application [Khan *et al.* (2016)] has high real-time constraints in the order of

an hour, where as a traffic update [Hu *et al.* (2015)] application may have relatively low time constraints in the order of minutes and real-time person tracking [Shi and Jia (2017)] or real-time video streaming applications have time constraints of few seconds and some applications such as power grid monitoring may have very low latency constraints typically in milliseconds. Edge computing may be beneficial for the applications with low latency constraints. Researchers have been exploring different type of edge computing solutions such as edge servers, mobile base stations, cloudlets, mobile adhoc networks. These solutions incur additional infrastructure and maintenance cost. This research focuses on designing cost-effective edge computing solution that does not involve additional infrastructure and also provide results within the time constraints of the applications.

Volunteer computing may be a desirable choice for designing cost-effective edge computing solutions for executing the real-time applications of IoT environments. Increasing number of computing devices across the population provides a platform to utilize the idle resources on volunteer devices for executing application tasks. Although there are large number of devices within the IoT environments, devices, network connectivity, resource availability and context of data on the volunteer computing devices are continually changing leading to uncertainty. Providing reliable results may be difficult due to uncertainty. Additionally, many of the IoT environments are loosely coupled in nature which exposes them to various challenging issues that might hinder the real-time monitoring. Some of the key challenges of uncertainty are 1. unavailability of computing devices, 2. unavailability of relevant data and 3. real-time monitoring. Affects of these challenges can be explained as follows:

Unavailability of computing devices: Many factors such as low power on the devices, intermittent network connectivity, human mobility, dynamic resource usage

patterns, multipurpose usage of the devices may result in the unavailability of the computing devices. To analyze the problems caused due to unavailability of computing devices we consider the example of real-time person tracking that uses mobile crowd sensing for data acquisition and volunteer computing for executing application tasks. IoT environments may be dynamic in nature, that is devices enter and leave the environment voluntarily. Also, resources on these devices may become unavailable or the device owner may choose to withdraw donating resources. If a voluntary device executing the application tasks becomes unavailable, it results in application task failure. To withstand application task failure, every time a device becomes unavailable the task has to be offloaded onto a new device. Unavailability of the device may decrease the success rate of the application execution and result in data loss. Also, task reassignment introduces computational overhead. Hence to increase success rate of the application execution and reduce the burden of task reassignment it may be useful to determine the devices that may be available for task completion. In a real-time person tracking in mobile crowd sensing example if a device that is assigned face detection task runs out of resources the task assigned to the device may be incomplete and therefore, has to be assigned to another device for the successful execution of the application.

Unavailability of relevant data: Dynamic nature of the devices in an uncertain environment may result in the unavailability of the relevant (required) data. Data relevance is vital for enriching performance and reliable execution of the application. Many factors contribute to data relevance. Data context is one such factor that has a significant impact on data relevance. Data context can be used to extract meaningful information about an entity [Abowd *et al.* (1999)]. Data collected without considering the required context may not be useful for the application. Availability of required

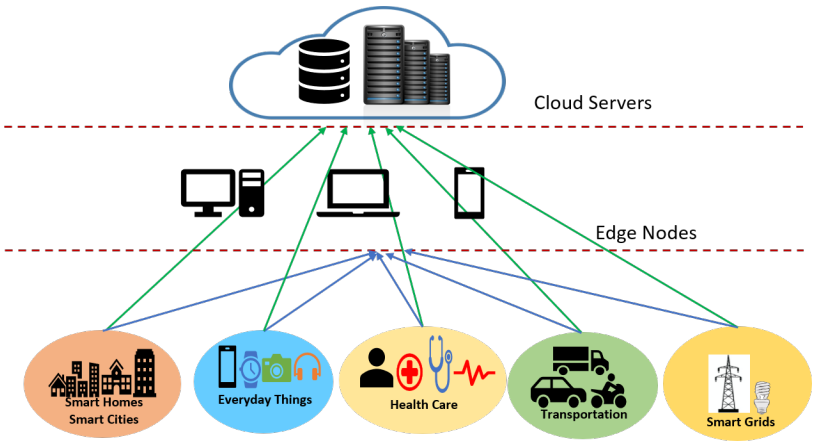


Figure 1.1: IoT Environments.

data context may be essential for improving the reliability of the application. Another factor that effects data relevance is data correctness. Data correctness improves reliability of the application. Security attacks may alter data, thereby effecting data correctness. Uncertain environments are more vulnerable to the security attacks as there is limited or no control on the devices that connects to the dynamic network. One of the most common security attacks that results in incorrect data is the false data injection (FDI) attack. Incorrect data leads to incorrect results thereby degrading reliability of the application. To improve reliability of the application it may be beneficial to acquire data from the devices that match the data context of the application. Importance of data context during an application execution is demonstrated using real-time person tracking example that exploits mobile crowd sensing for data collection. As mentioned earlier, in addition to the data context, data correctness is also crucial for reliability of the application. Linear state estimation process of large scale smart grid systems is used to illustrate the effect of incorrect data and integration of incorrect data detection methods.

Real-time Monitoring: Recent advancements in IoT devices such as faster data

sensing rates introduce the need for designing highly responsive applications to achieve real-time monitoring. Besides responsiveness, large number of devices (increased computational burden) and geographical distribution of the devices are major obstacles for realizing real-time monitoring in uncertain environments. Applications in uncertain environments can take advantage of edge computing solutions to handle highly responsive applications [Satyanarayanan (2017)]. Additionally, to manage large number of geographically dispersed devices and minimize the overhead of increased computational burden it may be feasible to design an efficient distributed management system. The highly responsive and critical process of linear state estimation in large scale smart grids systems is used to demonstrate the difficulties involved in achieving real-time monitoring in uncertain environments.

1.1 Solution Approach

Primary objective of this research is to design a system that optimizes for data collection and improves the success rate of task completion while meeting strict time constraints of real-time monitoring in uncertain environments. Figure 1.2 gives an overview of the proposed research. From the figure it may be observed that the proposed research consists of four solutions each addressing one fundamental operation of the IoT application. Fundamental operations of the IoT applications include: data acquisition, data correctness, computation offloading and real-time monitoring. This section provides brief description of the proposed solutions.

Data acquisition and data correctness are crucial for reliability of the application. State of the art systems collects data from all the available devices which results in large volumes of data. However, all this data may not be useful for application, and therefore it may be advantageous to collect only the data that is useful for the

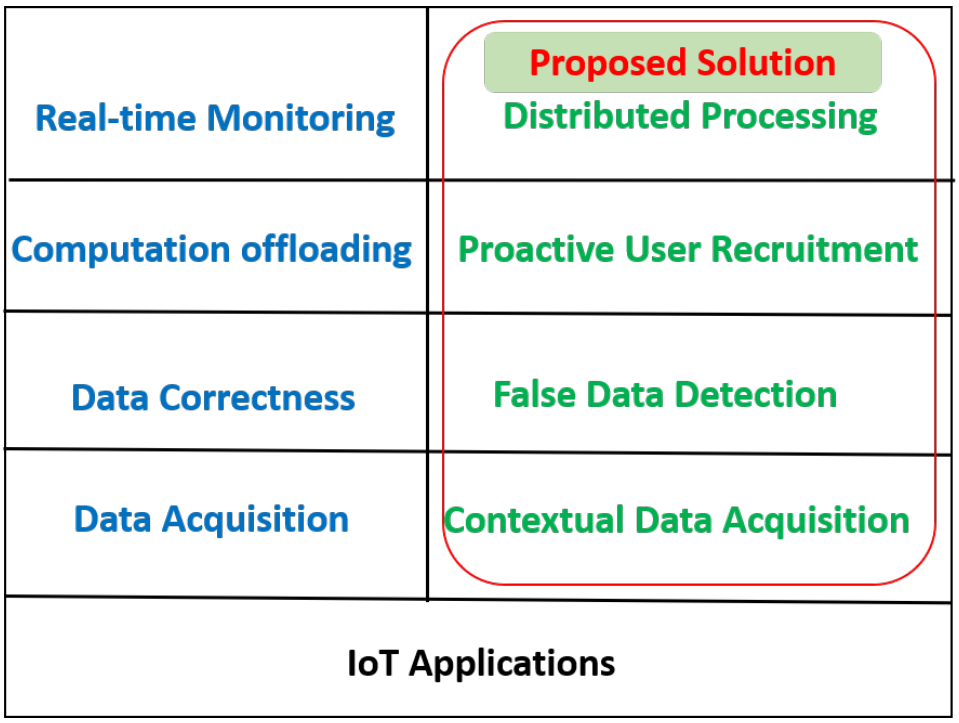


Figure 1.2: Overview of the Proposed Research.

application. To address this, proposed system introduces contextual data acquisition. *Contextual data acquisition* selects only the devices that match the required context of the application. On the other hand, data in uncertain environments are at high risk of being corrupted due to false data injection (FDI) attacks. To withstand FDI attacks and ensure data correctness proposed research integrates false data detection (FDD). FDD methods might increase the computational complexity of the application and make it challenging to meet the low latency constraints of the application. *Parallel processing methodologies are adopted to address the overhead of the computational complexity and ensure the reliability of the system.*

Real-time monitoring applications of the IoT environments exploit computation offloading. Computation offloading in this research is performed on the nearby volunteer computing devices. Volunteer computing devices are dynamic in nature and therefore

may face the challenge of tasks disruptions. Task disruptions halt the application tasks and result in incomplete tasks. *Proactive user recruitment* of the proposed research aims to handle the task disruptions and improve the success rate of task completion. Traditional user recruitment methods consider the current availability of the devices and offload computation on the available devices. Dynamic device usage patterns, mobility, and intermittent network connectivity may result in unavailability of the device thereby halting task execution. Proactive user recruitment approach is designed to select an optimized set of devices that are likely to complete the task and improve the reliability and task completion probability.

To, address high responsiveness and scalability of the large systems, proposed solution adopts a distributed approach. Although distributed solution is most popular approach to achieve scalability and reduce computational burden it often results in increased communication cost of the system. To mitigate the communication cost, the system is divided into sub-regions such that there is minimum connectivity between the sub-regions. Identifying a distributed solution with reduced communication cost is designed as a graph partitioning problem. This solution also includes a novel optimization technique to minimize communication cost. Linear state estimation of large-scale smart grid systems with stringent time constraints (typically in the order of milliseconds) is considered to present the challenges associated with the reliable distributed management in uncertain environments. A container based emulator is designed to depict distributed management in large scale IoT environments. Main contributions of this research can be summarized as follows:

1. A novel contextual data acquisition and the integration of false data detection focus on enhancing reliability of the applications.

2. A novel probabilistic model for resource and device availability prediction.

3. Proactive user recruitment module is designed to improve the success rate of the task completion.

4. A novel distributed management of volunteer compute resources enables real-time monitoring and support highly responsive applications.

5. A container based test bed to evaluate distributed management in large scale systems.

Chapter 2

RELATED WORKS

In this chapter, I present some of the prominent research works in the areas of edge computing, mobile crowd sensing and computation offloading that are important aspects of the proposed research. I discuss about the works that are directed towards real-time monitoring of large scale systems. Additionally, I present case study applications evaluated in this research: real-time person tracking, 360°video construction of a stadium event and linear state estimation of power grid systems. These applications are used to demonstrate the affects of key challenges on the IoT environments.

2.1 Edge Computing

Edge computing paradigm aims to move computation and storage closer to the data sources [Shi *et al.* (2016); Satyanarayanan (2017)]. Edge computing is gaining acceptance due to low latency, high availability and low cost. Real-time and collaborative applications like grid-monitoring, oil plant monitoring, real-time traffic monitoring, connected railways and smart grid applications can leverage edge computing for faster response services. Applications such as autonomous driving [Yuan *et al.* (2018)], vehicle to vehicle communications [Zhang *et al.* (2017)], disaster management [Sapienza *et al.* (2016); Higashino *et al.* (2017)] have exploited edge computing for experiencing reliable real-time monitoring.

Sensors deployed in IoT environments collect data at a very high measurement. For example, phasor measurements units (PMU's) installed on the power grids collect data at 30Hz. Another example for the sensor with high data rate is the mobile

phone camera. Continuous data collection provides a platform for developing real-time monitoring applications. Processing this data on the edge environment can help in achieving real-time monitoring.

Researchers have developed different types of edge computing solutions. Mobile Edge Computing [Sapienza *et al.* (2016)], Micro Clouds [Wang *et al.* (2017)] and Cloudlets[Satyanarayanan *et al.* (2009)] are some of the popular edge computing approaches. Edge computing solutions are designed to address various objectives like data management [Song *et al.* (2017)], energy management [Hou *et al.* (2018)], mobility management [Chen and Tsai (2018)].

2.2 Mobile Crowd Sensing

Mobile Crowd Sensing (MCS) has gained significant attention in recent times due to its capability to obtain contextual information without the need for additional application specific infrastructure. MCS is ideal for illustrating the dynamic availability of sensing devices. Availability of devices is uncertain for various reasons such as device owner mobility, dynamic resource usage, low power on devices and intermittent network connectivity.

Collaborative computing applications such as missing child [Satyanarayanan (2010)], real-time traffic monitoring [Hull *et al.* (2006)], participatory news reporting [Chen *et al.* (2016)], environment monitoring [Saremi *et al.* (2016)] and health care[De Rolt *et al.* (2016)] exploit MCS for contextual data acquisition. MCS applications can assist in real-time monitoring of the infrastructure. MCS applications listed in Table 2.1 have different contextual requirements and operate in real-time. Real-time operation of an application depends primarily on its computational complexity. Some of the MCS applications, such as person tracking presented in [Shi and Jia (2017)] involves

Table 2.1: MCS Applications.

Application Features				
Application Name	Contextual Requirements	Sensed data	Computation Requirements	Realtime Constraints
Traffic Monitoring [Hu *et al.* (2015)]	Location,	Accelerometer, GPS (≈30KBps)	Feature classification for stops and driving pattern, traffic forecasting	≈82s (minutes) [Goh *et al.* (2012)]
Event Summarization [Chen *et al.* (2016)]	Location, Orientation	Images, Videos (≈MB)	Detect event highlights triggered by increased photo capture activity	Post event summary (≈ hours)
Group Event[Bao and Roy Choudhury (2010)]	Location, Movement, Audio	Images, Videos (≈MB)	Event highlight identification based on audio changes	Post event summary (≈ hours)
Disaster Monitoring [Wu *et al.* (2016)]	Location, Accelerometer, Gyroscope.	Images (≈MB)	Maximum coverage analysis of destruction	Set by rescue personnel (≈minutes)
Person tracking [Shi and Jia (2017)], ContextAiDe	Location, Orientation, Accelerometer	Images (≈ few MB)	Estimate next location of person/perpetrator.	(≈few s)

image/video processing tasks which are computationally complex. These tasks need to process within the real-time constraints of the application. Uncertainty of computing device availability may have a huge impact on the real-time execution of the application. Therefore, it may be essential to have an optimized set of computing devices to accomplish real-time application execution.

2.3 Computation Offloading

Computation offloading augments constrained resources of IoT devices by delegating the compute tasks to external devices. Computation offloading can be used for energy saving, improve application performance and augment data storage. MAUI [Cuervo

et al. (2010)] and CloneCloud [Chun *et al.* (2011)] maximize energy savings on mobile device through code offload. CloneCloud [Chun *et al.* (2011)] uses VM images to optimize energy consumption of mobile device. Tao *et al.* (2018), use fine grain offloading to reduce energy consumption and improve application performance.

Performance of the offloading approaches can also affect the application performance. To handle this, researchers have investigated methods to improve offloading performance [Wu *et al.* (2017); Chen *et al.* (2018)]. Different type of mobile applications rely on computation offloading for executing complex tasks. For example, Sadeghi *et al.* (2016) offload complex tasks of signal processing to save battery life of the mobile device. Chang *et al.* (2017) have observed significant performance by offloading linear regression model training to edge servers. Some works have explored offloading neural network model inference [Ran *et al.* (2018)] and deep neural networks between fog devices and edge server [Teerapittayanon *et al.* (2017); Hu *et al.* (2019)]. For many years, researchers have been using computation offloading to volunteer devices (desktop clients) for solving complex problems. Seti@home [Anderson *et al.* (2002)] has laid the foundation for computation offloading by using the idle resources on volunteer devices for performing search for radio signals from extraterrestrial civilizations. Folding@home [Larson *et al.* (2009)] has been offloading computation to volunteer devices for simulating protein folding. Computation offloading and volunteer computing can prove to be an effective solution for disaster management. To deal with recent outbreak of COVID-19 by understanding the dynamics of the COVID-19 proteins, researchers have called for volunteer compute resources for executing offload workloads [Chodera (2020)].

