Time Synchronization and Task-Assignment Algorithms for Specific Domains of the Internet of Things

THESIS

Submitted in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

G SAI SESHA CHALAPATHI
ID. No. 2004PH400558P

Under the Supervision of

Prof. S GURUNARAYANAN
Professor, Department of Electrical and Electronics Engineering

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

May 2019

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the thesis entitled "**Time Synchronization and Task-Assignment Algorithms for Specific Domains of the Internet of Things"** and submitted by Mr. **G Sai Sesha Chalapathi** ID.No. **2004PH400558P** for the award of Ph.D. degree of the Institute embodies original work done by him under my supervision.

Signature of the Supervisor
Name of the Supervisor: DR. S GURUNARAYANAN
Designation: Professor,
Department of Electrical and Electronics Engineering ,
BITS Pilani, Pilani Campus

Date :_____

Acknowledgments

Firstly, I would like to acknowledge the contribution of some special people in my life who have not just inspired me to complete this difficult journey of completing this Ph.D. work but have contributed in many aspects of my life in a selfless way. I want to thank the Almighty God who has compassionately given me the strength, ability, intelligence and much more during this thesis work and during the past three decades of my education, which has culminated in this doctoral thesis.

I thank Prof. S. Gurunarayanan, my thesis supervisor who has not just guided me through this thesis work but has also been very helpful, tolerant and gave me the freedom to explore my ideas. I also thank Prof. K.R. Anupama, Professor, Department of EEE, K.K. Birla Goa Campus, BITS-Pilani for her guidance in various stages of this Ph.D. work and for lending the Wireless Sensor Network platform for carrying out the experiments for this work. I gratefully acknowledge Prof. Souvik Bhattacharyya, Vice-Chancellor, BITS-Pilani, Prof. Ashoke K. Sarkar, Director, BITS-Pilani, Pilani Campus for giving me the opportunity to carry out this thesis work. I would also like to thank Prof. B.N. Jain, the past Vice-Chancellor, BITS-Pilani and Prof. G. Raghurama, Director BITS-Pilani, KK Birla Goa Campus for giving me their support. I also thank Prof. V.K. Chaubey, Prof. Anu Gupta and Prof. Navneet Gupta, for their guidance and support especially during their tenure as the Head of Department, Department of EEE, Pilani Campus. I also thank

other departmental colleagues especially Dr. Vinay Chamola, Dr. Nitin Chaturvedi, Dr. Sainath Bitragunta and Dr. Ashish Mishra for their support.

I also thank Prof. Andreas Springer, Head, Institute for Communications Engineering and RF-Systems, Johannes Kepler University (JKU), Austria for giving me the opportunity to work at JKU during the summers of 2016 and 2017. My special thanks go to Dr. Bernhard Etzlinger, Senior Researcher, JKU for his guidance during my work at JKU and in bringing out the work in the form of a journal paper, which is a part of this thesis.

I thank my students Mr. Raunak Manekar, Mr. Akshay Madan, Mr. Bhavik Dhandhaliya, Mr. Deependra Singh, Mr. Sahil Chandna and Mr. Syed Irfan for helping me in performing the experiments for this thesis work. I also thank other students, staff, and others who have helped me directly or indirectly.

I gratefully acknowledge the support, sacrifices and help given by my wife during this thesis work. Also, I would like to thank my friends, their family members for the support that they have given me in completing this doctoral thesis. I also gratefully acknowledge the sacrifices and the hard work of my parents, my sister, and brother-in-law, which has led me to come to a stage where I am able to complete my doctoral studies. I also thank my in-laws and other family members for their help and support during this thesis work.

Though this thesis is a result of the contributions of many people; I am responsible for the content, omissions, and errors in this thesis.

Abstract

Over the past decade, technological advancements made in network technologies has led to the advent of the Internet of Things (IoT). IoT aims to network different devices, appliances, and objects making it possible for them to exchange data which in turn enables efficient management of resources. IoT requires the interaction of many heterogeneous technologies like ZigBee, Bluetooth, Wi-Fi, WiMAX, etc. The focus of this thesis will be to address two specific problems -one each in Wireless Sensor Network (WSN) and mobile edge computing, both of which are important domains of the IoT. This work proposes two different time synchronization protocols for WSNs and a latency aware task-assignment scheme for edge cloudlet network.

WSN is a network of devices called nodes, which are used to perform coordinated sensing, monitoring, and actuation tasks. Since WSN is an example of a distributed system, it requires time synchronization for many functionalities like data fusion, duty-cycled packet communication, time-based localization protocols, etc. A time synchronization protocol called E-SATS (Efficient and Simple Algorithm for Time Synchronization) for a cluster-based WSN is presented in Chapter 3. This protocol synchronizes the cluster members of a cluster to its cluster head using very simple computations. E-SATS is implemented on a WSN testbed and its performance is analyzed for different Line-of-Sight (LOS) conditions. Further, E-SATS is compared to some prominent existing Time Synchronization Protocols (TSPs) for clus-

tered WSNs. The experimental results show that E-SATS has higher synchronization accuracy while being computationally efficient and consuming significantly lesser energy as compared to other state-of-the-art protocols.

Decentralized TSPs perform synchronization in a distributed way. They are beneficial in WSN deployments which do not have a strict requirement of a particular network structure. Recently, message-passing based methods have been employed to achieve synchronization in WSNs in a distributed way. A Mean-Field based message-passing method called Integrated Cooperative Synchronization (ICS) is proposed in Chapter 4 to synchronize the WSN nodes. Existing message-passing methods operate in two phases. The first phase is called the measurement phase in which the time measurements are collected during packet exchanges among the nodes. The second phase is called the message-passing phase in which the clock parameters of the nodes are estimated to achieve time synchronization of these nodes. Coordination of these two phases is highly challenging, especially in a large network. ICS mitigates this problem by integrating the measurement and message-passing phases. It selects an extended factorization of the underlying joint a-posteriori distribution of the clock parameters and uses an appropriate message scheduling for link initialization. ICS is conceptually much simpler than conventional message-passing methods while achieving similar accuracy and reduced computational complexity.

Chapter 5 of this thesis focuses on another domain of IoT, viz., edge computing. Over the past decade, mobile devices have been used to perform computationally intensive tasks. However, the computational

capability of these devices is limited due to memory, power and portability constraints. Cloud services have been used to enhance the performance of these resource-limited devices by offloading their computationally intensive tasks on to the cloud servers. However, when cloud servers are involved in processing, the latency and cost of computation increases. To mitigate these problems, devices with high computational resources, called edge devices or cloudlets, can be deployed in the locations close to the mobile users/devices. Due to easier access and nearness of the cloudlets, the cost and latency in processing the offloaded tasks decreases. Chapter 5 presents a task-assignment scheme in a multi-cloudlet network connected via wireless Software-defined Network (SDN) routers. This cloudlet network serves the task offload requests from mobile devices in a given locality. The aim of the proposed scheme is to minimize latency in processing the offloaded tasks and thus enhance the Quality of Service (QoS) for mobile devices. The optimality of the proposed scheme is proved mathematically. Also, an admission control policy is employed to maintain this optimality even in heavily loaded networks. Numerical simulations are performed for two scenarios of small and large networks and the performance for varying traffic and network parameters are evaluated. The results demonstrate that the proposed task-assignment scheme offers reduced latency compared to state-of-the-art task-assignment approaches and hence improves the QoS offered to mobile devices.

Overall, this thesis proposes computationally efficient time synchronization protocols for cluster-based WSNs and dynamic WSNs requiring decentralized synchronization protocols. It also proposes a novel scheme for assigning the tasks offloaded by the mobile devices onto the edge cloudlets while optimizing latency for processing these tasks.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

Term	Definition
FG	Factor Graph
ICS	Integrated Cooperative Synchronization
IoT	Internet of Things
LOS	Line-of-Sight
MAC	Medium-Access Control
MEC	Mobile Edge Computing
MF	Mean-Field
MIMO	Multi-Input-Multi-Output
NLOS	Non-Line-of-Sight
OFDMA	Orthogonal Frequency-Division Multiple Access
QoS	Quality of Service
RSSI	Received Signal Strength Indicator
TDMA	Time-Division Multiple Access
TSP	Time Synchronization Protocol
SDN	Software-Defined Network
VM	Virtual Machine
WSN	Wireless Sensor Network

Chapter 1

Introduction

Over the past decade, there has been a phenomenal increase in the number of devices connected together through the internet. The devices connected to the internet today are not limited to just computers or servers as it was the case a few years ago, but also include devices like smartphones, home appliances, industrial controllers, etc. This change was made possible by the advent of a new domain of technology called the *Internet of Things* or more commonly known as the *IoT*. The IoT paradigm allows objects to communicate data and information with each other and with end users over the Internet, which is expected to lead to efficient management of resources. For instance, the vision of 'smart cities' [Batty *et al.* 2012] has been pursued extensively in the recent past by the scientific community [Lin *et al.* 2017], [Ejaz *et al.* 2017], [Sotres *et al.* 2017] as well as the local governmental authorities in countries like Italy, Mexico, etc., to provide better public administration and unified, easily accessible, and transparent services to the citizens. Apart from this, IoT finds its application in agriculture [Elijah *et al.* 2018] , military surveillance [Suri *et al.* 2016], building energy management [Minoli *et al.* 2017], education [Ali *et al.* 2017], disaster management [Ray *et al.* 2017], etc.

IoT is made functional by a combination of many underlying technologies such as sensors, communication networks, Application Program Interfaces (APIs), back-end servers with data analytics, remote data/service access technologies, etc. Thus, it requires interface among many heterogeneous technologies to achieve seamless flow of data across these different entities of the IoT network. This interfacing is indeed challenging to achieve. Apart from this interfacing, there is a need to make the existing technologies "IoT-ready", i.e., suitable to be used in the IoT networks. Also, the interfacing of heterogeneous technologies paves way for newer problems which were never encountered before. For instance, mobile devices generate large amount of data in applications like image processing, speech recognition, etc. This data cannot be processed easily on these resource constrained devices. With the availability of internet connectivity on these mobile devices, this data is offloaded on to the cloud servers for processing. This data is processed on the cloud servers and the results are sent to the mobile devices. The process of offloading the data and receiving the results by the mobile device has a considerable latency. Therefore, we have to now invent methods to reduce this latency so that the Quality of Service (QoS) of the applications being executed on the mobile device can be improved.

Wireless Sensor Network (WSN) and Mobile Edge Computing (MEC) are two important domains of the IoT. WSN consists of a network of nodes which cooperatively perform sensing of event(s) or phenomena. MEC provides high computation capabilities at the edge of a network. These computational capabilities can be utilized by mobile devices to perform computationally intensive tasks. Mobile devices like smartphones, mobile tablets PCs, sensor nodes, etc., have memory, power and portability constraints and thus cannot perform computationally intensive tasks themselves. Thus they make use of edge devices in MEC networks to perform these tasks. We will elaborately introduce these domains in later sections of this chapter.

WSNs and MEC are used together in many IoT applications. For instance, Smart Street Lighting System (SSLS) [Zanella *et al.* 2014] which was implemented in Padova, Italy as a part of the Smart City Project uses both WSNs and edge computing nodes in it network. In this system, there is a WSN node on every street light. Each node is equipped with sensors to monitor the ambient light, air quality, temperature and humidity. The sensor data from the sensor nodes is collected by a node called the gateway which provides a link between the WSN and traditional Wireless Area Network (WAN). It also performs many functionalities like data storage, protocol translation, etc. This gateway is an example of 'edge' computing node. These edge devices provide an intermediate layer between the WSN and the cloud in this network. The focus of this doctoral thesis will be to address two specific problems in these two domains. The objectives of this thesis are:

1. To develop an efficient time synchronization protocol for clustered WSN and to implement it on a WSN testbed.
2. To carryout the performance evaluation of this synchronization protocol in various Line-of-Sight (LOS) environments.
3. To develop an accurate and efficient decentralized time synchronization protocol for a WSN (without any restriction on the topology of the network).
4. To develop a task-assignment scheme in a network of edge cloudlets to optimize the latency in processing the tasks offloaded on to them by the mobile devices.

The following subsections introduce WSNs and edge computing and introduce the specific problems addressed in both these domains in this thesis.

1.1 Wireless Sensor Networks (WSN)

WSN consists of a network of nodes which are generally capable of performing the following tasks:

- sensing some events or phenomena in a location
- processing the sensed data
- communicating with each other over the wireless

In addition to the above functionalities, in certain applications, the WSN nodes are also interfaced with actuators to perform some action based on the detection of some events. Thus, WSN nodes cooperatively perform the sensing, monitoring and actuation tasks in a given location. Applications like home automation [Ghayvat *et al.* 2015], agriculture [Ojha *et al.* 2015], personal health monitoring [Milenković *et al.* 2006], environmental monitoring [Othman & Shazali 2012], asset tracking [Kim *et al.* 2008], etc., require distributed sensing, processing of data, and suitable actuation. In such scenarios, WSNs emerge as the suitable platform for designing the solutions. Several WSN based IoT implementations have already been reported. The Smart Street Light application implemented in Padova (mentioned in the last section) is one such example. Some of the features of WSN which make them suitable for IoT applications are low cost (suitable for large scale deployment of devices), bidirectional communication (suitable for sensing and actuation) and low bandwidth requirements [Song *et al.* 2014].

The architecture of a WSN node is shown in Fig. 1.1. A WSN node consists of sensor(s) and actuator(s), a processing unit, communication interface and a power source. We briefly describe each of these constituents in the following:

- *Sensor and actuators:* Sensors and actuators are the interfaces of the WSN to the external world. The sensors, as mentioned before, sense a particular phenomenon and generate sensor data. The sensor data is processed either

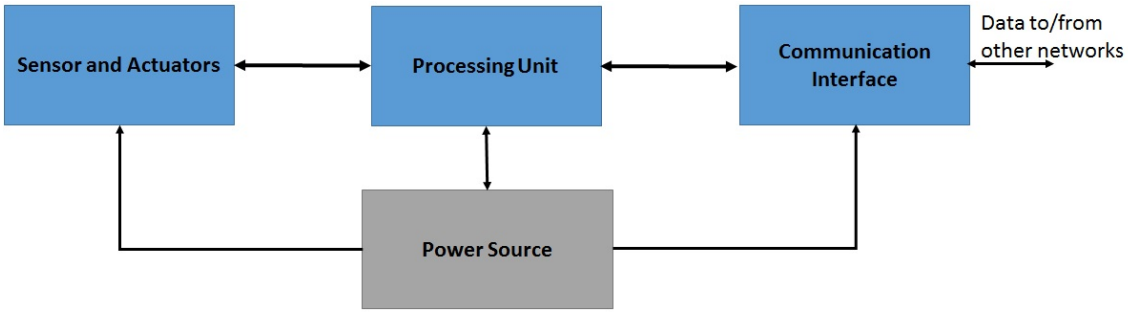


Figure 1.1: Architecture of a WSN node

locally on the node where the data was generated and/or remotely on other device(s). Based on the data generated, some decisions are taken which might involve some actuation like rotating a motor, display related tasks, etc., and the relevant WSN node controls the actuator.

- *Processing unit:* The Processing Unit (PU) provides the intelligence to the WSN node. It processes the sensor data partially or completely to monitor a location and take decisions. The PU on a WSN node is typically a microcontroller or a microprocessor. The choice of a PU for a node depends on the application running on the node, the nature of the sensor data generated by the sensors, etc. The PU also performs some pre-processing of the sensor data to minimize the data to be transmitted. Some networks protocols are also executed on the PU for data communication through the network.

- *Communication Interface:* Generally radio signals are used for wireless communication between the node and the network. However, certain WSN networks use acoustic signals, optical signals, etc., [Gkikopouli *et al.* 2012] for wireless data communication. WSNs are characterized by low data rates for data communication. WSN nodes are equipped with transceivers which are generally used in simplex mode. Of all the tasks of a WSN, data transmission and reception is one of the most power consuming tasks. Thus, a WSN node generally turns off its radio to conserve its energy when not in use.

- *Power Source*: Most of the WSN nodes in the network rely on batteries for their power supply requirements. However, there have been implementations where WSNs have used solar energy [Yi *et al.* 2009], temperature gradient [Lu & Yang 2010] or vibrations [Lu *et al.* 2016] to power themselves. Thus, a WSN is typically an energy-constrained network. WSNs are deployed in a harsh environments like industrial plants [Krishnamurthy *et al.* 2005], mines [Li & Liu 2009], volcanoes [Werner-Allen *et al.* 2005], etc., where frequent human intervention is difficult or impossible. Therefore, WSN nodes are designed to be robust and energy efficient.

A typical WSN uses many protocols/algorithms for various functionalities like the localization, time-synchronization, routing, clustering, etc. The first three objectives of this thesis focus on the time synchronization protocols in WSNs. Therefore, a brief overview of time synchronization protocol is provided in the next section.

1.2 Time Synchronization Protocols in Wireless Sensor Networks

WSNs are an example of distributed network where the nodes run in a coordinated way to carry out sensing and monitoring tasks. Thus, it is very important for the participating nodes to have a common notion of time of the network. It is possible to achieve time synchronization among the WSN nodes if they are all equipped with devices like Global Positioning System (GPS) modules. But it is not possible to have a GPS module on every node as it increases the cost of each node tremendously. Further, GPS connectivity is very limited in indoor locations, in cloudy conditions, etc. [Elson *et al.* 2002]. Therefore we need a *Time Synchronization Protocol (TSP)* to achieve the synchronization among the nodes.

WSN nodes maintain the local time at each node using crystal oscillators. But large scale deployment of WSN nodes prohibits usage of high accuracy crystal oscillators. Thus, WSN nodes use cheap crystal oscillators for running their local clocks. These cheap crystal oscillators bring inaccuracies in the local time due to crystal aging, environmental factors like temperature, humidity, etc. [Sivrikaya & Yener 2004]. Therefore, a TSP is essential to synchronize the local clocks of the WSN nodes. However, we cannot use generic synchronization protocols like Network Time Protocol (NTP) [Mills 1994] used in other distributed networks. This is because specific characteristics of WSNs like limited energy availability, low bandwidth, possibility of frequent changes in the network (due to ad-hoc joining and leaving of the nodes), etc., mandate usage of synchronization protocols specifically designed for WSNs.

TSPs provide a common reference of time either for the entire network or for some part(s) of the network. The common time reference provided by a TSP could be either (i) an absolute time like Coordinated Universal Time (UTC)/local time for a zone or (ii) relative/local time which is specific to the network or part of the network or (iii) give a basis to decide the chronology of some events in the WSN. The kind of time reference provided by a TSP depends on the requirements of the applications that use it. TSP plays an important role in applications/functions like data gathering and fusion, time-based localization, TDMA based communication, power management protocols, etc. Time synchronization is all the more important for a WSN used in an IoT network as the sensor data recorded by a WSN node is monitored and analyzed at remote locations. Thus, it is important to timestamp such sensor data according to a common clock valid across this IoT network. A general approach followed to achieve synchronization for this scenario is that all the WSN nodes are synchronized to the *sink node* (also called as Base Station or Coordinator), which acts as an interface between the WSN and other networks. This sink node is synchronized to the internet time and it translates the times-

tamps generated at WSN nodes to the internet time.

Time synchronization in WSN is particularly challenging as the nodes are usually deployed in harsh environments. They are deployed for unattended operation targeted to work with minimal or no maintenance, generally for a few months to a few years. Thus the synchronization protocol must be **robust** to node failures and provide a synchronization of required accuracy. Also, since a WSN is an ad-hoc network which involves joining and leaving of nodes in a dynamic manner, the TSP should be **scalable** with increasing network size without significant degradation in its performance. Also since a WSN is largely a battery operated network, the TSP should be **energy efficient**. A TSP is called energy efficient if it requires very few packets for achieving the synchronization and/or requires infrequent resynchronizations. Further, WSNs are often used in critical applications like health monitoring, surveillance, etc., which require immediate reporting of emergency events. Therefore the TSP protocol must be **immediate** and thus must be computationally efficient. A detailed discussion about the characteristics and requirements of a TSP in WSNs can be found in [Sivrikaya & Yener 2004].

There are several TSPs which have been proposed and even used in WSNs like RBS [Elson *et al.* 2002], TPSN [Ganeriwal *et al.* 2003], FTSP [Maróti *et al.* 2004], SLTP [Nazemi Gelyan *et al.* 2007], L-SYNC [Jabbarifar *et al.* 2010], etc. Most of these protocols have been implemented and tested either partially or fully on a simulator only. There are very few protocols which have been demonstrated on a hardware testbed. Simulator-based works cannot give a complete picture of the performance of a TSP as they make many assumptions at a high level of abstraction, do not consider packet loss and its effect on synchronization accuracy, etc. [Djenouri & Bagaa 2016]. We therefore require a TSP which has been proved suitable for WSN on a hardware platform, which is simple and efficient (in terms of the computations involved) and accurate. In addition, most of the works do not

describe the environment in which the experiments have been performed. The environmental conditions, i.e., whether the communicating nodes are Line-of-Sight (LOS) or Non Line-of-Sight (NLOS) significantly affect the performance of a TSP, especially in the IoT applications which have NLOS environments (like smart home application). So we aim to address these issues in this thesis by proposing a TSP called *Efficient and Simple Algorithm for Time Synchronization* (E-SATS). E-SATS has been designed for WSN with a cluster-based topology and is able to achieve micro-second level synchronization accuracy with simple computations. Thus, E-SATS is aptly suited for resource-constrained WSN nodes. Micro-second accurate TSP is very essential for many automotive, health and industrial applications [Elsts *et al.* 2016]. To prove the merit of E-SATS in realistic scenarios in different LOS conditions, its performance is been tested on a densely deployed large size WSN testbed. A detailed presentation of the work on E-SATS is made in Chapter 3 of this thesis.

E-SATS is an example of centralized TSP which requires coordination among the nodes to form a specific network topology. Cluster-based topology has several advantages. For example, it increases the energy-efficiency of operations like data gathering, data dissemination, etc. It also increases the lifetime of the network by increasing energy efficiency of these operations. However, formation and maintenance of specific network structure becomes a significant overhead for large sized WSN deployments [Etzlinger *et al.* 2014]. Also, node failures which are very common in WSNs, can drastically effect the performance of centralized TSPs [Leng & Wu 2011]. Thus, over the last few years, there has been a growing interest in a class of synchronization protocols called decentralized TSPs in which the synchronization is performed in a decentralized way. In decentralized TSPs, a node performs time synchronization with the help of its neighbors. Also, it is scalable and is more resilient to node failures. Examples of such TSPs are [Etzlinger *et al.* 2014], [Leng & Wu 2011], [Etzlinger *et al.* 2013b], [Schenato & Fiorentin 2011], [Zennaro

et al. 2011], etc. Consensus based synchronization techniques like [Schenato & Fiorentin 2011] and [Zennaro *et al.* 2011] synchronize the nodes in the network by achieving a consensus of the network time parameters among the nodes. Further, there are some TSPs like [Etzlinger *et al.* 2014] and [Leng & Wu 2011] which use statistical techniques like belief propagation and TSPs like [Etzlinger *et al.* 2013a] use Mean-field technique. The TSPs which use such statistical techniques achieve higher synchronization accuracy as compared to other methods. But they have high computational complexity. Further, they require two distinct phases namely the measurement phase and message-passing phase [Etzlinger *et al.* 2014], [Etzlinger *et al.* 2013a] [Zennaro *et al.* 2013], [Leng & Wu 2011]. In the measurement phase, all the nodes exchange packets with their neighbors recording their time stamps of transmission and reception of the packets. In the message-passing phase, the nodes achieve synchronization by exchanging the estimates of their local clock parameters estimated based on the timestamps recorded in the measurement phase. The coordination of these two phases in a large-sized WSN requires additional mechanisms and is highly challenging.

So we propose a new TSP based on mean-field technique called *Integrated Cooperative Synchronization (ICS)* which integrates both these phases without significant reduction in the synchronization accuracy. Further, ICS makes a drastic reduction in the computational complexity compared to conventional mean-field technique. It does so by selecting an extended factorization of the underlying joint a-posteriori distribution of the clock parameters and using an appropriate message scheduling for link initialization. Due to its computational simplicity and reduced memory requirements, it is suitable for resource constrained WSN nodes.

In the next section, we introduce edge computing, another domain of the IoT which has been explored in this thesis.

1.3 Edge Computing

Rapid advances made in mobile computing have enabled proliferation of mobile devices in a variety of tasks like video calling, gaming, image processing applications, etc. However, such capabilities have come with new challenges. Many of these applications are computationally intensive, which require hardware platforms with high computational power. Mobile devices like smartphones, tablet PCs, etc., cannot handle such intensive computations due to their memory, power and portability constraints. Trying to execute such tasks on the mobile device itself leads not only to the slowing down of the device but also to quicker battery energy drainage. Therefore, these tasks are offloaded on to the cloud servers. Cloud servers have features like large data-storage capacity, high computation capabilities and can process a variety of computationally intensive tasks offloaded on to them. Thus, cloud services have been seen as a viable solution to overcome the above-mentioned constraints of mobile devices [Nakamura *et al.* 2007], [Chun *et al.* 2011], [Xie *et al.* 2013]. Cloud platforms perform these offloaded tasks and provide the results to the mobile devices quickly.

However, along with the above-mentioned advantages of using the cloud services, there are certain drawbacks of using this option. Firstly, the mobile device requires an internet connection to offload its computation request on to the cloud server and to receive the results of its computation. Internet connectivity is generally limited and in the absence of Wi-Fi hotspots, the user has to rely on cellular networks for offloading and receiving the results. This could incur data usage charges. Secondly, usage of cloud services would incur additional subscription charges levied by the cloud service providers on their users. Thirdly, when the user is using the cloud services from far off locations, high computation tasks like face recognition and image processing based applications can experience high latency in getting the results. These factors envisage the need for devising a new

solution to these problems of mobile computing and this has led to the advent of *edge* or *fog computing* in the recent years.

In edge computing, the offloaded computations are performed on a computationally powerful device, which is present at a location near the offloading mobile device. Such computationally powerful devices are called *cloudlets* [Satyanarayanan *et al.* 2009] or *edge devices* [Mahmud *et al.* 2018]. These cloudlets have advantages like small size, easy installation and low cost. However, they are computationally less powerful than the cloud servers. Cloudlets can mitigate the problems involved in using the cloud servers by introducing an additional layer that lies in between cloud servers and mobile devices.

However, the computational resources on these cloudlets must be managed efficiently to ensure that the mobile users who are offloading their tasks on to the cloudlets experience optimum Quality of Service (QoS). One of the most important aspects of QoS is the *latency* experienced by the users to receive the results of the offloaded tasks.

In this thesis, a network of cloudlets is considered to serve the offloaded task requests from the mobile devices. This ensures that no single cloudlet is heavily loaded by task requests. In such a scenario, it is pertinent problem as to how to identify a suitable cloudlet such that the latency in processing the offloaded tasks is reduced. Thus, one of the works done as a part of this thesis focuses on latency aware optimal task-assignment scheme in edge cloudlets. A brief introduction to latency aware task-assignment schemes in edge cloudlets is given in the next section.

1.4 Latency Aware Task-Assignment Schemes in Edge Cloudlets

As mentioned before, one of the objectives of having edge cloudlets is to reduce the delay experienced by the mobile devices offloading their tasks on to the cloud servers. Edge cloudlets are generally deployed in enterprises like offices, universities and even public places like shopping malls, airports, etc., for serving applications like image processing, speech processing, etc. Thus, these edge cloudlets generally experience variable traffic during the day. One of the most common strategies to handle the network traffic of incoming requests from mobile devices is that the cloudlet in the close vicinity of a mobile device serves the task offloaded by the mobile device. Though this strategy is simple to implement, it leads to imbalance in the load experienced by the cloudlets when more mobile devices are nearer to one or few cloudlets. The load experienced by the cloudlets directly affects the latency in processing the offloaded task requests. Thus, we need to employ a more sophisticated task-assignment scheme to process the offloaded tasks.

Deviating from the traditional approach of assigning the offloaded tasks to the nearest cloudlet, the Latency Aware task-assignment scheme (LATA) discussed in this thesis decides which cloudlet among a network of cloudlets will process an incoming task request from a mobile device. Thus, a task request received at a cloudlet can be served by any cloudlet in the network. These cloudlets are connected through a wireless Software Defined Network (SDN). LATA identifies a cloudlet in the network and assigns the task request to it so that the overall latency in processing the requests is optimized. To identify such a cloudlet in the network, LATA takes into account the maximum service rate of each cloudlet, the current load at the cloudlets and the distance of each cloudlet from the requesting mobile device. In addition, it is proved mathematically that LATA gives an opti-

mal solution to latency aware task-assignment problem. A detailed presentation of LATA is made in Chapter 4 of this thesis.

1.5 Organization of the Thesis

This thesis focuses on time synchronization protocols in WSN and latency aware task-assignment scheme in edge cloudlets. Chapter 2 presents a literature survey of the work done in the above-mentioned domains. The TSPs in WSN with a cluster-based topology are surveyed first where the specific features and limitations of the existing TSPs are highlighted. Further, a survey of some of the decentralized TSPs for WSNs is presented. Then the existing works in edge cloudlets with a special focus on the latency aware task-assignment schemes are discussed. Also, the gaps in the existing schemes which provides a motivation to formulate and implement a new task-assignment scheme are identified.

Chapter 3 presents a newly proposed TSP called *E-SATS*, which achieves synchronization in a cluster-based WSN. It also presents the mathematical basis of this protocol and discusses the methodology used by E-SATS to synchronize the nodes. Further, the experimental setup used used in this work is presented. Finally, this chapter also discusses the experiments carried-out to evaluate the performance of E-SATS in various LOS conditions and analyzes their results.

Chapter 4 presents a new decentralized TSP called Integrated Cooperative Synchronization (ICS) for WSNs. It first gives a brief background of the conventional Mean-Field (MF) technique used to achieve synchronization in WSNs. It then presents ICS which is based on the MF technique but integrates the measurement and message-passing phases generally dealt separately in the MF-based synchronization technique. The results of the evaluations of ICS in a WSN is then presented.

Chapter 5 presents LATA, a new latency aware task-assignment scheme in a network of edge cloudlets. This scheme assigns the offloaded task to a cloudlet in a way that the latency in processing these tasks is optimized. Mathematical proofs to show the optimality of this scheme are also discussed. Further, the evaluation of this scheme for varying number of cloudlets in the network and a comparison with existing task-assignment schemes are also presented.

Chapter 6 presents a summary of the work carried out in this thesis and the conclusions derived from this work. It also mentions some of the potential directions for future research in the domains explored in this thesis.

Chapter 2

Literature Survey

This chapter presents a review of the existing time synchronization protocols (TSPs) in Wireless Sensor Networks (WSNs) and task-assignment schemes in edge cloudlets. Firstly, we briefly describe the clock models used in the existing TSPs for WSNs and some of the major challenges encountered in designing a TSP for a WSN. We then present some of the important state-of-the-art TSPs in clustered WSNs highlighting their features, advantages and limitations. We then focus on the decentralized TSPs in WSNs where we discuss the existing approaches for achieving synchronization in a distributed manner. We also highlight the motivation for a new decentralized TSP which is implemented in this thesis. We then turn our focus to edge computing where we give an overview of the recent research on edge computing especially the task-assignment schemes used to assign the offloaded tasks on to the edge cloudlets.

2.1 Time Synchronization Protocols in WSNs- the Preliminaries

In this section, we briefly described some important concepts which form as the background for the survey of the existing TSPs for WSNs presented in Sections 2.2 and 2.3.

2.1.1 Delays during packet transmission and reception

As mentioned in the previous chapter, WSNs are typically deployed in harsh environments like mines, industrial plants, volcanoes, etc. Wireless communication in such environments experiences packet loss which leads to loss of timing data, need for re-transmissions, increase in synchronization error, etc. [Ting *et al.* 2015], [Schenato & Fiorentin 2011].

A TSP in a WSN typically involves exchange of packets among the participating nodes. These packets contain timing information like the local clock of the sender, reception time of the previous packets, etc. There are different delays which a packet experiences in the course of this transmission-reception operation. These delays can be categorized as deterministic delays and non-deterministic delays. Deterministic delays consist of the following:

- Transmission delay: Time taken for the packet to be transmitted by the transmitter.
- Propagation delay: Time taken by the packet to traverse through the medium.
- Reception delay: Time taken for the packet to be received by the receiver.

Transmission and reception delays are deterministic as they can be calculated by knowing the packet size and the transmission and reception speed of the transmitter and the receiver, respectively. Similarly, the propagation delay can be deter-

mined by knowing the distance between the transmitting and the receiving nodes and the speed of the wireless signal in the medium. Non-deterministic delays consist of the following:

- Send time: Time taken for the packet to reach the Medium-Access Control (MAC) layer of the transmitter node which depends on the current processor load at the transmitting node.
- Access time: Time taken for the transmitter to get an access to the transmission channel.
- Receive time: Time taken for the application on the receiver node to be notified about the incoming packet.

These delays are elaborately discussed in [Maróti *et al.* 2004] and [Sivrikaya & Yener 2004]. A synchronization protocol has to take into account these delays as they directly influence the accuracy of the synchronization achieved.

2.1.2 Clock Models

As mentioned before, the local clock of a node is maintained using a crystal oscillator. Two common models are used to model the local clock of a node. The first is the offset only model and the second one is the skew offset model.

In the offset only model which is used in [Ganeriwal *et al.* 2003], [Dai & Han 2004], [Noh *et al.* 2008], the local time of a node k at time t denoted as $C_k(t)$ is given by

$$C_k(t) = t + \beta_k, \tag{2.1}$$

where β_k is the offset of the local clock of the node k from the reference clock (referred to as global time). Since the crystal oscillators of all the nodes do not tick at the same rate, this offset-only model cannot give an accurate synchroniza-

tion of the clocks. The nodes which are synchronized using a synchronization protocol based on this clock model will experience a large synchronization error very quickly. Thus, such nodes require frequent re-synchronizations [Djenouri & Bagaa 2016].

In the skew-offset model used in [Elson *et al.* 2002], [Etzlinger *et al.* 2017], [Meyer *et al.* 2018], [Chalapathi *et al.* 2016], the local time of a node k at time t denoted as $C_k(t)$ is given by

$$C_k(t) = \alpha_k t + \beta_k \tag{2.2}$$

where α_k and β_k are the skew and offset of the local clock of node k, respectively with respect to the global clock. As compared to the offset-only model, this model captures the difference in the rate of local clock with respect to the global clock in addition to the offset. Thus, the nodes synchronized using a synchronization protocol based on this model will have higher synchronization accuracy and they require less-frequent re-synchronizations as compared to a synchronization protocol based on the offset-only model [Djenouri & Bagaa 2016]. It is to be noted that in reality, the skew of a node's clock is not constant with time, i.e., $\frac{d\alpha_k}{dt} \neq 0$. However, since skew changes very slowly over time, it is taken as a constant during the synchronization process [Sichitiu & Veerarittiphan 2003], [Sivrikaya & Yener 2004]. The skew-offset clock model has been considered in both the TSPs which were implemented as a part of this thesis Chapters 3 and 4. In the next section, a brief discussion on factor graph is presented. Factor graphs are used in depicting the formulation of the TSP presented in Chapter 4.

2.1.3 Factor Graphs in Message Passing based TSPs

In a WSN having a large number of nodes, a significant overhead (in terms of the number of computations performed, packets exchanged, etc.) is involved in forming a specific network structure like spanning-trees, mesh, etc. Thus it

is desirable to perform synchronization in a distributed or decentralized way to avoid this overhead. Research community has explored distributed synchronization schemes extensively over the past few years and some of them are surveyed in Section 2.3. Many of these works including the work presented in Chapter 4 of this thesis have used a Bayesian approach (i.e., using Bayes rule) to formulate the synchronization problem. An a-posterior probability density function (PDF) results by using a Bayesian approach to synchronization problem (this is shown in Chapter 4). A maximum a-posterior (MAP) estimate of the clock parameters is obtained to determine the clock parameters of the WSN node and synchronize the WSN nodes. Estimating these clock parameters is computationally intensive and thus many contemporary works [Etzlinger *et al.* 2014], [Ahmad *et al.* 2012], [Zennaro *et al.* 2013], [Leng & Wu 2011] have adopted a graphical approach using Factor Graphs (FGs) [Kschischang *et al.* 2001]. FGs have been used to represent the synchronization problem that is presented in Chapter 4. Thus, we give a brief introduction to FGs below.

Let us consider a five-variable function $g(x_1, x_2, x_3, x_4, x_5)$ which can be written as follows:

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5), \tag{2.3}$$

where A, B, C, D and E are the domains of the functions f_A, f_B, f_C, f_D and f_E respectively. The factor graph for this function is represented in Fig. 2.1. In this figure, the circles are called the variable nodes and the rectangles are called the factor nodes. The factor nodes represent the functions into which the function g factorizes into. The function g is sometimes called the global function and the functions into which it factorizes (in this case f_A, f_B, f_C, f_D and f_E) are called local functions [Ahmad *et al.* 2012]. The variable nodes represent the arguments of the global function. As we see from Eq. (2.3), each local function is dependent

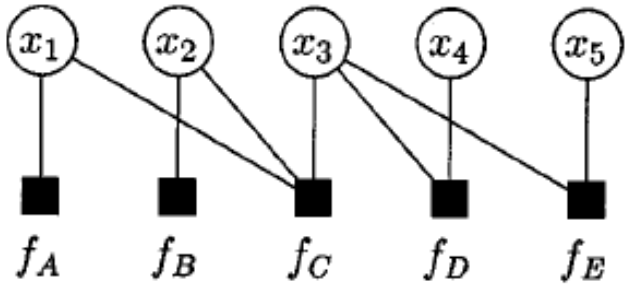


Figure 2.1: Example of a Factor Graph for the function $g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1)f_B(x_2)f_C(x_1, x_2, x_3)f_D(x_3, x_4)f_E(x_3, x_5)$. Source: [Kschischang *et al.* 2001].

on some of the variables of the global function. In the FG, each local function is joined to all the variables which are its arguments. Thus, FG is a bipartite graph which represents the factors of a global function and the variables each factor is dependent upon.

FGs are used to determine the marginal function of the global function. [1] To obtain marginal functions, each vertex of the FG is imagined to be a processor and the edges connecting these vertices are imagined to be the channels by which these vertices communicate with each other by sending "messages". The messages sent between these vertices are used to compute the marginal function. Messages communicated between two processors correspond to an appropriate description of a marginal function.

For computing a marginal function using FGs, the Sum-Product Algorithm (SPA) [Kschischang *et al.* 2001] is used. The SPA uses an update rule for its operation. Let $\mu_{x\to f}$ represent the message from variable node x to the factor node f and $\mu_{f\to x}$ be the message from f to x. Let $n(x)$ represent the set of neighbors of the vertex x in the FG. Then the update rule says that the message from a variable to

[1] The marginal function $g_i(x_i)$ is obtained by summing $g(x_1, x_2, x_3, x_4, x_5)$, i.e., $g_1(x_1) = \sum_{x_2\in B}\sum_{x_3\in C}\sum_{x_4\in C}\sum_{x_5\in D} g(x_1, x_2, x_3, x_4, x_5) = \sum_{\sim\{x_1\}} g(x_1, \ldots, x_5)$. This is also called summary of x_1 and $\sum_{\sim\{x_1\}}$ is called the summary operator.

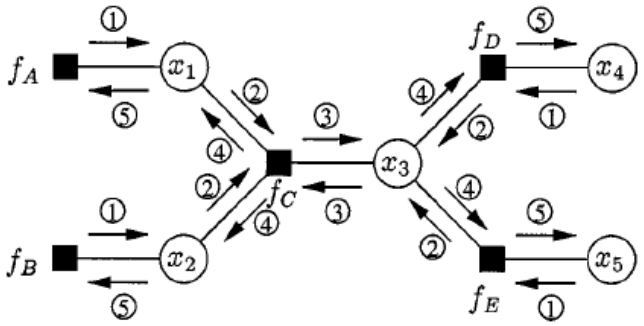


Figure 2.2: Example of a Factor Graph for illustrating calculation of the marginal. Source: [Kschischang *et al.* 2001].

a factor is given by

$$\mu_{x\to f} = \prod_{h\in n(x)\backslash\{f\}} \mu_{h\to x} \tag{2.4}$$

In the above equation, $n(x)\backslash\{f\}$ represents all neighbors of the node x except f. Further, the message from a factor to a variable is given by

$$\mu_{f\to x} = \sum_{\sim\{x\}} \Big(f(X) \prod_{z\in n(f)\backslash\{x\}} \mu_{z\to f}(z) \Big), \tag{2.5}$$

where X is the set of all the arguments of f or in other words $n(f)$. We can compute the marginal $g_i(x_i)$ as the product of two messages in two opposite directions on any edge connected to the node x_i. For example, in Fig. 2.2, the marginal $g_1(x_1)$ can be computed by considering the messages sent on any edge connected to x_1. Therefore, $g_1(x_1)$ is given by

$$g_1(x_1) = \mu_{f_A\to x_1}\mu_{x_1\to f_A} \tag{2.6}$$

$$= \mu_{f_C\to x_1}\ \mu_{x_1\to f_C} \tag{2.7}$$

We will use these concepts to estimate clock parameters in Integrated Cooperative Synchronization (ICS) protocol presented in Chapter 4.

A discussion on the existing TSPs for cluster-based WSNs is presented in the next

section.

2.2 Time Synchronization in Cluster-based WSNs

In a cluster-based WSN, nodes in the network form many groups of nodes called clusters. Each cluster consists of a leader node called a *cluster head* and few other nodes called *cluster members*. A typical cluster-based WSN is depicted in Fig. 2.3. In this figure, we find that there are three clusters, each with a cluster head and a few cluster members forming a cluster. Organizing a WSN into clusters is advantageous especially in saving the energy required by a node to transmit some information to a sink or a base station. The base station (which is also shown in Fig. 2.3) is the coordinator of the WSN and it acts as a gateway between the WSN and other networks. A base station is responsible for coordinating the network activities like data-collection, data-analysis, data-logging, node-association and disassociation, etc., whenever required in a specific application. It also provides protocol translation if the WSN has to communicate with another network using a different protocol.

In the absence of cluster-based topology, each node has to transmit a separate packet to the base station either in a hop-by-hop manner or by directly transmitting it to the base station. Direct transmission to the base station by every node will increase the traffic in the network and energy consumption. In a cluster-based network, the cluster head collects sensor data from all the cluster members, collates them and then transmits the collated sensor data to the base station. Thus, a single packet from a cluster head consists of information from all the nodes in a cluster. Therefore, this scheme reduces the energy consumed and traffic in the network for sending sensor data from all the nodes to the base station as compared to a scenario in which every node sends its sensor data to the base station separately.

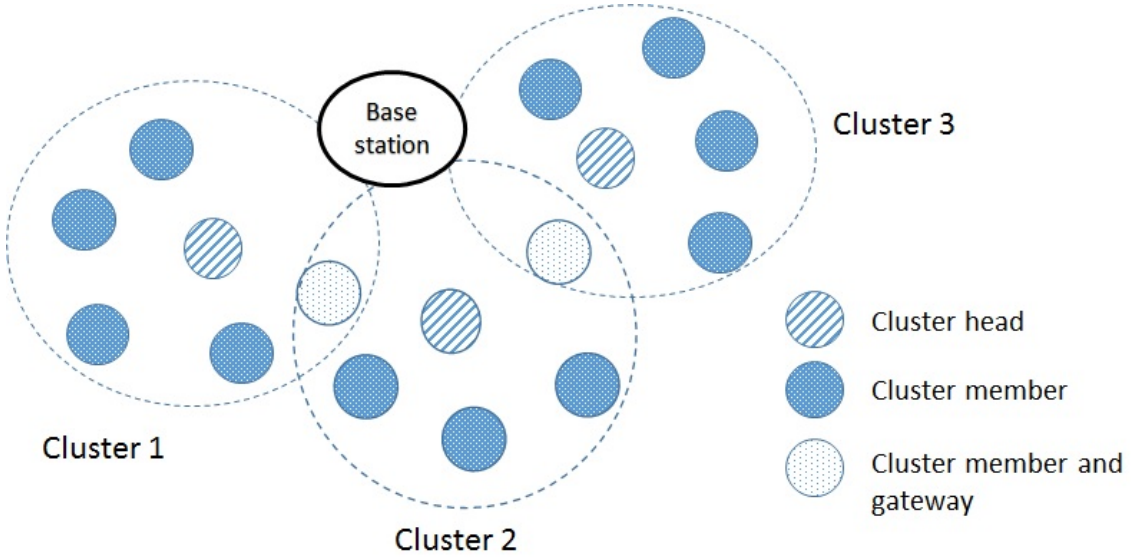


Figure 2.3: A typical cluster-based WSN

It has been shown in works like [Manjeshwar & Agrawal 2001], [Heinzelman *et al.* 2000], [Yang *et al.* 2004] that cluster-based topology is a very energy-efficient topology for WSNs particularly for operations like data-gathering, data dissemination, route-formation, etc., which are the most common operations in typical WSN deployments. Thus, it is pertinent to design time synchronization protocol specific to cluster-based WSNs. A TSP for a clustered WSN can take advantage of the structured organization of the network which leads to energy efficiency, faster synchronization, reduced network traffic, etc. A brief overview of the current state-of-the-art TSPs for clustered WSNs is presented in the following subsections.

The existing TSPs in clustered WSN can be categorized into four broad categories:

1. Hierarchical-based methods
2. Consensus-based methods
3. Regression-based methods
4. Other methods

Each of the above-mentioned methods are elaborated and an overview of the existing TSPs in each of these categories is presented in the following sub-sections. Also, each of these existing protocols are analyzed by mentioning their advantages and limitations.

2.2.1 Hierarchical-based Method

In hierarchical-based method, there exits a hierarchy among the cluster heads. As mentioned before, a base station generally acts as the gateway of the WSN to other outside networks. The cluster heads directly associated with the base station route the packets coming from the children cluster heads and send them to the base station. There are two prominent TSPs in this category: Cluster-based Hierarchical Time Synchronization (CHTS) and Time-Synchronization Algorithm Based on Cluster (CSSN). We elaborate on these two protocols below:

2.2.1.1 Cluster-based Hierarchical Time Synchronization (CHTS) for Multi-hop WSNs

CHTS [Kim *et al.* 2006] assumes that the network consists of two kind of nodes- High Performance oscillator equipped Nodes called HPNs and Low Performance oscillator equipped Nodes called LPNs. A network typically consists of a few HPNs (about 5-10% of total number nodes in the network) and the rest are LPNs. The network consists of a reference node called Cluster Head Reference (CHR) apart from cluster heads and cluster members. The cluster heads and CHR are HPNs, whereas the cluster members are LPNs.

A cluster head tree is constructed between the CHR and cluster heads with the CHR at the root. The CHR broadcasts a *CH Discovery* packet to initiate the discovery of cluster heads. This packet also contains a *hop* field which specifies the number of hops the sender is away from the CHR. The cluster heads which hear

the packet from the CHR identify CHR to be its parent and also note the *hop* field. These nodes re-broadcast the packet by incrementing the *hop* field by one. When a cluster head receives a *CH Discovery* packet it notes the *hop* field as well as the power level of the packet received, i.e., the Received Signal Strength Indicator (RSSI) value. If a cluster head receives the *CH Discovery* packets from two or more nodes, it selects a node with minimum hops from the CHR as its parent cluster head. If the *hop* field of two or more nodes is same and minimum, the node with higher RSSI is chosen as the parent cluster head. The idea is to minimize the number of hop counts to the CHR because the synchronization error increases with increase in the number of hops. This phase is followed by cluster member tree construction. Each cluster head sends a *cluster member discovery packet*. A cluster member after receiving these packets from multiple cluster heads chooses the node with highest RSSI value as its cluster head. A cluster head can communicate to all its cluster members directly, i.e., all the cluster members are within the radio range of the cluster head. However, the cluster head may not be within the radio range of a cluster member. In such cases, a cluster member chooses another cluster member which is closer to the cluster head as its parent. Thus, a hierarchical tree of cluster members is formed with the cluster head at the root. Thus the cluster may consist of two or three levels of cluster members called the first, second and third groups of cluster members. After this cluster member tree construction phase, the synchronization phase begins.

In the synchronization phase, CHTS very intelligently chooses an explicit synchronization method to synchronize the cluster heads to the CHR and an implicit synchronization method to synchronize most of the cluster members to its cluster heads. The cluster heads are synchronized first to the CHR. The CHR synchronizes the cluster heads which are directly connected to it, i.e., which are one hop away from it. The CHR broadcasts a packet to initiate this synchronization of these cluster heads. These cluster heads on receiving this packet back-off for a

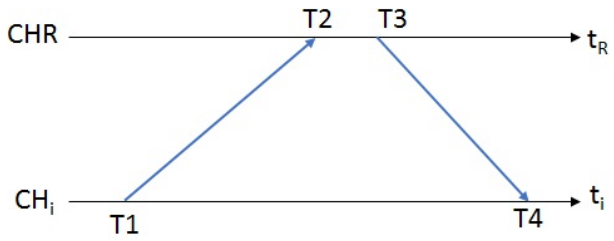


Figure 2.4: Illustration of two-way-exchange between CHR and CH_i

random amount of time. Then each cluster head sends a packet to the CHR one at a time. Let us say a cluster head CH_i sends a packet to CHR. This packet is time stamped as T1 by CH_i (as per CH_i's local clock) to record the time of transmission. The CHR receives this packet at T2 and sends an acknowledgment to CH_i at T3, where T2 and T3 are recorded as per CHR's local clock. The CH_i receives this acknowledgment at T4 (as per CH_i's local clock). The offset of CH_i is calculated as

$$Offset_{h,i} = \frac{(T2 - T1) - (T4 - T3)}{2} \tag{2.8}$$

The above procedure of packet sent by CH_i to CHR and reply sent by CHR to CH_i is called a *two-way exchange* as the packets are sent and received by both the nodes as illustrated in Fig. 2.4. In this figure, t_R and t_i refer to the time axis of local clock of CHR and CH_i respectively. This procedure is done by all the cluster heads directly connected with the CHR which we term as first-level cluster heads in the further discussion. The children cluster heads of the first-level cluster heads overhear their parent transmitting the packet with time stamp T1 and they take this as synchronization initiation packet. They back-off for random amount of time and they follow the same synchronization procedure as followed by the first-level cluster heads with CHR. Using this procedure, each cluster head calculates its offset with respect to the parent cluster head. This synchronization procedure is followed until the terminal cluster heads are synchronized.

After a cluster head gets synchronized with its parent cluster head, it synchronizes its cluster members by employing an implicit synchronization method as

mentioned before. It chooses a particular cluster member in the first-group of its cluster members and initiates a *two-way exchange* with it. It broadcasts a packet at T1 and the chosen cluster member say CM_j receives it at T2 and replies at T3. The packet is received by the cluster head at T4. The cluster head calculates the offset of CM_j using

$$Offset_{m,j} = (-1) * \frac{(T2 - T1) - (T4 - T3)}{2} \tag{2.9}$$

This offset and the value of T2 is broadcasted by cluster head which is heard by all the first-group and second-group cluster members. It is to be noted that all the second-group cluster members can hear the packets of cluster head but cannot directly transmit to the cluster head and thus they are in the second-group of cluster members. So all the first and second-group cluster members hear the packet broadcasted by the cluster head. Also the first-group cluster members (other than CM_j) and the second-group cluster members could hear the packet previously broadcasted by the cluster head with the timestamp T1. On receiving this packet, these first-group and second-group cluster members record the time of reception of this packet as $T2_{self}$. Thus when the cluster head broadcasts T2 and $Offset_{m,j}$, other cluster members can calculate their offset as

$$Offset_{m,k} = Offset_{m,j} + T2 - T2_{self}, \tag{2.10}$$

where k refers to a cluster member other than CM_j.

An advantage of CHTS is that it reduces the hop count of a cluster head to the CHR during the cluster-tree formation phase. During this phase, if a cluster head receives a reply from a potential cluster head parent with lesser hop counts to the CHR than its parent, it changes its parent and broadcasts this information to other nodes who might change their own parent cluster head. Though this reduces the hop-count of a cluster head to the CHR, it leads to increased in network traffic whenever change of cluster head parent takes place.

Further, CHTS requires two different kind of nodes- HPNs and LPNs. Though the number of HPNs in the network is 5-10% of total number of nodes in the network, but such specific requirement of cluster heads to be HPNs increases the monetary cost of the network. CHTS uses implicit synchronization method among the cluster members which reduces the number of packets required to synchronize the cluster members in a cluster. However, one of the major disadvantage about CHTS is that it is an offset-only synchronization procedure. In this synchronization, the skew of the clock is not estimated. Therefore it requires frequent re-synchronizations among the nodes.

Also, it uses a timer to keep track of the re-synchronization period. The period of the timer is calculated using the non-deterministic offset. This contradicts the very nature of this non-deterministic offset, i.e., it cannot be exactly determined or calculated. Thus it cannot be used to ascertain the period of this timer. CHTS does not account for the deterministic and non-deterministic delays that occur during any transmission. Thus it will not be able to achieve very good synchronization accuracy as is evident from the results of shown in this work.

2.2.1.2 Time-Synchronization Algorithm Based on Cluster (CSSN)

CSSN [Kong *et al.* 2010] forms a spanning-tree for the cluster heads just like CHTS. Thus CSSN forms multiple levels of cluster heads staring with the base station at the root. The base station, which is assigned level-0, broadcasts a *Sync_start* packet. The cluster heads who hear this packet assign themselves as level-1 and they send a *Sync_req* packet along with transmission timestamp of this Sync_req packet (T1). They perform two-way exchange with the base station just like in CHTS and calculate their offset with respect t to the base station. The cluster heads which have not been assigned a level yet and which hear the Sync_req packet of a level-1 node assign themselves as level-2. They also get synchronized with a higher level cluster head and thus all the cluster heads get synchronized.

During the synchronization, the offset is calculated in the similar way as it is done in CHTS.

The cluster heads then synchronize their cluster members. A cluster head constructs a set $\mathcal{S}$ consisting of cluster members used as reference broadcast nodes. This set $\mathcal{S}$ is constructed using the information about the cluster members available with the cluster head. A particular cluster member is selected from $\mathcal{S}$ and it is assigned a serial number beginning with 0. This cluster member, say CM_s, broadcasts a packet at transmission power such that the cluster head can hear this packet. This packet also contains the serial number of CM_s. All other cluster members record the reception time of this packet from CM_s and the serial number. The cluster head on receiving the broadcast from CM_s broadcasts its own reception time and the serial number to all the cluster members. Each cluster member now records the reception timestamp of the cluster head along with its own reception timestamp for the broadcast from CM_s along with the serial number. The cluster head removes CM_s from $\mathcal{S}$ and selects another cluster member from $\mathcal{S}$ to act as reference broadcast. The cluster head also assigns this newly selected reference broadcast node a serial number one more than the serial number assigned to previous reference broadcast node. Thus the above procedure is repeated until the $\mathcal{S}$ is empty. Finally each cluster member performs linear least square method on the reception timestamps recorded previously for the broadcasts from different reference broadcast nodes. Using this method, each cluster member calculates its clock offset with respect to the cluster head.

CSSN reduces the number of packets used in synchronizing the cluster members to the cluster head by using the reference broadcast mechanism. A major limitation with CSSN is that it only calculates offset (but not skew) while performing synchronization and thus it achieves very low synchronization accuracy of the order of milliseconds. It also does not account for the various deterministic and non-

deterministic delays which further worsen its synchronization accuracy. Further, since the synchronization of cluster heads use only one measurement to synchronize with the parent cluster head, the accuracy of synchronization will be badly affected by these deterministic and non-deterministic delays [Maróti *et al.* 2004] [Lim *et al.* 2016]. Also it does not have any mechanism to reduce the hop counts from a cluster head to the base station. This leads to increase in the synchronization error as we traverse from the base station to the terminal cluster heads.

2.2.2 Consensus-based Method

The TSPs discussed in the previous subsection attempted to estimate the skew and offset of their local clock with respect to a reference clock. The following are some of the problems with this approach to achieve time synchronization in a WSN:

i) The skew and offset of the WSN nodes are not actually known due to the involvement of deterministic and non-deterministic delays during the packet exchanges.

ii) It is sometimes not preferable to correct the local clock of a node to maintain the continuity of time. [Wu *et al.* 2015].

iii) It is disadvantageous to have a single node as a reference for other nodes as seen in hierarchy-based methods because a failure of the reference node leads to disruption of time-synchronization of the whole network.

Thus a class of TSPs called consensus based synchronization protocols has evolved. Some of the examples of consensus-based synchronization protocols for WSNs are ADMM [Zennaro *et al.* 2011], CMTS [Wang *et al.* 2017d], Average TimeSync [Schenato & Fiorentin 2011] and CCTS [Wu *et al.* 2015]. In the consensus-based method, the nodes maintain a virtual clock which they try to synchronize during

every synchronization phase.

Consider a set $\mathcal{N}$ consisting of all nodes in a WSN. Considering the skew-offset model representation of the local clock of node i (mentioned in Eq. (2.2)) we get the local time of i at time t, i.e., $C_i(t)$ using the following:

$$C_i(t) = \alpha_i t + \beta_i, \tag{2.11}$$

where α_i and β_i are the skew and offset of i with respect to the global clock. Let the skew-compensation parameter and offset-compensation parameter of the virtual clock of this node i be $\hat{\alpha}_i$ and $\hat{\beta}_i$ respectively. These parameters relate the local clock of a node with the virtual clock. The virtual clock of node i denoted by $\hat{C}_i(t)$ is defined as

$$\hat{C}_i(t) = \hat{\alpha}_i C_i(t) + \hat{\beta}_i = \hat{\alpha}_i \alpha_i t + \hat{\alpha}_i \beta_i + \hat{\beta}_i, \tag{2.12}$$

where $\hat{\alpha}_i \alpha_i$ and $\hat{\alpha}_i \beta_i + \hat{\beta}_i$ are the skew and offset of the virtual clock.

So by applying the consensus method, it is aimed to synchronize the virtual clocks of all the nodes i.e.,

$$\left.\begin{array}{ll} \lim_{t\to\infty} & \hat{\alpha}_i \alpha_i = \hat{\alpha}^C \text{and} \\ \lim_{t\to\infty} & (\hat{\alpha}_i \beta_i + \hat{\beta}_i) = \hat{\beta}^C, \end{array}\right\} \forall i \in \mathcal{N} \tag{2.13}$$

where $\hat{\alpha}^C$ and $\hat{\beta}^C$ are respectively the values to which the skew and offset of the virtual clock converge to after applying the consensus method. An overview of two consensus based TSPs for cluster-based WSNs namely- Cluster-Based Consensus Time Synchronization for Wireless Sensor Networks (CCTS) [Wu *et al.* 2015] and Cluster-Based Maximum Consensus Time Synchronization for Industrial Wireless Sensor Networks (CMTS) [Wang *et al.* 2017d] is presented in following subsections.

2.2.2.1 Cluster-Based Consensus Time Synchronization for Wireless Sensor Networks (CCTS)

CCTS [Wu *et al.* 2015] performs synchronization in two phases, viz., *intra-cluster time synchronization* which is followed by *inter-cluster time synchronization*. In CCTS, two virtual clocks, viz., intra-cluster virtual clock and network virtual clock are maintained by the cluster head. The intra-cluster virtual clock's skew and offset compensation parameters are updated by the cluster head and its cluster members during the intra-cluster synchronization in a cluster to achieve synchronization among these nodes. Similarly, the network virtual clock's skew and offset compensation parameters are updated by the cluster heads in the inter-cluster time synchronization phase to achieve a network-wide synchronization.

In the intra-cluster time synchronization, the cluster head first broadcasts its own local clock, skew-compensation parameter and intra-cluster virtual clock to the cluster members. It also records the transmission time-stamp of this packet as $C_h(t)$. Each cluster member after receiving this packet, records the time-stamp of reception of its local clock $C_j(t)$, its skew-compensation parameter and intra-cluster virtual clock and sends this information to the cluster head. The cluster head calculates the average of the skew-compensation parameters and the intra-cluster virtual clocks of the cluster members as $\hat{\alpha_h}^{av}(l)$ and $\hat{C_h}^{av}(l)$, where l represents the iteration index. The $\hat{\alpha_h}^{av}(l)$ is used to update the skew-compensation parameter of the intra-cluster virtual clock. The cluster head sends this updated parameter to the cluster members which then update their own skew-compensation parameter of the intra-cluster clock maintained at the cluster member. The cluster head then updates the offset-compensation parameter of the intra-cluster clock $\hat{\beta}_h(l)$ using $\bar{\hat{C}}_h^{av}(l)$ and sends it to the cluster members. The cluster members upon receiving this updated offset-compensation parameter from the cluster head update their own offset-compensation parameter of the intra-cluster clock main-

tained at each cluster member. This updation is performed in the form of iterations where $l = 1, 2, 3, \ldots$. These iterations are performed until the intra-cluster virtual clock converges.

After synchronizing the cluster members, the cluster head will participate in the synchronization of the inter-cluster virtual clock. The cluster heads of any two clusters having a common cluster member (called the gateway nodes as shown in Fig. 2.3) are referred to as "overlapping clusters". These overlapping clusters exchange the information of their clock parameters related to the inter-cluster virtual clock during inter-cluster synchronization and achieve a consensus on these clock parameters by iterative message exchanges.

CCTS achieves a network-wide synchronization using inter-cluster synchronization and it has been tested in simulations for a large network. It also uses a skew-offset model which therefore needs less frequent re-synchronization compared to offset-only model. However, CCTS has a very slow convergence and requires a large number of iterations [Wang *et al.* 2017d]. The reason for this slow convergence is attributed to the fact that CCTS first synchronizes the skew (in both the virtual clock synchronization phases) and then synchronizes the offset [Wang *et al.* 2017d]. The large number of iterations (as seen in Fig. 9 and Fig. 6 of CCTS [Wu *et al.* 2015]) used by CCTS to achieve microsecond accurate synchronization cannot be afforded by energy-constrained WSN nodes. Also, CCTS does not capture the deterministic and non-deterministic delays that occur in the communication of packets and thus it does not give good synchronization accuracy in practical WSNs which we will prove in Section 5.5 of this thesis.

2.2.2.2 Cluster-Based Maximum Consensus Time Synchronization for Industrial Wireless Sensor Networks (CMTS)

CMTS [Wang *et al.* 2017d] is another recently proposed consensus-based synchro-

nization protocol for cluster-based WSNs. CMTS, like CCTS, has two different synchronization phases, i.e., within a cluster (intra-cluster) and between the clusters (inter-cluster). CMTS maintains a single virtual clock at each node. To achieve consensus, CMTS uses the maximum value of skew and offset compensation parameters of the virtual clock of all the nodes in a cluster instead of the average value (as used in CCTS). By using maximum value, it achieves faster convergence compared to CCTS. Once all the cluster members are synchronized to their cluster head in a cluster, this synchronized cluster head serves as a time reference to the neighboring clusters. The inter-cluster synchronization is then achieved using gateway nodes. An improved version of CMTS called 'Revised-CMTS' is also presented in the same work[Wang *et al.* 2017d]. Revised-CMTS considers the communication delays and assumes that these communication delays have an upper-bound. It performs simultaneous synchronization of skew and offset of the virtual clocks. Though CMTS and Revised-CMTS have faster convergence than CCTS, they still need a large number of broadcasts in inter-cluster synchronization (as seen from Fig. 8 and Fig. 9 of Revised-CMTS [Wang *et al.* 2017d]). It needs around 2000 broadcasts for a 50 node network (as seen in Fig. 9 of Revised-CMTS [Wang *et al.* 2017d]) which is quickly draining the batteries of energy-constrained WSN nodes. We will test Revised-CMTS on a testbed and discuss its performance in Section 5.5 of this thesis.

2.2.3 Regression-based methods

Regression-based methods achieve synchronization among the nodes using linear regression method. Many TSPs use this method to achieve synchronization. SLTP [Nazemi Gelyan *et al.* 2007] and L-SYNC [Jabbarifar *et al.* 2010] are two prominent cluster-based TSPs of this category. We briefly describe these protocols below.

2.2.3.1 Scalable Lightweight Time Synchronization Protocol (SLTP)

SLTP [Nazemi Gelyan *et al.* 2007] consists of a *configuration phase* and a *synchronization phase*. Configuration phase is the cluster formation phase of SLTP in which all the nodes divide themselves into clusters. In the static mode where the nodes are stationary, a node called *eager node* begins the cluster formation by declaring itself to be a cluster head. It then broadcasts a packet for informing the neighboring nodes that it is a cluster head node. All the nodes which hear this packet become its cluster members. These new cluster members broadcast their status that they are cluster members. Any node which hears this packet and has not yet become a cluster head or cluster member becomes a cluster head. Each of these cluster heads broadcast a packet informing about their status. Thus this procedure repeats till each node in the network has become either a cluster head or a cluster member. A node which hears from more than one cluster head becomes a cluster gateway. In the dynamic mode where the nodes are mobile, only cluster heads are chosen in the configuration phase. The cluster members and cluster gateways are chosen in the synchronization phase.

In the synchronization phase, the cluster heads synchronize their respective cluster members by broadcasting their local times at random time intervals. The cluster members then use linear regression method to find their relative skew and offset with respect to the cluster head. In the dynamic mode, the configuration phase must be executed before every synchronization phase. Thus, there is a significant overhead in terms of performing the configuration phase for every synchronization in the dynamic mode.

2.2.3.2 Large Degree Clustering based Time Synchronization (L-SYNC)

L-SYNC [Jabbarifar *et al.* 2010] forms clusters in such a way that the overlapping between the clusters is minimized. It uses two algorithms called *SLACE-3*

and *ACE-UD* to achieve large size clusters with minimal overlapping. The ration behind these algorithms is that by achieving optimum number of clusters, the synchronization accuracy can be increased. Just like SLTP, L-SYNC uses a linear regression method to synchronize the cluster members to the cluster head in each cluster.

Both SLTP and L-SYNC simulate their performance over large-sized networks. However, they exhibit very high synchronization error of the order of milliseconds. Both SLTP and L-SYNC do not take into consideration of the non-deterministic and deterministic delays in the packet exchanges. Also, L-SYNC has very high overhead especially for the cluster formation phase which makes it unsuitable to large size networks and thus this protocol is not scalable with network size.

2.2.4 Other Methods

This category of TSPs consist of TSPs which cannot be categorized into any of the previously described categories. A prominent TSP for cluster-based WSN of this category is PC-Avg [Mamun-Or-Rashid *et al.* 2005].

2.2.4.1 PC-Avg

PC-Avg [Mamun-Or-Rashid *et al.* 2005] uses average of the local time of the cluster members to synchronize the cluster members. In this protocol, the cluster members send their local time to the cluster head periodically. The cluster head then calculates the average of the local times of the cluster members, sends this information to the cluster members and thus synchronizes the cluster members. This protocol exhibits a high synchronization error and also does not consider the non-deterministic and deterministic delays involved in communication. It also requires frequent re-synchronizations as it is based on the offset-only clock model.

2.2.4.2 PulseSS

Pulse-coupled synchronization and scheduling (PulseSS) [Gentz *et al.* 2016] aims to achieve simultaneous synchronization and scheduling in cluster-based WSNs. In this protocol, each node maintains a fine clock and a coarse clock. The time period of the fine clock is 'T' and the coarse clock advances by one for every 'L' periods of the fine clock. A node sends two beacons (or signals) to its cluster head called the start and end beacon to start and end the communication. The cluster head sends out acknowledgments when these beacons are received. These beacons are used to signal to other nodes that the cluster head is busy and they are also used in synchronization. The synchronization is achieved by these acknowledgments sent by the cluster head. This protocols also compensates for the propagation delays that occur in the communication. However, this protocol updates the phases (i.e., offset) of the clock only. Thus, it uses a offset-only model which necessitates more frequent re-synchronization. Also this protocol ignores the deterministic and non-deterministic delays in communication.

2.2.5 Research Gaps

The limitations of each of the existing protocols were mentioned previously in the respective subsections. Overall, the problems with the existing TSPs for clustered WSNs are as follows:

- All these protocols are simulator-based protocols and their suitability to practical WSNs has not been proven.
- These protocols (except Revised-CMTS) do not consider the deterministic and non-deterministic delays during the packet transmission and thus exhibit very high synchronization error.
- They do not consider the effects of the environment in which the nodes are

Table 2.1: Summary of existing works on time synchronization protocols for cluster-based WSNs

S. No.	Name of the existing work	Method of Synchro-nization	Deter. and Non-deter. delays	Clock model used	Synch. Accu-racy	Comm. traffic
1	CHTS [Kim *et al.* 2006]	Hierarchical	not accounted	offset-only	low	high
2	CSSN [Kong *et al.* 2010]	Hierarchical	not accounted	offset-only	low	high
3	CCTS [Wu *et al.* 2015]	Consensus	not accounted	skew-offset	medium	high
4	Revised CMTS [Wang *et al.* 2017d]	Consensus	assumes delays are bounded	skew-offset	medium	high
5	SLTP [Nazemi Gelyan *et al.* 2007]	Regression	not accounted	skew-offset	low	medium
6	L-SYNC [Jabbarifar *et al.* 2010]	Regression	not accounted	skew-offset	low	medium
7	PC-Avg [Mamun-Or-Rashid *et al.* 2005]	Other methods	not accounted	offset-only	low	low
8	PulseSS [Gentz *et al.* 2016]	Other methods	not accounted	offset-only	low	low

Note: Meaning of the short forms used in the above table: Deter and Non-deter. delays= deterministic and non-deterministic delays, synch. accuracy= synchronization accuracy achieved, comm. traffic= communication traffic in the network during the synchronization

deployed, i.e., whether the nodes are Line-of-Sight (LOS) or Non Line-of-Sight (NLOS).

The works surveyed in this section are summarized in table 2.1. We address the above-mentioned issues in a new TSP called E-SATS presented in Chapter 3. A survey of some of the important decentralized TSPs is presented in the next section.

2.3 Decentralized Time Synchronization Protocols for WSNs

In cluster-based WSNs, we need to organize the WSN nodes into clusters. The formation and maintenance of such a network structure proves to be a significant overhead especially large networks [Etzlinger *et al.* 2014], [Leng & Wu 2011]. Also centralized TSPs are not scalable and are vulnerable to node failures. Thus, decentralized TSPs have attracted significant interest in the recent years. In decentralized TSPs, the synchronization is done locally rather than at a central node for the entire network. In these TSPs, each node calculates its clock parameters by exchanging packets with its neighbors thus removing the need of centralized coordination or processing. There are many decentralized TSPs, viz., the consensus-based techniques like [Schenato & Fiorentin 2011], [Garone *et al.* 2015], Belief Propagation (BP) methods like [Leng & Wu 2011], [Etzlinger *et al.* 2014] and Mean-Field (MF) methods like [Etzlinger *et al.* 2013a], [Etzlinger *et al.* 2014].