In this research, I intend to utilize volunteer computing devices mainly mobile phones available on the edge environment to offload computationally complex tasks such as face detection and data intensive tasks such as linear state estimation to the edge environment.

2.4 Case Studies

2.4.1 Real-time Perpetrator Tracking

Th real-time perpetrator (i.e., person/s responsible for causing harm to the public) is used to illustrate the challenges of uncertain environments. Real-time tracking and identification of the perpetrator can not only lead to the capture of the individual sooner but may also reduce casualties. This application takes advantage of the widely available set of devices for obtaining data and executing the application tasks. Users available within a certain radius of the place of activity can be selected to perform the application tasks. Image collection, face detection, and face recognition are main tasks of the applications that can be performed by the selected devices.

Image collection is a sensing task, face detection and face recognition are compute tasks. Time constraints of this application tasks are relatively low in the order of seconds. Application tasks requirements are very important to collect data that it is useful for the application and also to select suitable devices for offloading computation. Therefore, it may be essential to define the requirements of the application tasks for this example. Requirements of the image collection task are the location of the event, battery state, network connectivity, camera availability, accelerometer. Requirements of the face detection and face recognition tasks are network connectivity, battery state and compute resources. Real-time perpetrator example is designed as a mobile application and is executed whenever there is an event of interest. The application

identifies the location of event and image or silhouette as the initial input. It then requests information such as camera availability, accelerometer, device orientation, battery state, compute resources, location is requested from devices that are present nearby the initial location. The accelerometer is used to minimize movement artifacts in the image capture. Orientation denotes angle of the device. Location radius defines the search area of the perpetrator; it is the radial distance in meters from the location of the event. Figure 2.1 presents a clear view of these requirements.

Executing these application tasks is challenging due to high communication cost and computation complexity. Application tasks exhibit high communication requirements as they require images to be transferred to the cloud or edge devices at a very fast rate. Additionally, application tasks such as face recognition are computationally complex. The set of user devices may change rapidly following the change in location of the perpetrator, and the user recruitment algorithm has to be executed more often. In such a scenario, the operational overhead may interfere with the real-time operation of the application. A novel user recruitment approach is proposed to select an optimized set of devices and assign application tasks for execution. Some of the devices of the selected set are assigned sensing task, and some of the devices are assigned computing tasks. Devices selected for image collection captures and uploads pictures to the fog nodes or the volunteer computing devices selected. Fog nodes and volunteer computing devices use a silhouette matching algorithm to identify the perpetrator in the images. Details of the proposed approach are discussed in Chapter 5.

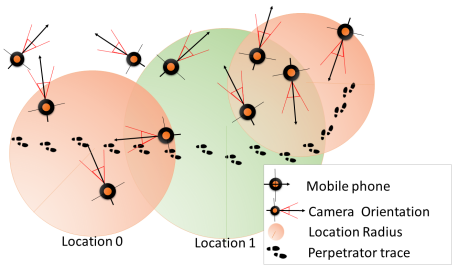


Figure 2.1: Perpetrator Tracking MCS App.

2.4.2 Real-time 360° Video of Stadium Event.

Smart Stadium where on entry all individuals are provided with a Google glass, which connects to their smart phones. The glass has capabilities to record their viewing space and stream it to their smartphones. A smart application can be envisioned where any individual can request a real-time 360°video feed of the playground. This video stream is composed by capturing video from strategic positions and stitching them in real time. Video capture and video stitching are the sensing and compute tasks of this application. Application requirements of video capture tasks is to location of the device in the stadium. The application selects candidate individuals primarily based on their location. The glass of each selected individual captures video snippets from their viewing space, and sends them to local devices for processing. Using distributed computation offloading a 360°frame which is further encoded into a video stream in a cloud server.

2.4.3 Smart Grid Systems

Smart grids systems is a well-known large-scale system with extensive usage. Efficient and effective real-time monitoring is vital for reliable grid management. State estimation is one of the critical processes for achieving the real-time monitoring of the grid.

State estimation has significantly evolved over the years and this evolution can be attributed to the type of sensing devices deployed for acquiring grid measurements. Traditional state estimation utilizes SCADA measurements collected from the grid to compute state vector. Deployment of phasor measurement units (PMU) on the grid has led to the initiation of Linear State Estimation (LSE). There has been considerable growth in the use of PMU data for wide-area monitoring of the grid [NAS (2016); Yang *et al.* (2007)]. For example, there are about 2,400 PMUs deployed in China, 2,000 in North America and 1,800 in India [Nuthalapati and Phadke (2015)]. Also, real-time grid measurements from PMUs facilitate dynamic failure handling and fault management [Wu and Giri (2006)]. LSE exploits the benefits of PMU data to present a dynamic picture of the smart grid. As complete observability is a prerequisite for LSE, LSE solvers need to process large sets of PMU data. For instance, an LSE solver for a grid with 100 PMUs and 20 measurements for each PMU at 30Hz sampling rate should handle over 10GB of PMU data in an hour [Patel *et al.* (2010)]. Also, LSE solvers operate under strict time constraints typically in the order of milliseconds (30 times a second) [Jones *et al.* (2013)]. Increase in the number of PMUs and the size of grid results in huge data making it challenging for LSE solvers. Another challenging factor of LSE is the communication latency introduced due to the geographical dispersion of PMUs. Installing additional communication infrastructures such as fiber optic networks and high compute capable infrastructures such as fast processors are a *possible* solution. However, this may incur large installation and operation costs.

Recently, researchers have been focusing on the use of inexpensive solutions such as cloud computing to mitigate the cost and to estimate the state of the power system in real-time. LSE is therefore ideal for discussing the problems that arise in real-time

monitoring of large-scale systems. LSE of smart grid systems can also be used to investigate the effect of data correctness on the reliability of the grid. Chi-square tests is an important tool for false data detection [Handschin *et al.* (1975)]. Researchers have proposed FDD approaches to mitigate the effect of security attacks on the state estimation process in SCADA based systems [Zhang *et al.* (2015); Liu *et al.* (2011)].

Chapter 3

SYSTEM MODEL

In this chapter, a detailed description of the devices and workflow that enables real-time monitoring of the IoT environments is provided. IoT environments include different types of devices: sensors, actuators, smart phones, laptops, servers, fog nodes. Figure 3.1 gives an overview of the hierarchical events in an IoT environment and introduces the proposed solutions.

3.1 Devices

1. ***Sensors*** measure physical parameters like temperature, pressure, human physiological signals etc., from the physical environment or a person or an object.

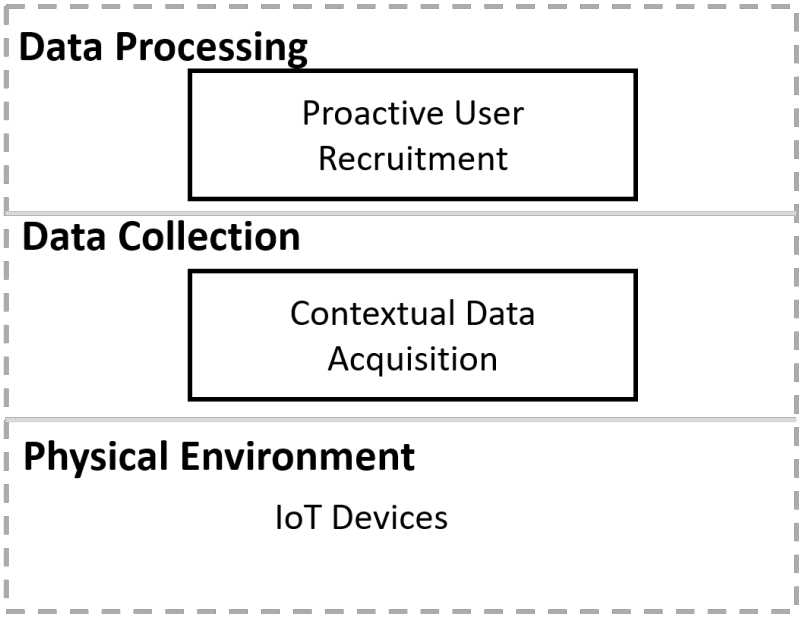


Figure 3.1: IoT Architecture.

Smart phones, ECG sensors, Brain sensors, surveillance cameras, thermostats are very few of the large number of sensors in use these days.

2. ***Actuators*** produce response signal(action) on the environment or on themselves. Examples of actuators include an alarm system, movement of robot.

3. ***Edge nodes:*** act as intermediary compute resources for data processing and communication. Smart phones, laptops and other volunteer devices are being used for data processing.

4. ***Fog servers:*** are powerful compute resources that are placed at the edge to mitigate the latency in data processing.

3.2 Data Collection

Numerous data collection techniques have been investigated to address the challenges introduced by constrained resources on the sensing devices, to reduce the communication cost and to improve the data quality generated by the sensing devices. Contextual data acquisition proposed in this research focuses on enriching the data quality. Contextual requirements of the application are defined and the data is collected only from the devices that match these requirements. This research uses mobile crowd sensing for collecting data, group of sensing devices present in same area can produce similar sensed data. Proposed data collection eliminates redundant data through optimized device collection. Contextual data acquisition not only improves data quality but also reduces processing of data that is not useful for the application.

3.3 Data Processing

Continuous data collection results in large volumes of data and the physical environments depend on powerful compute resources such as cloud servers for processing this data. In environments where the applications have very low latency constraints such as linear state estimation of power grid systems and where the compute infrastructure is sparse like disaster management scenarios dynamic infrastructures that are close to data sources are needed. This research aims at using volunteer computing devices to create a dynamic infrastructure at the edge for data processing. Given the uncertainty of device availability it is challenging to carry out reliable execution of the application. To address this we propose proactive user recruitment system.

Chapter 4

PREDICTIVE MODELING

As discussed in chapter 1 proposed research exploits volunteer devices for executing compute and data intensive tasks. Volunteer devices are dynamic in nature and can lead to uncertainty. Uncertainty in IoT environments may have severe effect on the application tasks completion. Incorporating predictive modeling can help in obtaining in device and resource availability. Device and resource availability information can be used to select devices for offloading application tasks. Predictive modeling uses mathematical models for predicting future time step values. Time series, statistical, probabilistic and machine learning models are some of the predictive modeling approaches. To predict device and resource availability, I analyze usage patterns such as battery level, network connectivity and human mobility patterns such as device location.

4.1 Usage Model Prediction

Location, network connectivity, and battery state resources are dependent on user mobility and device usage. LifeMap dataset is used to the evaluate resource prediction. LifeMap data is fine grained mobility data that is collected for a period of over two months [Chon *et al.* (2012)]. It has been observed that the device resources follow a regular pattern. Various mathematical approaches such as time series prediction, machine learning techniques, and Markov models [Gambs *et al.* (2012); Kostakos *et al.* (2016); Vanrompay *et al.* (2007)] can be

Table 4.1: States of Network Connectivity (ν) and battery (β).

State	Network State	Battery State	*Soc* range
ν_0	WiFi	β_0	$0 \leq Soc \leq 20$
ν_1	3G	β_1	$20 < Soc \leq 40$
ν_2	No Connectivity	β_2	$40 < Soc \leq 60$
		β_3	$60 < Soc \leq 80$
		β_4	$80 < Soc \leq 100$

used for predicting the contexts of location, battery, and network connectivity respectively. In this work, I have proposed combination of Markov model and time series prediction for estimating the resource values. Proposed prediction scheme uses a simple model that can capture different usage patterns changes in the day. These scheme runs continuously on the user devices. Resource prediction can improve user recruitment thereby improving the performance of applications.

4.1.1 Location Prediction

Analyzing user mobility patterns can help in learning about device availability in dynamic environments. Knowing about device availability can be useful for reliable execution of contextual applications. History of location data can be used to learn user mobility patterns. A transition matrix that consists of the recently visited locations of the user to understand user mobility. Proposed location model, uses five recent locations that form the transition matrix. GPS location data used from the database updates the transition matrix every 10 mins updating the probability of transition from the previous location to the current location. To capture the hourly pattern a separate transition matrix for

each hour of the day is maintained. This matrix updates the location transitions of the same hour every day and thus captures the daily pattern of the user. It also updates the transition matrix for frequently visited set of locations during the hour. Location for next time step is estimated using the transition matrix, and the estimated value is considered in the decision of user recruitment. Location for the next time step is defined as function of the hourly state and the current location of the state. Location prediction function is given by equation 4.1, where L_{t+1} is the device location for next time step, λ is the location state for the given hour and L_t is the current device location. Current location state is used to lookup the transition matrix and predict the next location state.

$$L_{t+1} = G(\lambda, L_t). \tag{4.1}$$

$$N_{t+1} = G(\eta, N_t). \tag{4.2}$$

4.1.2 Network State Prediction

Availability of network connectivity on the offload device has huge impact on the application task completion. Network connectivity can help to communicate results of the application task to the leader device or the server. Network states shown in Table 4.1 are used to build a transition model which is updated every 10 mins. A separate transition matrix is used for each hour of the day. These matrices help to capture the diurnal pattern of network transitions. Network state is predicted similar to the location state prediction. A matrix for the given hour and current network state is used to predict the network state in next window. Network state prediction is given by equation 4.2, N_{t+1} is network state for next time window, η is the network state for the given hour derived from the daily usage patterns and N_t is the current network state.

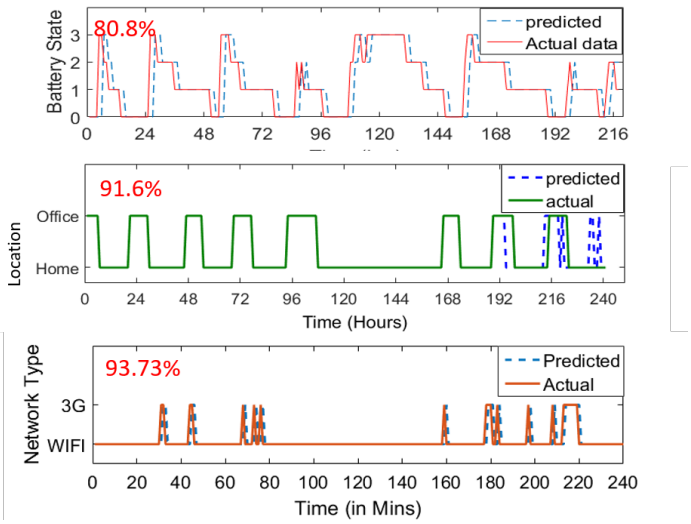


Figure 4.1: Resource Prediction Results.

4.1.3 Battery State Prediction

Prediction model uses different battery states in Table. 4.1 charging status of the battery for predicting the next state. Further in our model, a separate transition matrix are constructed for charging and discharging. Battery data updates the transition matrix at an interval of 10 mins. Current battery level and the transition matrix that is chosen based on the charging state of the battery is used to predict battery level in the next time step. Battery level at the next time step is the function of the current battery state and charging state of the device and is represented by equation 4.3. In equation 4.3, B_{t+1} is the battery state of next time step, B_C is 1 if the phone is connected to charging and 0 if phone is discharging, B_t is the battery state at time step t. B_t is used to look up the transition matrix to decide the next battery state.

$$B_{t+1} = G(B_c, B_t). \tag{4.3}$$

4.2 Results

LifeMap dataset is loaded on the mobile phones used in the experiments. Predicted values of the resources are used by the proactive user recruitment algorithm discussed in chapter 5 for optimally selecting the volunteer devices. Resource prediction can aid proactive user recruitment in selecting optimal offload devices. Optimal device selection can improve the task completion rate and also help in reducing energy consumption and data transfer. Results of these metrics are discussed in detail in proactive user recruitment evaluation. Figure 4.1 presents results of the resource prediction. Proposed prediction models have achieved an accuracy of 80.8% for battery state, 93.73% for network state and 91.6% for location.

Chapter 5

PROACTIVE USER RECRUITMENT FOR UNCERTAIN ENVIRONMENTS

Mobile crowd sensing and volunteer computing systems are highly dynamic and may result in uncertainty of device availability. Mobility of device owners, low power on devices, dynamic resource usage and intermittent network connections all contribute to the unavailability of devices. Facilitating real-time operation in such uncertain environments require careful optimization of different components of applications such as computing, sensing, communication distribution, and fault tolerance. This research focuses on the design of optimized user recruitment (device selection) for sensing and computing task called proactive user recruitment. Fundamental objective of proactive user recruitment is to improve data relevance and success rate of task completion. Proactive user recruitment is designed as a part of ContextAiDe architecture [Pore *et al.* (2018)]. Figure 5.1 presents ContextAiDe architecture.

ContextAiDe architecture is a combination of API and middleware that facilitates:

a) specification of contextual and computational requirements of application tasks and task distribution among the volunteer devices, b) continuous context and resource monitoring on the volunteer devices, c) distributed execution of application tasks, and d) stochastic optimization of operational and computational overheads.

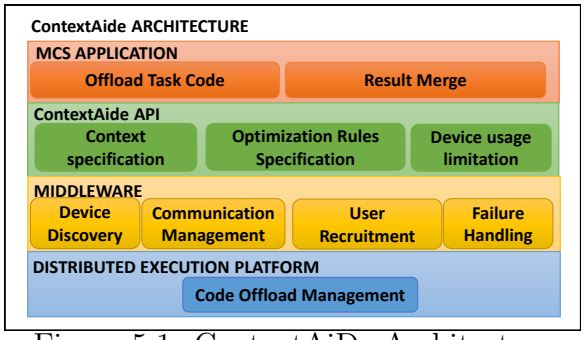


Figure 5.1: ContextAiDe Architecture.

5.1 User Recruitment

User recruitment is crucial for enriching the user experience and improving the success rate of task completion. Researchers have investigated different aspects of user recruitment that can aid in improving the performance of mobile crowd sensing and volunteer computing applications. Reddy et al. focus on time, location and user behavior for the selection of volunteer devices [Reddy *et al.* (2010)]. Liu et al. consider the available energy of the device in the decision of task assignment [Liu *et al.* (2017)].

CARDAP [Jayaraman *et al.* (2013)] uses activity recognition and energy efficiency for sensing task recruitment. A new DSE (Distance, Sociability, Energy) aware recruitment policy for distributed MCS monitoring apps with relaxed time constraints [Fiandrino *et al.* (2017)]. CATA scheme evaluates similarity in contexts required by applications while recruiting a device [Hassani *et al.* (2015)]. Proactive user recruitment selects an optimal set of devices that match contextual and computational requirements based on predicted resource availability. Device usage and human mobility patterns are analyzed for predicting resources. Further, proactive user recruitment approach iteratively revises selected set of devices with dynamically changing device availability.

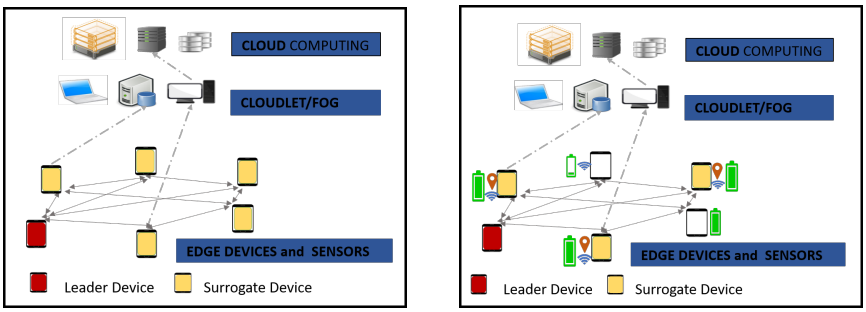


Figure 5.2: User Recruitment in (a)State of the art systems,(b)ContextAiDe.

Figure 5.2 gives an overview of user recruitment approaches of the state-of-art systems and proposed proactive user recruitment. From Figure 5.2, it can be seen that one of the devices in the edge environment is configured as a leader device. Leader device hosts application and devices that are ready to donate processing and sensing resources are configured as volunteer devices. User recruitment of the state-of-art systems recruit all the available devices while proactive user recruitment recruits only the devices that meet requirements of the application.

When a leader device is ready to request resources from the volunteer devices, it communicates through a common channel and establishes a one-one communication link with the volunteer devices. Smartphones, sensors, personal devices, and fog nodes may be the different types of volunteer devices available. Leader device requests for resource information from the available devices and recruits optimized set of volunteer devices that match the requirements of the application.

Application task type may be necessary for determining the selection of devices. Smartphones and sensors may be selected for data acquisition. Smartphones or personal devices may be selected to offload low latency and less complex tasks. Fog nodes may be utilized for processing delay tolerant or compute intensive or computationally complex tasks.

5.2 Proactive User Recruitment

Proactive user recruitment approach is designed as a two stage model. As the availability of resources on volunteer devices is associated with some uncertainty, stage 1 uses expected values of resources to choose the optimal set of devices. User recruitment decision is revisited in stage 2 based on the real-time resource values available on the volunteer devices.

Stage 1: This stage determines the optimized set of volunteer devices using stochastic models. For this, user recruitment approach computes expected values of the resources based on usage history and mobility of the user that are discussed in chapter 4. Expected values are used to reduce the failures in case of resource unavailability caused due to mobility, network connectivity and battery level on the device. Expected values are computed using the prediction models discussed in chapter 4. In this stage, as an initial step, all the devices with expected values matching the application requirements are selected. To determine the optimized set of devices from the selected devices that maximize the success of task completion proposed approach considers the number of volunteer devices to complete the task and time constraints of the task.

Stage 2: In this stage, the optimization decision of stage 1 is refined based on

the current resource values available on the devices. The objective function for device selection is the same as stage 1. However, during this stage, optimized set of stage 1 is validated with the correct resource values on the selected set of devices. If all the devices in the selected set are available and executing the offloaded tasks, then there may not be any task reassignment as it incurs additional overhead. However, if one or more devices of the optimized set become unavailable, then the user recruitment approach tries to find a new set of devices that match the application requirements.

5.2.1 Problem Formulation

This section presents an overview of the formulation of optimization followed by proactive user recruitment to select the optimized set of devices for executing the application tasks in uncertain environments. Volunteer devices that are available for performing application tasks are represented by set X as in Eq.5.1 where, $N = |X|$. A device x_i from set X may or may not be selected by the proactive user recruitment based on the required resource availability.

$$X = \{x_1, x_2, x_3, \ldots, x_N\} |\ x_i \in \{0, 1\}. \tag{5.1}$$

As presented in chapter 1, application requirements are the basis for the solutions proposed in this research. In this work, application requirement is referred to as context and is given by Def. 1. Collection of multiple contexts associated with the device is referred to as a **Context Set** and is represented using Ω.

Definition 1 ***Context:*** *Context is a key-value pair $\{\psi, C\}$ where ψ is an attribute/resource of the mobile device, and C is a value taken by the attribute ψ on the device.*

Given a request for S volunteer devices for executing the application tasks, the user recruitment approach first selects N' devices that match the requirements of the application ($N' \leq N$ and $N' > S$). This set is denoted by X_H. Proactive user recruitment approach selects devices that match the application requirements. Application requirement matching and device recruitment are evaluated using a context matching index (Def.2) and context sense index (Def.3).

Definition 2 ***Context Matching Index:*** *Context Matching Index $\mathcal{I}$, is defined as the ratio of context distance of a given context to the acceptable deviation ϵ.*

$$\mathcal{I} = \frac{\delta(C_d^p, C_r^p)}{\epsilon}. \tag{5.2}$$

Definition 3 ***Context Sense Index (CSI):*** *Context Sense Index for a mobile device is defined as the sum of weighted context matching index for each context C_d^p in Ω_d.*

$$\Upsilon_d = \sum_{k=1}^{K} w_k.\mathcal{I}_k, \tag{5.3}$$

where w_k is the weight associated with the context $\{\psi_k^p, C_k^p\}$ and is used to set priorities for the context. Sum of the weights for a given context set is 1.

Proactive user recruitment aims to select an optimized set of devices from the set X_H using two-stage optimization: **Stage 1:** As mentioned in section 5.2, Stage 1 determines the optimized set of volunteer devices that match the application requirements by using stochastic models of the historical trends for resource and operational overheads. Proactive user recruitment approach assumes that device selection for performing application task is associated with a penalty of

P_i, given as:

$$P_i = \Upsilon'_i \textbf{ s.t. } \{ \delta(E[r_i], C_r) \leq \epsilon \; \forall \; C \; \subset \; \Omega' \},$$
$$P \gg 1 \; Otherwise. \tag{5.4}$$

where $E[r_i]$ and C_i^r are the expected and current values of resource r on device i. The objective function for finding the optimized set of volunteer devices is formulated as follows:

$$min \sum_{i=0}^{N'} x_i.P_i, \quad x_i \in \{0,1\}, \; x_i \subset X_H \tag{5.5}$$
$$\textbf{s.t.} \qquad \sum_{i=0}^{N} x_i \geq S$$
$$\tau_i^s(t) + \tau_i^t(t) < \tau^T$$
$$R_i(t) - R_i^\Theta(t) > R_i^T.$$

where P_i is the penalty associated with the selection of device as given in Eq. 5.4, S is the number of devices required for executing the application tasks. $\tau_i^s(t)$ is time taken for task completion and $\tau_i^t(t)$ is the data transfer time and τ^T is the execution time constraint. Both $\tau_i^s(t)$ and $\tau_i^t(t)$ represent the time varying operational overheads. Such overheads also vary for different volunteer devices and can be application specific. A data-driven approach is utilized for modeling these overheads and is discussed in detail in section 5.4.3. R_i is the current resource availability, and R_i^Θ is the amount of resource usage on the volunteer device. R_i^T is the limit set on the resource usage by the device owner. R_i^Θ is obtained from observing the previous usage history of the person. Stochastic models are used to estimate resources R_i^Θ and are discussed in more detail in section 4.1.

Stage 2: In this stage, the optimization decision of stage 1 is refined based on the actual context and resource values available on the devices. If the resources or context on one or more devices in the optimized set of stage 1 become unavailable, proactive user tries to find a new set of devices. Unlike stage 1 of optimization where we use the expected values in stage 2, we use the actual resource values. The new optimized set of volunteer devices is given by X'_H from the set X_H using the current resource and context values on the device for assigning the application tasks. The penalty for a device in stage 2 optimization is given by:

$$\begin{aligned} P_i &= \Upsilon_i \ \mathbf{s.t.}\{\ \delta(C_i^p, C_r) \leq \epsilon \ \forall \ C \ in \ \Omega\}, \\ &= P \gg 1 \ Otherwise. \end{aligned} \tag{5.6}$$

where C_i^p, C_r are the current and required value of resource on device i and the objective function is given by

$$min \sum_{i=0}^{N} x_i.P_i, \mid x_i \in \{0,1\}, \tag{5.7}$$

where x_i is device selected for application task and P_i is penalty associated with device i.

In addition to application requirements, the proposed user recruitment approach also considers user-defined preferences for selecting devices.

5.3 Implementation

In this section, I present implementation details of the proactive user recruitment approach and MCS application integration into ContextAiDe architecture. ContextAiDe architecture is designed as an Android application. Application

requirements, optimization rules and time constraints of an MCS application are specified. Code offloading is used for distributing the sensing and computing tasks. All participating devices have the ContextAiDe app installed. The device hosting the MCS application is configured as a leader. As mentioned in 5.2 user recruitment module on the leader device discovers available devices and recruits optimized set of volunteer devices for executing application tasks. Real-time perpetrator tracking application discussed in section 2.4.1 is used to illustrate the working of the proactive user recruitment approach.

5.4 User Recruitment Components

In this section, the modules that are used by the proactive user recruitment algorithm to obtain an optimized set of devices for executing application tasks are discussed in detail.

5.4.1 MCS Application Requirements

Context and resource requirements of MCS application can be specified through API. Requirements of the real-time perpetrator tracking application are GPS location, orientation, battery level, availability of WiFi, user activity (stationary, moving). Leader device requests contextual and resource information from the nearby available devices connected to it. Exchange of Contextual and resource availability information between the volunteer and leader is designed using a serialized object.

Optimization Rules: enable to select the devices that are within a certain radius of the perpetrator. The real-time requirements such as execution time and device usage constraints are specified in the optimization rules.

Volunteer Device limits: Device owner can set limits on the resource usage. User preference limits set on the resources such as battery level, data usage, and sensors are considered in runtime while matching of context on the device and execution of the task.

5.4.2 Middleware Functions

Volunteer device discovery discovers nearby available devices and is accomplished through the publish-subscribe model. PubNub Java API is used to implement volunteer device discovery. All the participating devices subscribe to a common channel. Whenever app request is initiated, the leader device writes the IP address to the channel. The volunteer devices read this message, and they connect to the leader device over a TCP socket connection.

Execution Module: Leader device sends the DEX of the sensing and processing task to the devices selected by the user recruitment algorithm. Volunteer devices start the sensing task by invoking senseData() method. Data collected is processed locally invoking processData() or it may be sent to the processing/fog/cloud server. In perpetrator tracking example, the sensing task involves taking pictures in a fixed time interval. The processing task on mobile includes detecting images with faces while the fog server task is recognition of the perpetrator.

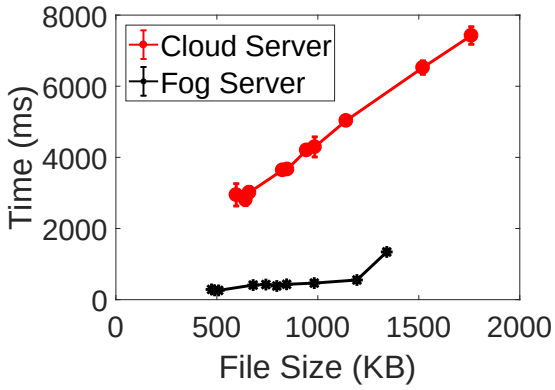


Figure 5.3: File Transfer Time

Optimized User Recruitment Algorithm The first step in the user recruitment process is to obtain the contextual information of the connected volunteer devices. In this step, volunteer devices find the expected values for the user dependent contexts such as the battery (dependent on usage patterns) and the location and WiFi (dependent on user mobility). The expected values are based on a prediction window of 10 minutes as determined by our usage model. For contexts that are not user dependent such as camera availability volunteer device obtains the current value.

As the execution progresses, on each device if the context deviation or resource usage exceeds beyond limits or the context or resources become unavailable, stage 2 is initiated. Even though stage 1 optimization decision is based on 10 minutes window, in stage 2, optimization is performed with updated actual context values. Optimization algorithm ensures continuous availability of required context on the devices that are recruited. In both stages, the resource limitations and real-time constraints are obtained by profiling the tasks, data transfer delays (see section 5.4.3).

5.4.3 Profiling MCS Application Tasks

This section profiles the sensing, communication, and processing time for Real-time perpetrator tracking example. The cloud VM is set up in Google Cloud with 8vCPUs and 32 GB memory. A fog server is setup on the local network with Intel i7 Desktop 3.5GHZ Quad-core CPU, 16GB RAM.

Profiling File transfer time One of the standard methods to upload camera images to the server is saving the file on the device before sending it to the server using multipart file upload. Figure 5.3 shows the time incurred to send the original camera images to the Cloud and the Fog server. Different file sizes are obtained by using multiple resolution setting of Camera to capture images ranging from 500KB to 2MB which are sent using lossless compression. File transfer time is defined as the time between the start of data transfer and receipt of the acknowledgment.

Profiling Data Sensing Time The time for sensing is noted on the Android device from the receipt of sensing request to the time data is available on the device for transfer. For different file sizes, the sensing time was observed as 2203.9 ± 397 ms.