As discussed before in Section 2.2.2, consensus-based methods achieve synchronization by agreeing to a common notion of time, i.e., they achieve a **consensus** on the current time of the common clock. Average TimeSynch (ATS) [Schenato & Fiorentin 2011] is a very simple consensus based technique. It is a cascade of two consensus techniques- one each for skew and offset. In ATS, each node makes a pseudo-periodic broadcast. This broadcast is periodic with respect to the local clock of the nodes. However, with respect to a global clock which gives the global time (as mentioned in Section 2.1.2), these local clocks have a different clock frequency, i.e., relative skew is not unity. Therefore, the inter-arrival intervals of two subsequent broadcasts are not equal and thus these broadcasts are called pseudo-periodic broadcasts. Based on these broadcasts, particularly the time of arrival of the broadcasts from the neighbor nodes, each node calculates the relative drift (or skew) with respect to each of its neighbor. It then uses these relative drifts to

perform a drift compensation to converge the value of skew of the virtual clock. Then, the nodes perform an offset compensation to converge the offset of the virtual clock. Though ATS is a simple protocol, it has slow convergence speed [Etzlinger *et al.* 2014], thereby requiring many rounds of packet transmissions. Further, ATS assumes instantaneous reception of the transmitted packets neglecting the deterministic and non-deterministic delays in packet communication.

RoATS [Garone *et al.* 2015] is another consensus method which tries to mitigate the short comings of ATS by assuming that the network delays during the communication are bounded. However, it exhibits high synchronization error of about 19.6ms for a 20-node and 100-node networks. Also it does not compensate for the propagation delay. It has been shown in [Lim *et al.* 2016] that the propagation delay, which is generally neglected by most of the TSPs, plays an important role in achieving highly accurate time synchronization. It is of special importance to estimate the propagation delay and to compensate for its effect, especially when sub-microsecond accurate time synchronization is required. Sub-microsecond synchronization accuracy is essential for many distributed network based control and event analysis applications [Lim *et al.* 2016].

LSTS [Tian 2017] is another TSP which improves ATS. Similar to RoATS, LSTS also assumes bounded delays in its synchronization scheme. However, it uses least-squares estimation method for synchronization process. This protocol also exhibits poor synchronization accuracy just like RoATS and it also does not compensate for the propagation delays in the communication.

Apart from consensus based methods, there are other methods in which the nodes estimate the clock parameters (i.e., skew and offset) by exchanging messages among themselves. These methods are commonly called as *message-passing* methods. They differ from the consensus methods in that instead of using clock parameters of each other, they use a global statistical model. This model contains

the time measurements of the nodes. Therefore, the message-passing methods consider the errors which arise from the non-deterministic and deterministic delays also. To further discuss about the message-passing methods which are of interest in this thesis, we consider a Bayesian estimator as given below. Let the vector $\boldsymbol{\vartheta}_j$ contain the clock parameters of the WSN node j, i.e., clock skew α_j and offset β_j. The maximum a-posterior estimate of the clock parameters, $\hat{\boldsymbol{\vartheta}}_j$ is given by

$$\hat{\boldsymbol{\vartheta}}_j = arg \max_{\hat{\boldsymbol{\vartheta}}_j} p(\hat{\boldsymbol{\vartheta}}_j|\mathbf{T_1}) \tag{2.14}$$

$$= arg \max_{\hat{\boldsymbol{\vartheta}}_j} \int p(\boldsymbol{\vartheta}_j|\mathbf{T_1}) d\boldsymbol{\vartheta}_{\bar{j}} \tag{2.15}$$

where $\mathbf{T_1}$ contains all the time measurements during the packet exchanges between the nodes, $\boldsymbol{\vartheta}_{\bar{j}}$ denotes the integration over all $\boldsymbol{\vartheta}_i$ (i.e., clock parameters vector of all the nodes in the network) except $\boldsymbol{\vartheta}_j$. It is a computationally intensive task to perform the marginalization in Eq. (2.15). Thus, message-passing methods are used to reduce the computational complexity and arrive at an approximate value of this marginal. This approximate value of the marginal is called as the belief denoted by $b(\boldsymbol{\vartheta}_j)$, i.e.,

$$b(\boldsymbol{\vartheta}_j) \approx p(\boldsymbol{\vartheta}_j|\mathbf{T_1}). \tag{2.16}$$

Factorization of the joint probability density function involved in the marginalization operation performed in Eq. (2.15), is essential for the convergence of the message-passing methods. Factor Graphs (FGs) introduced in Section 2.1.3 are used to visualize this factorization structure and it helps in interpreting the messages which are passed over the edges of the FG as probability density functions.

Although there are many message-passing methods, we focus on two message-passing techniques, viz., Belief Propagation (BP) and Mean-Field (MF) message-passing. BP and MF are two variational inference techniques used in Bayesian

statistics. Variational inference techniques are used to find an approximate probability density functions for the latent (or hidden) variables given the observations, i.e., when posterior probability is available. BP and MF methods comprise of the rules that govern the message-passing over the function and variable nodes of the FG. BP was initially used for machine learning using Pearl's algorithm [Pearl 1988]. MF, on the other hand, was initially used for a simplified theory of ferromagnetism But it was later used in neural networks. A free energy formulation is used in [Yedidia *et al.* 2005] to show the unified interpretation of BP and MF. A significant difference between BP method and MF method is that MF is guaranteed to converge while BP is not. Also, BP is computationally more intensive than MF method and gives more accurate computation of the marginal when compared to that of the MF method.

BP-based message-passing method has been used in many works for time synchronization in WSNs. BP based synchronization for the offset-only clock model based synchronization, is proposed in [Leng & Wu 2011]. However, as discussed before, offset-only model based synchronization protocols require more frequent re-synchronizations than the protocols based on skew-offset model. A synchronization protocol based on skew-offset model using BP and MF is proposed in [Etzlinger *et al.* 2014]. This work shows that the non-deterministic delays in the communication follows a Gaussian distribution and it analyzes the convergence behavior of both BP and MF for the time synchronization problem. It has been shown in this work that MF method performs the synchronization by each node exchanging its belief with its neighbors. The message that has to be sent to each node is same and thus a broadcast mechanism can be used to perform the message exchanges. In contrast to this, BP requires a separate computation (and transmission) of the message to each neighbor because the message to each neighbor is different. Thus it requires more computations and packet transmissions than MF method.

An MF-based simultaneous synchronization and ranging protocol is proposed in [Etzlinger *et al.* 2013a]. In this work, MF method is used to estimate the clock parameters and distances between the nodes in a distributed way. In [Etzlinger *et al.* 2017], BP method is used for simultaneous localization and synchronization. BP-based simultaneous localization and synchronization is proposed in [Yuan *et al.* 2016]. However, this work assumes dense deployment of anchor nodes (i.e.,nodes who provide the time reference) to linearize the likelihood function. Also this work does not consider the clock skew.

2.3.1 Research Gaps

A common approach followed by message-passing methods to achieve synchronization is that all the nodes first exchange packets containing the timing information. The timing information represents the time of transmission and reception of the packets at the transmitting and receiving nodes respectively. This phase is referred to as the measurement phase. The clock parameters are then estimated using the time measurements obtained in the measurement phase. This phase is referred to as the message-passing phase. The message-passing phase can begin only after all the nodes in the network have completed the measurement phase. Thus, there is a need to synchronize the measurement and message-passing phases so that all the nodes in the network perform each phase together and there is no overlap in these phases in any part of the network. This synchronization not just becomes challenging but also becomes an additional overhead for the network. Further, since the measurements obtained in the measurement phase are used in each iteration of the message-passing phase, the computational complexity involved in clock parameter estimation increases tremendously. (This point will be explained in more detail in Chapter 4.) Further, any new node joining the WSN has to wait until the next measurement phase to estimate its clock parameters. In such scenarios, we need to perform measurement phase periodically

and this further increases the overhead incurred due to the synchronization protocol. Thus, we require a simpler message-passing based TSP which can integrate both these phases, i.e., the measurement and message-passing phases. This will enable using each message exchanged for both the purposes, i.e., to obtain time measurements and also to obtain the estimate of clock parameters. A message-passing TSP which addresses the above-mentioned issues will be presented in Chapter 4.

A overview of task-assignment schemes in edge computing in presented in the next section.

2.4 Task-Assignment Schemes in Edge Computing

Mobile computing has made information and data processing ubiquitous. Yet, by its very nature, mobile hardware is inferior to stationary hardware in terms of performance because of limited computing power, storage and battery capacity of the former [Satyanarayanan 1996]. Cloud technology has found its application in recent years in mobile computing, commonly referred to as *Mobile Cloud Computing* (MCC). MCC is used especially for offloading computationally intensive tasks from the mobile devices on to the cloud servers [Qi & Gani 2012], [Barbera *et al.* 2014]. Apart from the challenges of a heterogeneous network [Sanaei *et al.* 2014], the MCC technology has associated limitations of high network latency and high transmission power involved in connecting with the cloud [Qureshi *et al.* 2011]. To mitigate these shortcomings, researchers have explored the efficient use of a network of supporting devices called *cloudlets* [Satyanarayanan *et al.* 2009], [Jararweh *et al.* 2013]. Cloudlets are also referred to as *fog devices* [Bonomi *et al.* 2012], [Stojmenovic 2014] or *edge devices* [Ahmed & Ahmed 2016].

Cloudlets have been used to support the cloud services to mobile devices in many real time applications few of which are given below. An architecture based on

edge computing is proposed in [Taleb *et al.* 2017a] to achieve ultra-low latency and network congestion reduction for the upcoming 5G mobile systems for application in smart cities. A novel method to manage data streams at the mobile edge to enhance scalability in IoT architecture is presented in [Sun & Ansari 2016]. Recently, a vehicle control system is proposed in [Sasaki *et al.* 2016], where resources are allocated dynamically. In this system, computation is switched between the edge devices and the cloud according to the network conditions to overcome instability in vehicle control caused by long latencies in the absence of cloudlets. For widely used mobile device applications like gaming, face recognition, etc., that involve heavy computation and require low latencies, the use of cloudlets has been demonstrated to be technically feasible and beneficial compared to direct communication with cloud servers [Soyata *et al.* 2012]. Surveys on edge computing are presented in [Mach & Becvar 2017], [Mao *et al.* 2017], [Yu *et al.* 2018], [Taleb *et al.* 2017b] while [Kumar *et al.* 2013] presents an early version of a survey on computational offloading in mobile systems. A survey on computational offloading in Mobile Edge Computing (MEC) especially in terms of the current state of standardization and current work on various aspects of offloading like mobility management, decision making in offloading, resource allocation on the cloudlets, etc., is presented in [Mach & Becvar 2017].

Recently, researchers have also explored the effect of combining task-offloading decisions with several other network parameters. In [Wang *et al.* 2017a], the problem of task offloading is addressed along with strategies employed for content caching in cellular networks using edge computing. The same authors have focused on managing task offloading along with interference management in [Wang *et al.* 2017b]. Many works like [Satyanarayanan *et al.* 2009] propagate the concept of Virtual Machines (VMs), which will be invoked on the cloud/cloudlet devices to execute the offloaded tasks from the mobile users. In [Plachy *et al.* 2016], a strategy to find an optimal placement of these VMs while optimizing the commu-

nication costs in an MEC environment is presented. This scheme finds an optimal trade-off between VM migration cost and reducing the communication cost from the VM to the mobile user.

Another important issue is to deal with in cloudlets is the dynamic mobility of the mobile users in a cloudlet network. This opens up other interesting problems-as to where should the services requested by a mobile device be deployed/run in an MEC network and where should the service be migrated to cope up with user mobility and/or network changes. In [Wang *et al.* 2017c], the problem of service placement in MEC is explored and the authors of this work coined MEC as Mobile-micro-cloud (MMC).

In offloading the resource intensive tasks on to the cloudlets, one of the primary considerations is to minimize latency by managing the network traffic to provide better QoS to the users. The traditional way of task offloading is to offload the task from the mobile device to the nearest available cloudlet with an intention to minimize the communication delay (the Round-Trip-Time (RTT)) between the mobile device and destined cloudlet. This approach however does not take into consideration of the current workload at the nearest cloudlet and thus leads to poor latency during heavy traffic conditions.

In an MEC environment, the devices are mobile, energy limited, and multiple cloudlets are available with possibly distributed specialized resources. These factors create a need for specific decisions regarding choice of cloudlet to offload the task, proportion of computation to be offloaded, and task distribution [Satyanarayanan 1996]. These details are also discussed in [Huerta-Canepa & Lee 2010]. More recently, [Zhang *et al.* 2016b, Sardellitti *et al.* 2015, Muñoz *et al.* 2015, You *et al.* 2017, Kao *et al.* 2017] have dealt with some or all of these issues for mobile cloud computing in different directions. In [Zhang *et al.* 2016b], energy of both computation and data transmission are minimized with latency as constraint,

while [Sardellitti *et al.* 2015] addressed a similar problem in Multi-Input-Multi-Output (MIMO) multi-cell systems. In [Muñoz *et al.* 2015], a joint optimization of energy consumption and latency by delivering a framework in Femto Access Point (FAP) network using MIMO radios is explored. In [You *et al.* 2017], an optimization of energy consumption using Time-Division Multiple Access (TDMA) and Orthogonal Frequency-division Multiple Access (OFDMA) based resource allocation to the mobile users is presented. In [Kao *et al.* 2017], a special scenario where the application can be represented as a serial tree task graphs is addressed. The authors of this work focus on optimizing the latency in computing the offloaded applications composed of many tasks/routines by mapping different tasks on to multiple computing nodes/devices. However, they focus on a scenario where the application can be expressed as serial trees, which may not be always possible. In [Zhang *et al.* 2016a], strategies to maximize the benefit of mobile cloud computing resources and the utilities on mobile devices (which in this case are vehicles) with latency constraints are proposed. In [Mao *et al.* 2016], an online algorithm to jointly decide CPU cycle frequencies of mobile devices, transmit power and offloading decision for minimizing latency and task failure is proposed. A generic problem of task offloading for intermittent connection between mobile device and cloudlet is considered in [Zhang *et al.* 2014]. This intermittent connection is modeled and solved to minimize the communication and computation costs. Here the application is divided into phases and the decision is made whether to execute each application phase locally or to offload on to a nearby cloudlet. A different case of [Zhang *et al.* 2014] is discussed in [Truong-Huu *et al.* 2014], where cost constraints are similar, but now a set of parallel tasks are to be processed on cloudlets. [Liu *et al.* 2016] have proposed a strategy to schedule the task at a mobile device either at the mobile device locally or at the cloudlet based on the state of the transmission unit and computation unit of the mobile device. This work analyzes the average power consumption of the mobile device and average delay of each task

at the mobile device and comes up with optimization problem to minimize the delay with power as the constraint.

It is to be noted that these works have considered cloudlets that are stand-alone and primarily serve mobile devices in their service area. In [Sun & Ansari 2017], a task-assignment scheme in a multi-cloudlet network is presented. In this scheme, the cloudlets are connected by SDN switches. The scheme focuses on identifying the mobile device generating the maximum traffic and that is assigned to the cloudlet which can serve it with minimum response time. This is followed by doing task-assignment for the mobile device having the next highest traffic in the same way. This goes on until all the mobile devices have been assigned to the cloudlets. There is another scheme proposed in [Mukherjee *et al.* 2019], which reduces the network latency in a multi-cloudlet network where the cloudlets are connected to each other. The tasks offloaded to one cloudlet can be served on any of the cloudlets in the network. The tasks to be offloaded in this scheme are offloaded to the nearest cloudlet and are served by them. However, when the nearest cloudlet is not able to serve this request, the task is processed on a cloudlet, which offers the minimum latency to process the request. However, this scheme does not account for the current load at the cloudlets which significantly affects the overall network latency.

2.4.1 Research Gaps

Many of the existing task-assignment schemes consider only a single cloudlet for processing an offloaded task from a mobile device. This leads to load balancing problem and increases the latency in processing the offloaded tasks. Other existing schemes which consider cloudlet network to process the offloaded tasks, do not consider the current load of the cloudlets while evaluating the latency that an offloaded task will experience if it is processed on a particular cloudlet. Thus it

leads to non-optimal task-assignment among the cloudlets.

The above-mentioned issues are addressed in the task-assignment scheme presented in Chapter 5. This work improves the network latency by also considering the cloudlet load and develops a framework for optimal task offloading in a multi-cloudlet network. A cloudlet network is considered in this work, which is connected by wireless SDN switches that enables the cloudlets to cooperatively serve the tasks offloaded by the mobile devices. Networking the devices via SDN switches allows separation of the control and data pane and gives greater flexibility for routing of the offload requests among the cloudlets. This ensures reduced latency for mobile devices, thereby improving QoS experienced by them and also balancing the load at the cloudlets.

2.5 Summary

In this chapter, an overview of some of the prominent TSPs for clustered-based WSNs, decentralized synchronization protocols for WSNs and task-assignment schemes for edge cloudlets has been presented. The shortcomings of the existing approaches is summarized below:

- Most of the existing TSPs for cluster-based WSNs are simulator-based implementations and thus there is no credible proof to show their suitability for a real world WSN deployments.
- Further, most of the existing TSPs use an offset-only model in their protocol which makes frequent re-synchronization of the nodes thereby making these protocols inefficient in terms of energy consumption. Some of the protocols which use the skew-offset clock model require too many packets for achieving synchronization among the nodes.
- Decentralized TSPs for WSNs have been proposed to avoid the overhead of

network structure formation. Message-passing based decentralized TSPs operate in two phases, viz., measurement phase and message-passing phases to estimate clock parameters of the nodes. Such operation in two distinct phases require additional mechanisms to indicate the completion of one phase so that other phase can be begun. Need of such addition mechanism not just forms an overhead but also becomes challenging in a huge network.

- In edge cloudlets, most of the existing task-assignment schemes consider a single cloudlet to process the incoming task offloaded by a mobile device. Even those schemes which consider a network of cloudlets to process these tasks do not consider current load of the cloudlets while assigning the tasks to them. Thus, they end up making non-optimal task allocation which increases the latency experienced by mobile devices thereby degrading the QoS.

The above-mentioned research gaps provides the motivation for the works presented in this thesis and the above-mentioned shortcomings are addressed through these works. We first present a new TSP for a cluster-based WSN in the next chapter.

Chapter 3

E-SATS: An Efficient and Simple Time Synchronization Protocol for Cluster-based Wireless Sensor Networks

3.1 Introduction

Cluster-based topology is a suitable topology for many Wireless Sensor Networks (WSNs) used in Internet of Things (IoT) applications. For example, the Smart Street Lighting System (SSLS) (mentioned in Chapter 1) implemented in Padova, Italy [Zanella *et al.* 2014], uses a WSN node on every street light for monitoring the lighting condition, air quality and weather conditions like temperature and humidity. In such applications, it is advantageous to group the nodes which are in close spatial proximity into clusters. The sensor data from different nodes in a cluster can be collated or fused at a cluster head. This operation is known as data fusion. The cluster head then forwards this collated sensor data to the sink

node (or also referred to as the base station node) in a hop-by-hop manner. This approach will reduce the number of packets needed to transmit the sensor data from each node to the sink node. The sensor data from each node is accompanied by its corresponding time information. Thus, it is very essential to synchronize all the nodes to a common clock to ensure that the timestamp accompanying the sensor data is coherent. Incoherent timestamps will lead to erroneous analysis of sensor data.

As mentioned in Section 2.2, there are many existing time synchronization protocols specifically for cluster-based WSNs like SLTP [Nazemi Gelyan *et al.* 2007], L-SYNC [Jabbarifar *et al.* 2010], CCTS [Wu *et al.* 2015], Revised-CMTS [Wang *et al.* 2017d], etc. However, almost all of them are simulator-based time synchronization protocols. Also, they do not consider the effect of LOS conditions during the analysis of their protocols. Therefore a time synchronization protocol called *Efficient and Simple Algorithm for Time Synchronization* (ESATS) is presented in this chapter that addresses the above-mentioned issues.

The contributions made in this chapter are as follows:

1. A simple and efficient synchronization protocol for a cluster-based WSN named E-SATS is presented.
2. E-SATS is shown to achieve micro-second level synchronization on a hardware testbed.
3. The effect of LOS conditions on the accuracy of E-SATS is also analyzed.
4. E-SATS is also shown to have significantly better synchronization accuracy, lower computational complexity and lower energy consumption compared to the existing synchronization protocols used in cluster-based WSNs.

The organization of this chapter is as follows. Section 3.2 presents a summary of the drawbacks of some of the existing TSPs for cluster-based WSNs which

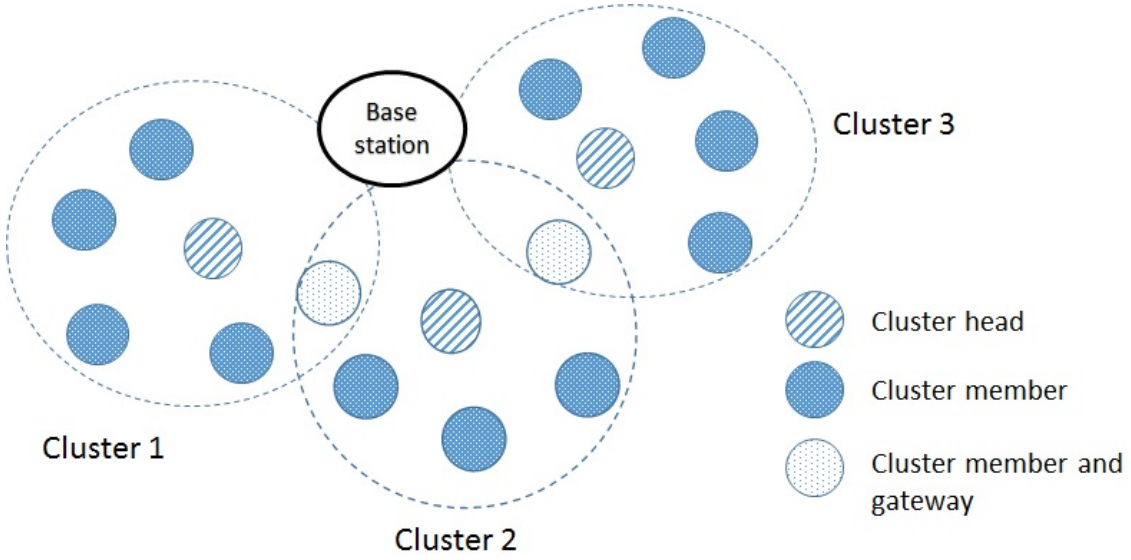


Figure 3.1: A typical network considered.

were reviewed in Section 2.2. Section 3.3 discusses the network model, clock model and the mathematical basis of E-SATS. Section 3.4 describes the testbed and the methodology used for the evaluation of E-SATS. Section 3.5 presents the results and analysis of the experiments carried out on the testbed in different LOS conditions. This section also presents a comparison of computational complexity and energy consumption of E-SATS with some of the existing synchronization protocols. Finally, Section 3.6 presents the conclusions.

3.2 Related work

This chapter focuses on time synchronization in WSNs with a cluster-based topology. A typical cluster-based WSN is shown in Fig. 3.1. A cluster consists of a cluster head and a few cluster members. There are a few cluster members called the gateway nodes which are cluster members of more than one cluster. These gateway nodes help in inter-cluster communication. Except for the gateway nodes, a cluster member communicates only to its cluster head via a unicast communication.

In this chapter, E-SATS will be compared with regression-based method (used in works like L-SYNC [Jabbarifar *et al.* 2010] and SLTP [Nazemi Gelyan *et al.* 2007]), CCTS [Wu *et al.* 2015] and Revised-CMTS [Wang *et al.* 2017d]. These protocols have been already discussed in Section 2.2. Therefore, we summarize some of the major drawbacks of these protocols which is a motivation to present a new synchronization protocol called E-SATS which addresses these drawbacks.

SLTP and L-SYNC do not consider the deterministic and non-deterministic delays that occur in the exchange of synchronization packets. Thus, they have high synchronization errors of the order of milliseconds. CCTS also does not consider these delays which occur during communication. Also, it exhibits very slow convergence rates due to which it requires many iterations to achieve synchronization. Revised-CMTS improves CCTS by considering communication delays with the assumption that the delays that occur during packet transmission and reception are bounded. However, Revised-CMTS also exhibits slow convergence and thus it requires many iterations to achieve synchronization among the nodes. Thus, both CCTS and Revised-CMTS have high overhead in terms of the number of packet transmissions during the synchronization which cannot be afforded by energy constrained WSNs.

A common drawback of all the above-mentioned synchronization protocols is that they are all simulation-based works, i.e., they have not been tested on real WSN platform. As mentioned in Section 1.2, simulation-based works do not give a complete and accurate understanding of the synchronization protocol in practical WSN deployment. Also, these works do not take into account the LOS conditions among the nodes in practical deployments. The performance of a synchronization protocol is greatly affected by the LOS conditions in which it operates (which will be proved in this chapter). Therefore, in this chapter, we present E-SATS which is a simple yet accurate time synchronization protocol tested on WSN testbed

in different LOS conditions. The preliminary version of E-SATS is presented in [Chalapathi *et al.* 2016] and [Chalapathi *et al.* 2019a]. E-SATS presented in this chapter has been reported in [Chalapathi *et al.* 2019b].

An elaborate discussion of E-SATS is presented in the next section.

3.3 Efficient and Simple Algorithm for Time Synchronization (E-SATS)

Before describing the E-SATS algorithm, we first describe the clock model and the network considered in this work.

3.3.1 Clock Model and the Network Considered

The network considered in this work is shown in Fig. 3.1. WSN nodes in this network are organized into clusters each of which has a cluster head and few cluster members. There are gateway nodes, which, as explained before, are part of more than one cluster and they help in communication between their cluster heads. The cluster heads are connected to a node called the base station or the sink node. The base station, as mentioned in previous chapter, acts as an interface between the WSN and other networks like the Internet when the WSN is used in IoT applications.

A static network is considered in E-SATS, i.e., the nodes in the network are stationary. Also, the links between any two nodes are symmetric links, i.e., if node p can communicate with node q, then q can also communicate with p. This is a reasonable assumption as WSN radios are usually capable of adjusting their transmission power levels to communicate with a particular node which is within its maximum radio communication range (the communication range when the maximum power level is used) [Wu *et al.* 2015]. Generally, WSN nodes use a

transmission power level lower than the maximum transmission level to minimize collisions with the packets transmitted by other nodes and also to conserve energy. Some other works like [Etzlinger *et al.* 2017], [Meyer *et al.* 2018], [Wu *et al.* 2015] have also made this assumption of symmetric links. The nodes which have higher computation and memory capabilities are identified as the cluster heads.

Let there be n nodes in a network and the set $\mathcal{N}$ represent all those n nodes. The cluster heads in the network are represented by $\mathcal{D} \triangleq \{d_1, d_2, ..., d_h\}$, where h is the total number of cluster heads in the network. Let the total number of cluster members in the whole network be u. Therefore, $n = h + u$. The set $\mathcal{T}_i$ represents the cluster members of the $i$thcluster head , i.e., d_i. Thus

$$\mathcal{T}_i = \{m \mid m \text{ is a cluster member of } d_i,\ d_i \in \mathcal{D}\}. \tag{3.1}$$

The number of cluster members of d_i is given by M_i. In the rest of the chapter, the jth cluster member of cluster head d_i is denoted as m_{ij}.

The skew-offset model discussed in Section 2.1.2 is considered in E-SATS. Each cluster member is synchronized to its cluster head. From Eq. (2.2), the local time of a cluster member m_{ij} with respect to its cluster head d_i at time t, represented as $C_{ij}(t)$, is given by

$$C_{ij}(t) = \alpha_{ij}\, C_i(t) + \beta_{ij}, \tag{3.2}$$

where $C_i(t)$ is the local time of the cluster head d_i. In the above equation, α_{ij} and β_{ij} are the relative skew and relative offset, respectively, of node m_{ij} with respect to d_i. The relative skew can be formally defined as the rate of the clock being considered with respect to a reference clock. The relative offset is the time difference of a clock being considered and the time at the reference clock at a

Table 3.1: Notation Summary

Notation	Meaning
$\mathcal{N}$	Set representing all nodes in the network
n	total number of nodes in the network
$\mathcal{D}$	Set representing all the cluster heads
h	Total number of cluster heads in the network
u	Total number of cluster members in the network
i	index of the cluster head
d_i	ith cluster head
$\mathcal{T}_i$	Set representing all the cluster members of d_i
M_i	Number of cluster members in the ith cluster
j	index of the cluster member
k	index of the iteration of communication between a cluster member and its cluster head
m_{ij}	jth cluster member of ith cluster head (i.e. d_i)
α_{ij}	Relative skew of m_{ij} with respect to d_i
β_{ij}	Relative offset of m_{ij} with respect to d_i
ζ_{ik}^{j}	deterministic delays in communication between m_{ij} and d_i in the kth iteration
$\eta_{ik}^{j}, \omega_{ik}^{j}$	non-deterministic delays in communication between m_{ij} and d_i in the kth iteration
Q	Total number of iterations in Synchronization phase
S	Total number of common event packets sent by *tester node* in Synchronization evaluation phase

given point of time. If we plot the time recorded on the clock being considered (as Y-coordinate) with respect to the time at the reference clock as X-coordinate, the relative skew is the slope of the line joining these points and relative offset is the X-intercept of this line. Such a plot is shown in Fig. 3.2. We will elaborate about this figure in Section 3.3.3. The notations used in this chapter are summarized in Table 3.1.

In E-SATS, there is a cluster formation phase when the network is initialized. This phase is followed by a synchronization phase.

3.3.2 Cluster Formation Phase

The cluster heads initiate the cluster formation phase. A cluster head broadcasts a *CM_assignment* packet containing its *nodeID*. Note that the cluster head can adjust the transmit power level according to the cluster area and the cluster size that is desired. When the nodes (which are not cluster heads) hear this packet, they identify themselves to be the cluster members of this cluster head and store the *nodeID* of the cluster head. If a cluster member hears the *CM_assignment* packets of more than one cluster head, it becomes a cluster member of all these cluster heads and identifies itself to be a gateway node. Further, the gateway cluster member unicasts the information about the *nodeID* of the cluster heads it is associated with to all its cluster heads. This helps a cluster head in identifying the gateway node to the neighboring cluster heads. Since the nodes are stationary, this cluster formation phase is performed only while initializing the network. However, a newly joining node can initiate a cluster discovery by sending a *CH_Discovery* packet. A cluster head which hears this packet, responds to this node by unicasting its acknowledgment to allow this node to join its cluster. If the newly joining node receives an acknowledgment from more than one cluster head, it will become a gateway node and it sends this information to all its cluster heads.

3.3.3 Synchronization Phase

The cluster head (say d_i) then synchronizes its cluster members by sending a *Synch_msg* packet at time $T_{1,1}$ as per its local clock. This packet contains the timestamp $T_{1,1}$ and is broadcasted to all its cluster members . On receiving this packet, the cluster members record the reception time. For example, cluster member m_{ij} will record the reception time as $T^j_{2,1}$. All the cluster members then respond to this *Synch_msg* by sending an acknowledgment packet at $T^j_{3,1}$ after backing off for a pre-determined amount of time. Note that this back-off time is different for each

cluster member so that collisions of the acknowledgment packets sent by them to d_i can be avoided. The acknowledgment sent by the cluster member contains $T_{1,1}$, $T^j_{2,1}$, $T^j_{3,1}$. The cluster head knows the back-off time observed by each cluster member. The cluster head d_i receives this packet at $T^j_{4,1}$. This iteration of cluster head sending the *Synch_msg* and cluster member sending an acknowledgment is repeated Q number of times. Each iteration collects four-time stamps for each cluster member m_{ij}, i.e., $T_{1,k}$, $T^j_{2,k}$, $T^j_{3,k}$ and $T^j_{4,k}$ (where the subscript k represents the iteration index and the superscript j represents the index of the cluster member). Note that the time stamp $T_{1,k}$ does not have the superscript j because it is common for all the cluster members. The back-off time observed by each cluster member is chosen in such a way that the duration of each iteration is kept small. The message exchanges for two cluster members m_{ij} and m_{is} with their cluster head are depicted in Figure 3.2. The superscript in the timestamps specifies the node to which the timestamp corresponds to. Thus, in E-SATS, in the ith cluster with cluster head d_i a total of M_i+1 packets are required in each iteration (1 broadcast packet and M_i acknowledgment packets from the cluster members). In contrast, SATS (the preliminary version of E-SATS) which was proposed in [Chalapathi *et al.* 2016] needs $2M_i$ packets in each iteration. This is because in SATS, a cluster head unicasts the *Synch_msg* and synchronizes the cluster members one-by-one. Thus, E-SATS is more energy efficient than SATS.

The timestamps collected for an iteration k of cluster member m_{ij} are related to each other as follows:

$$T^j_{2,k} = \alpha_{ij}(T_{1,k} + \zeta^j_{ik} + \eta^j_{ik}) + \beta_{ij}, \tag{3.3}$$

$$T^j_{3,k} = \alpha_{ij}(T^j_{4,k} - \zeta^j_{ik} - \omega^j_{ik}) + \beta_{ij}. \tag{3.4}$$

In the above equations, ζ^j_{ik} represents the deterministic delay and, η^j_{ik} and ω^j_{ik} represent the non-deterministic delays measured by local clock of d_i. Eqs. (3.3)

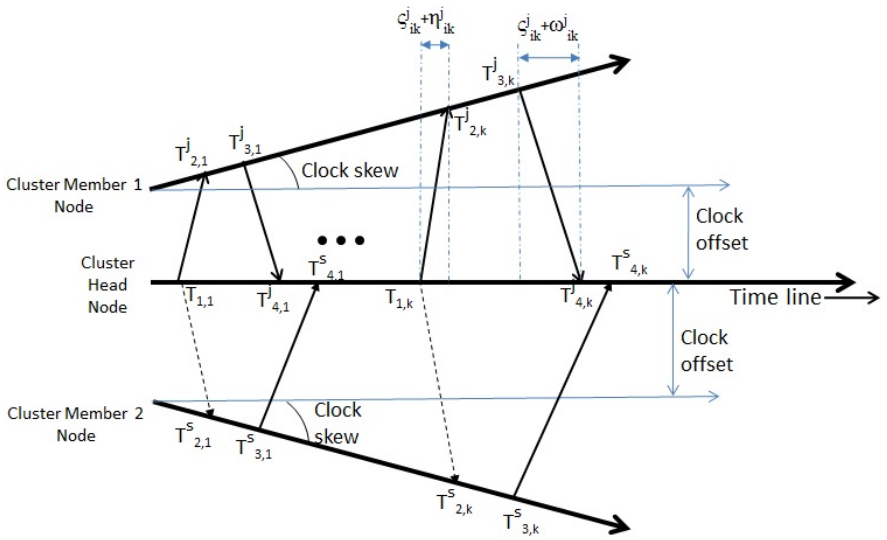


Figure 3.2: Synchronization message exchanges between a cluster head and two cluster member nodes m_{ij} and m_{is}.

and (3.4) can be rewritten as follows:

$$T^j_{2,k} = \alpha_{ij} T_{1,k} + \beta_{ij} + \alpha_{ij}(\zeta^j_{ik} + \eta^j_{ik}), \tag{3.5}$$

$$T^j_{3,k} = \alpha_{ij} T^j_{4,k} + \beta_{ij} - \alpha_{ij}(\zeta^j_{ik} + \omega^j_{ik}). \tag{3.6}$$

Let us consider the case where the timestamps of cluster member m_{ij} and the timestamps of cluster head d_i are plotted in Cartesian plane as shown in Fig. 3.3. The Y-axis represents the local time at m_{ij} and the X-axis represents local time at d_i. From Eq. (3.5), it can be seen that since α_{ij}, ζ^j_{ik} and η^j_{ik} are positive quantities, the line $L1 = \alpha_{ij} T_{1,k} + \beta_{ij}$ lies below the points $(T_{1,k}, T^j_{2,k})$. Also it can be observed from equation Eq. (3.6) that the line $L2 = \alpha_{ij} T^j_{4,k} + \beta_{ij}$ will lie above the points $(T^j_{3,k}, T^j_{4,k})$. So a good approximate estimate of the relative skew and offset of m_{ij} with respect to d_i can be obtained by fitting a line (say L3) that passes in between

L1 and L2. To obtain this line L3, two points A1 and A2 are identified such that

$$A1 = \{0.5(T^j_{4,b} + T_{1,b}),\ 0.5(T^j_{2,b} + T^j_{3,b})\}, \tag{3.7}$$

$$A2 = \{0.5(T^j_{4,a} + T_{1,a}),\ 0.5(T^j_{2,a} + T^j_{3,a})\}, \tag{3.8}$$

where $b = arg\ \underset{1\leq k\leq Q}{min}(T^j_{4,k} - T_{1,k} - \text{back-off time of } m_{ij})$ and $a = arg\ \underset{1\leq k\leq Q, k\neq b}{min}(T^j_{4,k} - T_{1,k} - \text{back-off time of } m_{ij})$.