Profiling Execution Time of MCS Tasks Execution of the perpetrator tracking MCS application involves running of face detection application on the mobile device and face recognition application on the Fog/Cloud server. The time for running the MCS task is time between the start of processing the file and availability of results. Average time incurred for varying file sizes is 1161 ms with a standard deviation of 331.69 ms for face detection task while face recognition task requires average 860 ms with a standard deviation of 101.8 ms.

For evaluation run (Section 5.5.3), we estimate the time starting from request for image capture to the time when results of face recognition are generated as a combined time for sensing and data transfer time for a single image. Time incurred using the file saving method is estimated to be 8.6s for cloud and 4.71s for Fog server processing.

5.5 Evaluation and Results

Proactive user recruitment approach is evaluated using Real-time perpetrator tracking application on an edge environment that includes a small group of mobile devices, local fog server, and a cloud server. Evaluation run uses two execution environments: 1. It runs on the actual set of mobile devices described in this section and 2. It is a simulated set of mobile devices that use historical user data of LifeMap data set collected in a free moving environment [Chon *et al.* (2012)] in section 5.6.

5.5.1 Experimentation Setup

For experiment evaluation, we used 14 mobile devices, one notebook, a fog server, and a cloud server. Specification details of these devices are listed as follows.

(a) 6 One Plus One phones, Qualcomm Snapdragon 2.5GHz Quad core CPU, 3GB RAM, 64GB Storage, Android 5.1.1

(b) 4 Nexus 5 phone, Qualcomm Snapdragon 2.3GHz Quad core CPU, 2 GB RAM, 32GB Storage, Android 5.1.1

(c) 2 LG g2 phone, Qualcomm Snapdragon 2.26GHz Quad core CPU, 2 GB RAM, 16GB Storage, Android 5.1.1

(d) 1 Nexus 7 tablet, Qualcomm Snapdragon 1.5GHz CPU, 2 GB RAM, 32GB Storage, Android 5.1.1

(e) 1 Moto G5 plus phone, Qualcomm Snapdragon 2.0GHz CPU, 4 GB RAM, 64GB Storage, Android 7.0

(f) Fog Server: Intel i7 Desktop 3.5GHZ Quad core CPU, 16GB RAM

(g) Cloud server in GoogleCloud with 8vCPUs and 32 GB memory

All the mobile devices have ContextAiDe App installed. Real-time perpetrator tracking app is offloaded to the volunteer devices. One of the devices acts as the leader device and initiates application task request by publishing its IP address on the ContextAiDe publish-subscribe channel.

5.5.2 Applications on Heterogeneous Devices

As listed in the experimental setup , ContextAiDe architecture uses different types of devices. From the section 2.4.1 we know that Image collection, Face detection and Face recognition are the application tasks of Real-time perpetrator tracking example. Based on the task type and the profiling results , image collection and face detection are assigned to mobile devices and face recognition is assigned to the Fog or Cloud server to process data from users in the smallest time to make the results available to the leader device. *Image Collection* task captures images at the rate specified in the runtime contextual requirements. *Face Detection* is invoked as soon as the file captured is saved and image data containing faces from the images obtained is sent to the Fog or Cloud. Open source python based face recognition library which internally uses deep learning models is used to recognize the perpetrator [King (2009)]. Using the model of perpetrator's face application task can detect if any of the volunteer devices captured the perpetrator's image.

Table 5.1: Perpetrator Location Tracking Results

Plot(XS,CA)	Value	Description
(a) tracked	Location	Location tracked by both methods indicates perpetrator tracked within search area
(b) 542,221	Radius	plot indicates search radius (here mean).
(c) 86%,31.7%	Devices	Number of devices (here percentage)
(d) 7.5MB,2.8MB	Data	Average data sent by sensing devices to fog server every minute.
(f) 59.6J, 22.6J	Energy	Average Mobile Energy Consumption every minute.
(g) 17.9s, 14.8s	Time	Average time incurred for processing a request.

5.5.3 Perpetrator tracking Evaluation

This evaluation is performed using 14 android devices available at a given location at the time of the event with changing context values. Initial reference location and search radius are provided at the beginning of the run. At a given location, proactive user recruitment optimally selects devices as volunteer devices that acquire images from a camera. MCS app components involve two processing components, 1. detecting images that contain faces (runs on a mobile device which captures images.) 2. recognize the face in the image sent (runs on the Fog server).

Once a perpetrator is detected, this app obtains the next reference location and adjusts the search radius based on movement of the perpetrator (See Figure 2.1). The devices are chosen optimally based on predicted context and available context abbreviated as CA. This approach is compared to existing strategy (XS) where devices are selected with optimal context but the search radius is fixed, and the next location is the last known location of the perpetrator. Figure 5.4 shows 30 mins run of tracking a perpetrator. Table 5.1 compares the perfor-

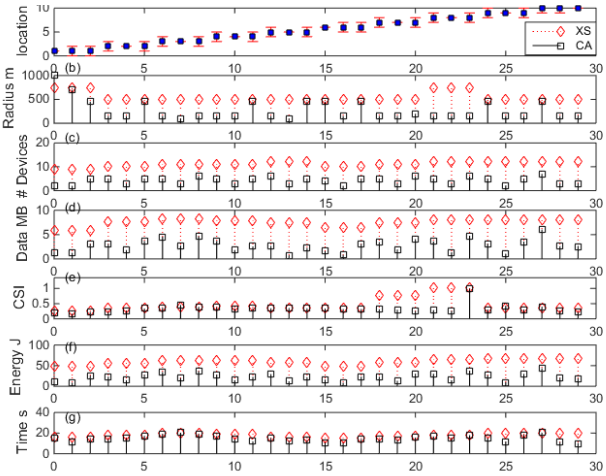


Figure 5.4: Perpetrator Tracking Scenario.

mance of contextAiDe and XS for a single request which tracks perpetrator through location 1 to the 10.

5.6 Proactive User Recruitment Evaluation

This evaluation is designed using real-life mobile device traces from LifeMap Dataset Chon *et al.* (2012). 10 requests of Real-time perpetrator tracking app of an average duration of 30 mins are generated to track a person moving in the campus area. In this evaluation proactive user recruitment is compared with user recruitment of prior research works to analyze for different performance factors like accuracy, energy savings, data transfer, and incomplete requests.

LifeMap Dataset: This dataset has sensor data acquired from 11 people on the campus of Yonsei University. Data from 11 users is obtained during different periods; we use data from 8 users for one month when data from maximum users are available. We obtain a data trace of GPS locations, battery level, connectivity, and user activity with a time granularity of 2 mins. The user

recruitment algorithm is evaluated to study different performance aspects like accuracy, energy savings, data transfer in the following section.

5.6.1 Strategies for User Recruitment

This section presents different user recruitment approaches.

1. Existing Strategy (**XS**): This user recruitment scheme is similar to Context Aware Task Allocation (CATA) Hassani *et al.* (2015) where context optimized selection is performed for each new task request. Next location is given by one of the devices that recognize the perpetrator while the search radius is fixed.

2. Ideal (**I**): This recruitment revises the selection decision by performing optimization at fixed time intervals to select optimally context matching devices. In the evaluation presented, the time interval is set for 2 mins. This recruitment scheme chooses devices with best CSI in every time step to provide optimal user selection.

3. ContextAiDe's Current Available Context (**CA**): This approach uses currently available context and resource values on the device. Optimization is performed for user recruitment when the required context becomes suddenly unavailable, or context request is initiated.

4. **ContextAide's Stochastic (CAS)**: This approach employs a proactive user recruitment scheme discussed in section 5.2. Stage 1 uses the predicted resource and context values of the device for offload decision. Context and resource values are predicted using a stochastic approach.

5. ContextAiDe's Cloud(**CAC**): In this approach, sensing tasks are assigned to devices that match application requirements, and the processing tasks are offloaded to cloud.

ContextAiDe's user recruitment schemes are designed to adapt to dynamically changing context requirements, e.g., location or search radius may change based on the movement of the perpetrator at runtime. New location and search radius are obtained based on the previous evaluation.

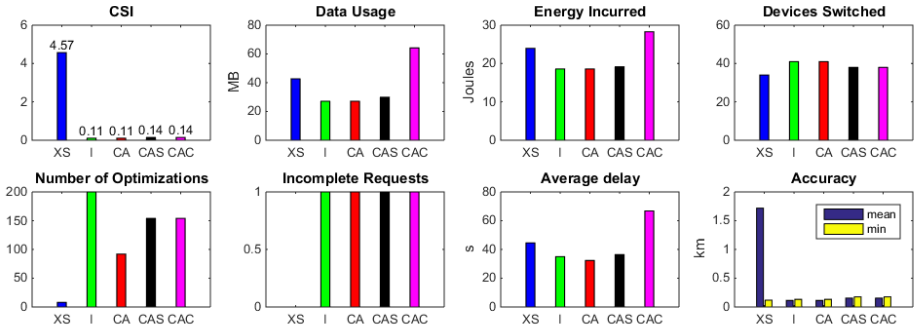


Figure 5.5: Performance of Proactive User Recruitment w.r.t. Other Recruitment Approaches.

5.6.2 Evaluation Using Data Traces.

ContextAiDe user recruitment strategies discussed in Section 5.6.1 are evaluated based on different parameters presented in this section.

(1.) CSI: Lower CSI value indicates that data acquired matches the required context.

(2.) Data Usage: Sensed data acquired on the mobile device is data sent to fog/cloud server for processing after local processing except in CAC where data is processed in Cloud. Amount of data transferred by different algorithms is compared.

(3.) Energy: Energy usage is estimated on the basis of data sensing, processing and data transfer that are executed on the mobile device (§5.5.3) while processing each application request.

(4.) Number of Optimizations: In a given sequence of application requests, optimization initiated depends on the user recruitment algorithm used. The number of optimizations indicate the operational overhead of the algorithm.

(5.) Devices switched: Sum of new devices that were recruited during execution.

(6.) Incomplete requests: Context-aware recruitment of devices causes some of the requests to be incomplete. The total number of requests that remains incomplete over the sequence of requests.

(7.) Delay: Average time incurred to accomplish the MCS task.

(8.) Accuracy: It shows the distance between the actual and the estimated location of the perpetrator

Figure 5.5 shows that CA successfully generates context relevant data shown by lower CSI and accuracy plot. CAS using predicted resources results in better energy and delay performance. Data used in CAS is 24.8% lesser, and energy is 37.8% lower than existing strategy (XS) for accurately tracking the perpetrator. Time incurred is improved by 43%. Performance improvement is achieved using CSI optimization indicated by the accuracy plot. Close monitoring of context and proactive decision based on expected context values results in savings of

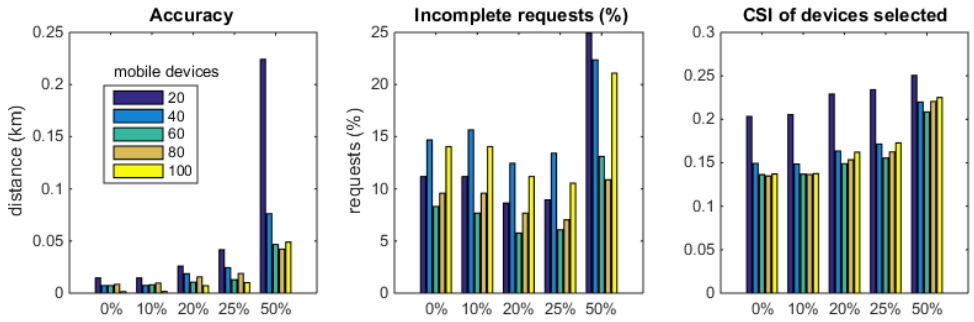


Figure 5.6: Performance of ContextAiDe in Varying Uncertainty Scenarios.

data usage, energy consumption and time incurred. However, this comes at the cost of accuracy. Minimum distance achieved in CA, CAS is a little more than XS and I strategy but the mean value of XS is much higher than CA/CAS. Alternatively, using the cloud server in CAC strategy incurs much higher energy (33%) and delay (50%) than CAS.

To evaluate the performance of CAS strategy, we set up a simulation with the different number of mobile users for executing Real-time perpetrator tracking. The mobility, WiFi, battery activity are modeled according to the LifeMap usage data. Performance of CAS is evaluated for variation in uncertainty that is associated with Contextual data. Figure 5.6 shows the variation in accuracy, percentage of incomplete request (due to unavailability of context) and average CSI which shows context relevance of data obtained. For lower uncertainty (<20%) accuracy is as desired less than 40 m. Percentage of requests abandoned, and CSI of devices selected remain steady for lower uncertainty (<25%). However, as uncertainty increases, higher values of CSI are noted for high at the cost of increased incomplete requests and lower accuracy.

5.7 Contact Tracing

In addition to the presented usecases context monitoring and contextual data acquisition of the ContextAiDe architecture can also be used for disaster management. In this research mobility patterns of the mobile device owners has explored for contact tracing of COVID-19 management. Contact tracing of COVID-19 management is defined as the process of identifying individuals who may have in contact with the infected persons. Mobility data may be used by COVID-19 contact tracing for drawing different insights. Insights considered in this research are as follows:

(a) Identifying if a individual has been in contact with infected persons.

(b) Identifying high risk locations.

(c) Understanding COVID-19 spread map.

The contact tracing usecase is designed as mobile application and workflow of the application is presented in the figure 5.7. Mobility data is organized in individual and aggregate granularity for identifying a person's contact with infected persons and for identifying high risk locations. Based on the user mobility patterns user receives an exposure alert or alert if the user is likely to visit high risk location. Further, the information collected is also used to analyze the COVID-19 spread. Three different data sets are used for this use case: 1. LifeMap dataset,2. COVID-19 cases dataset for India and 3. Google mobility data. LifeMap dataset is used to analyze mobility patterns of the user and predict if the user has visited any places as the infected personnel at the same time. This also helps in predicting if the user next visiting location is a

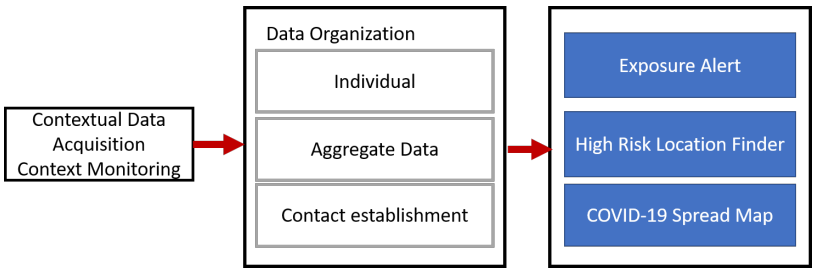


Figure 5.7: Context Monitoring for COVID-19 Management.

high risk location or not. COVID-19 cases dataset is used to understand the disease spread map of the given region. Google mobility data is an aggregate data that helps in finding places visited by large population at the same time. This places are marked as high risk locations. To understand the COVID-19 spread map graph $G(V, E)$ is constructed with the given dataset where the nodes(V) represent people contracted with COVID-19, edge is drawn between patients if the person had come in contact with the already infected person. Spread map has different metrics: lowest number of infections related to a person, maximum number of infections related to one person, maximum length of the contact trace. For a given COVID-19 cases dataset the contact trace path length is 5 and the maximum number of infections spread by single person is 56. This work has benefited from the context monitoring of the ContextAiDe architecture by sending alerts to the users based on the location prediction.

5.8 Conclusion

Proposed Proactive user recruitment approach can be used by different MCS applications. This solution improves the success rate of task completion and enriches the data obtained through crowd sensing. Results show that data con-

text optimization based on user recruitment designed reduced the data communicated by 25% while there is 43% improvement in time incurred compared to existing algorithms. The energy requirement is reduced by 37.8% while the improved contextual content seen by similar accuracy results for perpetrator tracking. Using continuous monitoring of the context on user devices proactive user recruitment adapts to the changes of context on devices and improves the selection of devices. Additionally, learning the behavior of usage dependent contexts and stochastic optimization gives a desirable performance considering smaller levels (<20%) of context uncertainty.

Chapter 6

IMPACT OF FALSE DATA DETECTION ON LATENCY SENSITIVE APPLICATIONS

This chapter outlines significance of data correctness on real-time applications. Linear state estimation of power grid systems is considered to understand the challenges of latency sensitive applications of large scale systems.

6.1 Linear State Estimation Formulation

In this section, the formulation of Linear State Estimation is presented. Linear state estimation is non-iterative estimator that uses time synchronized complex voltage and current measurements from PMUs to estimate the states of the grid. PMU based LSE of the power grid is given by the following equations [Jones *et al.* (2013)]:

$$\hat{\boldsymbol{x}} = \boldsymbol{H}\boldsymbol{Z}, \text{ where } \boldsymbol{Z} = \begin{bmatrix} V_m \\ I_m \end{bmatrix}, \boldsymbol{H} = (\boldsymbol{B}^T\boldsymbol{W}^{-1}\boldsymbol{B})^{-1}\boldsymbol{B}^T\boldsymbol{W}^{-1}. \tag{6.1}$$

In Eq.6.1, V_m and I_m represent voltage and current measurements, $\boldsymbol{Z}$ is called the measurement vector, $\hat{\boldsymbol{x}}$ is state vector, $\boldsymbol{B}$ is given by Eq.6.2 and $\boldsymbol{W}$ is the covariance matrix.

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{II} \\ \boldsymbol{yA} + \boldsymbol{y}_s \end{bmatrix}, \tag{6.2}$$

$\boldsymbol{II}$ is voltage incidence matrix, $\boldsymbol{y}$ and $\boldsymbol{y}_s$ are series and shunt admittance matrices. $\boldsymbol{A}$ is the current incidence matrix.

6.2 False Data Detection

In this research, a classical detector is used for incorporating false data detection into a linear state estimation process [Kosut *et al.* (2011)]. During an FDI attack, adversary alters grid measurements by adding an attack vector to the PMU measurements. Altered grid measurement vector can be represented as:

$$\boldsymbol{Z_a} = \boldsymbol{Z} + \boldsymbol{\alpha}, \tag{6.3}$$

where $\boldsymbol{Z_a}$, $\boldsymbol{Z}$ are modified and actual measurement matrices respectively. $\boldsymbol{\alpha}$ is the attack vector. The classical detector used to incorporate FDD is given by the following equations.

$$\boldsymbol{R_a} = \boldsymbol{Z'} - \boldsymbol{Z_a}, \tag{6.4}$$

$$\boldsymbol{J(\hat{x})} = \boldsymbol{R_a^T} \Sigma_e^{-1} \boldsymbol{R_a} \geq \boldsymbol{\tau}, \tag{6.5}$$

where $\boldsymbol{R_a}$ is residual vector, $\boldsymbol{Z'}$ is obtained from last state estimate, Σ_e is the covariance, $\boldsymbol{J(\hat{x})}$ is the detector and $\boldsymbol{\tau}$ is an acceptable threshold of the grid. False data detection prevents incorrect state estimate computation.

6.3 False Data Detection For Reliability In Uncertain Environments.

As stated in chapter 1, data correctness helps in improving the reliability of the application. Security attacks often hamper data correctness in uncertain environments. Like other uncertain environments smart grid systems also suffer from the security attacks. This work demonstrates the difficulties caused due to security attacks on PMU measurements in smart grid systems. During a security attack, attackers compromise the network and inject false data into the PMU measurements. False data injection (FDI) attacks may cause the Linear State Estimation (LSE) application to estimate incorrect state. LSE is a crucial

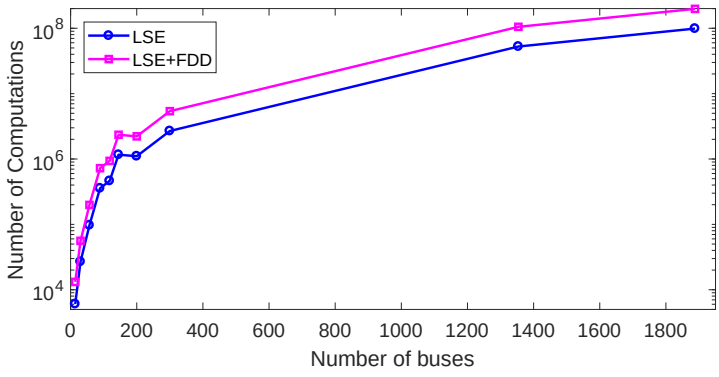


Figure 6.1: Number of Complex Number Computations for LSE and FDD.

application for reliable grid monitoring and planning. Incorrect state estimate may degrade the reliability of the power grid and result in sub-optimal power flow in the grid. Liu et al. show that FDI attacks modify the state estimation results and have a severe effect on the security of the power grid systems [Liu *et al.* (2011)]. Therefore, it may be essential to integrate false data detection (FDD) in the LSE. However, integrating FDD may increase the computational overhead of LSE thereby making it difficult to meet the real-time constraints for larger grid sizes. Figure 6.1 presents a comparison between the number of complex number computations of LSE and LSE with FDD for different grid sizes. This work uses the classical detector FDD approach described in[Kosut *et al.* (2011)] to detect FDI attacks on the linear state estimation.

Possible solutions that can be explored to minimize the overhead introduced by FDD are a) parallel processing and b) distributed LSE. Distributed LSE divides the grid into smaller substations to reduce computation time, aggregating the result in the control center and then performing FDD. The computation involved in the LSE problem, in general, may be scalable when implemented in

a distributed manner. However, in the context of PMU based state estimation, distributed LSE may require infrastructure upgrades and may have increased communication cost. Unlike the centralized LSE, distributed LSE may additionally, require two substations to exchange power flow data in one of the Tie lines connecting them Chatterjee *et al.* (2015). This results in increased communication in addition to the time taken to compute the Sub LSE. Additionally, it is observed that communication latency is exceedingly higher than the LSE computation time. For a distributed solution to be better than a centralized approach, we have to derive the optimal number of partitions or sub LSEs. Further, the partitions cannot be selected randomly. The configuration of each partition and the order in which sub-LSEs are evaluated affect the communication latency. Moreover, sharing power flow information between substations has serious security concerns, which needs to be addressed before a distributed implementation.

To solve computation overhead and achieve real-time monitoring, a novel Cloud hosted parallelized LSE-FDD (CLSE-FDD) solver that utilizes fast GPUs to scale with an increasing number of PMUs is proposed. Main contributions of CLSE-FDD are: *1. Analysis of the overhead of FDD on the LSE problem, 2. leveraging parallel processing of GPU to minimize the overhead of FDD and scale for large grid sizes, and 3. exploiting data level parallelism on the LSE and FDD operations.* CLSE-FDD is evaluated for different test systems using the CPU and GPU solvers. Results show that the GPU CLSE-FDD solver is up to 19x faster than the CPU CLSE-FDD for larger test system sizes. GPU CLSE-FDD application can easily scale in excess of 1,500 PMUs.

6.4 Cloud hosted LSE-FDD System Model

CLSE-FDD application aggregates data obtained from geographically distributed PMUs to construct the grid measurement vector. Grid measurement vector is sent to GPU for computing state estimate and detecting FDI attacks. CLSE-FDD considers following design assumptions:

(a) Each bus of the grid has a PMU installed that measures bus voltages and line currents. In practice, to achieve complete observability, all buses may not need PMUs. However, this assumption allows us to evaluate the worst-case computational complexity.

(b) PMU data measurement rate is 30 Hz hence introducing a real-time constraint of ≈33 ms for LSE.

(c) Communication latency for the CLSE-FDD application is assumed to be ≈10 ms[Maheshwari *et al.* (2013)].

6.4.1 Performance overhead of False Data Detection

To explain the overhead introduced by FDD on LSE, consider a power grid with $\boldsymbol{N}$ buses and $\boldsymbol{M}$ transmission lines. Every bus on the grid has one or more transmission lines and is associated with one voltage measurement. Each transmission line has two current measurements. It is also known that for the power grid $\boldsymbol{N < M}$. LSE of the grid is given by Eq 6.1, where $\boldsymbol{H}$ is a matrix of size $\boldsymbol{N \times 2M}$, and $\boldsymbol{Z}$ is the measurement matrix of size $\boldsymbol{2M \times 1}$. The complexity of the LSE($\hat{\boldsymbol{x}}$) computation, is $\boldsymbol{O(N.M)}$. Time complexities of individual operations of FDD are given in Table 6.1. From Table 6.1 it is

Table 6.1: Complexities of LSE & FDD Operations.

Operation	LSE	FDD
$\hat{x}$	$O(N.M)$	$O(N.M)$
R_a	NA	$O(M+N)$
R_a^T	NA	$O((M+N)^2)$
Σ_e^{-1}	NA	$O(M^3)$
$J(\hat{x})$	NA	$O(M)$

N= Number of buses and M=Number of branches.

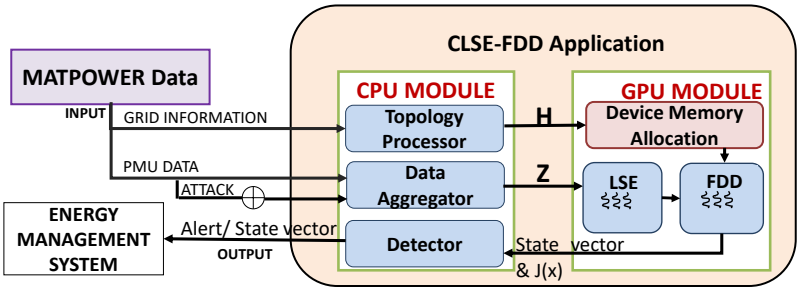


Figure 6.2: Architecture of CLSE-FDD.

evident that the complexity of LSE-FDD is more than LSE. This may inhibit the LSE application from meeting the strict real-time constraints of the state estimation process.

6.5 Architecture of CLSE-FDD

CLSE-FDD application architecture consists of two modules: the CPU module and the GPU module. Tasks that cannot be parallelized or the tasks that suffer from the overhead of CPU-GPU data transfer are executed in CPU module. Tasks that are compute intensive and can be parallelized are executed in the GPU module. These tasks leverage parallel processing techniques to meet the low-latency constraints of LSE and scale with increasing PMU installations. Detailed representation of the modules is shown in Figure 6.2.

6.5.1 CPU Module

Topology Processor processes grid configuration information and computes pseudoinverse matrix $\boldsymbol{H}$ in Eq. 6.1 that can be used in state estimation computation. It also sets $\boldsymbol{\Sigma_e^{-1}}$ that can be used for FDD.

Data aggregation task is executed every 33 ms after receiving the voltage and current data of PMU. This task is designed to process data and build measurement matrix $\boldsymbol{Z}$ of Eq. 6.1.

The *detector* determines the presence of FDI in PMU data and is invoked every time a new state estimate vector is available. Results of the LSE and FDD tasks of the GPU module serve as input to this task. The output of this task is sent to EMS.

6.5.2 GPU Module

Linear State Estimation is invoked every time a new measurement vector is constructed with a new set of PMU data. This task takes measurement vector and the topology processor results as input and computes state estimate vector $\hat{\boldsymbol{x}}$.

False Data Detection is executed after LSE task of the GPU module. Results of the topology processor ($\boldsymbol{\Sigma_e^{-1}}$, $\boldsymbol{Z}$) and LSE task ($\hat{\boldsymbol{x}}$) are used to compute FDD operations as described in Section 6.2. $\boldsymbol{J(\hat{x})}$, $\hat{\boldsymbol{x}}$ are sent to Detector task.

6.6 Implementation

Cloud hosted LSE is implemented as a 'C' application [Chakati *et al.* (2017)]. As state estimation involves the processing of a large number of measurements that are usually complex numbers, using regular math library may degrade the performance. To overcome this and handle complex numbers effortlessly we use GSL[Galassi *et al.* (2007)]. As a first step towards including FDD, Cloud hosted LSE application implemented in 'C' is extended to compute the FDD elements using GSL [Chakati *et al.* (2017)]. It has been observed that the execution time of the CLSE-FDD is increased by 2.4x on an average across different grid sizes as presented in Figure 6.4a. Increase in execution time may hamper the scalability of CLSE-FDD application. CLSE-FDD application execution time is minimized and scalability is achieved by utilizing the parallel processing capabilities of GPU. GPU module of CLSE-FDD application is implemented using CUDA. To efficiently process large sets of complex grid measurements on the GPU I use the CUBLAS library [NVIDIA (2017)]. Test data for different grid configurations listed in Table 6.2 is obtained from MATPOWER. Tasks described in Section 6.5 can be classified into two phases: 1) Initialization Phase and 2) Computation Phase. Figure 6.3 lists different computations performed in these phases.

6.6.1 Initialization Phase

This phase is executed first time the CLSE-FDD application starts or if there is a change in a grid configuration. Current incidence $\boldsymbol{A}$, admittances $\boldsymbol{Y}$, $\boldsymbol{Y_s}$ given in Eq. 6.2 are computed in this phase. Also, covariance matrices $\boldsymbol{W}$ and $\boldsymbol{\Sigma_e}$ that are used in computing pseudoinverse matrix $\boldsymbol{H}$ and FDD, respectively

are set in this phase. $\boldsymbol{H}$ and $\boldsymbol{\Sigma_e^{-1}}$ are copied to the GPU only once at the start of the process. To reduce memory allocation time, GPU memory for the matrices $\boldsymbol{Z}$, $\hat{\boldsymbol{x}}$, $\boldsymbol{R_a}$, $\boldsymbol{R_a}^{\boldsymbol{T}}$,$\boldsymbol{\Sigma_e^{-1}}$ is allocated only once at the start of the process.

6.6.2 Computation Phase

This phase is invoked for every new set of PMU measurements. PMU data obtained from MATPOWER is used to construct grid measurement matrix $\boldsymbol{Z}$. Pseudoinverse matrix $\boldsymbol{H}$ and grid measurement matrix $\boldsymbol{Z}$ are taken as input to compute the state estimate vector. $\boldsymbol{H}$, $\boldsymbol{Z}$ matrices are represented as one-dimensional row major order `cuDoubleComplex` vectors on GPU. CUBLAS library function `cublasZgemm` is used to achieve data level parallelism. `cublasZgemm` computes the state estimate vector $\hat{\boldsymbol{x}}$.

$\boldsymbol{Z}$, $\hat{\boldsymbol{x}}$, $\boldsymbol{\Sigma_e^{-1}}$ are used for performing FDD operations. Figure 6.3 presents the workflow of the CLSE-FDD application and demonstrates interdependency between the FDD operations. Interdependent FDD operations are executed sequentially on the GPU. However, individual computations on each of these operations are optimally parallelized by the cuBLAS routines. Detector vector $\boldsymbol{J}(\hat{\boldsymbol{x}})$ is communicated from GPU to CPU host. $\boldsymbol{J}(\hat{\boldsymbol{x}})$ is compared against the acceptable threshold set for the grid to detect an FDI attack. If $\boldsymbol{J}(\hat{\boldsymbol{x}})$ is greater than $\boldsymbol{\tau}$ threshold then attack is detected and the EMS is alarmed, otherwise $\hat{\boldsymbol{x}}$ is sent to EMS.

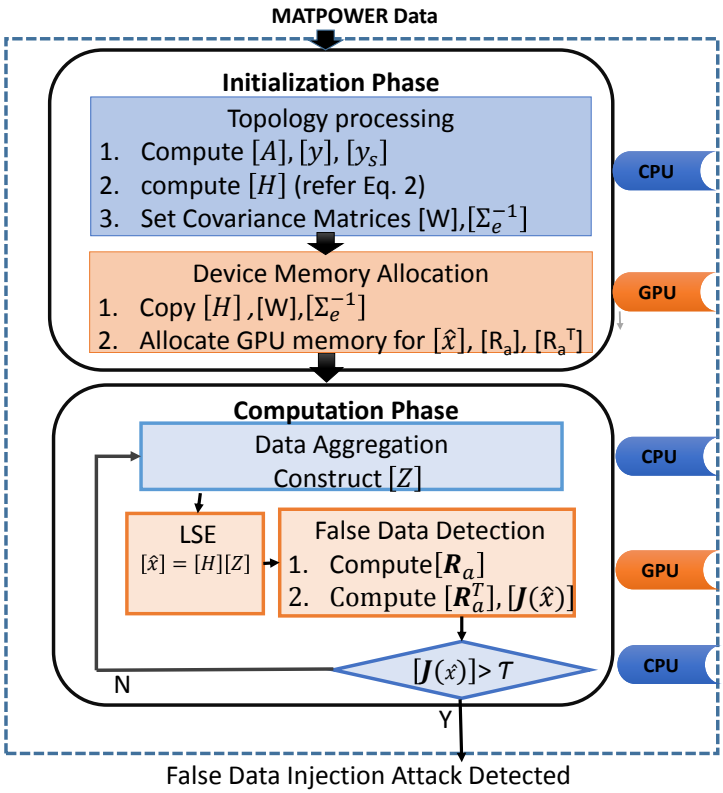


Figure 6.3: Workflow of CLSE-FDD Application.