The point A1 represents the iteration for which the deterministic and non-deterministic delays are minimum. Similarly, A2 represents the iteration for which these delays are the next minimum. Thus, the line L3 passes through the points A1 and A2. The slope of L3 gives the relative skew and its y-intercept gives the relative offset of node m_{ij}. So the estimated values of relative skew ($\hat{\alpha}_{ij}$) and relative offset ($\hat{\beta}_{ij}$) are obtained by the following equations:

$$\hat{\alpha}_{ij} = \frac{(T^j_{2,b} + T^j_{3,b}) - (T^j_{2,a} + T^j_{3,a})}{(T^j_{4,b} + T_{1,b}) - (T^j_{4,a} + T_{1,a})}, \tag{3.9}$$

$$\hat{\beta}_{ij} = \frac{(T^j_{2,b} + T^j_{3,b})}{2} - \hat{\alpha}_{ij}\frac{(T^j_{4,b} + T_{1,b})}{2}. \tag{3.10}$$

Though this is a simple and approximate method to calculate relative skew and offset, it gives an accuracy comparable to more sophisticated and computationally intensive maximum likelihood estimates as shown in [Chaudhari *et al.* 2008]. In E-SATS we use the gateway nodes to do the time translation whenever a packet is sent from one cluster to another. We do not employ a network-wide synchronization when it is not required. Thus by performing the inter-cluster synchronization on a need basis, we reduce the overhead involved in network-wide synchronization when it is not needed. Algorithm 3.1 summarizes the E-SATS algorithm.

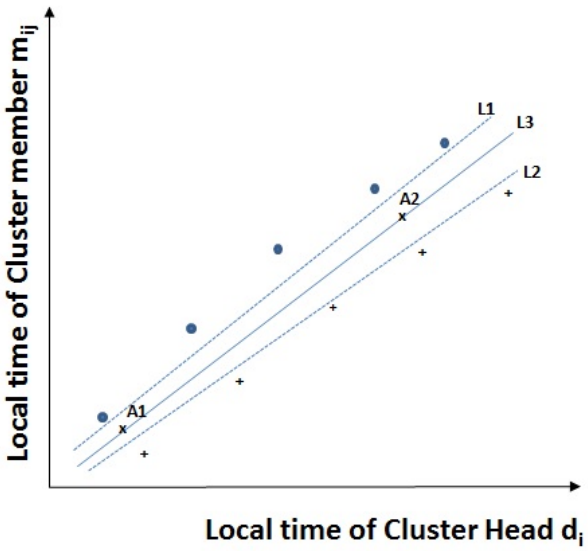


Figure 3.3: A diagram to illustrate the approximate estimate used to calculate relative skew and offset.

3.3.4 Use case for E-SATS

As mentioned before E-SATS will be useful in WSN networks in which cluster-based topology is used. Cluster-based topology is beneficial in applications where the nodes do not move frequently. We can consider the Smart Street Lighting System (SSLS) application [Zanella *et al.* 2014] implemented in Padova, Italy. As mentioned in the beginning of this chapter, each street light is equipped with a sensor node which has sensors to monitor lighting condition, air quality, temperature and humidity at a given location. A group of WSN nodes which are in close proximity to each other can be clustered together. We can also employ the cluster head rotation feature to avoid the draining of the battery of a single node which becomes a cluster head. The WSN nodes report the sensor data periodically and the sensor data is collated by the cluster head and forwarded to the Base Station which further forwards the sensor data to the city's data centre through the internet. The data centre is equipped with cloud servers to analyze the sensor data to make predictions about weather, to host information about weather and air quality for remote access, etc. It is very important for the sensor nodes

Algorithm 3.1 E-SATS Algorithm

Cluster formation Phase

1: Each cluster head d_i (where $d_i \in \mathcal{D}$) broadcasts *CM_assignment* packet along with its *nodeID*.
2: A node p (where $p \in \mathcal{N}$ and $p \notin \mathcal{D}$) hears this packet and assigns itself as cluster member of d_i, i.e., $\mathcal{T}_i = \mathcal{T}_i \cup \{p\}$.
3: If p hears *CM_assignment* packet of more than one cluster head, p becomes a gateway node and joins the cluster of all those cluster heads .
4: p sends the information of its cluster heads to all $d_l \in \mathcal{D}$ where $p \in \mathcal{T}_l$.

Synchronization Phase

1: **for** i=1:h **do**
2: **for** k=1:Q **do**
3: d_i broadcasts a *Synch_msg* at $T_{1,k}$
4: **for** j=1:M_i **do**
5: Node m_{ij} (where $m_{ij} \in \mathcal{T}_i$) receives it at $T^k_{2,j}$.
6: Node m responds at time $T^k_{3,j}$ after backing-off for a predetermined time.
7: d_i receives the packet at $T^k_{4,j}$
8: **end for**
9: d_i waits till it receives the acknowledgment from all cluster members .
10: **end for**
11: **for** j=1:M_i **do**
12: calculate $\hat{\alpha}_{ij}$ and $\hat{\beta}_{ij}$ using Eqs. (3.7), (3.8), (3.9) and (3.10)
13: **end for**
14: **end for**

to employ a time synchronization protocol so that sensor data that they report is also accompanied by an accurate time stamp (of the sensor data) which will help in accurate analysis of the sensor data. Since the sensor nodes are stationary in this application, we do not require to run the cluster formation phase except for the cluster head election algorithm. Also as it will be demonstrated in Section 3.5 that E-SATS is able to achieve microsecond-level synchronization accuracy which

is more than sufficient for this application.

3.4 WSN Testbed and Methodology Used

E-SATS is implemented on a WSN testbed consisting of TelosB WSN motes [Tel]. TelosB motes have MSP-430, a 16-bit microcontroller and CC2420 [CC2], an IEEE 802.15.4 compliant transceiver. There were a total of 30 WSN nodes used for testing E-SATS and comparing it with other synchronization protocols. These nodes were organized into 6 clusters and they were turned on one-by-one. In the current implementation, all the 30 nodes used in the network are identical, i.e., they have same computational and memory capabilities. Thus, 6 nodes were chosen and programmed as cluster heads. Since both the cluster members and cluster heads are battery powered, a cluster-head rotation scheme can be considered so that each node in a cluster can become a cluster head in turns and thus a single node is not overloaded. However, such a cluster head rotation is not implemented in the current evaluations.

To evaluate the performance of E-SATS, the WSN nodes were deployed in two different environments:

Line-of-Sight (LOS): In this kind of deployment, all the nodes can communicate LOS without any obstacles between them.

Mixed-LOS: In this kind of deployment, some of the nodes can communicate LOS without any obstacles between them while others are Non-Line-of-Sight (NLOS).

3.4.1 LOS environment

The image shown in Fig. 3.4 shows the deployment of nodes for the LOS environment experiments. The area of each cluster in this deployment is 20 sq. m. and the

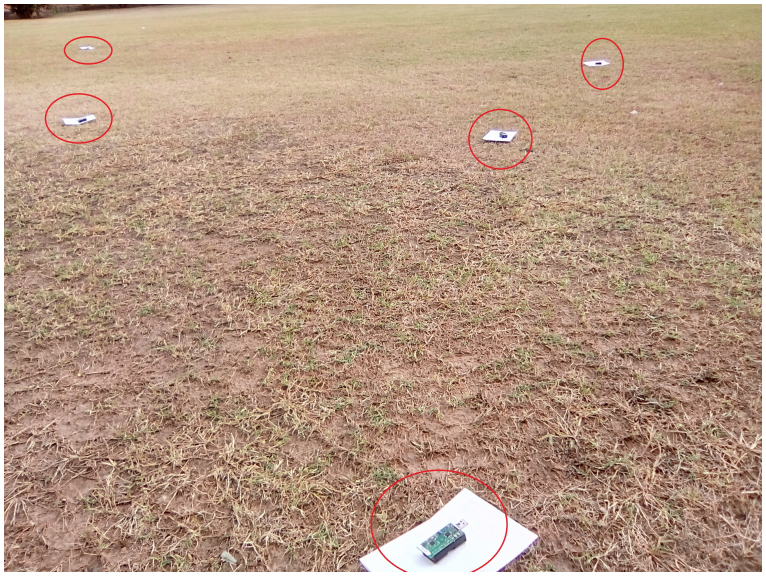


Figure 3.4: Picture showing part of the network (one cluster) deployed with WSN nodes encircled.

distance between two clusters is about 4 meters. Such a dense deployment is used to cause intense traffic which would evaluate the behavior of the protocol even in challenging situations. The clusters were activated one-by-one, i.e., initially only one cluster was active and gradually other clusters were powered-on one-by-one. When more than one cluster was operational, the packet exchanges in one cluster caused packet drops in other cluster(s).

3.4.2 Mixed-LOS environment

In this experiment, the cluster head could communicate to some of its cluster members with LOS while other cluster members were NLOS to the cluster head. Some of the nodes were deployed inside a lab while others were deployed in a gallery which is separated from the lab by thick concrete walls. Fig. 3.5a shows an indicative diagram of the deployment, while the image in Fig. 3.5b shows some of the nodes deployed in the lab. The presence of thick concrete walls will increase the communication delays among the nodes which are NLOS. In this experiment also, the clusters were turned on one-by-one and the performance of E-SATS along

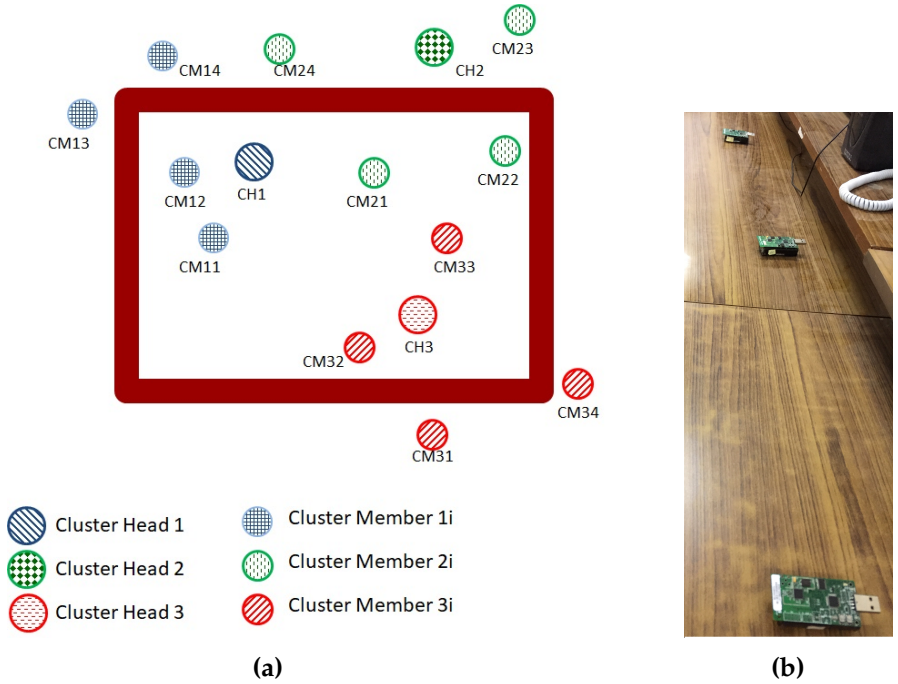


Figure 3.5: Set-up for mixed LOS environment (a) An indicative diagram depicting part of the deployment for mixture of LOS and NLOS environment (b) A picture showing some of the nodes deployed in the lab.

with other protocols was analyzed in each case.

3.4.3 Methodology Used

The methodology used to implement E-SATS is described in this subsection. The same methodology was followed for LOS environment and mixed-LOS environment. E-SATS was implemented in two phases, i.e., cluster formation phase and synchronization phase. Further, a synchronization evaluation is performed to calculate the synchronization error after the nodes were synchronized.

3.4.3.1 Cluster Formation phase

The cluster formation phase is performed as described in Section 3.3.2. Each cluster has a *Cluster_ID* for identification of the cluster. In the current implementation, there were 4 cluster members in each cluster. The energy consumed by E-SATS to perform the cluster formation is also estimated. Along the lines of [Shi *et al.* 2015],

[Marcelloni & Vecchio 2009], the information about the supply voltage, the current consumption and the data rate of transmission and reception operations of the radio of a node from the datasheets of CC2420 [CC2] has been used to estimate the energy consumption. If the supply voltage during a transmission/reception operation is V_{Rad}, current consumed by the radio is I_{Rad} and the time taken for this transmission and reception operation is t_{Rad}, the energy consumed by the radio during the transmission/reception operation can be obtained by fundamentals of electrical science which in this case is given

$$E_{Rad} = V_{Rad} \times I_{Rad} \times t_{Rad}. \tag{3.11}$$

The main challenge is to know the values of the supply voltage, current consumed during a transmission and reception operation. A detailed experimental analysis of transmission and reception operations for TelosB motes has been presented in [Amiri 2010]. The TelosB mote uses a voltage of 2.92V for transmission and 2.88V for reception [Shi *et al.* 2015], [Amiri 2010]. Also the size of the header and footer for any data transmission is 18 bytes for TelosB mote [Amiri 2010], [Abdelaal & Theel 2013]. The CH broadcasts *CM_assignment* packet as explained in Section 3.3.2 with its *nodeID* and a sequence number. Thus, the length of this packet along with the above-mentioned header and footer is 20 bytes.

The CC2420 radio can be programmed to have different transmission energy levels [CC2]. A transmission energy of -15dBm is used for the transmissions during these experiments. At this transmission level, the radio consumes 9.9mA for a transmission and 18.8mA for reception [CC2]. Since the radio uses a data rate of 250 kbps, it takes 6.4 ms each for both transmission and reception operations. Thus, we can calculate the energy consumed for a transmission and reception of this packet as 18.5 μJ and 34.65 μJ respectively. The total energy consumption for the 30-node network for all the packet transmissions and reception is 1289.18

μJ. It must be noted that this calculation is an approximate estimation without accounting for the energy consumed by the processor of the nodes which will be much smaller in this case.

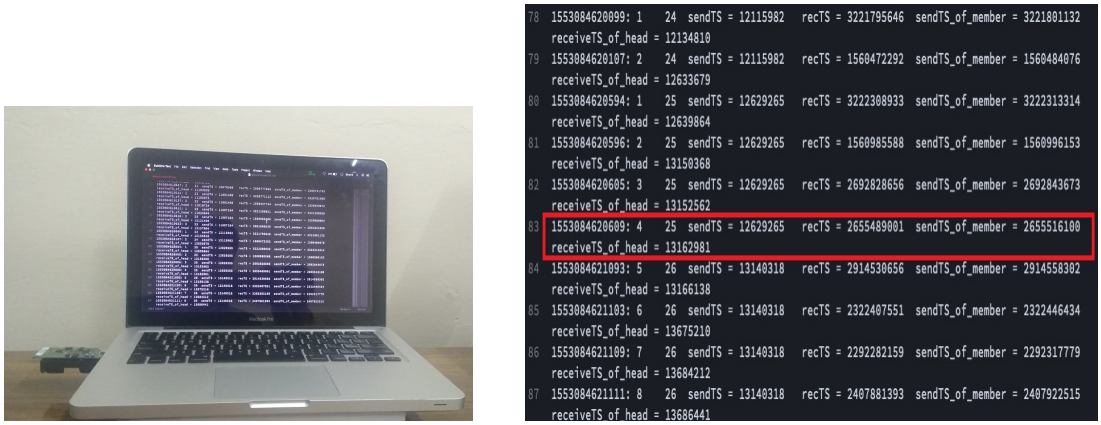
3.4.3.2 Synchronization phase

After clusters were formed, the cluster heads synchronize their respective cluster members by broadcasting the *Synch_msg* to them. Since the *Synch_msg* was broadcasted by the cluster head, all the cluster members in that cluster were able to hear this packet. They recorded the reception of this packet and responded to it by backing-off for a predetermined quantum of time. The cluster head then calculated the relative skew and relative offset of each cluster member (as discussed in Section 3.3.3) and unicasted these values to the corresponding cluster member. In the synchronization phase, 17 iterations were performed, i.e., Q=17. For monitoring, testing and debugging, the timestamps collected by the cluster head were also forwarded to the base station node which was connected to a PC as shown in Fig. 3.6a. Figure 3.6b shows one of the two-way exchange iteration of a cluster member encircled. The nodes were synchronized once every 1000 seconds.

As mentioned in Section 3.3.3, during the synchronization phase, the cluster head broadcasts a *Synch_msg* packet to synchronize the cluster members for every iteration. The cluster members backoff for a pre-determined amount of time and then send their acknowledgment to cluster head. This backoff time for the 4 cluster members in a cluster was 1ms, 5ms, 10ms and 15ms. These back-off times were observed in all the clusters.

3.4.3.3 Synchronization Evaluation

After the cluster members were synchronized to the cluster head, a good method to evaluate the synchronization error is that both d_i and m_{ij} observe a common event and record the occurrence of this event as per their local clock. These times-



(a) **(b)**

Figure 3.6: (a) The timestamps collected by each cluster head forwarded to the Base Station connected to a computer. (b) Timestamps of an iteration encircled in screenshot.

tamps will be used to calculate the synchronization error. Let us say that the common event was observed by d_i and m_{ij} at $t'_{e,k}$ and $t''_{e,k}$, respectively, as per their respective local clocks for the k-th common event packet (we generate the common event packets multiple times). Then m_{ij} can estimate the local time of d_i when d_i observed this event using Eq. (3.2). Let us say this estimate is $\hat{t}''_{e,k}$. The difference of $\hat{t}''_{e,k}$ and $t'_{e,k}$ will give the magnitude of synchronization error. Thus, synchronization error of the k-th common event is given by

$$e_k = |\hat{t}''_{e,k} - t'_{e,k}|. \tag{3.12}$$

Since the difference in the propagation time of this common-event packet to different nodes is very small, it will be a realistic assumption to say that this packet was observed by all the nodes in a cluster at the same time. This method of evaluating the synchronization error was used in these experiments while evaluating the synchronization error of E-SATS and comparing it with other synchronization schemes.

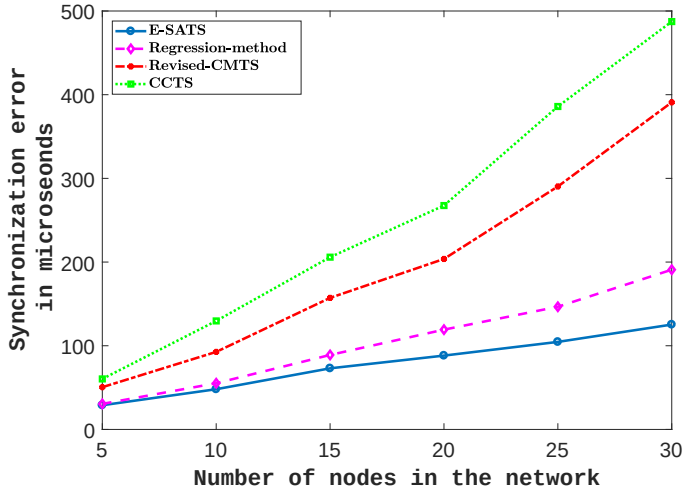
A WSN node which was not a part of the network was used to generate the common event by broadcasting a packet to all the nodes in the cluster. This node

is called *tester-node*. The transmission power of the tester-node was adjusted such that it can communicate with all the nodes in the cluster. This tester-node was used to test the synchronization accuracy of each cluster one-by-one. The tester-node sends 8 common-event packets and the average synchronization error was calculated. The synchronization evaluation was performed 10 seconds after the synchronization phase was completed.

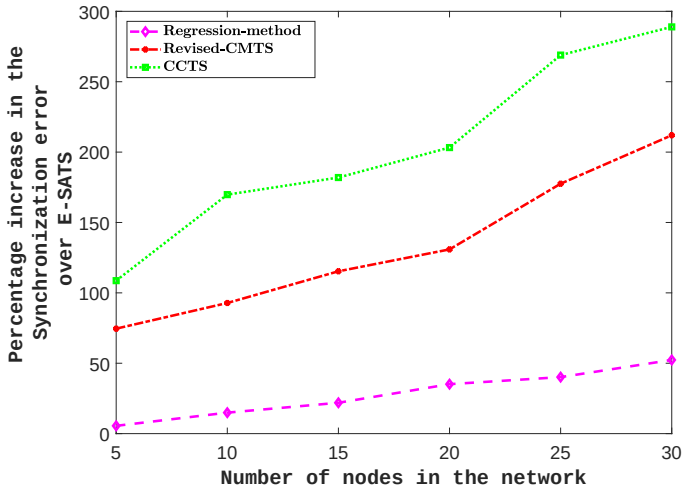
3.5 Results and Analysis

3.5.1 Performance Analysis in Terms of Synchronization Error

As described in the previous section, the WSN nodes were deployed in an LOS environment and in a mixed-LOS environment. The performance of E-SATS is compared in terms of synchronization error (in microseconds) with the regression-based method, CCTS [Wu *et al.* 2015] and Revised-CMTS [Wang *et al.* 2017d]. It must be noted that E-SATS performs synchronization within a cluster only. As explained in Section 3.3.3, it performs synchronization among the clusters through the gateway nodes only when there is a packet exchange from one cluster to another. Thus, to have a fair comparison of E-SATS with CCTS and Revised-CMTS, only the intra-cluster synchronization phases of Revised-CMTS and CCTS are compared with E-SATS. The regression-based method is the core of the synchronization schemes like SLTP [Nazemi Gelyan *et al.* 2007] and L-SYNC [Jabbarifar *et al.* 2010]. The synchronization error observed in LOS-environment is shown in Fig. 3.7a and the percentage of increase in synchronization error of other protocols over E-SATS is shown in Fig. 3.7b. In these figures, the regression-based method is referred to as 'regression-method' for brevity. We observe that when there are only 5 nodes in the network, i.e., one cluster, the synchronization error for all four synchronization protocols compared here are very close to each other. However, as more clusters are added to the network, the difference in the synchronization



(a)



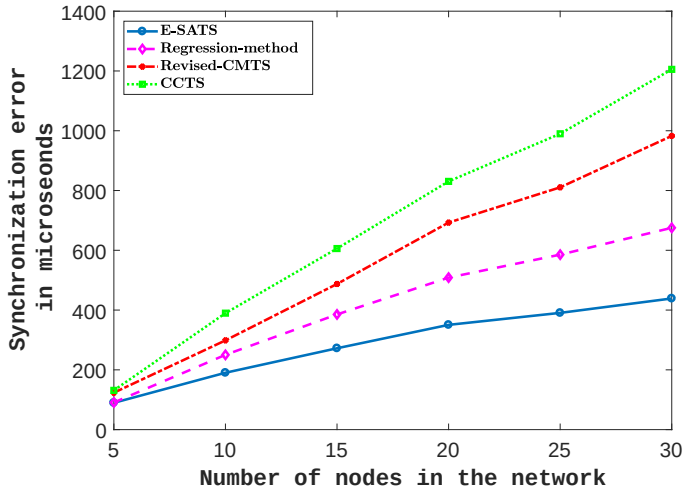
(b)

Figure 3.7: Results in LOS environment (a) Synchronization error of different synchronization protocols for varying network sizes (b) Percentage increase in synchronization error of other synchronization protocols over E-SATS for varying network sizes.

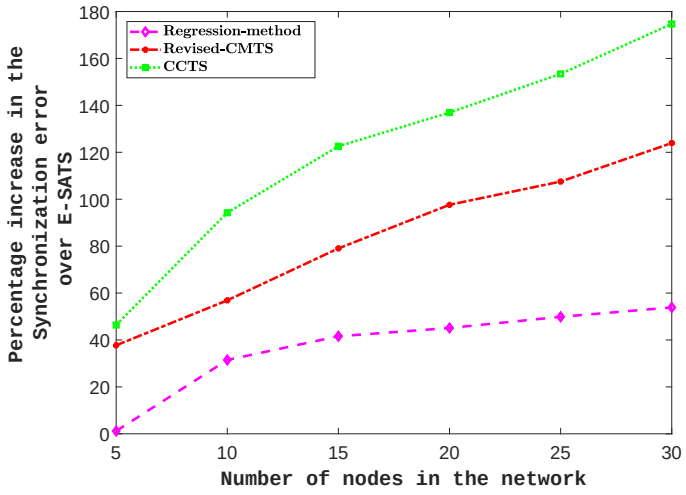
error widened. As the number of nodes in the network increases, the network traffic (i.e., number of packets exchanged) increases causing increased delays (especially non-deterministic delays). To have a fair comparison among the protocols being analyzed, the number of iterations have been kept the same while evaluating them. Regression-method takes into account all the packets to calculate the relative skew and offset. Thus, it will be affected by the delays in all the iterations and consequently its synchronization error is more than that of E-SATS. On the

other hand, E-SATS is affected by just two iterations which have minimal delays and thus it shows the least synchronization error. The reason for Revised-CMTS showing an increase in the synchronization error can be attributed to two facts. Firstly, it uses an upper bound of the communication delays in its computations which is just an estimated value. Thus, its results depend on the accuracy of this estimated value. Since the communication delays vary from iteration to iteration, it will lead to inaccurate estimation of compensation parameters. Secondly, as the results of Revised-CMTS [Wang *et al.* 2017d] also show, it takes a long time for Revised-CMTS to converge as the network size increases. As mentioned in Revised-CMTS [Wang *et al.* 2017d], it converges when minimum and maximum delays occur in successive iterations and the number of iterations we have taken may not be sufficient for it to converge. However, we cannot afford to perform a large number of broadcasts to allow the nodes to converge as it would drain the batteries of the WSN nodes. In the case of CCTS, it does not account for communication delays in its model and these delays significantly affect the synchronization accuracy. Therefore, as expected, CCTS has large synchronization error in a practical scenario. Further, CCTS also has the problem of large convergence time. The difference in the synchronization error of E-SATS and the consensus methods (both CCTS and Revised-CMTS) increases with increase in network size. It can be noted that E-SATS shows better synchronization accuracy compared to other three protocols for all the network sizes considered.

The results of the experiments performed in the mixed-LOS environment are shown in Figs. 3.8a and 3.8b. The synchronization error (in microseconds) of E-SATS, regression-method , Revised-CMTS [Wang *et al.* 2017d] and CCTS [Wu *et al.* 2015] is shown in Fig. 3.8a. The percentage of increase in synchronization error of other protocols over E-SATS is shown in Fig. 3.8b. In mixed-LOS environment also, when there are only 5 nodes in the network, the difference in the synchronization errors of all the four protocols was close to each other. How-



(a)



(b)

Figure 3.8: Results in mixed-LOS environment ((a) Synchronization error of different synchronization protocols for varying network sizes (b) Percentage increase in synchronization error of other synchronization protocols over E-SATS for varying network sizes.

ever, as the number of clusters in the network increases, the difference in the synchronization error also increases due to the reasons mentioned before. However, we see from Fig. 3.8b that improvement in E-SATS over CCTS for the 30 node network in mixed-LOS environment is 175% compared to an improvement of 289% observed in LOS environment. For Revised-CMTS, this improvement in the mixed-LOS environment was 124% compared to 212% in LOS-environment. It can be seen from the above results that in a NLOS environment, all the syn-

chronization schemes experience increased synchronization errors. This can be attributed to the fact that there is an increase in packet loss in NLOS communication. The synchronization error of E-SATS for a 30-node network was 438.82 μs in the mixed-LOS environment compared to 125.3 μs in LOS environment. The performance gain, i.e., improvement of E-SATS over the other protocols is lesser in the mixed-LOS environment compared to LOS-environment due to this increased synchronization error. Thus, we see a wide gap in the synchronization accuracy of a protocol in LOS and mixed-LOS environments. Thus, we conclude that NLOS communication significantly affects the performance of synchronization protocols and increases the synchronization error. However, E-SATS shows better synchronization accuracy than other protocols in both environments.

3.5.2 Comparison of Energy Consumption and Computational Complexity

The computational complexity of E-SATS, regression-method [Jabbarifar *et al.* 2010] [Nazemi Gelyan *et al.* 2007], CCTS [Wu *et al.* 2015] and Revised-CMTS [Wang *et al.* 2017d] is shown in Table 3.2. Note that this table is not specific to a particular LOS environment considered in the previous section and does not include computations due to packet drops. In this table, Q represents the number of iterations used in each synchronization phase, h represents the total number of clusters and u represents the total number of cluster members inn each cluster.

To understand how the number of arithmetic operations involved in E-SATS have been evaluated, let us consider a two-way exchange between a cluster member d_i and its cluster head m_{ij}. During the synchronization phase, for each two-way exchange, one subtraction (which can be seen as an addition) is required to calculate the duration of a two-way exchange which is computed by $(T^j_{4,k} - T_{1,k})$. Thus Q two-way exchanges require Q addition (or subtraction) operations. Further,

$2(Q-2)$ subtractions are required for comparison operations for determining the two-way exchange with minimum delays. Also, 4 additions are required in Eqs. (3.7) and (3.8) and 3 more in Eqs. (3.9) and (3.10). Thus a total of $3Q+3$ addition operations are required for synchronizing each cluster member by a cluster head. This synchronization of m_{ij} with d_i also requires a total of 7 multiplications, 4 of which are required in Eqs. (3.7) and (3.8) and the rest of the 3 multiplications are required in Eq. (3.10). Note that in Eq. (3.10), division by 2 in both the terms are taken as multiplication with 0.5. Further, one division operation is required in Eq. (3.9). To obtain the total number of operations for the whole network, we multiply by the above-obtained numbers with total number of members in each cluster, i.e., u and the total number of clusters in the network given by h. Thus we can arrive at the expressions mentioned in Table 3.2. Likewise, the number of additions, subtractions and divisions in regression-based method, CCTS and Revised-CMTS have been calculated.

Figure 3.9 shows the number of addition operations required in E-SATS and other protocols for different cluster sizes. E-SATS uses 1296 additions in a 30 node-network while the regression-based method requires 1608, CCTS requires 2346 and Revised-CMTS requires 3672 additions. Thus, E-SATS requires just 55% of the number of additions required by CCTS and 35% of that required by Revised-CMTS. E-SATS mainly requires addition (or subtraction) operations to identify

Table 3.2: Computational Complexity of E-SATS and other protocols (refer Table 3.1 for notation meanings)

Protocol	**Additions**	**Multiplications**	**Divisions**
E-SATS	(3Q+3)hu	7hu	hu
Regression-based method [Nazemi Gelyan *et al.* 2007] [Jabbarifar *et al.* 2010]	(4Q-1)hu	(2Q+6)hu	2hu
CCTS [Wu *et al.* 2015]	Qh(5u+3)	Qh(4u+4)	(3Qh)
Revised-CMTS [Wang *et al.* 2017d]	(9Qhu)	(5Qhu)	Qhu

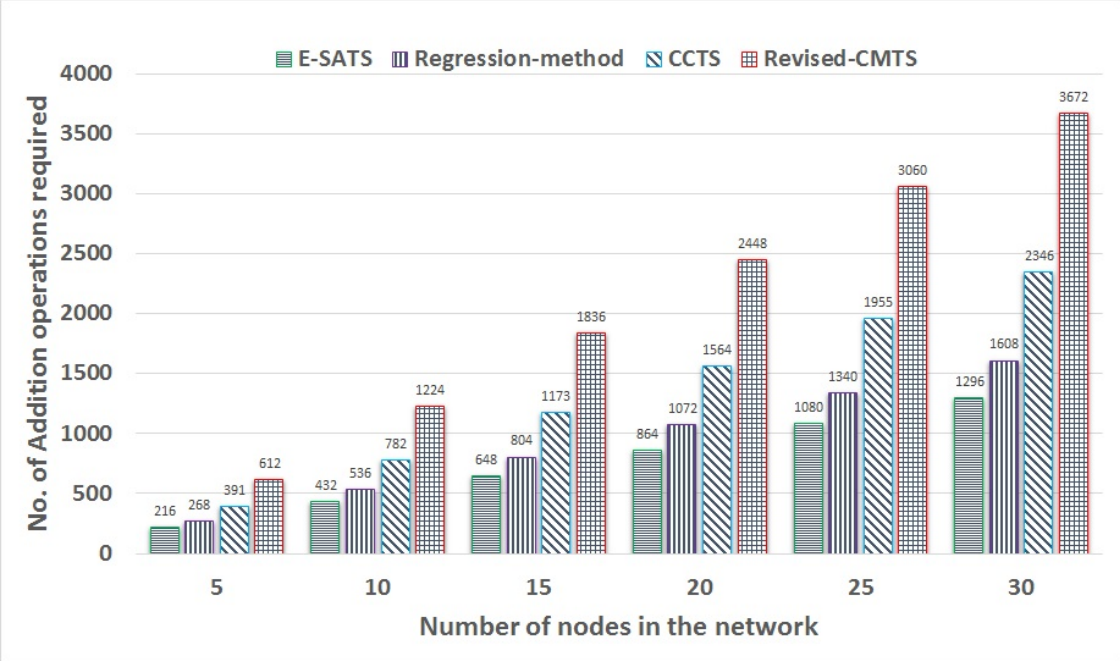


Figure 3.9: Number of addition operations in E-SATS and other protocols for varying network sizes.

two iterations with minimal duration (as discussed in Section 3.3.3). Once these iterations are identified, the computations involved in calculating the relative skew and offset are very simple. Thus, E-SATS requires comparatively lesser number of operations.