6.7 Evaluation and Results

CPU and GPU execution times of the CLSE-FDD are compared and it is found that high data rates of PMU require the data to be processed by the CLSE-FDD in the order of few milliseconds. Time taken for LSE-FDD is given by $\boldsymbol{T}$ as shown in Eq. 6.6. Communication time is the time to send PMU data to PDC plus, the time to send data from PDC to CLSE-FDD application. Execution time is the time required to complete the LSE and FDD computations.

$$\boldsymbol{T} = \textbf{communication time} + \textbf{execution time}. \tag{6.6}$$

Existing research shows that communication time is the major setback for LSE solver. However, for larger grid sizes execution time can also become a bottleneck [Chakati *et al.* (2017)]. CLSE-FDD application is designed to address execution time bottleneck. As mentioned in Section 6.4 real-time constraint of LSE is 33 ms, communication time is considered to be ≈10 ms. Therefore, the execution time constraint may be approximated to be ≈23 ms.

6.7.1 Experiment Set up

CLSE-FDD solution is hosted on a server in ASU data center. CLSE-FDD described in Section 6.6 utilizes the GPU accelerator GTX 680 available on the server to achieve additional speedup using parallel processing. PMU data and grid configuration information obtained from MATPOWER are given as inputs to the CLSE-FDD application. CLSE-FDD application is evaluated for different test systems available from MATPOWER. These test systems present close representation of real grid environments. The test system sizes and the number of line currents on the test system are listed in 6.2.

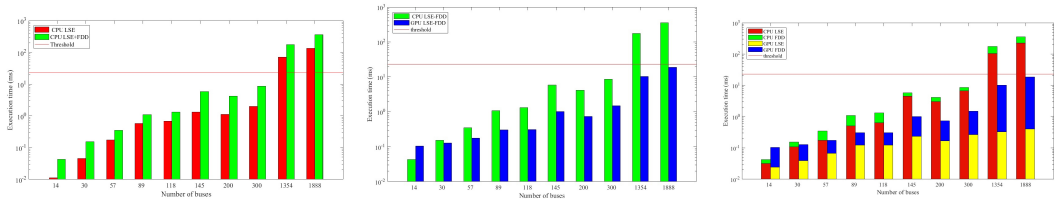
6.7.2 Results

Table 6.2 presents speedup of the GPU LSE as compared to the sequential execution of LSE for different test systems. From the data reported in Table 6.2 it may be also inferred that for larger grid sizes GPU-LSE gives speedup greater than 10x. As observed in Figure 6.4a, the CPU implementation of LSE-FDD shows an increase in execution time which may be attributed to the computational overhead introduced by the FDD. Speed up achieved by GPU implementation may aid in meeting the real-time constraints of LSE and LSE-FDD. From

Table 6.2: GPU-LSE Speedup.

Grid size (N)	No of Edges (M)	Speed Up of GPU LSE
14	20	2.447
30	41	1.59
57	78	4.359
89	206	6.734
118	186	7.943
145	429	4.032
200	245	4.957
300	409	5.283
1354	1749	11.591
1888	2330	11.835

Figure 6.4a it may be observed that the CPU implementation of CLSE-FDD can scale up to 300 PMUs on the grid. To scale for larger number of PMUs, the GPU implementation of CLSE-FDD may be more suitable. Figure 6.4b presents the execution time of LSE-FDD computation for the CPU and GPU implementations. For smaller test system sizes CPU CLSE-FDD outperforms the GPU implementation. However, as the test system size increases, GPU performance becomes better than CPU implementation. For larger systems, GPU implementation of LSE-FDD achieves a speedup of up to 19x (as observed in Figure 6.4c). Also, from the figure, it is clear that the GPU implementation of CLSE-FDD can easily support in excess of 1,500 PMUs installed in the grid. Figure 6.4c presents the execution times for CPU and GPU implementations of LSE and LSE-FDD for the different test systems.



(a) CPU 1.LSE, 2.LSE-FDD. (b) CLSE-FDD 1. CPU, 2.GPU. (c) LSE, LSE-FDD 1.CPU 2.GPU.

6.8 Observations

LSE-FDD detects the FDI attack and notifies the EMS to avoid the sub-optimal power flows in the grid. LSE-FDD implemented in cloud hosted CPU-GPU configuration can result in a speedup of 19x and can support 5 times more PMUs as compared to a CPU only configuration. However, a major bottleneck is the communication latency between CPU and GPU which needs to be further optimized. In this solution, I have demonstrated the significance and difficulties involved in integrating security measures such as false data detection. A GPU based cloud hosted LSE-FDD application is implemented and its advantages are presented.

Another potential approach towards fast LSE-FDD computation is to adopt a distributed architecture. A careful analysis shows that this approach can incur significant communication cost. Given a grid of size N, centralized LSE solver may require a total of (N+N/3) communications (N/3 is assumed to be the number of PDCs on the grid Jones *et al.* (2013), that is, three PMUs connect to one PDC). For a distributed LSE with K partitions (Sub LSE stations), the solver may require N+2K (N communications to send PMU data to Sub LSE stations, K communications between the Sub LSEs to exchange information about the Tie-Lines and K communications to send Sub LSE result to the con-

trol center that merges the Sub LSEs to obtain the final state vector). Finding the optimal partitions to improve communication cost in distributed LSE is an open research problem and is explored in the next solution of this thesis.

Chapter 7

DISTRIBUTED LINEAR STATE ESTIMATION

Achieving real-time monitoring in large scale systems is challenging due to increased computational burden and geographical dispersion of devices. Smart grid systems are spread across wide geographical area and include large number of devices that generate large volumes of data. Processing large volumes of data within strict time constraints may be difficult. This chapter presents an overview of the Distributed Linear State Estimator (Distributed LSE) that can aid in achieving real-time monitoring in smart grid systems. This chapter also discusses the design of power grid emulator using dockers to evaluate Distributed LSE for different test systems.

7.1 Distributed Linear State Estimation for Smart Grid Systems.

Traditional centralized LSE process deployed at the control center receives data from PMUs and compute the state estimate. To overcome the communication latency and meet the real-time constraints of LSE and to withstand the faster data measurement rates of PMU's centralized LSE systems utilize high-speed fiber optic networks. Increase in size of the power systems network and increase in the number of PMUs increases the computational burden of the LSE making it challenging to meet the real-time constraints. Also, from our earlier works, it has been observed that the communication latency is a significant bottleneck for the performance and scalability of LSE. Installing a high-speed fiber optic

network will result in huge infrastructure costs. To scale with large grid sizes, to minimize the overhead of communication cost and latency and to adhere to strict real-time constraints it may be feasible to propose a Distributed Linear State Estimator.

Reducing the complexity of the problem by dividing it into subproblems is one of the fundamental approaches of distributed solutions. However, this may increase communication cost between the distributed entities. Distributed solutions that incur minimum communication cost between the subproblems may offer better results than the centralized solutions. The primary objective of the proposed solution is to find optimal divisions of the grid that may be adopted by the Distributed LSE. To accomplish this, proposed solution uses grid partition strategies that address the issues of communication cost and optimality of the distributed LSE.

Design of a distributed solution generally incorporates a pre-processing step of dividing the centralized problem into subproblems. Many research works have devised the pre-processing step as graph partitioning problem. I follow a similar approach in the design of a distributed LSE by considering the smart grid network as a graph and partition it into sub-networks. These sub-networks are called as sub-LSE stations in Distributed LSE. As mentioned earlier, Distributed LSE may have increased communication cost. For example, partitioned linear state estimation proposed in [Chatterjee *et al.* (2015)] shows that sub-LSE stations should exchange the minimum power flow information in at least one Tie line connecting them to obtain complete state estimation of the grid. This may be a significant overhead for the adoption of Distributed LSE. For a Distributed LSE to surpass the centralized LSE, optimal sub-LSE stations that

have the minimum number of Tie lines needs to be derived. Main contributions of the proposed work are : 1. An offline grid partitioning algorithm that divides the smart grid network into an optimal number of sub-networks referred as sub-LSE stations that have minimum Inter Partition Communication (IPC), 2. Distributed LSE that utilizes the result of the partition algorithm to compute the sub-state estimates in parallel at the sub-LSE stations. Performance of proposed distributed LSE is compared with the traditional centralized LSE systems for different test systems.

7.2 Problem Formulation

Distributed LSE design is formulated as a graph partitioning problem. In this work, I aim to efficiently partition the smart grid into sub-networks to minimize inter-partition communication (IPC). Smart grid is represented as a graph $G = (V, E)$ where V is the set of nodes on the grid, E is the set of transmission lines on the grid. Smart grid graph G is divided into K subgraphs where each subgraph is completely observable and computes local state estimate. I_{ij} gives the IPC between the partitions P_i and P_j, objective of the partitioning is to reduce I_{ij} and is given by Eq. 7.1.

$$Minimize(\Sigma I_{ij}) \tag{7.1}$$

such that

$$\boldsymbol{V_i} \subset \boldsymbol{V}$$
$$\boldsymbol{E_i} = \{(\boldsymbol{v_i}, \boldsymbol{v_j}) | \boldsymbol{v_i}, \boldsymbol{v_j} \boldsymbol{\epsilon} \boldsymbol{V_i}\}$$
$$\boldsymbol{V_i} \leq \boldsymbol{c} \; -\forall i \epsilon 1, \ldots, k$$
$$\boldsymbol{I_{ij}} = \boldsymbol{\{(v_i, v_j) \mid (v_i, v_j) \epsilon E, v_i \epsilon V_i \& v_j \epsilon V_j\}}$$
$$\bigcup_{\boldsymbol{i=1}}^{\boldsymbol{k}} \boldsymbol{V_i} = \boldsymbol{V}$$

where V_i, E_i are the set of nodes and edges in subgraph P_i. I_{ij} is the numbers of edges of the graph that spans between subgraphs P_i an P_j. c is a constraint on the maximum number of nodes a subgraph V_i can hold.

7.2.1 Sub-Graph Size

As presented in section 7.2, c is the constraint on the size of the subgraph. Partitioning algorithm strictly enforces the constraints. Based on the observation from our previous work that the sequential execution of state estimation supports up to 300 PMUs, therefore, we set the constraint size c for the subgraph to be 300 [Chakati *et al.* (2017)]. For smaller test systems the constraint size c is determined based on the average degree($a\delta$) of the graph and is given by Eq. 7.2

$$\boldsymbol{c = \frac{N}{a\delta}}. \tag{7.2}$$

7.2.2 Communication Cost and Data Transfer

Distributed LSE is compared with traditional centralized LSE based on the following performance metrics: the size of the data transferred, communication cost and communication latency.

Size of data transferred is the total number of measurements communicated across all the levels of the LSE process. Data transferred in a traditional centralized LSE is the sum of data transferred between PMU and PDC layers and data transferred between the PDC layer and control center. For a smart grid of N nodes and M transmission lines, there are $N + 2M$ measurements. The size of data transferred from PMU to PDC layer is $(N + 2M)$, the same data is

transferred from the PDC layer to control center. Therefore, the size of the data transferred in a traditional centralized LSE is given by Eq. 7.3. Data transferred in a Distributed LSE can be defined as the sum of the data transferred from the PMU layer to PDC layer, data transferred between sub-LSE stations and sub-LSE stations to the control center. Size of data transferred from PMU to PDC layer is $N + 2M$, size of data transferred between sub-LSE areas is given by I. Total size of data transferred in a Distributed LSE is given by Eq. 7.4.

$$\boldsymbol{D_t = 2(N + 2M)} \tag{7.3}$$

$$\boldsymbol{D_d = (N + 2M) + I + N} \tag{7.4}$$

where I is the number of inter partition communications.

Cut Percentage is used to measure the performance of the partitioning algorithm and defined as follows:

$$\boldsymbol{cut = I/M} \tag{7.5}$$

Communication cost is the total number of communications required across different levels of the grid. This work considers following assumptions for traditional and distributed LSE:

(a) Each bus of the grid has a PMU installed that measures bus voltages and line currents.

(b) Traditional LSE considers that there are $N/3$ PDCs.

(c) Distributed LSE considers that the grid is divided into K partitions.

Traditional LSE and Distributed LSE require $(N + N/3)$ and $(N + 2K)$ communications respectively.

Communication latency is the total time incurred for data transfer and is given by Eqs. 7.6

$$C_t = T_p + T_c \tag{7.6}$$

where T_p is the time to transfer data from PMUs to PDCs and T_c is the time taken to transfer data from PDC to control center.

7.3 Proposed Solution

This work aims to design an optimal Distributed LSE that scales for larger grid sizes and minimizes communication overhead. Distributed LSE decomposes grid into sub-LSE stations and computes the local sub-state estimate at each of the sub-LSE. Distributed LSE consists of two components: 1. partitioning algorithm and 2. LSE computation.

7.3.1 Partition Algorithm

Fundamental objective of this partition algorithm is to divide the grid into sub-networks called the sub-LSE stations, such that there is minimum connectivity between the sub-LSE stations. Partitioning algorithm is designed as a two-step model. First step utilizes some of the well-known graph partitioning methods to divide the large graph into subgraphs. Second step applies optimization strategies to minimize IPC.

Initial Partitioning

Initial partitioning is the first step of the partition algorithm. During this step, following methods are used to divide the smart grid: 1) Fiedler Vector based

recursive bisection, 2) Kernighan-Lin method based recursive bisection, 3)BFS Method and 4) Recursive Minimum Cut Method. These methods follow the constraint size c defined in section 7.2.1 for all the subgraphs.

First method bisects the graph by computing the Fiedler vector for subgraphs in each iteration [Fiedler (1973)]. Second method recursively bisects the graph using the Kernighan-Lin method [Kernighan and Lin (1970)]. Third method determines an initial set of sub-LSE based on BFS traversal of the graph starting at the node with highest betweenness centrality. And the fourth method uses Stoer Wagner algorithm for obtaining the initial sub-LSE stations [Stoer and Wagner (1997)]. In the BFS approach, node with maximum betweenness centrality is considered as the starting node for the traversing through the graph. Current node is added to the existing sub-LSE station if its size is less than the constraint size. If the size of the partition is greater than or equal to the constraint size, then the node is added to the new sub-LSE station. All the methods maintain the constraint size c for sub-LSE stations. IPC of the initial sub-LSE stations can be reduced by using efficient optimization techniques. Optimization methods for reducing the IPC are discussed in section 7.3.1.

After obtaining the initial partitions, results of the methods that have the minimum number of inter-partition connections are given to the second step. Fiedler Vector based recursive bisection method results in minimum inter-partition communication. Results of this method are given to the partition optimization.

Partition Optimization

Initial partitioning may sometimes result in unbalanced partitions leading to high IPC and may also degrade the performance of the distributed solution. Therefore, it may be essential to balance the partitions, thereby reducing IPC. Additionally, balanced partitions can also be optimized for minimizing IPC. In this work, two novel optimization methods are proposed. **Method I:** In this method, all the vertices that have external connectivity greater than the internal connectivity are identified. This vertices are called **INodes** and are represented by I_n. Each vertex in I_n are iteratively moved to a new partition to minimize the IPC. For every vertex v in I_n connectivity score cs for all the sub-LSE's is computed. Connectivity score is the number of edges the node is connected to in a given partition. Vertex is moved to a new sub-LSE if the cs of a sub-LSE Pi is greater than the cs of the current sub-LSE. In the event, where cs of P_i is the same as the current sub-LSE, the vertex is moved to the new sub-LSE if the IPC reduces. New set of INodes are obtained after each iteration and this is repeated until the IPC cannot be further minimized.

Method II: This method focuses on balancing sub-LSE areas by merging any two unbalanced sub-LSEs. Two sub-LSEs P_i and P_j are unbalanced if ratio of $|P_i|$ to $\left|P_j\right|$ is less than the threshold τ, where the number of nodes in P_j is greater than the number of nodes in P_i. Two sub-LSEs are merged only if they are connected and if the size of the merged sub-LSE follows the size constraint. Merging is terminated when there are no more sub-LSEs that are unbalanced or if all available sub-LSE's for merge operation violate the constraint size.

7.3.2 LSE Computation

The second component of Distributed LSE is LSE computation. LSE computation is performed at each of the sub-LSE stations obtained from the partition algorithm. Each sub-LSE computes the sub-state estimate and send it to the control center to obtain complete state vector. Detailed distributed LSE computation is explained in Section 7.4.

7.4 System Architecture

This section presents components of the proposed architecture and the work flow of the solution. System architecture is divided into two modules: offline module and online module. Offline module tasks do not have strict real-time constraints and are executed only when is a change in the grid configuration. On the other hand, tasks defined in the online module have to be executed for every new grid measurement, and these tasks have to always ensure that they meet real-time constraints.

7.4.1 Offline Module

This module is used for setting up the environment and computing the preprocessing functions. Tasks in this module are executed only once and do not have strict time constraints. Tasks of the offline module are Partition, Configuration and Topology Processor. Partition task takes the network topology of the system as input and divides the grid into sub-LSE's using the partitioning algorithm defined in section 7.3.1. Results of the partitioning algorithm are sent to the configuration task.

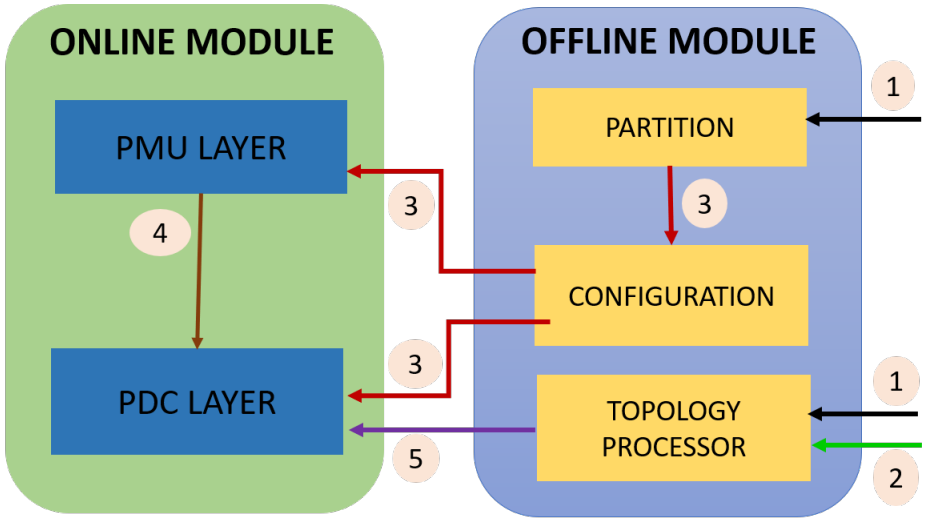
Configuration task communicates the grid partition information to the PMU and PDC layers of the Online module. In addition, this task also takes the network topology information to determine the connectivity of each node on the system and communicate the same to the PMU layer.

Topology processing task uses the network topology and network parameters to compute the H matrix of equation 6.1. This H matrix is sent to the PDC layer of the online module.

7.4.2 Online Module

The online module consists of two layers: PMU layer and PDC layer, PMU layer is composed of all the PMU's installed on the grid. PDC layer is the set of PDC's on the grid. PMU's of the PMU layer contact the configuration module to obtain information about the PDC to which it has to send the data and establishes a connection to the PDC. PMU's sense voltage and current measurements from the grid and send it to the connected PDC.

PDC communicates with configuration module to obtain configuration information such as how many and which PMU's will be connected to it. PDC's also communicate with the topology processor to obtain the H matrix and store it locally. Upon receiving the data from all the PMU's connected to it, PDC constructs partial measurement vector(z') and compute the sub-state estimate. PDC's use H and z' matrix to compute the sub-state estimate v_i. PDC's communicate with other PDC's to send the tie line information and also send the sub-state estimates to the control center for obtaining the state vector.



1. Network Topology
2. Network Parameters
3. Grid Partition Configuration
4. Grid Measurements [Voltage and Current]
5. Psuedoinverse Matrix [H]

Figure 7.1: Distributed LSE System Model.

7.5 Implementation

This section presents the implementation details of the partitioning algorithm and Distributed LSE.

7.5.1 Partition Algorithm

Partitioning algorithm is implemented in Python. NetworkX package is used for implementing the fundamental graph operations [Hagberg *et al.* (2008)]. Network topology is given as input to the first step of the algorithm; this step obtains initial partitions using all the three approaches discussed in section 7.3.1. Partitions are obtained based on the constraint size and partition method requested. After identifying the initial partitions of the grid, IPC across all the

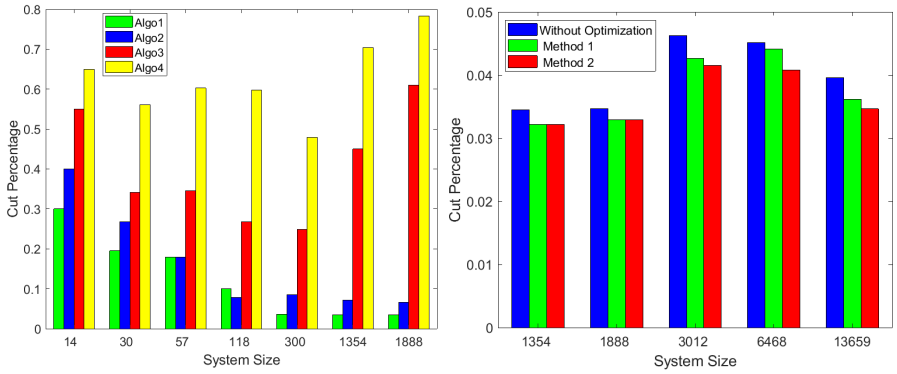
partitions is computed. Optimization is applied on the initial partitions obtained and the combination of the partition and optimization method that gives reduced IPC across all the test systems is considered to be the optimal solution. Partition sets of the methods that have lower IPC are selected as input to the optimization step.

7.5.2 Distributed-LSE

Distributed LSE is implemented using **'C'**. High performance GSL library is used in the implementation of Distributed LSE for handling complex number computations [Galassi *et al.* (2016)]. Distributed LSE application consists of the following modules: configuration module, topology processor, PMU Module and PDC Module. Configuration module uses the grid partition configuration obtained from the partition algorithm. PMU and PDC modules contact configuration module over TCP sockets and obtain the grid partition configuration. Topology processor uses network topology and network parameters as input and computes A, y, y_s, B and H given by Eqns. 6.1, 6.2. PMU module connects to the PDC using TCP socket, constructs z'. PDC module receives z' from all connected PMU's, uses the H result of the topology processor and computes sub-LSE estimate using GSL methods. Sub-LSE estimates are sent to the control center to merge and obtain the complete state vector.

7.5.3 Docker Container Based Power Grid Emulator

Docker container based power grid emulator is designed to illustrate working and scalability of the Distributed LSE. This emulator consists of three differ-



(a) (b)

Figure 7.2: Cut Percent after (a) Initial Partition and (b) Optimization.

ent types of containers: PMU Docker, PDC Docker and the Control Center Docker. PMU docker container represents the PMU's of the grid. PMU docker obtains the configuration information from the offline module and connects to its respective PDC docker. It then sends voltage and current measurements to the PDC docker using the socket I/O. PDC dockers represent the PDC on the grid they are also considered as the sub-LSE stations. PDC dockers obtain the partition configuration and results of the topology processor from the offline module. Configuration information is used to validate the PMU connections. PDC dockers obtain voltage and current measurements from the PMU dockers. These measurements are used to construct a measurement vector for that partition. Sub-state estimate of a partition is computed within the PDC docker by multiplying the results of topology processor with the measurement vector. Control Center docker is used to obtain the global state estimate. Results of Distributed LSE are presented in section 7.6.

7.6 Evaluation and Results

This section presents evaluation of the partitioning algorithm and the Distributed LSE for the standard test systems of the power grid systems. Distributed LSE is evaluated in comparison with the centralized LSE. Performance metrics used to evaluated Distributed LSE are data transfer and communication cost. Memory usage and net I/O on the docker container modules for the test systems are also evaluated. PMU data used in the execution of Distributed LSE and centralized LSE is obtained from MATPOWER [Zimmerman *et al.* (2011)].

7.6.1 Partition Algorithm

Figure 7.2a shows cut percentage of different methods after initial partition. From the figure, it may be observed that algorithm 1, Fielder vector based recursive bisection approach divides the grid into sub-LSE networks that have minimum IPC. Sub-LSE networks obtained by applying this method are given as input to the partition optimization module. Figure 7.2b shows cut percentage for different test systems after partition optimization. Partition optimization is applied only to larger test systems. From the results, it may be observed that proposed optimization methods reduce the IPC by 6% and 8% with respect to the initial partition. Figure 7.3 shows the partition of 14-Bus network before and after optimization.

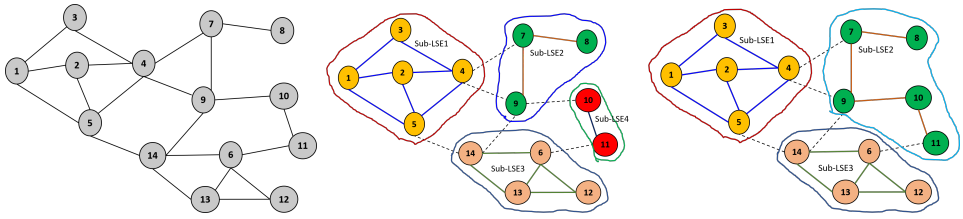
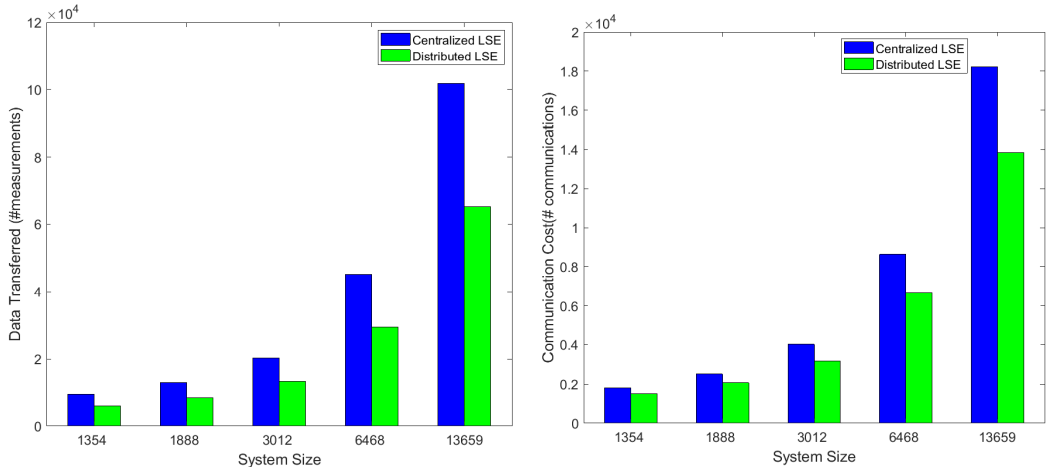


Figure 7.3: 14-Bus Network (a)Complete Graph (b)Sub-LSE after Initial Partition (c)Sub-LSE after Optimization.

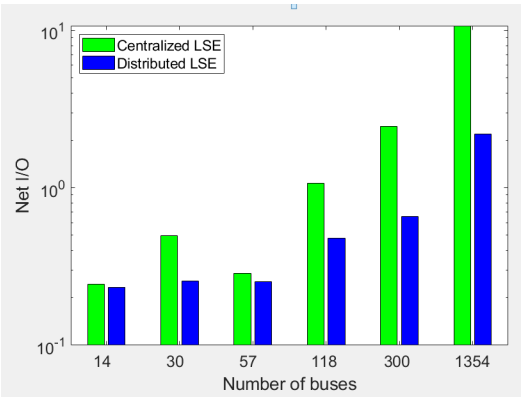
7.6.2 Distributed LSE

Figure 7.4a presents the data measurements transferred in Centralized LSE and Distributed LSE. It can be observed that data transfer and communication cost incurred by the Distributed LSE are reduced as compared to the Centralized LSE. Data transferred in Distributed LSE has reduced by an average of 35%. It may be further observed that Distributed LSE performance is increasing with the increase in system size.

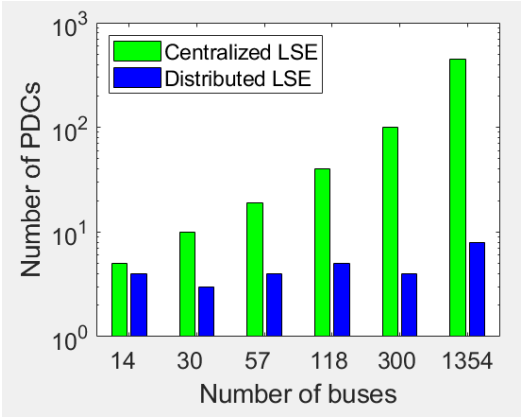


(a) Data Transfer (b) Communication Cost

Figure 7.5a and 7.5b present reduced net I/O and number of PDCs in Dis-



(a) Net IO



(b) Number of PDC's in DLSE

tributed LSE in comparison to centralized LSE for different test systems. Net I/O of Distributed LSE is reduced by an average of 25%.

7.7 Conclusion

Distributed LSE is designed to scale for large grid sizes. Proposed Distributed LSE results in reduced the communication cost and data transfer. Data transfer incurred in Distributed LSE is reduced by 35% compared to the centralized LSE across all the test systems and communication cost is reduced by an average of 20% across all the test systems. Another essential objective of the proposed Distributed LSE is to mitigate latency overhead by moving computation towards the edge. Docker based power grid emulator designed in this research depicts the distributed linear state estimation. Power grid emulator implemented in this research can be used to evaluate different aspects of linear state estimation.

Chapter 8

CONCLUSION

In this research, a reliable distributed management framework for latency sensitive application in uncertain environments is proposed. Data acquisition module of the framework is developed to collect contextual data that enables data enrichment and data reduction. False data detection module of the framework is used for ensuring data correctness and reliability of the application. Proactive user recruitment proposed in the framework is designed to find optimal set of volunteer devices which improves the success rate of task completion. To evaluate contextual data acquisition and proactive user recruitment modules of the proposed framework a real-time perpetrator tracking application is developed. The application offloads face recognition code to volunteer devices to find the location of perpetrator. Resource monitoring and prediction can be generalized and used for other mobile crowd sensing applications. For example, recent outbreak of COVID-19 can be controlled by having effective contact tracing approaches. Location monitoring and prediction proposed for the proactive user recruitment is extended for COVID-19 contact tracing. Distributed

Expanding use of machine learning models for IoT applications can benefit from extending ContextAiDe for optimal offloading of pre-trained machine learning models. Using ContextAiDe for mobile health applications is left as future work.

REFERENCES

" U.S. Department of Energy, North American SynchroPhasor Initiative", `https://www.naspi.org`, [Online; accessed 5-Dec-2016] (2016).

Abowd, G. D., A. K. Dey, P. J. Brown, N. Davies, M. Smith and P. Steggles, "Towards a better understanding of context and context-awareness", in "Handheld and Ubiquitous Computing", edited by H.-W. Gellersen (Springer Berlin Heidelberg, 1999).

Al-Ali, A. and R. Aburukba, "Role of internet of things in the smart grid technology", Journal of Computer and Communications **3**, 05, 229 (2015).

Al-Dweik, A., R. Muresan, M. Mayhew and M. Lieberman, "Iot-based multifunctional scalable real-time enhanced road side unit for intelligent transportation systems", in "2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE)", pp. 1–6 (IEEE, 2017).