Figure 3.10 shows the multiplication operations required by E-SATS and other protocols. E-SATS has an enormous advantage over other protocols in the number of multiplication operations required to perform synchronization. For a 30-node network, E-SATS requires just 168 multiplication operations which is just 17.5% of the multiplications required by regression-method and 8.2% of that required by CCTS and Revised-CMTS. E-SATS requires multiplication operations only for the computations in Eqs. (3.7), (3.8), (3.9) and (3.10) which are evaluated only once per synchronization cycle. Thus, it just requires 7 multiplication operations per cluster member. However, since other protocols perform multiplication operations during every iteration, they require much higher number of multiplication operations to achieve synchronization. Note that both Revised-CMTS and CCTS use the

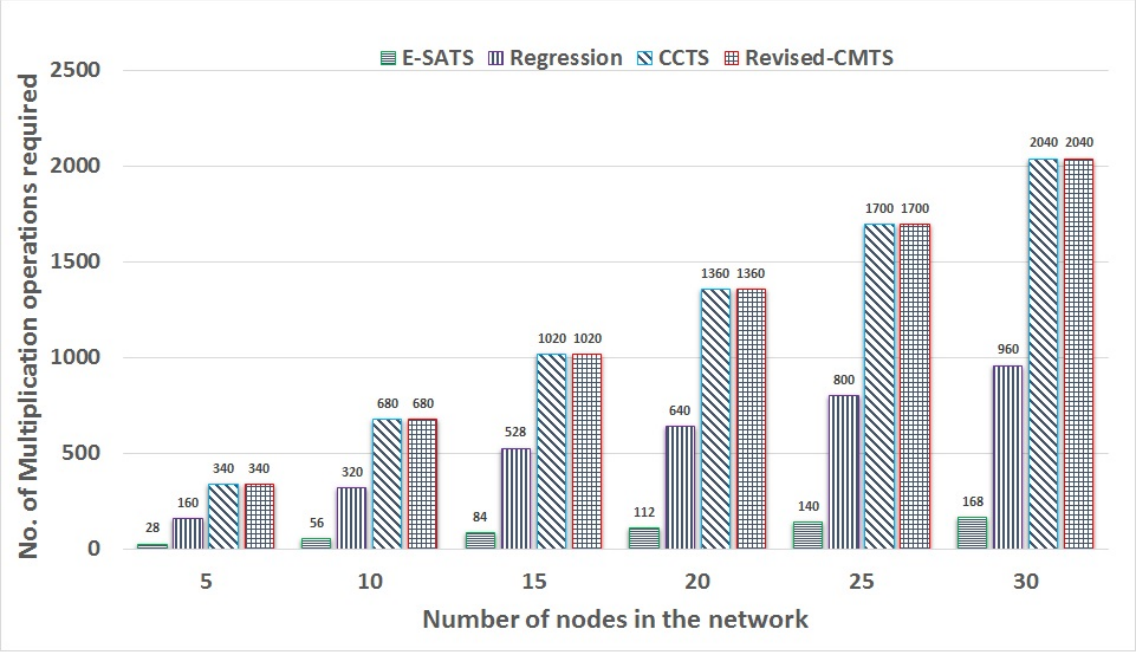


Figure 3.10: Number of multiplication operations in E-SATS and other protocols for varying network sizes.

same number of multiplication operations for the network used. However, these numbers will be different for other networks with different number of cluster members and cluster heads.

Figure 3.11 shows the division operations required by E-SATS and other protocols. In a 30 node network, E-SATS requires just 24 division operations which is the 50% of what is required by regression, 1.7% of that required by CCTS and 5.9% of what is required by Revised-CMTS. E-SATS just requires one division operation for each cluster member in Eq. (3.9). (The division by 2 in equation Eq. (3.10) is accounted in the number of multiplication as division by 2 can be seen as multiplication with 0.5.) Thus, in all the cases, i.e., from 5 node network to 30 node network, E-SATS requires the least number of additions, multiplications, and division operations.

The energy consumption of E-SATS, Regression-method, CCTS and Revised-CMTS for the transmissions and receptions during one synchronization phase in μ J is shown in Fig. 3.12. It must be noted that these energy estimations do not include any re-transmissions due to packet drops and it is not specific to any LOS

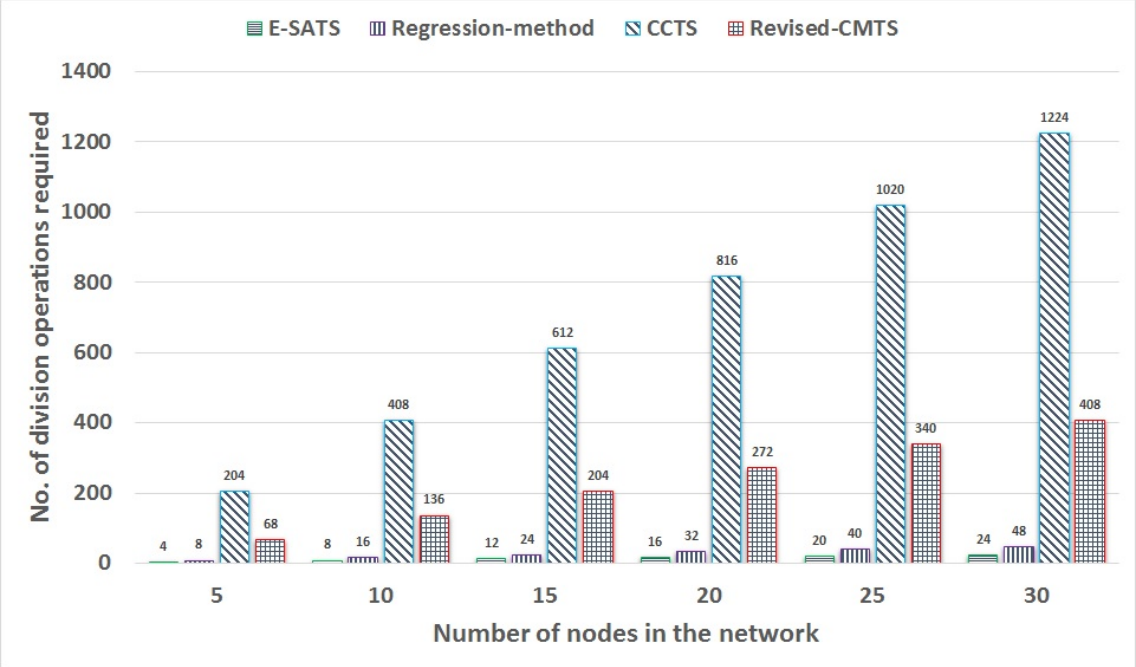


Figure 3.11: Number of division operations in E-SATS and other protocols for varying network sizes.

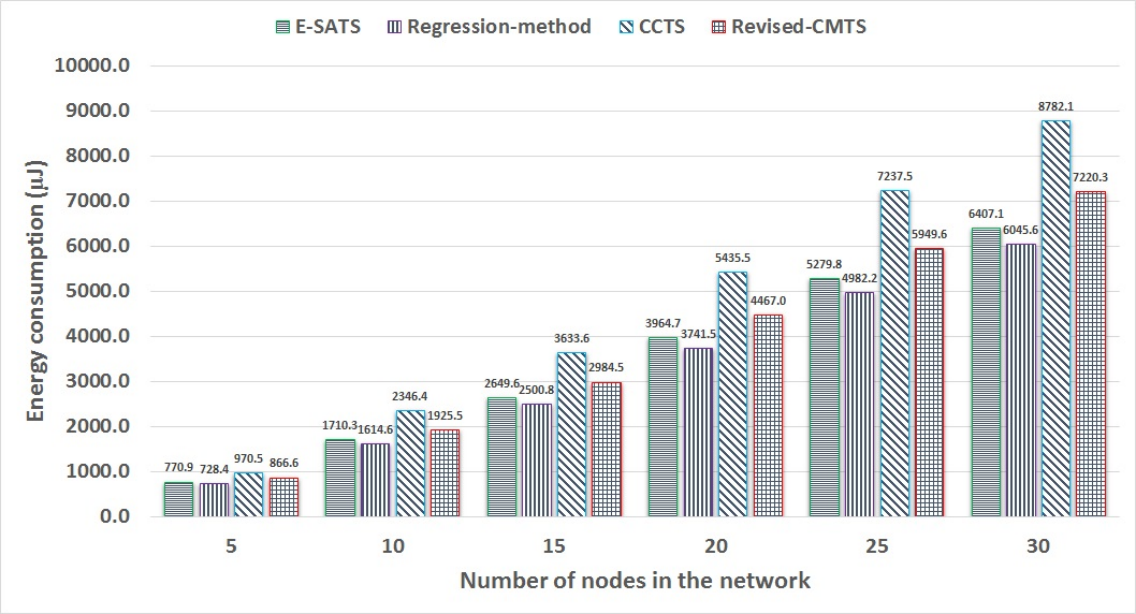


Figure 3.12: Energy consumption (in micro Joules (μJ)) of E-SATS and other protocols for the transmission and receptions during one synchronization cycle for varying network sizes.

conditions considered in the previous section. We have estimated the energy consumed for each protocol using the same method used in Section 3.4.3 to calculate the energy consumption for the cluster formation phase. The energy consump-

tion of E-SATS is slightly more than that of the regression-method. However, this difference is at most 361.4 μ J for a 30-node network. This difference is due to the fact that the regression-method uses smaller data payload for the packets exchanged though the number of packet transmissions and receptions in the regression-method and E-SATS are same. It must be noted that this graph shows the energy consumption of the radio only. The total energy consumption of a WSN node during a synchronization phase is roughly equal to the summation of the energy consumed by the radio (for transmissions and receptions) and the energy consumed by its processor (to perform the addition, multiplications and divisions). We have not shown the energy consumed by the processor of the node for these arithmetic operations and have just shown the number of operations required by each of the protocols during the synchronization phase. This is because arithmetic operations involving floating point numbers require variable number of clock cycles in MSP430 depending on the numbers involved in an operation [MSP]. To determine the exact number of cycles (and thus the energy consumption) of these operations will require computationally intensive methods like profiling which cannot be performed on the energy constrained WSN nodes. However, we note that most of these operations are floating point data operations requiring large number of cycles. For example, according to [MSP], a C library of MSP430 requires 427 cycles to perform a multiplication of an integer and a floating point number. The current consumed by MSP430 on TelosB mote which runs at 4.1MHz is 2.33 mA during a multiplication operation [Prayati *et al.* 2010]. So the regression-method which requires 960 multiplications (in 30-node network) compared to 168 multiplications of E-SATS will consume 569.615 μ J more energy than E-SATS (if the multiplications involve an integer and a floating-point number). We note that this excess energy consumed by the regression-method for multiplication operations (569.615 μJ) is much more than the energy it has saved compared to E-SATS in the transmission and reception operations (i.e., 361.4 μ J).

Further, a multiplication of two floating point numbers requires many more cycles which only increases the energy consumption of the regression-method compared to E-SATS. The multiplications involved in the calculations of these synchronization protocols mostly involve two floating point numbers. Thus, we can conclude that E-SATS consumes least energy among the compared synchronization protocols. Further, E-SATS offers better synchronization accuracy than the compared protocols.

3.6 Conclusion

In this chapter, a simple and efficient TSP called E-SATS has been proposed for WSNs with a cluster-based topology. E-SATS has been tested on a practical WSN testbed in an LOS environment and a mixed-LOS (mixture of LOS and NLOS) environment. It achieves micro-second level accurate synchronization in both the environments. This protocol has been found to outperform other state-of-the-art synchronization protocols for clustered WSNs in the two environments considered, both in terms of synchronization accuracy and also in terms of a number of computations. It also consumes lesser energy than the other protocols. These features make E-SATS a suitable synchronization protocol for resource constrained WSNs having cluster-based topology.

Chapter 4

Integrated Cooperative Synchronization for Wireless Sensor Networks

4.1 Introduction

As discussed before, clock synchronization is one of the basic building blocks for many applications in WSNs or the Internet of Things (IoT) [Wu *et al.* 2011]. WSNs generally involve dense deployment of large number of nodes. This high node density advocates the application of decentralized synchronization methods [Etzlinger *et al.* 2014], [Etzlinger *et al.* 2013a], [Schenato & Fiorentin 2011], [Zennaro *et al.* 2011], [Tian 2017]. From the above cited time synchronization protocols, delay compensated approaches discussed in [Etzlinger *et al.* 2014], [Etzlinger *et al.* 2013a], [Tian 2017] outperform approaches presented in [Zennaro *et al.* 2011], [Schenato & Fiorentin 2011] as the inaccuracies in that result from the communication delay between nodes are mitigated.

From an implementation perspective, the non-delay compensated method pre-

sented in [Schenato & Fiorentin 2011] is a very efficient method. In this method, each node records the time-of-arrival of packets broadcast by its neighbors. With the content of the received packet, i.e., the transmission timestamp and the current clock parameter estimate of the sender, the receiver node can update its own clock estimate with simple computations, and then broadcast it to the neighbors. This conceptual simplicity sets a benchmark for decentralized synchronization schemes.

In the message passing methods, the Mean-Field (MF) based synchronization protocols described in [Etzlinger *et al.* 2014] and [Etzlinger *et al.* 2013a] are of particular interest. This is because MF-method inherently supports a broadcasting mechanism of synchronization information while having only moderate computational complexity and guaranteed convergence. However, its biggest shortcoming compared to [Schenato & Fiorentin 2011] is the requirement of two operational phases that are necessary for practical implementation. First, each node has to collect time measurements with all its neighbors and store them for the second phase (requiring additional memory resources). Second, the nodes have to iteratively update and broadcast their local messages. In contrast, the heuristic method in [Lim *et al.* 2016] has similar complexity as [Schenato & Fiorentin 2011]. However, it uses a flooding principle that extracts a tree topology from the network, which is prone to error propagation.

In this work, we propose an MF message passing scheme called Integrated Cooperative Synchronization (ICS) that integrates the measurement phase into the message passing phase. The main novelty is that we rely on an extended factorization of the a-posteriori distribution of the clock parameters and a specific message passing schedule such that with each broadcast packet a node can collect a new measurement and do message passing computations. This mitigates the two major drawbacks of [Etzlinger *et al.* 2014, Etzlinger *et al.* 2013a] with only a small

loss in achievable accuracy. The proposed method has a similar implementation complexity as the consensus method presented in [Schenato & Fiorentin 2011].

The contributions made in this chapter are:

1. Existing message-passing based TSPs operate in two phases-measurement and message passing phases. These TSPs need an additional mechanism to synchronize these phases. Integrated Cooperative Synchronization (ICS) proposed in this chapter integrates both these phases thus removing the need of such a mechanism to synchronize these phases.

2. In existing message-passing based TSPs, newly joining nodes have to wait till next measurement phase to begin estimating their clock parameters. In ICS, every packet exchanged between two nodes is used both for collecting measurements and estimating clock parameters. Thus ICS reduces the latency for newly joining nodes to estimate their clock parameters.

3. ICS greatly simplifies the computations and memory required for clock parameter estimation as only one packet is used during clock parameter estimation.

4. When only one packet is used for clock-parameter estimation, it seems like a one-way message dissemination which makes synchronization of skew and offset impossible (explained later). ICS therefore provides a link initialization method by which this limitation of one-way dissemination can be overcome.

5. ICS is able to achieve sub-microsecond accuracy without any prior information of the clock parameters of the agent nodes.

The following section introduces the system model and the network considered in this work. Section 4.3 describes the conventional MF message passing method which is the basis of ICS. Different modes of ICS to achieve synchronization are

described in Section 4.4. Section 4.5 presents the results of the simulation performed to evaluate ICS and to compare it with other decentralized time synchronization protocols. Conclusions are presented in Section 4.6. The work presented in this chapter has been published in [Chalapathi *et al.* 2019d].

4.2 System Model and Network Description

4.2.1 Network, Clock and Timestamping Model

We consider a network containing $P = M + A$ stationary nodes, consisting of $M \geq 1$ synchronous master nodes and A asynchronous agent nodes. The node indices are collected in the corresponding disjoint sets $\mathcal{M} \triangleq \{1, 2, ...M\}$ $\mathcal{A} \triangleq \{M + 1, M + 2, ...P\}$ for master and agent nodes, respectively, that form the set of all nodes $\mathcal{P} = \mathcal{M} \cup \mathcal{A}$. The network topology is represented by the set $\mathcal{S}$ that contains all node pairs (p, q) that can directly communicate to each other. Moreover, we require that all links are symmetric, i.e., if $(p, q) \in \mathcal{S} \implies (q, p) \in \mathcal{S}$. Further, we define the neighbor set of node $p \in \mathcal{P}$ as $\mathcal{T}_p = \{q|(p, q) \in \mathcal{S}\}$, i.e., it contains all the nodes which are directly connected to the node p. We assume that there are no direct links between any two master nodes.

A stationary skew-offset clock model is considered, where the local time at a node p, $c_p(t)$, is related to the reference time t with clock skew α_p and offset β_p by

$$c_p(t) = \alpha_p t + \beta_p \,. \tag{4.1}$$

The clock parameters are collected in the vector $\boldsymbol{\vartheta}_p \triangleq \left[1/\alpha_p \;\; \beta_p/\alpha_p\right]^{\mathrm{T}}$. Agent nodes have unknown $\boldsymbol{\vartheta}_p$, whereas master nodes are assumed to have $\boldsymbol{\vartheta}_p = [1 \;\; 0]^{\mathrm{T}}$, i.e., absolute synchronization [Xiong *et al.* 2018] is considered.

Each node $p \in \mathcal{P}$ broadcasts N_p packets, each indexed by $n_p = 1, \ldots, N_p$. All

neighbors $q \in \mathcal{T}_p$ will receive the packets. Per link $(p,q) \in \mathcal{S}$, this yields N_p packets $p \to q$ and N_q packets $q \to p$. The n_pth packet $p \to q$ is transmitted at reference time $t_{p,0}^{(n_p)}$, and received at node q at $t_{pq,1}^{(n_p)} = t_{p,0}^{(n_p)} + \Delta_{pq} + \omega_{pq}^{(n_p)}$, where Δ_{pq} is the unknown communication delay that is assumed to remain constant and $\omega_{pq}^{(n_p)}$ is the measurement noise, modeled as normal Gaussian distribution [Etzlinger *et al.* 2014] with $\omega_{pq}^{(n_p)} \sim \mathcal{N}\big(0, \sigma_\omega^2\big)$. Timestamps are recorded at node p at transmission as $c_p\big(t_{p,0}^{(n_p)}\big)$, and measured at node q at reception as $c_q\big(t_{pq,1}^{(n_p)}\big)$. Using Eq. (4.1), the receive timestamp is

$$c_q\big(t_{pq,1}^{(n_p)}\big) = \alpha_q \, \frac{c_p\big(t_{p,0}^{(n_p)}\big) - \beta_p}{\alpha_p} + \alpha_q \, \omega_{pq}^{(n_p)} + \alpha_q \, \Delta_{pq} + \beta_q \, . \tag{4.2}$$

In the following, we will use a short notation $\tilde{c}_p^{(n_p)} \triangleq c_p\big(t_{p,0}^{(n_p)}\big)$ for the transmit timestamp and $c_{qp}^{(n_p)} \triangleq c_q\big(t_{pq,1}^{(n_p)}\big)$ for the receive timestamp .

4.2.2 Stochastic Model

To each node we assign a Gaussian prior distribution $p\big(\boldsymbol{\vartheta}_p\big)$ on the clock parameters, and on each link a Gaussian prior distribution $p\big(\Delta_{pq}\big)$ on the communication delay.

The approximate local likelihood function [Etzlinger *et al.* 2014] of $c_{qp}^{(n_p)}$ given the clock parameters $\boldsymbol{\vartheta}_p$ and $\boldsymbol{\vartheta}_q$ is

$$\begin{aligned} &p\big(c_{qp}^{(n_p)} | \boldsymbol{\vartheta_q}, \boldsymbol{\vartheta_p}, \Delta_{pq}\big) \\ &\qquad \propto \exp\Big(-\frac{1}{2\sigma_\omega^2} \parallel \mathbf{a}_{pq}^{(n_p)} \boldsymbol{\vartheta}_q + \mathbf{b}_{pq}^{(n_p)} \boldsymbol{\vartheta_p} - \Delta_{pq} \parallel^2 \Big) \, , \end{aligned} \tag{4.3}$$

where $\mathbf{a}_{pq}^{(n_p)} \triangleq \big[c_{qp}^{(n_p)} \;\; -1\big]$ and $\mathbf{b}_{pq}^{(n_p)} = \big[-\tilde{c}_p^{(n_p)} \;\; 1\big]$. Similarly, the likelihood of the n_qth $q \to p$ packet, $n_q \in \{1, \ldots, N_q\}$ can be derived by interchanging p and q in Eq. (4.3). The pairwise likelihood function of all measurements $\mathbf{c}_{pq} \triangleq [\{c_{qp}^{(n_p)}\}_{n_p=1}^{N_p}$,

$\{c_{pq}^{(n_q)}\}_{n_q=1}^{N_q}]^{\mathrm{T}}$ between p and q is

$$p(\mathbf{c}_{pq}|\boldsymbol{\vartheta}_q, \boldsymbol{\vartheta}_p, \Delta_{pq}) \propto \prod_{n_p=1}^{N_p} p(c_{qp}^{(n_p)}|\boldsymbol{\vartheta}_q, \boldsymbol{\vartheta}_p, \Delta_{pq}) \times \prod_{n_q=1}^{N_q} p(c_{pq}^{(n_q)}|\boldsymbol{\vartheta}_q, \boldsymbol{\vartheta}_p, \Delta_{pq})\,. \tag{4.4}$$

Collecting the pairwise likelihood function over all links and joining the prior distributions yields the global posterior function

$$\begin{aligned} p(\boldsymbol{\vartheta}, \boldsymbol{\Delta}|\mathbf{c}) &\propto \prod_{q' \in \mathcal{P}} p(\boldsymbol{\vartheta}_{q'}) \prod_{(p,q)\in\mathcal{S}} p(\Delta_{pq}) \\ &\times \Big(\prod_{n_p=1}^{N_p} p(c_{qp}^{(n_p)}|\boldsymbol{\vartheta}_q, \boldsymbol{\vartheta}_p, \Delta_{pq}) \prod_{n_q=1}^{N_q} p(c_{pq}^{(n_q)}|\boldsymbol{\vartheta}_q, \boldsymbol{\vartheta}_p, \Delta_{pq}) \Big) \qquad (4.5) \\ &= \prod_{q' \in \mathcal{P}} p(\boldsymbol{\vartheta}_{q'}) \prod_{(p,q)\in\mathcal{S}} p(\Delta_{pq})\, p(\mathbf{c}_{pq}|\boldsymbol{\vartheta}_q, \boldsymbol{\vartheta}_p, \Delta_{pq})\,, \qquad (4.6) \end{aligned}$$

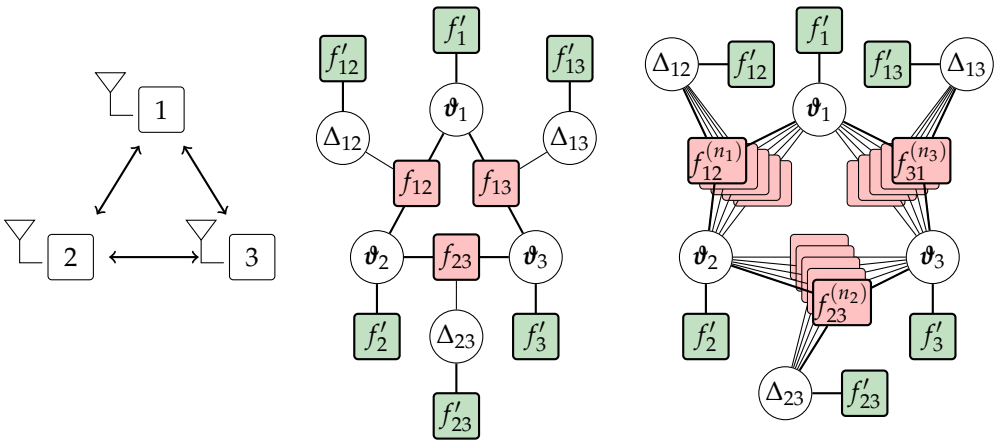
where the vectors $\mathbf{c}$ and $\boldsymbol{\Delta}$ collect all $\mathbf{c}_{pq}$ and Δ_{pq}, respectively, for $(p,q) \in \mathcal{S}$, and $\boldsymbol{\vartheta}$ collects all $\boldsymbol{\vartheta}_p$, $p \in \mathcal{P}$. From Eq. (4.6), an estimate of the clock parameters of node p is obtained by

$$\hat{\boldsymbol{\vartheta}}_p = \arg\max \int p(\boldsymbol{\vartheta}, \boldsymbol{\Delta}|\mathbf{c})\, \mathrm{d}{\sim}\boldsymbol{\vartheta}_p\,, \tag{4.7}$$

where $\sim\boldsymbol{\vartheta}_p$ denotes all entries of $\boldsymbol{\vartheta}$ except $\boldsymbol{\vartheta}_p$.

4.3 Conventional Mean Field Message Passing

Conventional MF message passing finds an approximate solution $b(\boldsymbol{\vartheta}_p) \approx \int p(\boldsymbol{\vartheta}, \boldsymbol{\Delta}|\mathbf{c})\, \mathrm{d}\sim\boldsymbol{\vartheta}_p$ to solve Eq. (4.7) in an iterative way [Etzlinger *et al.* 2014], [Etzlinger *et al.* 2013a]. MF synchronization has high accuracy and moderate computational complexity, but requires a measurement phase and message passing phase in practical im-



(a) (b) (c)

Figure 4.1: (a) Wireless network, (b) corresponding Factor Graph (FG) for conventional Mean-Field (MF) based cooperative synchronization and delay estimation, (c) FG with extended factorization for MF based integrated cooperative synchronization. The following short notation is used: $f'_p = p(\boldsymbol{\vartheta}_p)$, $f'_{pq} = p(\Delta_{pq})$, $f_{pq}^{(n_q)} = p(c_{pq}^{(n_q)})|\boldsymbol{\vartheta}_p, \boldsymbol{\vartheta}_q, \Delta_{pq})$ and $f_{pq} = p(\mathbf{c}_{pq}|\boldsymbol{\vartheta}_p, \boldsymbol{\vartheta}_q, \Delta_{pq})$.

plementation. Coordinating these phases becomes highly challenging in asynchronous networks and thus prevents a broad application.

In the following, we will shortly introduce the basic MF equations to highlight the shortcomings in its practical implementation. Thereafter, modifications needed for ICS are introduced. Motivated from practical observations, we consider the following simplifications: (i) all links have the same measurement covariance σ_ω^2 and (ii) no prior on the distances is available, i.e., $p(\Delta_{pq})$ has infinite covariance.

Conventional message passing uses the factorization shown in Eq. (4.6) to construct a factor graph (FG). For synchronization, this factor graph matches the topology of the network (see Fig. 4.1(a) and (b)). Thus, messages that are passed over the edges of the FG directly translate to packet transmissions between nodes. For example, parameters of messages between variable vertices $\boldsymbol{\vartheta}_1$ and $\boldsymbol{\vartheta}_2$ can be transmitted in packets between nodes 1 and 2.

On such a factor graph, MF operates iteratively after all measurements are obtained. The measurements are collected in the matrices $\mathbf{A}_{pq} \triangleq \left[\mathbf{a}_{qp}^{(1)\,\mathrm{T}}, \ldots, \mathbf{a}_{qp}^{(N_q)\,\mathrm{T}}\right.$,

$\mathbf{b}_{pq}^{(1)\,\mathrm{T}}, \ldots, \mathbf{b}_{pq}^{(N_p)\,\mathrm{T}}\big]^{\mathrm{T}}$ and $\mathbf{B}_{pq} \triangleq \big[\mathbf{b}_{qp}^{(1)\,\mathrm{T}}, \ldots, \mathbf{b}_{qp}^{(N_q)\,\mathrm{T}}, \mathbf{a}_{pq}^{(1)\,\mathrm{T}}, \ldots, \mathbf{a}_{pq}^{(N_p)\,\mathrm{T}}\big]^{\mathrm{T}}$. Starting from an initial guess of the communication delay and the clock parameter vector denoted by $\hat{\Delta}_{pq}^{(0)}$ and $\hat{\boldsymbol{\vartheta}}_p^{(0)}$ respectively, a node computes[1] at iteration i

$$\hat{\Delta}_{pq}^{(i+1)} = \frac{\mathbf{1}^{\mathrm{T}}\big(\mathbf{B}_{pq}\hat{\boldsymbol{\vartheta}}_q^{(i)} + \mathbf{A}_{pq}\hat{\boldsymbol{\vartheta}}_p^{(i)}\big)}{(N_p + N_q)} \tag{4.8}$$

$$\hat{\boldsymbol{\vartheta}}_p^{(i+1)} = \mathbf{P}_p^{-1}\Big(\mathbf{m}_{0,p} + \sum_{q \in \mathcal{T}_p} \mathbf{A}_{pq}^{\mathrm{T}}\big(\hat{\Delta}_{pq}^{(i)} - \mathbf{B}_{pq}\hat{\boldsymbol{\vartheta}}_q^{(i)}\big)\Big)\,, \tag{4.9}$$

where

$$\mathbf{P}_p = \mathbf{P}_{0,p} + \sum_{q \in \mathcal{T}_p} \mathbf{A}_{pq}^{\mathrm{T}}\,\mathbf{A}_{pq}\,, \tag{4.10}$$

and $\mathbf{P}_{0,p}$ and $\mathbf{m}_{0,p}$ are the precision matrix and weighted mean of the prior $p(\boldsymbol{\vartheta}_p)$. In the above equations, $\hat{\boldsymbol{\vartheta}}_p^{(i+1)}$ and $\hat{\boldsymbol{\vartheta}}_p^{(i+1)}$ denote the estimate of the propagation delay and the clock parameter vector respectively. After the computations are completed, the node broadcasts its estimate $\hat{\boldsymbol{\vartheta}}_p^{(i+1)}$ to its neighbors, and vice-versa receives their estimates. The method has two shortcomings. First, the iterations have to be coordinated among all nodes, and second, the measurements have to be collected before starting the message passing, requiring additional coordination of the network and memory for $(2\,N_p + 2\,N_q)$ timestamps that have to be stored at node p per neighbor q.

4.4 Integrated Cooperative Synchronization

In contrast to conventional MF, ICS uses the factorization shown in Eq. (4.5), in which each likelihood function obtained with a single packet exchange is con-

[1] Note that equations (4.8) and (4.9) are a simplified version of the message passing equations in [Etzlinger *et al.* 2013a].

sidered separately, i.e., it performs message passing on an extended FG (see Fig. 4.1(c)). In every transmitted packet, the transmission time stamp is appended, and by measuring the receive time, the receiver can construct a single packet likelihood directly after reception. If the transmitter additionally includes its current message parameters, this likelihood function can be directly used for message passing. Thereby, the message passing does not operate iteratively as a message is passed only once over every single edge.

Although the proposed modification seems trivial, the message scheduling requires careful consideration when no prior delay estimate is available. Therefore, we first introduce the computation rules for standard operation when previous accurate delay estimates are available, and thereafter discuss the special case when a new link is initialized, i.e., when no prior delay estimate is available. The transition between these modes is per node and depends on a simple local decision.

4.4.1 Mode I: Standard Operation

Nodes perform message computation for synchronization in the standard operation mode if an accurate estimate of the delay to the neighbor is available. This mode is motivated by observation that $\mathbf{A}_{pq}^{\mathrm{T}}\,\mathbf{A}_{pq} = \sum \mathbf{a}_{qp}^{(n_q)\,\mathrm{T}}\mathbf{a}_{qp}^{(n_q)} + \sum \mathbf{b}_{pq}^{(n_p)\,\mathrm{T}}\mathbf{b}_{pq}^{(n_p)}$ in Eq. (4.10) and $\mathbf{A}_{pq}^{\mathrm{T}}\,\mathbf{B}_{pq} = \sum \mathbf{a}_{qp}^{(n_q)\,\mathrm{T}}\mathbf{b}_{pq}^{(n_q)} + \sum \mathbf{b}_{qp}^{(n_p)\,\mathrm{T}}\mathbf{a}_{pq}^{(n_p)}$ in Eq. (4.9).

Selecting message passing only in the direction of packet transmission will reproduce the first parts of those sums.

Each node q broadcasts packets to its neighbors, where the n_qth packet includes the transmission timestamp $\tilde{c}_q^{n_q}$, its clock estimate $\hat{\boldsymbol{\vartheta}}_q^{(k_q)}$, and the delay estimates to all its neighbors $\hat{\Delta}_{pq}^{(k_{pq})}$, $p \in \mathcal{T}_q$.. Here, k_q and k_{pq} are the recursion indices for the clock parameter estimate and for the delay estimate, respectively. If a node p receives the n_qth packet from node q, it can directly perform the following

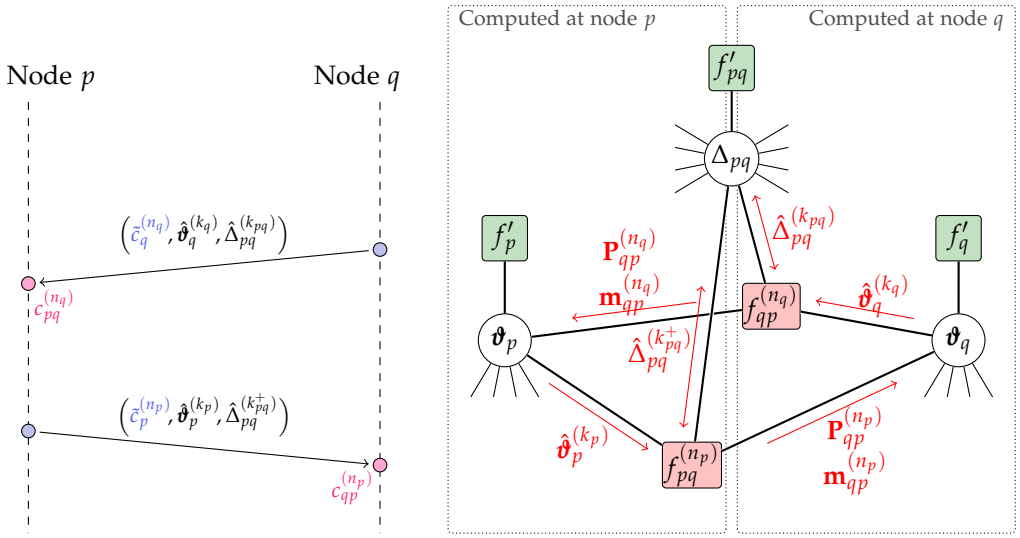


Figure 4.2: ICS standard operation between nodes p and q: (a) Packet exchange where broadcast packets include timestamp and estimates of clock parameters and delays to all neighbors at the time of transmission; (b) The corresponding messages passing on the FG where $k_{pq}^+ = k_{pq}+1$.

message passing computations on the branch of the FG that includes $f_{qp}^{(n_q)}$:

$$\mathbf{P}_{qp}^{(n_q)} = \mathbf{a}_{qp}^{(n_q)\mathrm{T}} \mathbf{a}_{qp}^{(n_q)}, \qquad \mathbf{m}_{qp}^{(n_q)} = \mathbf{a}_{qp}^{(n_q)\mathrm{T}} \left(\hat{\Delta}_{pq}^{(k_{pq})} - \mathbf{b}_{qp}^{(n_q)} \hat{\boldsymbol{\vartheta}}_q^{(k_q)}\right), \tag{4.11}$$

then updates its local variables

$$\mathbf{P}_p^{(k_p+1)} = \mathbf{P}_p^{(k_p)} + \mathbf{P}_{qp}^{(n_q)}, \qquad \mathbf{m}_p^{(k_p+1)} = \mathbf{m}_p^{(k_p)} + \mathbf{m}_{qp}^{(n_q)}, \tag{4.12}$$

and estimates

$$\hat{\Delta}_{pq}^{(k_{pq}+1)} = \frac{k_{pq}\hat{\Delta}_{pq} + \left(\mathbf{a}_{pq}^{(n_q)} \hat{\boldsymbol{\vartheta}}_p^{(k_p)} + \mathbf{b}_{pq}^{(n_q)} \hat{\boldsymbol{\vartheta}}_q^{(k_q)}\right)}{k_{pq}+1}, \tag{4.13}$$

$$\hat{\boldsymbol{\vartheta}}_p^{(k_p+1)} = \left(\mathbf{P}_p^{(k_p+1)}\right)^{-1} \mathbf{m}_p^{(k_p+1)}. \tag{4.14}$$

The updated parameters $\hat{\boldsymbol{\vartheta}}_p^{(k_p+1)}$ and $\hat{\Delta}_p^{(k_p+1)}$ are broadcast with the next packet. If a node is a master node, i.e. $p \in \mathcal{A}$, then the clock parameters are not updated

as in Eq. (4.14), but remain constant. Note that no timestamps from previous packets have to be stored, i.e., the message passing happens with the same packet as the measurement recording. The packet exchange and message passing for this phase of ICS is shown in Fig. 4.2.

4.4.2 Mode II: Link Initialization

We consider the initialization $\mathbf{P}_p^{(0)} = \mathbf{0}$, $\mathbf{m}_p^{(0)} = \mathbf{0}$ for all non-master nodes $p \in \mathcal{A}$ and $\hat{\Delta}_{pq}^{(0)} = 0$ for all links $(p,q) \in \mathcal{S}$. Thus, we assume not having any prior knowledge on the network, except the clock parameters of the master nodes.

When a node $p \in \mathcal{A}$ receives the first packet from a neighbor with accurate clock parameter estimates, i.e. from a master node (or a node q with a sufficiently large clock estimation index k_q, referred to as pseudo-master node), it would not be able to compute the standard mode equations. Firstly, $\mathbf{P}_p^{(1)}$ cannot be inverted in Eq. (4.14) after only one reception, i.e., $\mathbf{P}_p^{(1)} = \mathbf{P}_{qp}^{(1)}$, as it has only rank one (see Eq. (4.11)). This would not be a limitation if node p has already updated $\mathbf{P}_p^{(k_p)}$, $k_p > 1$ times with received packets from other (pseudo-)master nodes. Secondly, as the delay is not known before, $\mathbf{m}_{qp}^{(1)}$ in Eq. (4.11) will be inaccurate and severely perturb the network synchronization process.

Therefore, we propose that every node has one set of link initialization variables $\mathbf{P}_{p,\text{init}}^{(k'_p)}$ and $\mathbf{m}_{p,\text{init}}^{(k'_p)}$, and that the following procedure is executed when a node p receives a packet from a neighboring (pseudo-)master q.

1. Node p initializes $\mathbf{P}_{p,\text{init}}^{(0)} = \mathbf{0}$, $\mathbf{m}_{p,\text{init}}^{(0)} = \mathbf{0}$ and $k'_p = 0$
2. Node p updates the local initialization variables for the first two packets received from a (pseudo-)master q as in Eq. (4.11), with $\hat{\Delta}_{pq}^{(1)} = \hat{\Delta}_{pq}^{(2)} = 0$.
3. Node p then computes $\hat{\boldsymbol{\vartheta}}_{p,\text{init}} = \mathbf{P}_{p,\text{init}}^{(2)\,-1} \mathbf{m}_{p,\text{init}}^{(2)}$ and adds it to its next broadcast, indexed by n'_p. At this point, a deterministic error occurs on the phase

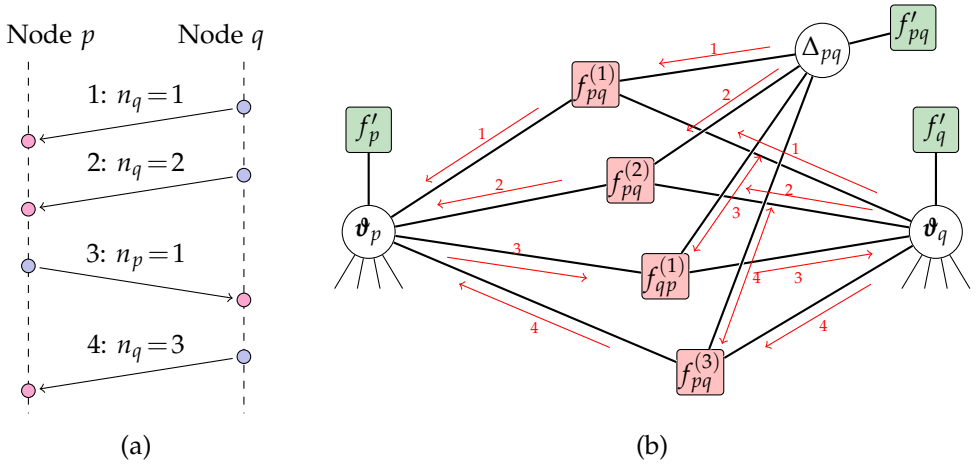


Figure 4.3: Link initialization mode: (a) Message passing between two nodes p and q with four packets exchanged between them; (b) Corresponding messages on the FG.

estimate (second entry of $\hat{\boldsymbol{\vartheta}}_{p,\text{init}}$) of the magnitude of the unknown Δ_{pq}.

4. Node q computes the delay estimate with $\hat{\Delta}^{(3)}_{pq} = \left(\mathbf{a}^{(n'_p)}_{qp}\hat{\boldsymbol{\vartheta}}_{p,\text{init}} + \mathbf{b}^{(n'_p)}_{pq}\hat{\boldsymbol{\vartheta}}^{(k_q)}_{q}\right)/2$, and broadcasts the result with its next packet.

5. Node p corrects its previous estimate to $\hat{\boldsymbol{\vartheta}}'_{p,\text{init}} = \hat{\boldsymbol{\vartheta}}_{p,\text{init}} - [0, \hat{\Delta}^{(3)}_{pq}]^{\text{T}}$ and its variable $\mathbf{m}^{(2)}_{p,\text{init}} = \mathbf{P}^{(2)}_{p,\text{init}}\hat{\boldsymbol{\vartheta}}'_{p,\text{init}}$. It then adds the initialization variables to its standard operation variables with $\mathbf{P}^{(k_p+1)}_{p} = \mathbf{P}^{(k_p)}_{p} + \mathbf{P}^{(2)}_{p,\text{init}}$ and $\mathbf{m}^{(k_p+1)}_{p} = \mathbf{m}^{(k_p)}_{p} + \mathbf{m}^{(2)}_{p,\text{init}}$.

6. Node p joins the standard operation mode, and executes equations (4.11)–(4.14) with the packet received in step 5.

These steps, graphically depicted in Fig. 4.3, require the cooperation of node p and q during the transmission of 4 packets.

4.4.3 Use case for ICS

A suitable application in which ICS can be employed as a time synchronization protocol is WSN-based animal habitat monitoring application [Handcock *et al.* 2009], [Sikka *et al.* 2006]. In ranches which are spread over large geographic areas, it is challenging to observe the grazing patterns of the animals, their so-

cial behaviour and other activities or events. Each of these animals (like cows, pigs, etc.) can carry a WSN node on its collar. These WSN nodes are equipped with relevant sensors like GPS sensors, accelerometers, acoustic sensors, etc. The nodes can also have actuators which generate acoustic and vibration signals [Sikka *et al.* 2006]. These sensors nodes provide the sensor data which is useful in analyzing the social behaviour of these animals, the grazing patterns, etc. The sensor data analysis can be used for increasing productivity in animal farming. Since the WSN nodes attached to the animals are mobile, the network topology keeps changing frequently. Thus, we cannot employ time synchronization protocol s which employ formation of a specific network structure like trees, clusters, etc. Also since such ranches might have large number of animals which implies that the node density is very high, it necessitates the use of decentralized time synchronization protocol. A time synchronization protocol is essential in this application to provide an accurate time stamp accompanying the sensor data which facilitates correct data analysis. Also, we might also employ localization algorithms to locate an animal. Many time-based localization protocols like [Etzlinger *et al.* 2017], [Meyer *et al.* 2018] use time synchronization protocol s as the basis for the localization protocols. Also the low latency involved in ICS for a new node or a displaced node to obtain the estimate of its clock parameters will help the nodes (animals) to get synchronized very quickly even when it moves from one location of the ranch to another. Thus ICS will be a useful protocol for this application. Next section presents the results of the simulations carried out to analyze the performance of ICS.

4.5 Results and Analysis

Simulations were carried out to evaluate the performance of ICS. MATLAB 2017R1 was used to perform these simulations. For evaluating ICS we used a network of

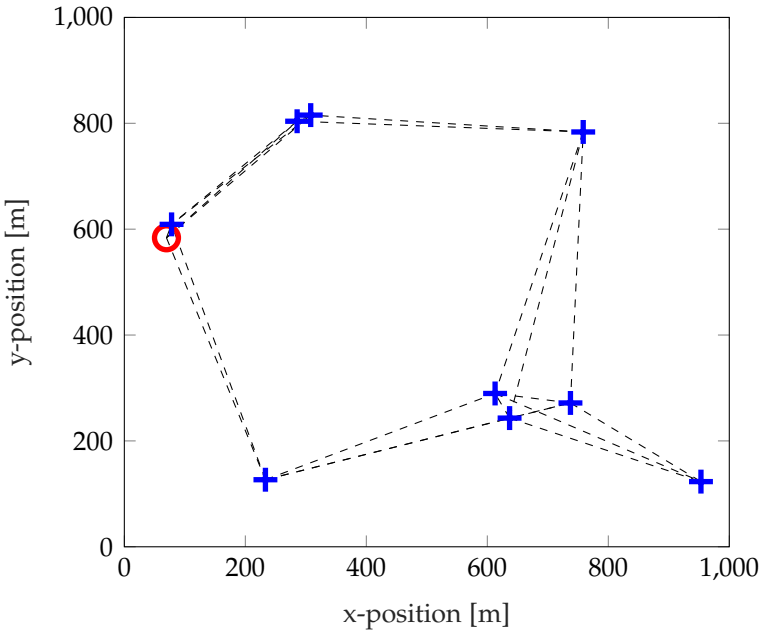


Figure 4.4: Topology of a wireless network with 10 randomly placed nodes. Red circles indicate master nodes, blue crosses agent nodes, and the dashed lines the communication links.

10 nodes randomly placed in a $1000 \times 1000\,\mathrm{m}^2$ field as shown in Fig. 4.4, where the connectivity is defined by the node range of 600 m. The noise standard deviation σ_ω was set to 93 ns as seen from [Etzlinger *et al.* 2014]. The communication delays Δ_{pq} are set to the propagation time, i.e., distance between the nodes divided by the speed of light. The clock skews were drawn by a normal distribution of standard deviation 100 ppm i.e. $\alpha_p \sim \mathcal{N}(1, 10^{-4})$. Similarly the standard deviation for the clock offset was selected to be 5 s, i.e $\beta_p \sim \mathcal{N}(0, 5)$. A node was selected as pseudo-master if it had $k_p = 20$ updates in the standard operation.

We compare the performance of ICS with the consensus method from [Schenato & Fiorentin 2011], referred to as ATS, and the conventional MF method from [Etzlinger *et al.* 2014] referred to as CMF using $K = 100$ measurements and $I = 4$ iterations. For evaluation, we depict 800 time intervals, where each node broadcasts one packet per interval. While ICS and ATS perform synchronization operations after each broadcast, CMF has to collect its measurements first. In Fig. 4.5, the RMSE of the clock phase (solid lines) and clock skew (dashed lines) are depicted.

While all methods achieve similar accuracy for clock skew with the selected parameters, the differences in the clock phase accuracy are significant. ATS, which is limited by the communication delay, achieves an accuracy of 27.5 μs, while ICS and CMF achieve 0.23 μs and 0.05 μs, respectively. Thus, ICS clearly outperforms ATS while having a similar complexity and achieves results close to CMF that has major drawbacks for ad-hoc network implementations.

Finally, we assess the latency and the computational complexity. For ICS we define the latency as the number of intervals until all nodes are pseudo-master nodes in standard operation, i.e., k_p times the network diameter (which is 3 in this network). ATS has no latency, as all nodes update their clock parameters from the first packet on. For CMF, the latency is defined by the duration of the message passing phase, i.e., K plus I. For the considered network we have 60, 1 and 104 intervals of latency for ICS, ATS and CMF, respectively. Thus, ATS has lowest latency compared to ICS and CMF. However, in the applications where synchronization accuracy outweighs latency, ICS is advantageous. Also from practical experience on WSN testbeds, the difference in this latency of ATS and ICS is in the order of few tens of seconds even for a network of about 30-50 nodes.

The computational complexity is assessed in the number of mathematical operations per received packet for ICS and ATS, or per iteration in the message passing phase for CMF. The resulting numbers are shown in Table 4.1. This table shows that while ICS requires only slightly more operations per packet than ATS, both are advantageous to CMF in the message passing phase.

A detailed explanation for these expressions for the number of additions, multiplications and divisions involved in ICS is presented below. We consider only the computational complexity in standard operation mode, as this constitutes the dominant part.

Eq. (4.11)***:*** requires 8 multiplications and 2 additions

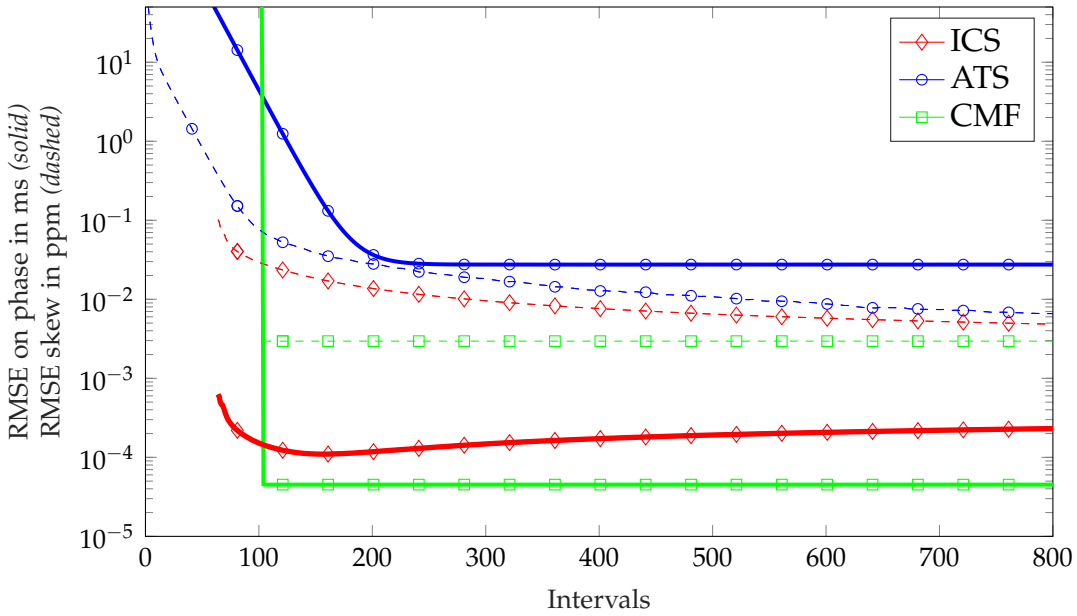


Figure 4.5: Numerical results with RMSE on clock phase (solid lines) and clock skew (dashed lines).

Eq. (4.14)***:*** requires 6 additions.

Eq. (4.12)***:*** requires 5 multiplication, 4 additions and 1 division.

Eq. (4.14)***:*** requires 6 multiplications, 3 additions and 1 divisions.

In total, ICS requires 19 multiplications, 15 additions and 2 divisions per node per received packet. In the similar manner, the number of addition, multiplication and division operations for CMF and ATS have been obtained.

Table 4.1: Per node computational complexity in number of operations: for ICS and ATS per received packet, and for CMF per iteration of the message passing phase.

Protocol	**Additions**	**Multiplications**	**Divisions**
ICS (proposed method)	15	19	2
ATS [Schenato & Fiorentin 2011]	10	7	1
CMF [Etzlinger *et al.* 2014]	$(\lvert\mathcal{T}_p\rvert(8K-6))+4K+8$	$(\lvert\mathcal{T}_p\rvert 8K)+4K+6$	2

4.6 Conclusions

While existing message passing based synchronization schemes yield excellent results, they have major shortcomings that prevent them from broad application. In order to mitigate those drawbacks, ICS proposes to use MF message passing on an extended factorization of the a-posteriori function and a specific message schedule, in order to achieve a simplistic and still highly accurate synchronization method with drastic reduction in computational complexity. Hereby, nodes have to perform only simple computations, and take local decisions for operating in a standard operation or in a link initialization phase. The above-mentioned features make ICS suitable for resource constrained WSN nodes. Despite convergence was observed in numerical evaluations, a theoretical analysis has to be carried out in the future work.

Chapter 5

An Optimal Delay Aware Task-Assignment Scheme for Edge Cloudlet Networks

5.1 Introduction

Over the past decade, there has been an increasing demand for mobile devices to perform computationally intensive tasks. However, the computational capability of these devices is limited due to memory, power and portability constraints. One of the feasible and attractive ways to enhance the performance of the resource-limited mobile devices is to offload their computationally intensive tasks on to the cloud servers when internet connectivity is available. However, when cloud servers are involved in processing, the latency and cost of computation increases. To mitigate these problems, devices with high computational resources, called cloudlets, can be deployed in the locations close to the mobile users/devices. The mobile devices can then offload their computationally intensive tasks on to them. Due to easier access and nearness of the cloudlets, the cost and latency in

processing the tasks decreases.

In this chapter, we focus on task-assignment problem in a multi-cloudlet network connected via a network of wireless Software-defined Network (SDN) switches, which services the task offload requests from mobile devices in a given locality. The computational resources of the cloudlets are used to serve the tasks offloaded by mobile devices in their vicinity [Kosta *et al.* 2012]. In edge computing network, the tasks from the mobile devices or other end devices likes smart objects are offloaded on to the cloudlets. In an edge computing, the tasks are handled by the cloudlets and the cloud servers are absent in edge computing network [Mahmud *et al.* 2018]. Edge computing network does not provide Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and other cloud services spontaneously [Mahmud *et al.* 2018], [Shi *et al.* 2016]. Traditionally, the nearest cloudlet serves the request of a mobile device. However, this is an inefficient approach because the nearest cloudlet may be heavily loaded at a given time while another cloudlet in the network may be lightly loaded as illustrated in Fig. 5.1. In this figure, we see that there are more mobile devices in the vicinity of *Cloudlet1* as compared to *Cloudlet2* or *Cloudlet3*. Thus, as per the traditional task-assignment scheme, Cloudlet1 will handle all the tasks offloaded by the mobile devices in its vicinity. Therefore, Cloudlet1 becomes heavily loaded (hence marked as red). The mobile devices served by such a heavily loaded cloudlet will thus experience high latency for completion of their computation tasks [Jia *et al.* 2016]. At the same time, since there are lesser mobile devices near Cloudlet3, it is lightly loaded (hence marked as green) and Cloudlet2 has a medium load (hence marked as yellow). We, therefore, observe that there is an imbalance in the load allocated to these cloudlets.

We can solve this load imbalance among the cloudlets by networking these cloudlets through Software Defined Network (SDN) switches to enable the network load to

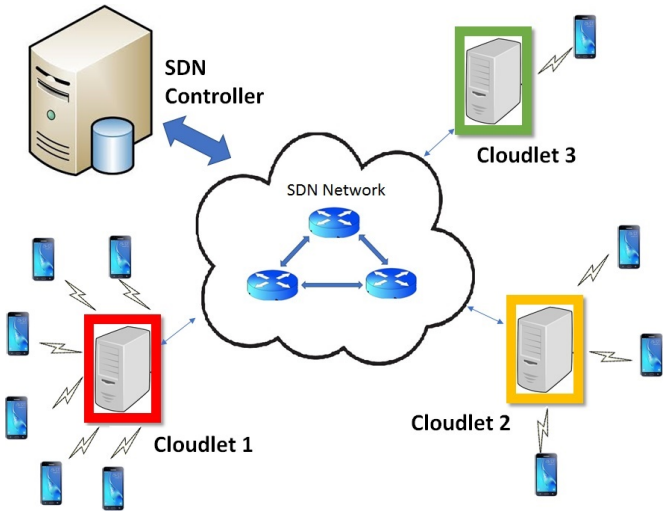


Figure 5.1: A cloudlet network with the different participating entities. Cloudlet1 is heavily loaded, Cloudlet2 is having medium-load and Cloudlet3 is lightly loaded.

be served in a cooperative manner. This is done to ensure a better quality of service and better utilization of resources by load balancing. The task-assignment scheme for edge cloudlets presented in this chapter called *Latency Aware task-assignment* (LATA), aims to reduce the latency in processing the tasks offloaded by the mobile devices. An optimal scheme is presented here and its performance is compared with other state-of-the-art offloaded task-assignment schemes presented in [Sun & Ansari 2017] and [Mukherjee *et al.* 2019]. A preliminary version of the current work was presented in [Chamola *et al.* 2017]. The work described in this chapter has been presented in [Chalapathi *et al.* 2019c]. The main contributions made in this chapter can be summarized as follows:

1. An optimal task-assignment scheme for a multi-cloudlet environment in which a wireless SDN network is used to connect the cloudlets is proposed in this chapter. An elaborate description of the mechanics of this scheme, mathematical proofs of its optimality and convergence are also presented.

2. An admission control for this task-assignment scheme to manage the admission of task-assignment requests when the number of requests is greater than what the network can handle is also proposed.

3. A performance analysis for different sized cloudlet networks- a small network consisting of three cloudlets and a large one consisting of ten cloudlets is also presented.

4. A thorough performance analysis of latency for different requests rates for these different sized networks with varying network environment parameters is also included.

5. The simulations show that this task-assignment scheme reduces the latency experienced by offloaded tasks in a multi-cloudlet scenario as compared to the existing state-of-the-art methods.

Rest of the chapter is organized as follows. The system model considered in this work is discussed in the next section. The problem formulation is discussed in Section 5.3 and the solution methodology in Section 5.4. The simulation results are discussed in Section 5.5 and the conclusions are presented in Section 5.6.

5.2 System Description

In this work, we consider a cloudlet network as depicted in Fig.5.1. In our model, the tasks offloaded by a mobile device on an overloaded cloudlet can be served on some other lightly loaded cloudlet in the network. The wireless SDN controller present in the network takes care of the offloaded task-assignment as per the scheme discussed in Section 5.4. Let us consider our cloudlet network as represented by the set of cloudlets $\mathcal{W}$ with $\mathcal{W} = \{G_1, G_2, \cdots, G_k, \cdots, G_{|\mathcal{W}|}\}$, where we refer to the k-th cloudlet as G_k. Let us denote the maximum rate of service that the k-th cloudlet offers as S_k^{max}. It is assumed that the task offloading requests from the mobile devices located at a location $y \in \mathcal{R}$ ($\mathcal{R}$ being the geographical region) follows a Poisson process that has an arrival rate of $\lambda(y)$. Poisson arrival process has been taken into account based on its wide acceptance for generating

traffic for mobile networks [Lee *et al.* 2014]. The analysis done later is not limited by the traffic's being Poisson in nature but also holds valid for any other type of distribution. We denote the average file size offloaded at location y as $\tau(y)$. Thus, we get the mobile task offload density denoted by $\gamma(y)$ as $\gamma(y) = \lambda(y)\tau(y)$, where the spatial task offload variability is captured by $\gamma(y)$. A number of factors determine the latency experienced by the tasks that the mobile devices offload on to the cloudlets for the execution. These factors are enumerated as: (i) the current load of the cloudlet in executing the offloaded task request, (ii) the maximum rate of service of the cloudlet in processing its request offers, and (iii) the distance between the serving cloudlet and the corresponding mobile device. Without loss of generality, let us consider the service rate that the cloudlet k offers to the mobile device located at y as

$$s_k(y) = \frac{S_k^{max}}{1 + (dis(G_k, y)/d_0)^{\alpha}} \tag{5.1}$$

where $dis(G_k, y)$ is the Euclidean distance of the k-th cloudlet from the mobile device that is at location y. In the above equation, we introduce the parameter α that helps us adjust the service rate according to different network scenarios. d_0 is the scaling factor for the distance. We arrive at the above formula in Eq. (5.1) based on the intuition that the service rate that the cloudlet offers is directly proportional to the maximum rate of service which that cloudlet can offer, and inversely proportional to the distance between the mobile device and that cloudlet. As compared to the computational delay at the cloudlets, the transmission delay and the propagation delay for our network are negligible (especially due to the fact that the cloudlets are in close proximity of the mobile devices). To illustrate this point, let us consider a wireless SDN network having 10 Gbps links. Note that such wireless SDN solutions are commercially available [Cis]. Further let us consider a scenario where the mobile user is at a distance of 100 m from the cloudlet which is processing the tasks offloaded on to it by the mobile user and let

the maximum computational service rate (S_{max}) of this cloudlet be 3000 KBps. Assuming that the size of a packet offloaded to be processed by the user is 40 KB, the the transmission delay using this SDN solution will be $T_D = \frac{40\times8\times10^3}{10\times10^9} = 32\mu s$.The effective service rate observed by the mobile user at 100m from this cloudlet as per Eq. (5.1) is 272.72KBps (with α taken to be unity). The computational delay for a 40KB packet offloaded on to this cloudlet by this mobile user will be $T_C = \frac{40}{272.72} = 0.1466705s = 146670.5\mu s$. Likewise, the propagation delay for a round trip of 200 m (i.e., twice of 100m) will be $T_P = \frac{200}{3\times10^8} = 0.66\mu s$. Thus the transmission and propagation delays are much lower than the computational delay. Therefore, in this work, the primary consideration for latency evaluation is the computational delay .

To specify task-assignment relationship between the mobile devices and the cloudlets, let us introduce a function $u_k(y)$ called task-assignment indicator function. This function takes the value 1 if the mobile user located at y is served by the cloudlet k and 0 otherwise. Also, let us define the cloudlet load by ρ_k,

$$\rho_k = \int_{\mathfrak{R}} \frac{\gamma(y)}{s_k(y)} u_k(y) dy. \tag{5.2}$$

As given by [Liu *et al.* 2014] the cloudlet load represents the time fraction for which the cloudlet k is engaged in serving its traffic requests.

Definition 1: $\mathfrak{Z}$ denotes the feasible set of the cloudlet loads $\rho = (\rho_1, \cdots, \rho_{|\mathcal{W}|})$. We define $\mathfrak{Z}$ as

$$\mathfrak{Z} = \Big\{\rho \,|\, \rho_k = \int_{\mathfrak{R}} \frac{\gamma(y)}{s_k(y)} u_k(y) dy,\ 0 \le \rho_k \le 1-\epsilon,\ \forall k \in \mathcal{W},$$
$$u_k(y) \in \{0,1\}, \sum_{k=1}^{|\mathcal{W}|} u_k(y) = 1,\ \ \forall k \in \mathcal{W},\ \ \forall y \in \mathfrak{R}\Big\},$$

with ϵ being an arbitrarily small positive constant. The assumption here is that an offloaded task of a particular mobile device is served in its entirety by only

one cloudlet in the network. Although a mobile device would be connected to the closest cloudlet in the network, the cloudlet that serves its offloaded task request is decided based on the task-assignment algorithm discussed in Section 5.4. The arrivals of offloaded tasks follow Poisson process, and thus the sum of arrivals of offloaded tasks is also a Poisson process. There is only one server at a cloudlet and the service process follows a general distribution. Thus, the traffic arrivals at the cloudlets are modeled as an M/G/1 queue. At the cloudlet k, we can give the average traffic flow by the fraction $\frac{\rho_k}{1-\rho_k}$ [Liu *et al.* 2014]. From the Little's Law, the latency that a traffic flow experiences is proportional to the average number of flows in the system [Kleinrock 1976]. Hence, at a given cloudlet, the total number of flows is considered as the k-th cloudlet's latency indicator, i.e., $\mathcal{L}_k(\rho_k)$, which is given by [Liu *et al.* 2014]

$$\mathcal{L}_k(\rho_k) = \frac{\rho_k}{1-\rho_k}. \tag{5.3}$$

From the above equation, we can see that as the value of ρ_k tends to 1, the latency indicator, $\mathcal{L}_k(\rho_k)$ increases exponentially approaching ∞. It is to be noted that the latency indicator being a relative indicator of the overall latency of the system, is unit-less. Several contemporary studies [Liu *et al.* 2014], [Han & Ansari 2014],[Liu *et al.* 2015] have used the above indicator to quantitatively analyze the performance of a system's latency, due to its ability to give a comprehensive view of the latency performance of the network by jointly capturing computational as well as queuing delay. It can be seen here that both these delays have been well captured in the expression for the latency given by Eq. (5.3), where ρ_k (given in Eq. (5.2)) is a function of the data traffic size and computational rate offered, and $\frac{\rho_k}{1-\rho_k}$ is a well known expression for queuing delay.

5.3 Problem Formulation

The problem [Pr1] denotes the problem to minimize the total network latency and is formulated as follows

$$\textbf{[Pr1]} \ \underset{\rho}{\text{minimize}} \quad \Phi(\rho) = \sum_{k=1}^{|\mathcal{W}|} \mathcal{L}_k(\rho_k)$$
$$\text{subject to:} \quad \rho \in \mathfrak{Z}$$

We present a task-assignment scheme called Latency Aware task-assignment (LATA) in this chapter. Using this scheme, the SDN controller aims to improve the overall QoS experience of the mobile devices by solving the above-mentioned optimization problem. It does the task-assignment for the service requests, i.e., it maps offloaded task requests to the serving cloudlet, for ensuring best latency performance in the network. As described in Section 5.4, we arrive at the optimal solution to [Pr1] by balancing the current load of the cloudlets (ρ) in the network. A description of LATA is presented in the next section.

5.4 LATA Optimal Task-Assignment Scheme

We present the task-assignment scheme called LATA in this section. This scheme aims to find the optimal solution of the problem [Pr1] that was formulated in the previous section. Let us now briefly describe how the task-assignment algorithm works. The detailed description will follow in sequel.

Each cloudlet estimates a variable called "resistance index" (described later) and advertises it to the SDN controller. The resistance index is calculated for each individual cloudlet periodically by evaluating their respective traffic loads. The wireless SDN controller then decides which cloudlet should be assigned the offloaded task with the aim to minimize the value of the objective function formulated in

Table 5.1: Notation Summary

Notation	Meaning
y	location of the mobile user
$\mathcal{W}$	set of Cloudlets
k	index of the cloudlet in $\mathcal{W}$
$\lambda(y)$	task offloading arrival rate per unit area
$\tau(y)$	average file size
$\gamma(y)$	$:=\lambda(y)\tau(y)$, mobile task offload density
S_k^{max}	maximum service rate offered by the k-th cloudlet
$s_k(y)$	the service rate offered by k-th cloudlet at y
α	parameter capturing distance dependency of service rate
$u_k(y)$	task-assignment indicator of k-th cloudlet at y
ρ_k	k-th cloudlet's load
$\mathcal{L}_k(\rho_k)$	latency indicator of k-th cloudlet
$\mathfrak{Z}$	Feasible set of cloudlet loads
$\tilde{\mathfrak{Z}}$	relaxed feasible set of cloudlet loads
$\Phi(\boldsymbol{\rho})$	objective function which is sum of latency indicators of all cloudlets
i	index of the time slot
$u_k^i(y)$	task-assignment indicator of k-th cloudlet at y in i-th time slot
ψ_k^i	resistance index of k-th cloudlet in time slot i
ρ_k^i	load sent by the k-th cloudlet to the SDN controller in i-th time slot
$T_k(\rho_k^i)$	load of k-th cloudlet in the i-th time slot
θ	admission control parameter

[Pr1].

We make the following assumptions in this task-assignment algorithm:

1. The time scale of the traffic arrival and departure processes is faster than the scale at which the resistance indices of the cloudlets are unicasted (to the SDN controller), so that our scheme converges. This assumption ensures that for the current set of resistance indices of the cloudlets, the SDN controller

makes the task offloading decisions before the cloudlets send the next set of indices into the network.

2. The resistance indices are sent to the SDN controller by all the cloudlets at the same time, i.e., the cloudlets in our network are synchronized.

LATA consists of two algorithms, one running at the SDN controller and another at the cloudlet. We will now describe these two algorithms.

5.4.1 Algorithm at the SDN Controller

As mentioned earlier, LATA aims at arriving at the optimal solution to the problem [Pr1] that has been formulated previously. The feasible set $\mathfrak{Z}$ defined in Section 5.2 is not a convex set as $u_k(y) \in \{0,1\}$. This is because $\mathfrak{Z}$ is not continuous and thus it is not differentiable. Thus, we will not be able to minimize the objective function $\Phi(\rho)$ subject to $\rho \in \mathfrak{Z}$. Thus, to make the set $\mathfrak{Z}$ continuous and differentiable, we modify the above constraint on $u_k(y)$ to $0 \leq u_k(y) \leq 1$, thus introducing convexity to the optimization problem [Pr1].

With such relaxation in place, $u_k(y)$ can be seen as the probability that task offloaded by a mobile device at location y is serviced by the cloudlet k. We can now define $\tilde{\mathfrak{Z}}$, the set of relaxed cloudlet loads given as

$$\begin{aligned}\tilde{\mathfrak{Z}} = \Big\{\rho \,|\, \rho_k = \int_{\mathfrak{R}} \frac{\gamma(y)}{s_k(y)} u_k(y) dy, 0 \leq \rho_k \leq 1-\epsilon, \forall k \in \mathcal{W}, \\ 0 \leq u_k(y) \leq 1, \sum_{k=1}^{|\mathcal{W}|} u_k(y) = 1, \forall k \in \mathcal{W}, \forall y \in \mathcal{R}\Big\}.\end{aligned} \tag{5.4}$$

The set $\tilde{\mathfrak{Z}}$ mentioned above is convex and its convexity has been proved in [Kim *et al.* 2012]. By applying the above-obtained relaxation in problem [Pr1], we arrive at a modified problem [Pr2] which can be formulated as

$$\textbf{[Pr2]} \ \underset{\rho \in \tilde{\mathfrak{Z}}}{\text{minimize}} \quad \Phi(\rho) = \sum_{k=1}^{|\mathcal{W}|} \mathcal{L}_k(\rho_k).$$

The new optimization problem [Pr2] has been formulated using $\tilde{\mathfrak{Z}}$. However, it can be seen from Theorems 1 and 2 presented further in this section that we get a deterministic task-assignment (which belongs to $\mathfrak{Z}$) based on the task-assignment algorithm presented later in this section.

We define a time slot as the time taken between any two consecutive updates of resistance index. In the beginning of the i-th time slot, the respective resistance indices are sent by the cloudlets to the SDN controller. The cloudlets are assigned the tasks offloaded by the mobile device based on the service rate they offer and the resistance indices that they have sent to the SDN controller. It is assumed that the resistance indices are sent by the cloudlets at the start of the time slot i. The term ψ_k^i denotes the resistance index sent by the cloudlet k at the beginning of time slot i which is defined as

$$\psi_k^i = \frac{\partial \mathcal{L}_k^i(\rho)}{\partial \rho_k^i} = \frac{1}{(1-\rho_k^i)^2}. \tag{5.5}$$

We use the function given below to assign the offloaded tasks of the mobile devices (for any device at location y) to the cloudlets.

$$u_k^i(y) = \begin{cases} 1 & \text{if } k = \underset{k \in \mathcal{W}}{\arg\max} \frac{s_k(y)}{\psi_k^i} \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

$u_k(y)$ is the task-assignment indicator function (defined in Section 5.2), which indicates whether or not the k-th cloudlet services the task requests offloaded by the device at y. The term $s_k(y)$ captures the service rate that the k-th cloudlet offers at location y. Thus, the computation complexity of each task-assignment process is of the order of $O(|\mathcal{W}|)$, i.e., the number of cloudlets in the network

Algorithm 5.1 Algorithm at the SDN Controller

Input: At the beginning of time slot i, the resistance index ψ_k^i, $\forall k \in \mathcal{W}$ and the service rate $s_k(y)$ at location $y, \forall y \in \mathcal{R}$, $\forall k \in \mathcal{W}$

Output: task-assignment indicator $u_k^i(y)$

1: Receive the resistance indices (ψ_k^i) for all the cloudlets at the beginning of the time slot.
2: Upon receiving a task offloading request at location y evaluate $s_k(y)$, $\forall k \in \mathcal{W}$ using Eq. (5.1).
3: Assign a cloudlet for the request at location y, i.e., calculate $u_k^i(y)$ using Eq. (5.6).

because the number of computations involved in each task assignment process increases linearly with increase in the number of cloudlets in the network.