Anderson, D. P., J. Cobb, E. Korpela, M. Lebofsky and D. Werthimer, "Seti@ home: an experiment in public-resource computing", Communications of the ACM **45**, 11, 56–61 (2002).

Bao, X. and R. Roy Choudhury, "Movi: mobile phone based video highlights via collaborative sensing", in "Proc. of the 8th international conference on Mobile systems, applications, and services", pp. 357–370 (ACM, 2010).

Catarinucci, L., D. De Donno, L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi and L. Tarricone, "An iot-aware architecture for smart healthcare systems", IEEE Internet of Things Journal **2**, 6, 515–526 (2015).

Centenaro, M., L. Vangelista, A. Zanella and M. Zorzi, "Long-range communications in unlicensed bands: The rising stars in the iot and smart city scenarios", IEEE Wireless Communications **23**, 5, 60–67 (2016).

Chakati, V., M. Pore, A. Pal, A. Banerjee and S. K. Gupta, "Challenges and trade-offs of a cloud hosted phasor measurement unit-based linear state estimator", in "Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), 2017 IEEE", pp. 1–5 (IEEE, 2017).

Chang, T., L. Zheng, M. Gorlatova, C. Gitau, C. Huang and M. Chiang, "Demo: Decomposing data analytics in fog networks", in "Proc. ACM Conference on Embedded Networked Sensor Systems (ACM SenSys' 17)", (2017).

Chatterjee, P., A. Pal, J. S. Thorp and J. De La Ree, "Partitioned linear state estimation", in "IEEE PES Innovative Smart Grid Technologies Conference(ISGT),", pp. 1–5 (IEEE, 2015).

Chen, H., B. Guo, Z. Yu and Q. Han, "Toward real-time and cooperative mobile visual sensing and sharing", in "IEEE INFOCOM 2016 - The 35th Annual IEEE Intl. Conf. on Computer Communications", pp. 1–9 (2016).

Chen, X., H. Zhang, C. Wu, S. Mao, Y. Ji and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning", IEEE Internet of Things Journal **6**, 3, 4005–4018 (2018).

Chen, Y.-S. and Y.-T. Tsai, "A mobility management using follow-me cloud-cloudlet in fog-computing-based rans for smart cities", in "Sensors", (2018).

Chodera, J., "Folding@home update on sars-cov-2 (10 mar 2020)", URL `https://foldingathome.org/2020/03/10/covid19-update/` (2020).

Chon, Y., E. Talipov, H. Shin and H. Cha, "CRAWDAD dataset yonsei/lifemap (v. 2012-01-03)", Downloaded from `https://crawdad.org/yonsei/lifemap/20120103` (2012).

Chun, B.-G., S. Ihm, P. Maniatis, M. Naik and A. Patti, "Clonecloud: elastic execution between mobile device and cloud", in "Proceedings of the sixth conference on Computer systems", pp. 301–314 (2011).

Cuervo, E., A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra and P. Bahl, "Maui: making smartphones last longer with code offload", in "Proceedings of the 8th international conference on Mobile systems, applications, and services", pp. 49–62 (ACM, 2010).

De Rolt, C. R., R. Montanari, M. L. Brocardo, L. Foschini and J. da Silva Dias, "Collega middleware for the management of participatory mobile health communities", in "IEEE Symposium on Computers and Communication (ISCC), 2016", pp. 999–1005 (IEEE, 2016).

Doukas, C. and I. Maglogiannis, "Bringing iot and cloud computing towards pervasive healthcare", in "2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing", pp. 922–926 (IEEE, 2012).

Elhoseny, M., G. Ramírez-González, O. M. Abu-Elnasr, S. A. Shawkat, N. Arunkumar and A. Farouk, "Secure medical data transmission model for iot-based healthcare systems", IEEE Access **6**, 20596–20608 (2018).

Fiandrino, C., F. Anjomshoa, B. Kantarci, D. Kliazovich, P. Bouvry and J. N. Matthews, "Sociability-driven framework for data acquisition in mobile crowdsensing over fog computing platforms for smart cities", Sustainable Computing, IEEE Transactions on **2**, 4, 345–358 (2017).

Fiedler, M., "Algebraic connectivity of graphs", Czechoslovak mathematical journal **23**, 2, 298–305 (1973).

Galassi, M., J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth and F. Rossi, "Gnu scientific library", Reference Manual edition **1** (2007).

Galassi, M., J. Theile and G. Team, "GSL-GNU Scientific Library", `https://www.gnu.org/software/gsl/`, [Online; accessed 5-Dec-2016] (2016).

Gambs, S., M.-O. Killijian and M. N. del Prado Cortez, "Next place prediction using mobility markov chains", in "Proceedings of the First Workshop on Measurement, Privacy, and Mobility", p. 3 (ACM, 2012).

Ganti, R. K., F. Ye and H. Lei, "Mobile crowdsensing: current state and future challenges", IEEE Communications Magazine **49**, 11 (2011).

Goh, C. Y., J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran and P. Jaillet, "Online map-matching based on hidden markov model for real-time traffic sensing applications", in "15th Intl. IEEE Conf. on Intell. Transp. Syst.", pp. 776–781 (2012).

Gope, P. and T. Hwang, "Bsn-care: A secure iot-based modern healthcare system using body sensor network", IEEE Sensors Journal **16**, 5, 1368–1376 (2016).

Guerrero-Ibanez, J. A., S. Zeadally and J. Contreras-Castillo, "Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and internet of things technologies", IEEE Wireless Communications **22**, 6, 122–128 (2015).

Hagberg, A. A., D. A. Schult and P. J. Swart, "Exploring network structure, dynamics, and function using networkx", in "Proceedings of the 7th Python in Science Conference", pp. 11 – 15 (2008).

Handschin, E., F. C. Schweppe, J. Kohlas and A. Fiechter, "Bad data analysis for power system state estimation", IEEE Transactions on Power Apparatus and Systems **94**, 2, 329–337 (1975).

Hassani, A., P. D. Haghighi and P. P. Jayaraman, "Context-aware recruitment scheme for opportunistic mobile crowdsensing", in "IEEE 21st Intl. Conf. on Parallel and Distributed Systems (ICPADS)", pp. 266–273 (2015).

Higashino, T., H. Yamaguchi, A. Hiromori, A. Uchiyama and K. Yasumoto, "Edge computing and iot based research for building safe smart cities resistant to disasters", in "2017 IEEE 37th Intl. Conf. on Dist. Comp. Systs. (ICDCS)", pp. 1729–1737 (2017).

Hou, W., Z. Ning and L. Guo, "Green survivable collaborative edge computing in smart cities", IEEE Trans. on Industrial Informatics **14**, 4, 1594–1605 (2018).

Hu, C., W. Bao, D. Wang and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge", in "IEEE INFOCOM 2019-IEEE Conference on Computer Communications", pp. 1423–1431 (IEEE, 2019).

Hu, S., L. Su, H. Liu, H. Wang and T. F. Abdelzaher, "Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification", ACM Trans. Sen. Netw. **11**, 4, 55:1–55:27 (2015).

Hull, B., V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan and S. Madden, "Cartel: a distributed mobile sensor computing system", in "Proc. of SenSys", pp. 125–138 (ACM, 2006).

Hussain, M., M. S. Alam, M. Beg *et al.*, "Fog computing in iot aided smart grid transition-requirements, prospects, status quos and challenges", arXiv preprint arXiv:1802.01818 (2018).

Jayaraman, P. P., C. Perera, D. Georgakopoulos and A. Zaslavsky, "Efficient opportunistic sensing using mobile collaborative platform mosden", in "Collaboratecom", pp. 77–86 (IEEE, 2013).

Jones, K. D., J. S. Thorp and R. M. Gardner, "Three-phase linear state estimation using phasor measurements", in "2013 IEEE PES General Meeting", pp. 1–5 (2013).

Kamruzzaman, M., N. I. Sarkar, J. Gutierrez and S. K. Ray, "A study of iot-based post-disaster management", in "2017 International Conference on Information Networking (ICOIN)", pp. 406–410 (IEEE, 2017).

Kernighan, B. W. and S. Lin, "An efficient heuristic procedure for partitioning graphs", The Bell system technical journal **49**, 2, 291–307 (1970).

Khan, M. A., H. Debnath, N. R. Paiker, N. Gehani, X. Ding, R. Curtmola and C. Borcea, "Moitree: A middleware for cloud-assisted mobile distributed apps", in "4th Intl. Conf. on MobileCloud", pp. 21–30 (IEEE, 2016).

King, D. E., "Dlib-ml: A machine learning toolkit", Journal of Machine Learning Research **10**, 1755–1758 (2009).

Kostakos, V., D. Ferreira, J. Goncalves and S. Hosio, "Modelling smartphone usage: A markov state transition model", in "Proc.of Intl. Conf. on Pervasive and Ubiquitous Comp.", UbiComp '16, pp. 486–497 (2016).

Kosut, O., L. Jia, R. J. Thomas and L. Tong, "Malicious data attacks on the smart grid", IEEE Trans. on Smart Grid **2**, 4, 645–658 (2011).

Larson, S. M., C. D. Snow, M. Shirts and V. S. Pande, "Folding@ home and genome@ home: Using distributed computing to tackle previously intractable problems in computational biology", arXiv preprint arXiv:0901.0866 (2009).

Li, L., H. Xiaoguang, C. Ke and H. Ketai, "The applications of wifi-based wireless sensor network in internet of things and smart grid", in "2011 6th IEEE Conference on Industrial Electronics and Applications", pp. 789–793 (IEEE, 2011).

Liu, C. H., B. Zhang, X. Su, J. Ma, W. Wang and K. K. Leung, "Energy-aware participant selection for smartphone-enabled mobile crowd sensing", IEEE Systems Journal **11**, 3, 1435–1446 (2017).

Liu, Y., P. Ning and M. K. Reiter, "False data injection attacks against state estimation in electric power grids", ACM Transactions on Information and System Security (TISSEC) **14**, 1, 13 (2011).

Maheshwari, K., M. Lim, L. Wang, K. Birman and R. van Renesse, "Toward a reliable, secure and fault tolerant smart grid state estimation in the cloud", in "IEEE PES ISGT", pp. 1–6 (2013).

Nuthalapati, S. and A. G. Phadke, "Managing the grid: Using synchrophasor technology [guest editorial]", IEEE Power and Energy Magazine **13**, 5, 10–12 (2015).

NVIDIA, "cuBLAS", `http://docs.nvidia.com/cuda/cublas/index.html`, [Online; accessed 5-Oct-2017] (2017).

Patel, M., S. Aivaliotis, E. Ellen *et al.*, "Real-time application of synchrophasors for improving reliability", NERC Report, Oct (2010).

Pore, M., V. Chakati, A. Banerjee and S. K. Gupta, "Contextaide: End to end architecture for mobile crowd sensing applications", (ACM, 2018).

Ran, X., H. Chen, X. Zhu, Z. Liu and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics", in "IEEE INFOCOM 2018-IEEE Conference on Computer Communications", pp. 1421–1429 (IEEE, 2018).

Ray, P. P., M. Mukherjee and L. Shu, "Internet of things for disaster management: State-of-the-art and prospects", IEEE Access **5**, 18818–18835 (2017).

Reddy, S., D. Estrin and M. Srivastava, "Recruitment framework for participatory sensing data collections", in "Intl. Conf. on Pervasive Computing", pp. 138–155 (Springer, 2010).

Sadeghi, K., A. Banerjee, J. Sohankar and S. K. Gupta, "Optimization of brain mobile interface applications using iot", in "2016 IEEE 23rd International Conference on High Performance Computing (HiPC)", pp. 32–41 (IEEE, 2016).

Sanchez, L., L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, "Smartsantander: Iot experimentation over a smart city testbed", Computer Networks **61**, 217–238 (2014).

Sapienza, M., E. Guardo, M. Cavallo, G. La Torre, G. Leombruno and O. Tomarchio, "Solving critical events through mobile edge computing: An approach for smart cities", in "2016 IEEE International Conference on Smart Computing (SMARTCOMP)", pp. 1–5 (IEEE, 2016).

Saremi, F., O. Fatemieh, H. Ahmadi, H. Wang, T. Abdelzaher, R. Ganti, H. Liu, S. Hu, S. Li and L. Su, "Experiences with greengps—fuel-efficient navigation using participatory sensing", IEEE Trans. on Mobile Computing **15**, 3, 672–689 (2016).

Satyanarayanan, M., "Mobile computing: the next decade", in "Proc. of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond", p. 5 (ACM, 2010).

Satyanarayanan, M., "The emergence of edge computing", Computer **50**, 1, 30–39 (2017).

Satyanarayanan, M., P. Bahl, R. Caceres and N. Davies, "The case for vm-based cloudlets in mobile computing", IEEE pervasive Computing **8**, 4, 14–23 (2009).

Shi, J. and W. Jia, "Real-time target tracking through mobile crowdsensing", in "Web Information Systems Engineering – WISE 2017", edited by A. Bouguettaya, Y. Gao, A. Klimenko, L. Chen, X. Zhang, F. Dzerzhinskiy, W. Jia, S. V. Klimenko and Q. Li, pp. 3–18 (Springer International Publishing, Cham, 2017).

Shi, W., J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: Vision and challenges", IEEE Internet of Things Journal **3**, 5, 637–646 (2016).

Song, F., Z.-Y. Ai, J.-J. Li, G. Pau, M. Collotta, I. You and H. Zhang, "Smart collaborative caching for information-centric iot in fog computing", in "Sensors", (2017).

Stoer, M. and F. Wagner, "A simple min-cut algorithm", Journal of the ACM (JACM) **44**, 4, 585–591 (1997).

Tao, X., K. Ota, M. Dong, H. Qi and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing", IEEE Wireless Communications Letters **6**, 6, 774–777 (2018).

Teerapittayanon, S., B. McDanel and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices", in "2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)", pp. 328–339 (IEEE, 2017).

Tonyali, S., K. Akkaya, N. Saputro, A. S. Uluagac and M. Nojoumian, "Privacy-preserving protocols for secure and reliable data aggregation in iot-enabled smart metering systems", Future Generation Computer Systems **78**, 547–557 (2018).

Vanrompay, Y., P. Rigole and Y. Berbers, "Predicting network connectivity for context-aware pervasive systems with localized network availability", in "WoSSIoT EuroSys", (2007).

Wang, S., R. Urgaonkar, T. He, K. Chan, M. Zafer and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs", IEEE Trans. Parallel Distrib. Syst. **28**, 4, 1002–1016 (2017).

Wu, H. and J. Giri, "PMU impact on state estimation reliability for improved grid security", in "IEEE PES Transmission and Distribution Conf. and Exhibition", pp. 1349–1351 (2006).

Wu, S., C. Niu, J. Rao, H. Jin and X. Dai, "Container-based cloud platform for mobile computation offloading", in "2017 IEEE international parallel and distributed processing symposium (IPDPS)", pp. 123–132 (IEEE, 2017).

Wu, Y., Y. Wang, W. Hu and G. Cao, "Smartphoto: A resource-aware crowdsourcing approach for image sensing with smartphones", IEEE Trans. on Mobile Computing **15**, 5, 1249–1263 (2016).

Yang, Q., T. Bi and J. Wu, "Wams implementation in china and the challenges for bulk power system protection", in "IEEE PES General Meeting", pp. 1–6 (2007).

Yuan, Q., H. Zhou, J. Li, Z. Liu, F. Yang and X. S. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution", IEEE Network **32**, 1, 80–86 (2018).

Zanella, A., N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of things for smart cities", IEEE Internet of Things journal **1**, 1, 22–32 (2014).

Zhang, J., M. Momtazpour, N. Ramakrishnan, G. Welch and S. Rahman, "Secure and adaptive state estimation for a pmu-equipped smart grid", in "2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)", pp. 1431–1436 (IEEE, 2015).

Zhang, K., Y. Mao, S. Leng, Y. He and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive offloading", IEEE Vehicular Technology Magazine **12**, 2, 36–44 (2017).

Zimmerman, R. D., C. E. Murillo-Sanchez and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education", IEEE Trans. on Power Sys. **26**, 1, 12–19 (2011).