5.4.2 Algorithm at the Cloudlet

At the end of the i-th time slot, load at the k-th cloudlet is represented as $T_k(\rho_k^i)$. Each cloudlet evaluates its load defined as per the following equation

$$T_k(\rho_k^i) = \min\left(\int_{\mathcal{R}} \frac{\gamma(y)}{s_k(y)} u_k(y)dy, 1-\epsilon\right), \tag{5.7}$$

where ϵ is an arbitrarily small positive constant. The cloudlets update their cloudlet load, $T_k(\rho_k^i)$, and use it to calculate the resistance index to be broadcasted to the SDN controller, using the load as per the below equation

$$\rho_k^{i+1} = \xi\rho_k^i + (1-\xi)T_k(\rho_k^i) \tag{5.8}$$

Here ξ is taken as the averaging factor with $0 < \xi < 1$.

The working of the algorithms on the SDN controller and at the cloudlet is depicted in Fig. 5.2.

5.4.3 Convergence of LATA Scheme

This subsection and the next prove the convergence and optimality of the algorithm for task-assignment that has been discussed above. We first show that the objective function Φ is convex in $\rho \in \tilde{\mathfrak{Z}}$. This will ensure that we can have an optimal task-assignment by minimizing the objective function.

Lemma 1: *When the cloudlet load ρ is defined in $\tilde{\mathfrak{Z}}$, the objective function $\Phi(\rho)$ is observed to be convex in ρ.*

Proof. This can be proven by showing $\nabla^2\Phi(\rho) > 0$.
We can write the objective function as

$$\Phi(\rho) = \sum_{k=1}^{|\mathcal{W}|} \mathcal{L}_k(\rho) = \sum_{k=1}^{|\mathcal{W}|} \frac{\rho_k}{1-\rho_k} \tag{5.9}$$

The 1^{st} and 2^{nd} order derivatives of the objective function evaluated with respect to ρ are as follows

$$\nabla\Phi(\rho) = \sum_{k=1}^{|\mathcal{W}|} \frac{1}{(1-\rho_k)^2} \tag{5.10}$$

$$\nabla^2\Phi(\rho) = \sum_{k=1}^{|\mathcal{W}|} \frac{2}{(1-\rho_k)^3} \tag{5.11}$$

Since the value of $\frac{2}{(1-\rho_k)^3} > 0$, the above evaluated 2^{nd} order derivative is also non-negative for all cloudlets. Thus, we have shown that our objective function is convex. □

As proved in Lemma 1, the objective function is convex. Thus, we can find an optimal load $\rho^* \in \tilde{\mathfrak{Z}}$ which corresponds to a unique optimal task-assignment, such that it minimizes the objective function, $\Phi(\rho) = \sum_{k=1}^{|\mathcal{W}|} \mathcal{L}_k(\rho_k)$. Next step to prove is that LATA is convergent. We will make use of Lemmas 2 and 3 (discussed further in the chapter) for proving the same. We begin with proving that $T_k(\rho^i)$,

Algorithm 5.2 Algorithm at the Cloudlet

Input: task-assignment $u_k^i(y), \forall y \in \mathcal{R}$ at the beginning of time slot i
Output: The resistance index ψ_k^{i+1}

1: Calculate the current load $T_k(\rho_k^i)$ using Eq. (5.7)
2: Evaluate ρ_k^{i+1} using Equation (8)
3: Calculate the resistance index ψ_k^{i+1} using the Eq. (5.5)
4: Advertise this ψ_k^{i+1} to the SDN Controller at the beginning of the next time slot.

and in turn $(\boldsymbol{\rho}^{i+1} - \rho^i)$, yields the direction of descent for $\Phi(\rho^i)$ at ρ^i (as it will be proved in Lemma 2 and 3). Further in Theorem 1, we will show that the cloudlet load will converge after a few iterations. It will be proved in Theorem 2 that the objective function Φ is minimized with the obtained cloudlet load.

Lemma 2: *Given $\rho^i \neq \rho^*$, $T_k(\rho^i)$ provides the direction of descent for $\Phi(\rho^i)$ at ρ^i.*

Proof. From Lemma 1, we know that when ρ is defined in $\tilde{\mathfrak{Z}}$, $\Phi(\rho)$ is a convex function of ρ. From this, we can easily prove Lemma 2 by showing that $< \nabla\Phi(\rho^i), T(\rho^i) - \rho^i > \leq 0$ (where $< m, l >$ denotes the inner product of the vectors m and l) [Boyd & Vandenberghe 2004]. For the cloudlet loads ρ_k^i and $T(\rho_k^i)$, let the task-assignment indicators be $u_k(y)$ and $u_k^T(y)$. The inner product can be then given as

$$
\begin{aligned}
&< \nabla\Phi(\rho^i), T(\rho^i) - \rho^i > \\
&= \sum_{k=1}^{|\mathcal{W}|} \frac{1}{(1-\rho_k^i)^2} \left(T_k(\rho_k^i) - \rho_k^i \right) \\
&= \sum_{k=1}^{|\mathcal{W}|} \frac{1}{(1-\rho_k^i)^2} \left(\int_{\mathcal{R}} \frac{\gamma(y)(u_k^T(y) - u_k(y))}{s_k(y)} dy \right) \\
&= \int_{\mathcal{R}} \gamma(y) \sum_{k=1}^{|\mathcal{W}|} \left(\frac{\frac{1}{(1-\rho_k^i)^2}}{s_k(y)} (u_k^T(y) - u_k(y)) \right) dy.
\end{aligned}
$$

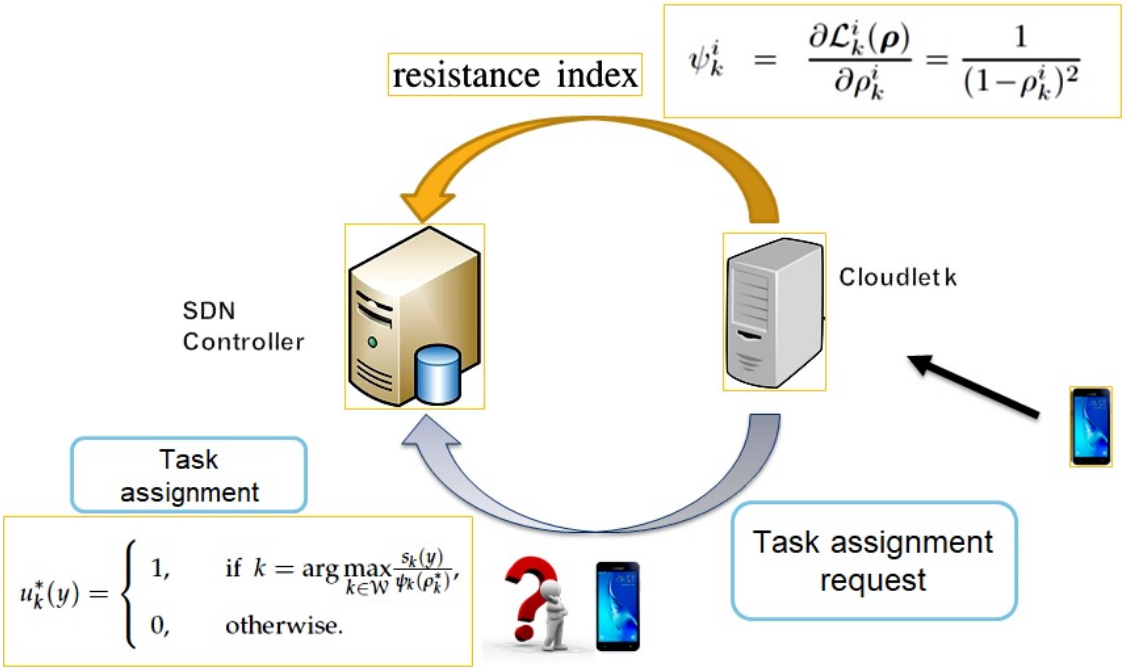


Figure 5.2: Illustrative representation of the process of task offloading from the mobile user on to the cloudlet with intervention of the SDN Controller.

We can see that

$$\sum_{k=1}^{|\mathcal{W}|} \frac{\frac{1}{(1-\rho_k^i)^2}(u_k^T(y) - u_k(y))}{s_k(y)} \leq 0$$

holds because $u_k^T(y)$ from (Eq. 5.6) will maximize the value of $\frac{s_k(y)}{\frac{1}{(1-\rho_k^i)^2}}$. Therefore, $< \nabla\Phi(\rho^i), T(\rho^i) - \rho^i > \leq 0$, thus proving the lemma. □

Lemma 3: $(\boldsymbol{\rho}^{i+1} - \rho^i)$ provides the descent direction to $\Phi(\rho^i)$.

Proof. To prove this, we consider the expression given below.

$$\begin{aligned} \rho_k^{i+1} - \rho_k^i &= \xi\rho_k^i + (1-\xi)T_k(\rho_k^i) - \rho_k^i \\ &= (1-\xi)(T(\rho_k^i) - \rho_k^i). \end{aligned} \tag{5.12}$$

We have earlier seen in Lemma 2 that $(T(\rho^i) - \rho^i)$ is a descent direction of $\Phi(\rho^i)$. Since $0 < \xi < 1$, we can say $(1-\xi) > 0$. Thus, $\rho^{i+1} - \rho^i$ also gives the descent direction of $\Phi(\rho^i)$. □

Next, we prove that our task-assignment scheme is optimal and convergent in Theorems 1 and 2.

Theorem 1: *The cloudlet load vector ρ will converge to $\rho^* \in \mathfrak{Z}$ (ρ^* is the optimal cloudlet load vector).*

Proof. Earlier in Lemma 1, it has been shown that $\Phi(\rho^i)$ is convex. Further, Lemma 3 shows that $(\boldsymbol{\rho}^{i+1} - \rho^i)$ gives a descent direction for $\Phi(\rho^i)$. Thus, the convergence of $\Phi(\rho^i)$ to ρ^* can be guaranteed. Let us prove this by contradiction. Suppose that $\Phi(\rho^i)$ does not converge to $\Phi(\rho^*)$, but rather to a different point; then ρ^{i+1} will again give a direction of descent which decreases $\Phi(\rho^i)$(as proven in Lemma 3), which contradicts the assumption of convergence that we began with. Additionally, as ρ^i is derived based on (Eq. 5.6) where $u_k(y) \in \{0,1\}$, ρ^* belongs to set $\mathfrak{Z}$. □

5.4.4 Optimality of the LATA Scheme

We establish the optimality of the LATA scheme through the following theorem. Note that this theorem uses results of the lemmas proved in the previous subsection.

Theorem 2: *Given a non-empty set $\mathfrak{Z}$ and given that the cloudlet load ρ has a convergence in ρ^*, the task-assignment corresponding to ρ^* minimizes the objective function $\Phi(\rho)$.*

Proof. Suppose that the task-assignment corresponding to ρ^* is $u^* = \{u_k^*(y) | u_k^*(y) \in \{0,1\}, \forall k \in \mathcal{W}, \forall y \in \mathcal{R}\}$ and the task-assignment corresponding to ρ is $u = \{u_k(y) | u_k(y) \in \{0,1\}, \forall k \in \mathcal{W}, \forall y \in \mathcal{R}\}$ with $\rho \in \mathfrak{Z}$ being the load vector of some cloudlet. We have already seen that $\Phi(\rho)$ is convex over ρ, and now to prove this theorem, we show that $< \nabla\Phi(\rho^*), \rho - \rho^* > \geq 0$. Please note that for

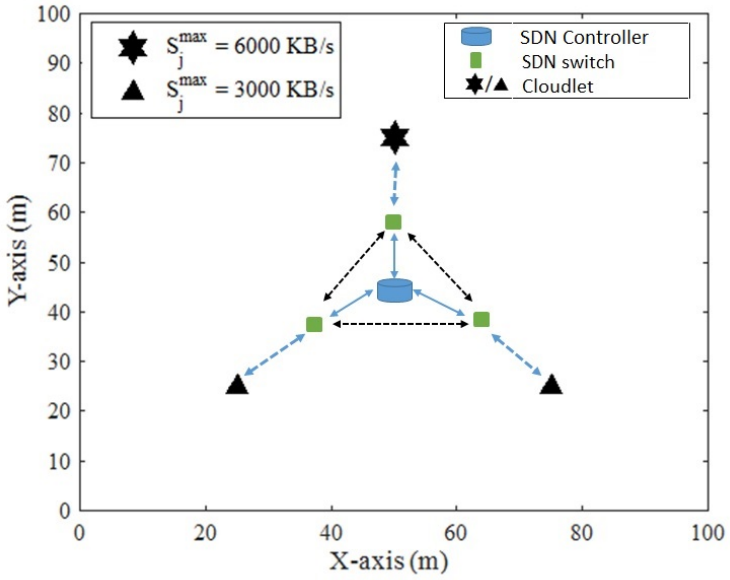


Figure 5.3: The topology of the network with three cloudlets used in our simulations.

the purpose of clarity, we substitute $\frac{\partial \Phi(\rho^*)}{\partial \rho_k^*}$ as $\psi_k(\rho_k^*)$ in the following proof.

$$
\begin{aligned}
< \nabla\Phi(\rho^*), \rho - \rho^* > &= \sum_{k=1}^{|\mathcal{W}|} \psi_k(\rho_k^*)\,(\rho - \rho^*) \\
&= \sum_{k=1}^{|\mathcal{W}|} \left(\int_{\mathcal{R}} \frac{\gamma(y)(u_k(y) - u_k^*(y))}{s_k(y)\psi_k^{-1}(\rho_k^*)} dy \right) \\
&= \int_{\mathcal{R}} \gamma(y) \sum_{k=1}^{|\mathcal{W}|} \frac{(u_k(y) - u_k^*(y))}{s_k(y)\psi_k^{-1}(\rho_k^*)} dy.
\end{aligned}
$$

However, we already know that the criterion to choose the optimal offloaded task-assignment is as under

$$
u_k^*(y) = \begin{cases} 1, & \text{if } k = \arg\max_{k \in \mathcal{W}} \frac{s_k(y)}{\psi_k(\rho_k^*)}, \\ 0, & \text{otherwise.} \end{cases},
$$

and thus we can deduce the following equation using the optimal task-assignment criterion,

$$
\sum_{k=1}^{|\mathcal{W}|} \frac{u_k^*(y)}{s_k(y)\psi_k^{-1}(\rho_k^*)} \leq \sum_{k=1}^{|\mathcal{W}|} \frac{u_k(y)}{s_k(y)\psi_k^{-1}(\rho_k^*)}. \tag{5.13}
$$

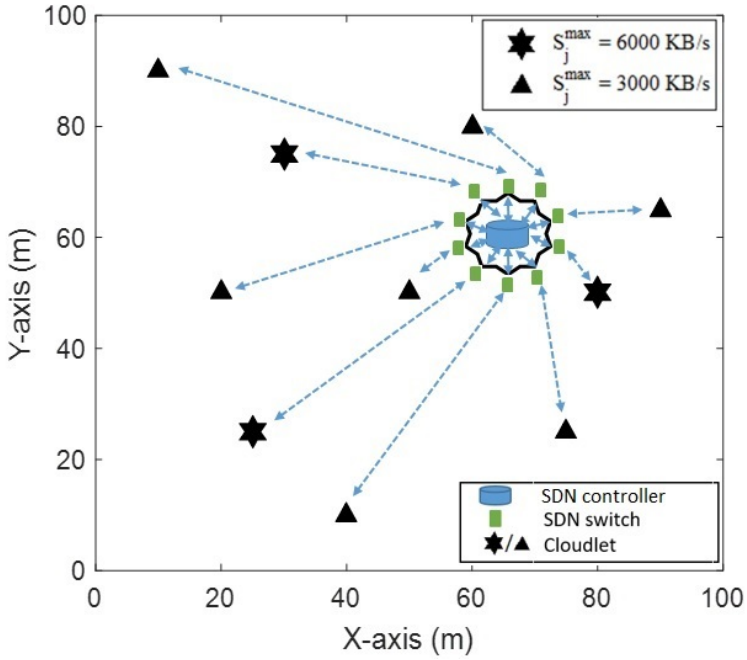


Figure 5.4: The topology of the network with ten cloudlets used in our simulations.

Hence, $< \nabla\Phi(\rho^*), \rho - \rho^* > \geq 0$ proving this Theorem. □

5.4.5 Admission Control

To ensure optimality and convergence of the LATA scheme, the cloudlet assignment problem is required to be feasible. This means that the traffic loads of the cloudlets should lie in the feasible set that has been defined in Definition 1 in Section 5.2. Note that when the traffic in the network is beyond its capacity to serve, the cloudlet assignment problem ceases to be feasible. Thus, the optimality and convergence property of the LATA scheme does not hold. This necessitates an admission control policy to ensure that the above-mentioned properties still hold in presence of exceedingly high task offload requests in the network. For admission control, let $\theta(y)$ be the coefficient for admission control of user at location y, such that $0 \leq \theta(y) \leq 1$ is the probability of a mobile device at location y getting admittance into the network. The SDN controller assigns $\theta(y)$ for a location y. Note that $\theta(y)$ is not dependent on cloudlet selection. This ensures that the integration of admission control does not change cloudlet selection of the mobile

users. The cloudlet serving the user is still evaluated based on Eq. (5.6). As a result of this admission control, the load of the k-th cloudlet at the end of ith time slot is updated in the following way

$$T_k(\rho_k^i) = \min\left(\int_{\mathcal{R}} \theta(y)\frac{\gamma(y)}{s_k(y)}u_k(y)dy, 1-\epsilon\right). \tag{5.14}$$

The cloudlet updates its load based on Eq. (5.8). Thus, the SDN controller restricts the loads of the cloudlets to ensure that the cloudlet assignment problem remains feasible. For this admission control, the relaxed feasible set becomes

$$\tilde{\mathcal{Z}} = \Big\{\rho \;|\rho_k = \int_{\mathcal{R}} \theta(y)\frac{\gamma(y)}{s_k(y)}u_k(y)dy, 0 \le \rho_k \le 1-\epsilon, \forall k \in \mathcal{W},$$
$$0 \le u_k(y) \le 1, \sum_{k=1}^{|\mathcal{W}|} u_k(y) = 1, \;\; \forall k \in \mathcal{W}, \;\; \forall y \in \mathcal{R}\Big\}.$$

As $0 \le \theta(y) \le 1$ is constant, Lemma 1 holds now also, and thus the set remains convex. Further as the integration of admission control does not modify the objective problem of clouldet assignment, thus Lemma 2 still stands true. This ensures that convergence and optimality proofs given previously still hold, thus enabling the traffic load to converge to the optimal solution even with admission control applied. The effect of variation in this parameter θ on the latency will be analyzed in the next section.

5.4.6 Use case for LATA

We consider the use case of *'Shopper Information and Navigation Application in a Shopping Mall'*. In huge shopping malls, there may be many hundreds and thousands of shoppers present at a given point of time. These shoppers might have queries like location and route to a particular store, current offers in different stores, information about the stores which stock a particular item which he is

looking for, etc. Such information can be provided to the shopper using a mobile application of the shopping mall. The shopper might use speech or even images to put his queries in the application. The speech or image must then be analyzed to provide the relevant information to the shopper on his mobile device. Using the cloud services for this application will increase the latency in processing the data and providing the results. Thus it is suitable to use cloudlets to do the speech or image processing and provide the relevant information to the shopper so that the latency involved can be reduced. However, we need to choose an appropriate cloudlet so that this latency is minimized to improve or retain the QoS to the user. We can deploy cloudlets at different locations of this shopping mall to perform these tasks. Many times, a particular cloudlet might be overloaded due to the tasks already offloaded on to it. Thus, we have to select an appropriate cloudlet to optimize the latency in processing the data (which is the task in this case) offloaded by the shoppers. Thus LATA can be employed in this application in selecting the appropriate cloudlet to process the offloaded tasks so that the latency experienced by the shoppers is optimized. There can be a central controller which runs the controller side algorithm to decide the appropriate cloudlet to offload the data. The cloudlets can send the information about their load and their resistance index and send these values to the central controller. This data will be used by the central controller to identify the optimal cloudlet to perform the offloaded task. The following section presents the results and analysis of the simulations performed to evaluate LATA.

5.5 Results and Analysis

The LATA algorithm described in this paper was evaluated through simulations carried out using MATLAB 2017R1 and the results of these simulations are discussed and analyzed in this sections. In these simulations, we consider two dif-

ferent cloudlet topologies (shown in Fig. 5.3 and Fig. 5.4) with three and ten cloudlets respectively, to carry out the performance analysis of our scheme. Note that these two different topologies have been considered for the performance analysis for different network sizes. As it can be seen from Fig. 5.3, there are three-cloudlets which provide service in an area of 100 m $\times$ 100 m whereas in Fig. 5.4 there are ten-cloudlets providing service in the given area. The cloudlets are randomly placed within the network. It is to be noted that the SDN switched network with the SDN Controller at the core connects the different cloudlets in the network. Two kinds of cloudlets are considered here: one with its maximum service rate being 3000 KB/s and the other having maximum service rate as 6000 KB/s. For generating the mobile task offload requests, Homogeneous Poisson Point Process (HPPP) is used. The performance is analyzed for various task arrival rates with the lowest traffic being 20 offload requests arriving in the network per second in the coverage area. The maximum traffic arrival rate has been taken to be 80 requests/s for the three-cloudlet network scenario and 160 requests/s for the 10 cloudlet network scenario. Note that we have taken the different maximum traffic arrival rates (of 80 and 160 requests/s) for these networks considering their traffic handling capabilities. Also, $dis(.)$ denotes the distance of the cloudlet from the mobile device in km. The parameter d_0 has been taken as 15 m (i.e., 0.015 km). The offload requests are considered here to range from 10 KB/packet to 70 KB/-packet by taking into account of the different computational demands of the users (e.g., 70 KB/packet traffic offloaded is from users which are requesting more computationally intensive tasks to be performed like 3D gaming whereas those with 10 KB/packet have lesser computationally intensive tasks offloaded like an application requesting to zip the streamed files into a folder). The location based task offload density ($\gamma(y)$) is calculated based on the model discussed in Section 5.2. The value of the averaging factor ξ (mentioned in Eq. (5.8)) is taken to be 0.95. Keeping this value of ξ, we note that the task-assignment algorithm converges

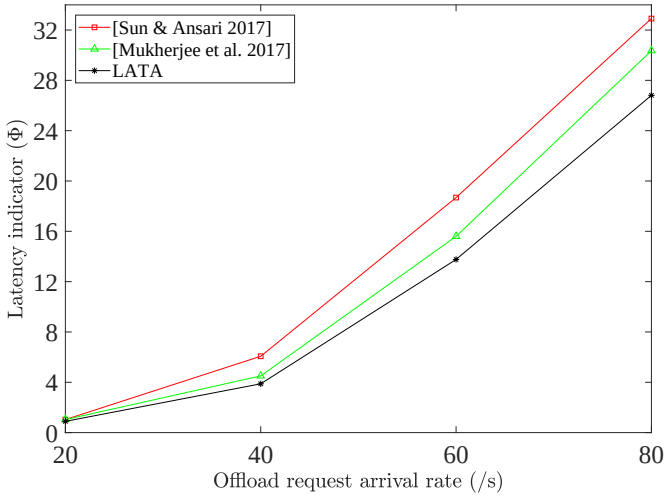


Figure 5.5: Latency variation for varying offload request arrival rates in a network of three cloudlets.

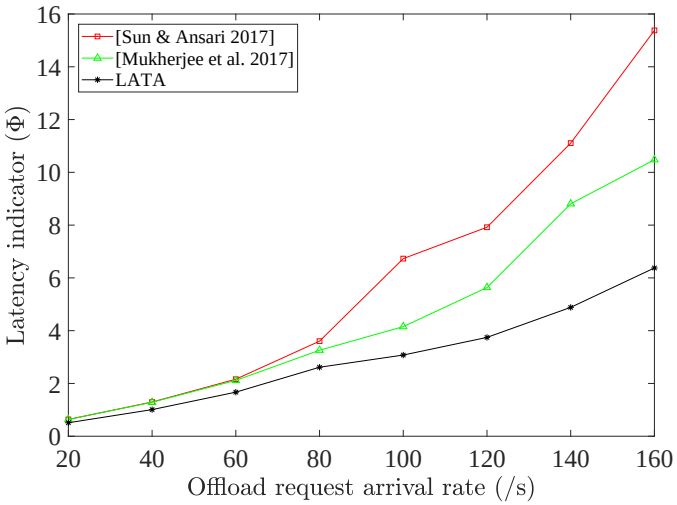


Figure 5.6: Latency variation for varying offload request arrival rates in a network of ten cloudlets.

within 10 iterations. The performance of the our scheme has been compared to the state-of-the-art schemes [Sun & Ansari 2017] and [Mukherjee *et al.* 2019] which have been briefly discussed in Section 2.4.

5.5.1 Latency Performance with Traffic Variation

We study how the network latency performance changes when the request arrival rates of the offloaded traffic are varied. The arrival rates of the offloaded requests

range from 20 to 80 offload requests/s (for the three-cloudlet network) and 20 to 160 offload requests/s (for the ten-cloudlet network). We calculate the latency indicator for the whole network (Φ) corresponding to these arrival rates of requests. As stated in Section 5.2, this latency indicator is a unit-less quantity. The numerical value of α has been taken as 1 for the results in this section. However the effect of α on latency has been studied in the next subsection. The results pertaining to the network having three cloudlets are shown in Fig. 5.5 whereas those pertaining to network of ten cloudlets are shown in Fig. 5.6. From the results, we observe that as the traffic load increases the latency increases in all three schemes; this is due to a higher load on the cloudlets. It can be seen from the results that the LATA scheme presented in this chapter has better performance than the schemes presented in [Mukherjee *et al.* 2019] and [Sun & Ansari 2017] since it achieves lower latency. Further, we also see that performance gain of LATA over the schemes in [Sun & Ansari 2017] and [Mukherjee *et al.* 2019] increases as the traffic is increased. For example, in the three-cloudlet scenario, for an average traffic arrival rate of 80 requests/s the latency for the scheme in [Sun & Ansari 2017] is 13% higher and for [Mukherjee *et al.* 2019] is 10% higher as compared to LATA. In the ten-cloudlet scenario, for average traffic arrival rate of 160 requests/s, the latency is 140% higher for [Sun & Ansari 2017] and 64% higher for [Mukherjee *et al.* 2019] as compared to the LATA scheme. The reasoning for this behavior is as follows. For the case of very high offload arrival rate, the schemes in [Sun & Ansari 2017] and [Mukherjee *et al.* 2019] prefer offloading many of the requests to the cloudlets closest to them which leads to some of the cloudlets getting overloaded (which have more number of users near them). This causes an increase in the overall latency of the network. For the scheme presented in [Mukherjee *et al.* 2019], the task is assigned to the nearest cloudlet if it is not overloaded, else it assigns the task to the remote cloudlet which is able to serve the task with least latency. Note that the model proposed by them does not take into consideration of the current load

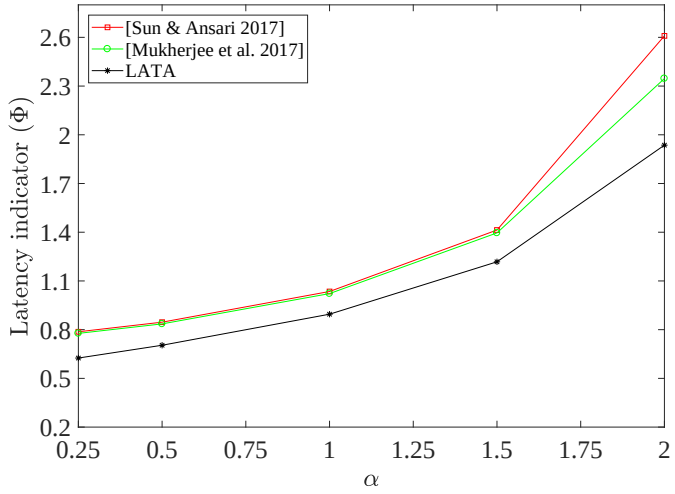


Figure 5.7: Latency behavior with different α values for a network with three cloudlets and traffic of 20 requests/s.

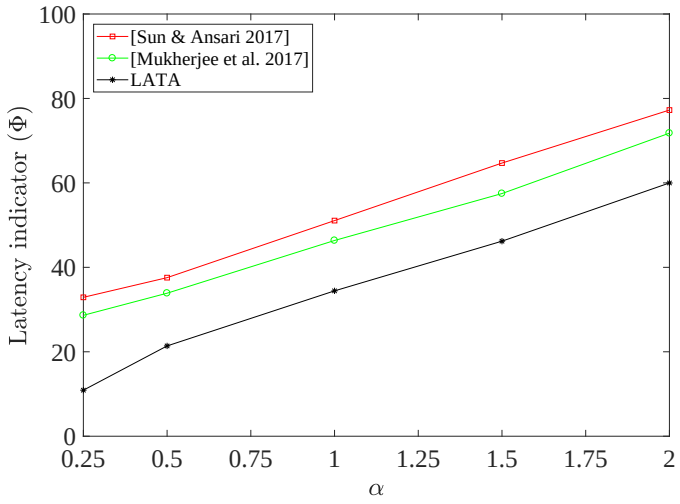


Figure 5.8: Latency behavior with different α values for a network with three cloudlets and traffic of 80 requests/s.

on the remote cloudlet while making the task offload decisions, and thus is unable to optimize the network level latency. However, in case of LATA, in addition to the distance of the cloudlet from the request, the current load of the cloudlets is considered for making the task-assignment decisions. This gives an improved latency on account of its load balancing nature.

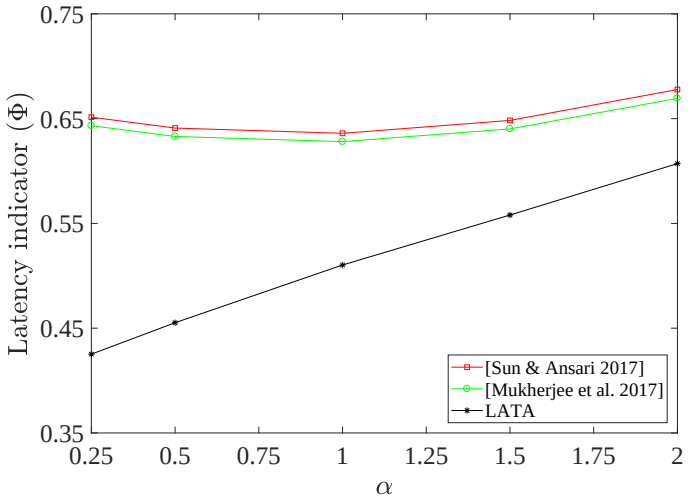


Figure 5.9: Latency behavior with different α values for a network with ten cloudlets and traffic of 20 requests/s.

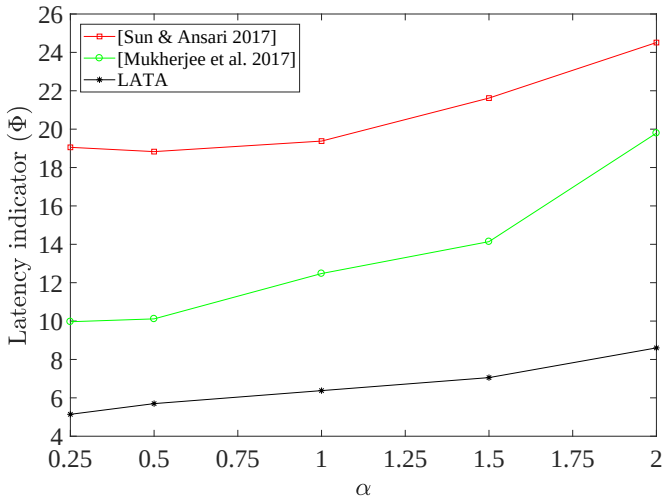


Figure 5.10: Latency behavior with different α values for a network with ten cloudlets and traffic of 160 requests/s.

5.5.2 Latency Performance with Varying Network Parameters

In this section, we study the latency performance as the network parameter α is varied. In the formulation shown in Eq. (5.1), the parameter α captures the quality of the network in terms of the latency associated with the distance of the requests from the cloudlet. With other parameters fixed, a higher value of α would indicate that the cloudlet provides a lower service rate at the location where the request is originated. We vary α from 0.25 to 2 for the two different network

scenarios (three-cloudlet and ten-cloudlet networks) and study the performance of the system. The analysis is done considering the two extreme values of the request rate, i.e., the minimum and the maximum requests/second. We fix the value of traffic to a certain value and see the latency performance on varying the network parameter α. Note that a higher value of this parameter (α) indicates a slower system. In Fig. 5.7 and Fig. 5.9, we show the change in the latency as α is varied for an average traffic of 20 request/s for three and ten-cloudlet network, respectively. In Fig. 5.8 and Fig. 5.10, we show the change in the latency for an average traffic of 80 and 160 request/s respectively, which are the upper limits of traffic taken for network with three and ten cloudlets, respectively. Note that in the three-cloudlet network, for its maximum traffic, the latency indicator's value for LATA increases from around 10 to nearly 60 (increase by 6 times) as α varies from 0.25 to 2. However, in the ten-cloudlet network for the maximum traffic, the latency indicator increases from 5 to 8.5 (increase by 1.7 times only). From this we conclude that the increase in latency with an increase in α is less emphatic for the ten-cloudlet network as compared to the three-cloudlet network. This is because the requests generated in the network are served by a greater number of cloudlets in ten-cloudlet network. Note that in all cases we find that LATA outperforms the schemes presented in [Sun & Ansari 2017] and [Mukherjee *et al.* 2019] because of making optimal cloudlet assignment based on the service rate, distance of the cloudlet from the user offloading the task as well as the current load on the cloudlet.

5.5.3 Latency Performance with Admission Control

We analyze the latency performance when the admission control parameter θ is varied. As mentioned in the previous section (Section 5.4.5), the parameter $\theta(y)$ depicts the probability with which a mobile device at a location y is admitted into the network. For example, $\theta(y)$ =1 would indicate that all the offload requests

arriving in the network at location y are admitted, whereas $\theta(y) = 0.7$ would indicate that requests arriving at that location are allowed admittance into the network with a probability 0.7 (in-turn denied admittance with probability 0.3). We have considered this parameter to be the same for the entire area served by the cloudlets (i.e. for all y). We vary this parameter while keeping the other parameters constant. For this analysis, we keep the parameter α mentioned in Eq. (5.1) as unity. The simulations for both three-cloudlet and ten-cloudlet networks for different arrival rates are performed. For a three-node network, the LATA algorithm for different average traffic arrival rates varying from 80 requests/s to 140 requests/s is simulated and the results are depicted in Fig. 5.11. Note that for higher traffic arrival request rates (i.e., 120 requests/s and 140 requests/s), when the admission probability is high (> 0.7), the latency becomes unacceptably high. This indicates that the network cannot manage such high traffic. However, if the network has such high traffic, the admission control parameter can be reduced to enable lower latency for the served users (this is however at the cost of some of the users being dropped/denied admission into the network). For example, when the arrival rate is 140 requests/s and θ is set as 0.5, the latency indicator for the users being served is limited to 21 (at the cost of around 50% of the requests being dropped).

The latency performance for the ten-cloudlet network for varying values of θ for average arrival rates varying from 160 requests/s to 400 requests/sec is depicted in Fig. 5.12. Note that when the offload request arrival rate is low, admission control is not required and the θ value can be kept as 1. However, when the traffic is high, admission control can be applied to reduce the latency experienced by the users served by the cloudlet network.

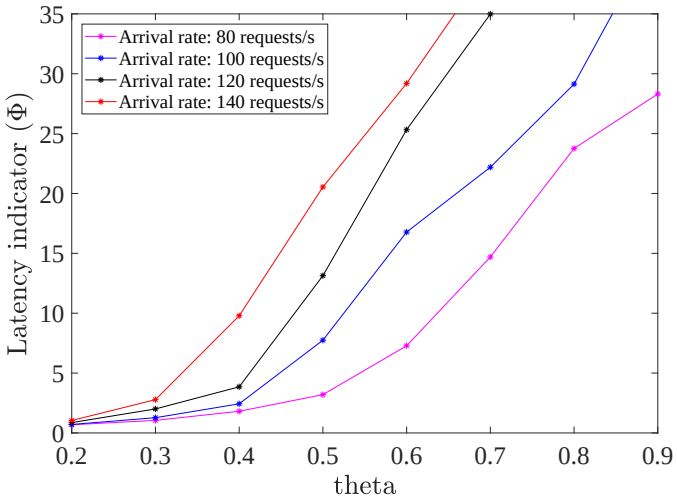


Figure 5.11: Latency behavior for different admission control parameter values for a network with three cloudlets.

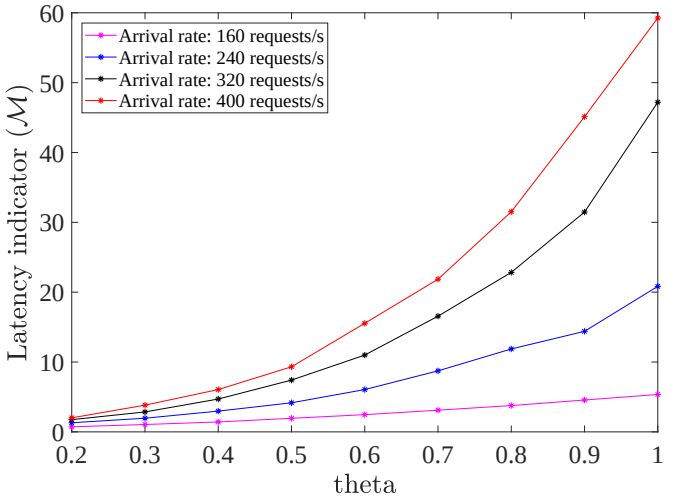


Figure 5.12: Latency behavior for different admission control parameter values for a network with ten cloudlets.

5.6 Conclusion

This chapter has presented a novel task-assignment scheme (named LATA) which makes task-assignment decisions for a network of cloudlets that serves computationally intensive tasks offloaded by the mobile devices. The task-assignment aims at reducing the network latency to enhance the QoS experienced by the mobile users served by the cloudlets. This task-assignment scheme converges to an optimal load for all the cloudlets which minimizes the overall latency in servicing

the offloaded tasks. The optimality and convergence property of this scheme have been proved mathematically. Through simulation results, it has been shown that this scheme gives a better performance in comparison to the existing schemes. An admission control policy to ensure that the task-assignment scheme remains optimal even when traffic in the network is more than what it can handle has also been presented.

Chapter 6

Conclusion and Future Work

WSN and edge computing are two major enabling technologies in the IoT paradigm. This thesis has addressed two important algorithms in WSNs and edge computing, viz., Time Synchronization Protocols (TSPs) in WSNs and latency aware task-assignment schemes in edge cloudlet network. Next section presents the conclusions of this thesis.

6.1 Conclusion

A TSP for cluster-based WSNs named E-SATS has been presented in Chapter 3. E-SATS has been proposed to address two major drawbacks of the existing TSPs for cluster-based WSNs. First, most of the existing protocols are simulator-based protocols which make many assumptions at high-level of abstraction which may not be valid for practical WSNs. Second, the existing TSPs do not consider the effect of the Line-of-Sight (LOS) conditions on the performance of a TSP. E-SATS uses fewer computations and consumes lesser energy (than other protocols) while achieving microsecond-level synchronization accuracy. E-SATS accounts for the deterministic and non-deterministic delays, which seriously affect the accuracy of

a synchronization protocol. It involves a cluster-formation phase by which the nodes in the network are organized into clusters. The cluster members were then synchronized to their respective cluster heads using simple two-way exchanges. This protocol has been implemented on WSN testbed of 30 nodes and its performance has been analyzed in two different environments, viz., Line-of-Sight (LOS) environment (where all the nodes in the WSN were LOS) and mixed-LOS environment (where half of the nodes were LOS and the rest were Non-Line-of-Sight (NLOS)). It has been compared to other state-of-the-art TSPs for clustered WSNs. The experimental results have shown that E-SATS achieved better synchronization accuracy compared to other protocols. Also, it has been shown analytically that E-SATS has higher computational and energy efficiency than other protocols. Since the performance of E-SATS has been proved on a WSN testbed, it gives a credible proof that E-SATS is ideal for resource-constrained WSN nodes which require a simple and energy efficient yet accurate TSP. Further, it has also been shown that as the number of nodes which are NLOS increases, the synchronization error of all the TSPs increases significantly.

A decentralized TSP based on a message-passing method called Integrated Cooperative Synchronization (ICS) has been proposed in Chapter 4. ICS is a Mean-Field based delay compensated method to evaluate maximum a-posterior estimate of the clock parameters. It integrates the measurement phase with message-passing phase thereby removing the need of a separate mechanism to synchronize these two phases. It uses an extended factorization of the a-posterior function and specific message-passing schedule to achieve synchronization. In the normal operation, this protocol uses every packet exchanged between two nodes both for obtaining the time measurements and for estimating the clock parameters. A link initialization phase has been proposed to handle the links where a prior estimate of the delay was not present. It has been shown that ICS achieves high accuracy (sub-microsecond accuracy) and uses simpler computations compared to the

existing message-passing methods.

A latency aware task-assignment scheme (LATA) for a network edge cloudlets to handle the tasks offloaded by mobile devices on to the edge cloudlets has been proposed in Chapter 5. This scheme makes task-assignment decisions in such a way that the overall network latency experienced by the offloaded tasks is optimized. The optimality of this scheme has been proven mathematically. The performance of this scheme in varying task offloading traffic conditions has been analyzed and compared with existing state-of-the-art task-assignment schemes for edge cloudlets. It has been shown that LATA outperforms the existing schemes by achieving reduced latency. Also, the latency performance has been assessed for varying network conditions. An admission control policy which maintains the optimality of this scheme in high task offloading traffic conditions has also been presented and its performance has been analyzed for varying values of network admission parameter.

Thus, two time synchronization protocols for WSNs and a task-assignment scheme for edge cloudlets have been presented in this thesis. The synchronization protocol proposed for cluster-based WSN, i.e., E-SATS has been proved to be computationally efficient, energy efficient while achieving higher synchronization accuracy of the order of microseconds. Thus, it is suitable for resource-constrained WSN nodes. The decentralized message-passing based synchronization protocol, i.e., ICS, has been proved to achieve high synchronization accuracy. The task-assignment scheme for the edge cloudlet network has achieved optimal latency for processing the tasks offloaded by edge cloudlets thereby improving the QoS experienced by the mobile users.

Next section presents the future research directions based on this doctoral research.

6.2 Future Work

The TSP proposed for clustered WSN achieves synchronization for a network which is static to a large extent. However, its performance in a network where the nodes are highly mobile has not been tested. The performance of E-SATS for such a dynamic network has to be tested and the cluster formation phase may have to be simplified further to reduce the overhead due to frequent cluster formation. Further, E-SATS achieves a synchronization accuracy of the order of 100s of microseconds. Such an accuracy is not sufficient for some industrial and control system application which requires sub-microsecond accuracy. Thus, there is a need to improve the synchronization accuracy that can be achieved in a cluster-based WSN.

ICS proposed in Chapter 4 shows its convergence in numerical evaluations. However, its convergence is required to be proved mathematically. ICS can be adapted for a cluster-based WSN and it can be compared with E-SATS in terms of synchronization accuracy, immediacy, etc. Further, this synchronization scheme can be extended to achieve simultaneous localization and synchronization. Localization protocols are used to find the location of the nodes whose position changes dynamically due to their mobility. Since the localization problem is closely related to synchronization, this extension is highly feasible.

LATA presented in Chapter 5 proposed a task-assignment scheme for edge cloudlets in which the tasks were offloaded on to a nearby cloudlet and the results of the offloaded task request were delivered to the mobile device through the same cloudlet regardless of which cloudlet served the task request. However, since the mobile devices change their position, the current location of the mobile device must be tracked so that a cloudlet closest to the current location of the mobile device can be used to deliver the results. Further, there is a need to incorporate energy-awareness in the task-assignment scheme to optimize the energy con-

sumed to serve the tasks requests. Energy-awareness is very essential when the cloudlets are energy constrained. In green cloudlet networks [**?**], the cloudlets are solar powered and hence are energy constrained. In such a scenario, it becomes important to optimize both energy and latency while serving the requests of mobile devices. Thus, LATA can be modified to incorporate energy-awareness.

References

[Abdelaal & Theel 2013] M. Abdelaal and O. Theel. *An efficient and adaptive data compression technique for energy conservation in wireless sensor networks*. In 2013 IEEE Conference on Wireless Sensor (ICWISE), pages 124–129, Dec 2013. 68

[Ahmad *et al.* 2012] A. Ahmad, D. Zennaro, E. Serpedin and L. Vangelista. *A Factor Graph Approach to Clock Offset Estimation in Wireless Sensor Networks*. IEEE Trans. Inf. Theory, vol. 58, no. 7, pages 4244–4260, July 2012. 20

[Ahmed & Ahmed 2016] A. Ahmed and E. Ahmed. *A survey on mobile edge computing*. In 2016 10th International Conference on Intelligent Systems and Control (ISCO), pages 1–8, Jan 2016. 45

[Ali *et al.* 2017] M. Ali, S. Lee and B. H. Kang. *An IoT-based CBL methodology to create realworld clinical cases for medical education*. In 2017 International Conference on Information and Communication Technology Convergence (ICTC), pages 1037–1040, Oct 2017. 1

[Amiri 2010] Moslem Amiri. Measurements of energy consumption and execution time of different operations on Tmote Sky sensor motes. Master's thesis, Masaryk University, Brno, Czech Republic, 2010. 68

[Barbera *et al.* 2014] M. V. Barbera, S. Kosta, A. Mei, V. C. Perta and J. Stefa. *Mobile offloading in the wild: Findings and lessons learned through a real-life experiment with a new cloud-aware system*. In IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, pages 2355–2363, April 2014. 45

[Batty *et al.* 2012] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis and Y. Portugali. *Smart cities of the future*. The European Physical Journal Special Topics, vol. 214, no. 1, pages 481–518, Nov 2012. 1

[Bonomi *et al.* 2012] Flavio Bonomi, Rodolfo Milito, Jiang Zhu and Sateesh Addepalli. *Fog Computing and Its Role in the Internet of Things*. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM. 45

[Boyd & Vandenberghe 2004] S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge Univ. Press, Cambridge, U.K., 2004. 112

[CC2] *CC2420- Datasheet*. http://www.ti.com/lit/ds/symlink/cc2420.pdf. Accessed: Apr 22, 2019. 65, 68

[Chalapathi *et al.* 2016] G. S. S. Chalapathi, R. Manekar, V. Chamola, K. R. Anupama and S. Gurunarayanan. *Hardware validated efficient and simple Time Synchronization protocol for clustered WSN*. In 2016 IEEE Region 10 Conference (TENCON), pages 2162–2166, Nov 2016. 19, 56, 60

[Chalapathi *et al.* 2019a] G. S. S. Chalapathi, Vinay Chamola and S. Gurunarayanan. *A Testbed Validated Simple Time Synchronization Protocol for Clustered Wireless Sensor Networks for IoT*. Journal of Intelligent and Fuzzy Systems, vol. 36, no. 5, pages 4531–4543, May 2019. 56

[Chalapathi *et al.* 2019b] G. S. S. Chalapathi, Vinay Chamola, S. Gurunarayanan and Biplab Sikdar. *E-SATS: An Efficient and Simple Time Synchronization Protocol for Cluster-based Wireless Sensor Networks*. IEEE Sensors, May 2019. Accepted, Available online. 56

[Chalapathi *et al.* 2019c] G. S. S. Chalapathi, Vinay Chamola, Chen-Khong Tham and S. Gurunarayanan. *An Optimal Delay Aware Task Assignment Scheme for Wireless SDN Networked Edge Cloudlets*. Future Generation Computing Systems, January 2019. under review. 101

[Chalapathi *et al.* 2019d] G. S. S. Chalapathi, B. Etzlinger, S. Gurunarayanan and A. Springer. *Integrated Cooperative Synchronization for Wireless Sensor Networks*. IEEE Wireless Communications Letters, vol. 8, no. 3, pages 701–704, June 2019. 85

[Chamola *et al.* 2017] V. Chamola, C. K. Tham and G. S. S. Chalapathi. *Latency aware mobile task assignment and load balancing for edge cloudlets*. In 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pages 587–592, March 2017. 101

[Chaudhari *et al.* 2008] Q. M. Chaudhari, E. Serpedin and K. Qaraqe. *On Maximum Likelihood Estimation of Clock Offset and Skew in Networks With Exponential Delays*. IEEE Transactions on Signal Processing, vol. 56, no. 4, pages 1685–1697, April 2008. 62

[Chun *et al.* 2011] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik and Ashwin Patti. *CloneCloud: Elastic Execution Between Mobile Device and Cloud*. In Proceedings of the Sixth Conference on Computer Systems, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM. 11

[Cis] *Cisco Catalyst 3650, 10 Gbps Wireless SDN Switch from Cisco*. https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-3650-series-switches/data_sheet-c78-729449.html. Accessed: Apr 22, 2019. 103

[Dai & Han 2004] Hui Dai and Richard Han. *TSync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks*. SIGMOBILE Mob. Comput. Commun. Rev., vol. 8, no. 1, pages 125–139, Jan. 2004. 18

[Djenouri & Bagaa 2016] D. Djenouri and M. Bagaa. *Synchronization Protocols and Implementation Issues in Wireless Sensor Networks: A Review*. IEEE Systems Journal, vol. 10, no. 2, pages 617–627, June 2016. 8, 19

[Ejaz *et al.* 2017] W. Ejaz, M. Naeem, A. Shahid, A. Anpalagan and M. Jo. *Efficient Energy Management for the Internet of Things in Smart Cities*. IEEE Communications Magazine, vol. 55, no. 1, pages 84–91, January 2017. 1

[Elijah *et al.* 2018] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow and M. N. Hindia. *An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges*. IEEE Internet of Things Journal, vol. 5, no. 5, pages 3758–3773, Oct 2018. 1

[Elson *et al.* 2002] Jeremy Elson, Lewis Girod and Deborah Estrin. *Fine-grained Network Time Synchronization Using Reference Broadcasts*. SIGOPS Oper. Syst. Rev., vol. 36, no. SI, pages 147–163, Dec. 2002. 6, 8, 19

[Elsts *et al.* 2016] A. Elsts, S. Duquennoy, X. Fafoutis, G. Oikonomou, R. Piechocki and I. Craddock. *Microsecond-Accuracy Time Synchronization Using the IEEE 802.15.4 TSCH Protocol*. In 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), pages 156–164, Nov 2016. 9

[Etzlinger *et al.* 2013a] B. Etzlinger, D. Bartel, W. Haselmayr and A. Springer. *Mean field message passing for cooperative simultaneous ranging and synchronization*. In Global Conf. Sig. and Inf. Process. (GlobalSIP), pages 583–586, Dec. 2013. 10, 40, 44, 82, 83, 87, 89

[Etzlinger *et al.* 2013b] Bernhard Etzlinger, Florian Meyer, Andreas Springer, Franz Hlawatsch and Henk Wymeersch. *Cooperative simultaneous localization and synchronization: A distributed hybrid message passing algorithm*. In Proc. Asilomar Conf. Sig., Syst., Comput., Pacific Grove, CA, Nov. 2013. 9

[Etzlinger *et al.* 2014] B. Etzlinger, H. Wymeersch and A. Springer. *Cooperative synchronization in wireless networks*. IEEE Trans. Signal Process., vol. 62, no. 11, pages 2837–2849, Jun. 2014. 9, 10, 20, 40, 41, 43, 82, 83, 86, 87, 95, 97

[Etzlinger *et al.* 2017] B. Etzlinger, F. Meyer, F. Hlawatsch, A. Springer and H. Wymeersch. *Cooperative Simultaneous Localization and Synchronization in Mobile Agent Networks*. IEEE Transactions on Signal Processing, vol. 65, no. 14, pages 3587–3602, July 2017. 19, 44, 57, 94

[Ganeriwal *et al.* 2003] Saurabh Ganeriwal, Ram Kumar and Mani B. Srivastava. *Timing-sync Protocol for Sensor Networks*. In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys '03, pages 138–149, New York, NY, USA, 2003. ACM. 8, 18

[Garone *et al.* 2015] Emanuele Garone, Andrea Gasparri and Francesco Lamonaca. *Clock synchronization protocol for wireless sensor networks with bounded communication delays*. Automatica, vol. 59, pages 60 – 72, 2015. 40, 41

[Gentz *et al.* 2016] R. Gentz, A. Scaglione, L. Ferrari and Y. . P. Hong. *PulseSS: A Pulse-Coupled Synchronization and Scheduling Protocol for Clustered Wireless Sensor Networks*. IEEE Internet of Things Journal, vol. 3, no. 6, pages 1222–1234, Dec 2016. 38, 39

[Ghayvat *et al.* 2015] Hemant Ghayvat, Subhas Mukhopadhyay, Xiang Gui and Nagender Suryadevara. *WSN- and IOT-Based Smart Homes and Their Extension to Smart Buildings*. Sensors, vol. 15, no. 5, pages 10350–10379, 2015. 4

[Gkikopouli *et al.* 2012] A. Gkikopouli, G. Nikolakopoulos and S. Manesis. *A survey on Underwater Wireless Sensor Networks and applications*. In 2012 20th Mediterranean Conference on Control Automation (MED), pages 1147–1154, July 2012. 5

[Han & Ansari 2014] T. Han and N. Ansari. *Powering mobile networks with green energy*. IEEE Wireless Communications, vol. 21, no. 1, pages 90–96, February 2014. 105

[Handcock *et al.* 2009] Rebecca N. Handcock, Dave L. Swain, Greg J. Bishop-Hurley, Kym P. Patison, Tim Wark, Philip Valencia, Peter Corke and Christopher J. O'Neill. *Monitoring Animal Behaviour and Environmental Interactions Using Wireless Sensor Networks, GPS Collars and Satellite Remote Sensing*. Sensors, vol. 9, no. 5, pages 3586–3603, 2009. 93

[Heinzelman *et al.* 2000] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan. *Energy-efficient communication protocol for wireless microsensor networks*. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, pages 10 pp. vol.2–, Jan 2000. 24

[Huerta-Canepa & Lee 2010] Gonzalo Huerta-Canepa and Dongman Lee. *A Virtual Cloud Computing Provider for Mobile Devices*. In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, MCS '10, pages 6:1–6:5, New York, NY, USA, 2010. ACM. 47

[Jabbarifar *et al.* 2010] M. Jabbarifar, A. S. Sendi, H. Pedram, M. Dehghan and M. Dagenais. *L-SYNC: Larger Degree Clustering Based Time-Synchronisation for Wireless Sensor Network*. In 2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications, pages 171–178, May 2010. 8, 35, 36, 39, 53, 55, 71, 75, 76

[Jararweh *et al.* 2013] Y. Jararweh, L. Tawalbeh, F. Ababneh and F. Dosari. *Resource Efficient Mobile Computing Using Cloudlet Infrastructure*. In 2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks, pages 373–377, Dec 2013. 45

[Jia *et al.* 2016] M. Jia, W. Liang, Z. Xu and M. Huang. *Cloudlet load balancing in wireless metropolitan area networks*. In IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pages 1–9, April 2016. 100

[Kao *et al.* 2017] Y. H. Kao, B. Krishnamachari, M. R. Ra and F. Bai. *Hermes: Latency Optimal Task Assignment for Resource-constrained Mobile Computing*. IEEE Transactions on Mobile Computing, vol. 16, no. 11, pages 3056–3069, Nov 2017. 47, 48

[Kim *et al.* 2006] Hyunhak Kim, Daeyoung Kim and Seong eun Yoo. *Cluster-based hierarchical time synchronization for multi-hop wireless sensor networks*. In 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), volume 2, pages 5 pp.–, April 2006. 25, 39

[Kim *et al.* 2008] Sun-Jin Kim, Jung Hae Seo, Jonnalagadda Krishna and Sun-Joong Kim. *Wireless sensor network based asset tracking service*. In PICMET '08 - 2008 Portland International Conference on Management of Engineering Technology, pages 2643–2647, July 2008. 4

[Kim *et al.* 2012] H. Kim, G. de Veciana, X. Yang and M. Venkatachalam. *Distributed alpha -Optimal User Association and Cell Load Balancing in Wireless Networks*. IEEE/ACM Transactions on Networking, vol. 20, no. 1, pages 177–190, Feb 2012. 108

[Kleinrock 1976] L. Kleinrock. Queueing systems vol. 2: Computer applications. Wiley-Interscience, New York, U.S.A., 1976. 105

[Kong *et al.* 2010] L. Kong, Q. Wang and Y. Zhao. *Time Synchronization algorithm based on Cluster for WSN*. In 2010 2nd IEEE International Conference on Information Management and Engineering, pages 126–130, April 2010. 29, 39

[Kosta *et al.* 2012] S. Kosta, A. Aucinas, Pan Hui, R. Mortier and Xinwen Zhang. *ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading*. In 2012 Proceedings IEEE INFOCOM, pages 945–953, March 2012. 100

[Krishnamurthy *et al.* 2005] Lakshman Krishnamurthy, Robert Adler, Phil Buonadonna, Jasmeet Chhabra, Mick Flanigan, Nandakishore Kushalnagar, Lama Nachman and Mark Yarvis. *Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea*. In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys '05, pages 64–75, New York, NY, USA, 2005. ACM. 6

[Kschischang *et al.* 2001] F.R. Kschischang, B.J. Frey and H.-A. Loeliger. *Factor graphs and the sum-product algorithm*. IEEE Trans. Inf. Theory, vol. 47, no. 2, pages 498–519, Feb. 2001. xii, 20, 21, 22

[Kumar *et al.* 2013] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu and Bharat Bhargava. *A Survey of Computation Offloading for Mobile Systems*. Mobile Networks and Applications, vol. 18, no. 1, pages 129–140, Feb 2013. 46

[Lee *et al.* 2014] D. Lee, S. Zhou, X. Zhong, Z. Niu, X. Zhou and H. Zhang. *Spatial modeling of the traffic density in cellular networks*. IEEE Wireless Communications, vol. 21, no. 1, pages 80–88, February 2014. 103

[Leng & Wu 2011] Mei Leng and Yik-Chung Wu. *Distributed Clock Synchronization for Wireless Sensor Networks Using Belief Propagation*. IEEE Trans. Signal Process., vol. 59, no. 11, pages 5404–5414, Nov. 2011. 9, 10, 20, 40, 43

[Li & Liu 2009] Mo Li and Yunhao Liu. *Underground Coal Mine Monitoring with Wireless Sensor Networks*. ACM Trans. Sen. Netw., vol. 5, no. 2, pages 10:1–10:29, Apr. 2009. 6

[Lim *et al.* 2016] Roman Lim, Balz Maag and Lothar Thiele. *Time-of-Flight Aware Time Synchronization for Wireless Embedded Systems*. In Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks, EWSN '16, pages 149–158, USA, 2016. Junction Publishing. 31, 41, 83

[Lin *et al.* 2017] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang and W. Zhao. *A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications*. IEEE Internet of Things Journal, vol. 4, no. 5, pages 1125–1142, Oct 2017. 1

[Liu *et al.* 2014] D. Liu, Y. Chen, K. K. Chai and T. Zhang. *Distributed delay-energy aware user association in 3-tier HetNets with hybrid energy sources*. In 2014 IEEE Globecom Workshops (GC Wkshps), pages 1109–1114, Dec 2014. 104, 105

[Liu *et al.* 2015] D. Liu, Y. Chen, K. K. Chai, T. Zhang and M. Elkashlan. *Two-Dimensional Optimization on User Association and Green Energy Allocation for HetNets With Hybrid Energy Sources*. IEEE Transactions on Communications, vol. 63, no. 11, pages 4111–4124, Nov 2015. 105

[Liu *et al.* 2016] J. Liu, Y. Mao, J. Zhang and K. B. Letaief. *Delay-optimal computation task scheduling for mobile-edge computing systems*. In 2016 IEEE International Symposium on Information Theory (ISIT), pages 1451–1455, July 2016. 48

[Lu & Yang 2010] Xin Lu and Shuang-Hua Yang. *Thermal energy harvesting for WSNs*. In 2010 IEEE International Conference on Systems, Man and Cy-

bernetics, pages 3045–3052, Oct 2010. 6

[Lu *et al.* 2016] Y. Lu, A. Savvaris, A. Tsourdos and M. Bevilacqua. *Vibration energy harvesters for wireless sensor networks for aircraft health monitoring*. In 2016 IEEE Metrology for Aerospace (MetroAeroSpace), pages 25–32, June 2016. 6

[Mach & Becvar 2017] P. Mach and Z. Becvar. *Mobile Edge Computing: A Survey on Architecture and Computation Offloading*. IEEE Communications Surveys Tutorials, vol. 19, no. 3, pages 1628–1656, thirdquarter 2017. 46

[Mahmud *et al.* 2018] Redowan Mahmud, Ramamohanarao Kotagiri and Rajkumar Buyya. Fog computing: A taxonomy, survey and future directions, pages 103–130. Springer Singapore, Singapore, 2018. 12, 100

[Mamun-Or-Rashid *et al.* 2005] M. Mamun-Or-Rashid, Choong Seon Hong and Chi-Hyung In. *Passive cluster based clock synchronization in sensor network*. In Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05), pages 340–345, July 2005. 37, 39

[Manjeshwar & Agrawal 2001] A. Manjeshwar and D. P. Agrawal. *TEEN: a routing protocol for enhanced efficiency in wireless sensor networks*. In Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001, pages 2009–2015, April 2001. 24

[Mao *et al.* 2016] Y. Mao, J. Zhang and K. B. Letaief. *Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices*. IEEE Journal on Selected Areas in Communications, vol. 34.12, pages 3590–3605, Dec 2016. 48

[Mao *et al.* 2017] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief. *A Survey on Mobile Edge Computing: The Communication Perspective*. IEEE Communications Surveys Tutorials, vol. 19.4, pages 2322–2358, Fourthquarter 2017. 46

[Marcelloni & Vecchio 2009] Francesco Marcelloni and Massimo Vecchio. *An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks*. The Computer Journal, vol. 52, no. 8, pages 969–987, 2009. 68

[Maróti *et al.* 2004] Miklós Maróti, Branislav Kusy, Gyula Simon and Ákos Lédeczi. *The flooding time synchronization protocol*. In Proc. 2nd Int. Conf. on Embedded networked sensor systems, pages 39–49, New York, NY, USA, Nov. 2004. ACM. 8, 18, 31

[Meyer *et al.* 2018] F. Meyer, B. Etzlinger, Z. Liu, F. Hlawatsch and M. Z. Win. *A Scalable Algorithm for Network Localization and Synchronization*. IEEE Internet of Things Journal, vol. 5, no. 6, pages 4714–4727, Dec 2018. 19, 57, 94

[Milenković *et al.* 2006] Aleksandar Milenković, Chris Otto and Emil Jovanov. *Wireless sensor networks for personal health monitoring: Issues and an implementation*. Computer Communications, vol. 29, no. 13, pages 2521 – 2533, 2006. 4

[Mills 1994] David L. Mills. *Precision Synchronization of Computer Network Clocks*. SIGCOMM Comput. Commun. Rev., vol. 24, no. 2, pages 28–43, Apr. 1994. 7

[Minoli *et al.* 2017] D. Minoli, K. Sohraby and B. Occhiogrosso. *IoT Considerations, Requirements, and Architectures for Smart Buildings—Energy Optimization and Next-Generation Building Management Systems*. IEEE Internet of Things Journal, vol. 4, no. 1, pages 269–283, Feb 2017. 1

[MSP] *Efficient Multiplication and Division Using MSP430™ MCUs*. http://www.ti.com/lit/an/slaa329a/slaa329a.pdf. Accessed: Apr 22, 2019. 80

[Mukherjee *et al.* 2019] A. Mukherjee, D. De and D. G. Roy. *A Power and Latency Aware Cloudlet Selection Strategy for Multi-Cloudlet Environment*. IEEE Transactions on Cloud Computing, vol. 7, no. 1, pages 141–154, Jan 2019. 49, 101, 120, 121, 124

[Muñoz *et al.* 2015] O. Muñoz, A. Pascual-Iserte and J. Vidal. *Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading*. IEEE Transactions on Vehicular Technology, vol. 64, no. 10, pages 4738–4755, Oct 2015. 47, 48

[Nakamura *et al.* 2007] Eduardo F. Nakamura, Antonio A. F. Loureiro and Alejandro C. Frery. *Information Fusion for Wireless Sensor Networks: Methods, Models, and Classifications*. ACM Comput. Surv., vol. 39, no. 3, Sept. 2007. 11

[Nazemi Gelyan *et al.* 2007] Sepideh Nazemi Gelyan, Arash Nasiri Eghbali, Laleh Roustapoor, Seyed Amir Yahyavi Firouz Abadi and Mehdi Dehghan. *SLTP: Scalable Lightweight Time Synchronization Protocol for Wireless Sensor Network*. In Mobile Ad-Hoc and Sensor Networks, pages 536–547, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 8, 35, 36, 39, 53, 55, 71, 75, 76

[Noh *et al.* 2008] K. Noh, E. Serpedin and K. Qaraqe. *A New Approach for Time Synchronization in Wireless Sensor Networks: Pairwise Broadcast Synchronization*. IEEE Transactions on Wireless Communications, vol. 7, no. 9, pages 3318–3322, September 2008. 18

[Ojha *et al.* 2015] Tamoghna Ojha, Sudip Misra and Narendra Singh Raghuwanshi. *Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges*. Computers and Electronics in Agriculture, vol. 118, pages 66 – 84, 2015. 4

[Othman & Shazali 2012] Mohd Fauzi Othman and Khairunnisa Shazali. *Wireless Sensor Network Applications: A Study in Environment Monitoring System*. Procedia Engineering, vol. 41, pages 1204 – 1210, 2012. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012). 4

[Pearl 1988] Judea Pearl. Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. 43

[Plachy *et al.* 2016] J. Plachy, Z. Becvar and E. C. Strinati. *Dynamic resource allocation exploiting mobility prediction in mobile edge computing*. In 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pages 1–6, Sept 2016. 46

[Prayati *et al.* 2010] A. Prayati, Ch. Antonopoulos, T. Stoyanova, C. Koulamas and G. Papadopoulos. *A modeling approach on the TelosB WSN platform power consumption*. Journal of Systems and Software, vol. 83, no. 8, pages 1355 – 1363, 2010. 80

[Qi & Gani 2012] H. Qi and A. Gani. *Research on mobile cloud computing: Review, trend and perspectives*. In Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on, pages 195–202, May 2012. 45

[Qureshi *et al.* 2011] S. S. Qureshi, T. Ahmad, K. Rafique and Shuja ul islam. *Mobile cloud computing as future for mobile applications - Implementation methods*

and challenging issues. In 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, pages 467–471, Sept 2011. 45

[Ray *et al.* 2017] P. P. Ray, M. Mukherjee and L. Shu. *Internet of Things for Disaster Management: State-of-the-Art and Prospects*. IEEE Access, vol. 5, pages 18818–18835, 2017. 1

[Sanaei *et al.* 2014] Z. Sanaei, S. Abolfazli, A. Gani and R. Buyya. *Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges*. IEEE Communications Surveys Tutorials, vol. 16, no. 1, pages 369–392, First 2014. 45

[Sardellitti *et al.* 2015] S. Sardellitti, G. Scutari and S. Barbarossa. *Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing*. IEEE Transactions on Signal and Information Processing over Networks, vol. 1, no. 2, pages 89–103, June 2015. 47, 48

[Sasaki *et al.* 2016] K. Sasaki, N. Suzuki, S. Makido and A. Nakao. *Vehicle control system coordinated between cloud and mobile edge computing*. In 2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pages 1122–1127, Sept 2016. 46

[Satyanarayanan *et al.* 2009] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies. *The Case for VM-Based Cloudlets in Mobile Computing*. IEEE Pervasive Computing, vol. 8, no. 4, pages 14–23, Oct 2009. 12, 45, 46

[Satyanarayanan 1996] M. Satyanarayanan. *Fundamental Challenges in Mobile Computing*. In Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing, PODC '96, pages 1–7, New York, NY, USA, 1996. ACM. 45, 47

[Schenato & Fiorentin 2011] Luca Schenato and Federico Fiorentin. *Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks*. Automatica, vol. 47, no. 9, pages 1878 – 1886, Sept. 2011. 9, 10, 17, 31, 40, 82, 83, 84, 95, 97

[Shi *et al.* 2015] L. Shi, J. Yuan, S. Yu and M. Li. *MASK-BAN: Movement-Aided Authenticated Secret Key Extraction Utilizing Channel Characteristics in Body Area Networks*. IEEE Internet of Things Journal, vol. 2, no. 1, pages 52–62, Feb 2015. 67, 68

[Shi *et al.* 2016] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu. *Edge Computing: Vision and Challenges*. IEEE Internet of Things Journal, vol. 3, no. 5, pages 637–646,

Oct 2016. 100

[Sichitiu & Veerarittiphan 2003] M. L. Sichitiu and C. Veerarittiphan. *Simple, accurate time synchronization for wireless sensor networks*. In 2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003., volume 2, pages 1266–1273 vol.2, March 2003. 19

[Sikka *et al.* 2006] P. Sikka, P. Corke, P. Valencia, C. Crossman, D. Swain and G. Bishop-Hurley. *Wireless ad hoc sensor and actuator networks on the farm*. In 2006 5th International Conference on Information Processing in Sensor Networks, pages 492–499, April 2006. 93, 94

[Sivrikaya & Yener 2004] F. Sivrikaya and B. Yener. *Time Synchronization in Sensor Networks: A Survey*. Netwrk. Mag. of Global Internetwkg., vol. 18, no. 4, pages 45–50, July 2004. 7, 8, 18, 19

[Song *et al.* 2014] Zhenyu Song, Mihai T. Lazarescu, Riccardo Tomasi, Luciano Lavagno and Maurizio A. Spirito. High-Level Internet of Things Applications Development Using Wireless Sensor Networks. Springer International Publishing, Cham, 2014. 4

[Sotres *et al.* 2017] P. Sotres, J. R. Santana, L. Sánchez, J. Lanza and L. Muñoz. *Practical Lessons From the Deployment and Management of a Smart City Internet-of-Things Infrastructure: The SmartSantander Testbed Case*. IEEE Access, vol. 5, pages 14309–14322, 2017. 1

[Soyata *et al.* 2012] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon and W. Heinzelman. *Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture*. In 2012 IEEE Symposium on Computers and Communications (ISCC), pages 000059–000066, July 2012. 46

[Stojmenovic 2014] I. Stojmenovic. *Fog computing: A cloud to the ground support for smart things and machine-to-machine networks*. In 2014 Australasian Telecommunication Networks and Applications Conference (ATNAC), pages 117–122, Nov 2014. 45

[Sun & Ansari 2016] X. Sun and N. Ansari. *EdgeIoT: Mobile Edge Computing for the Internet of Things*. IEEE Communications Magazine, vol. 54, no. 12, pages 22–29, December 2016. 46

[Sun & Ansari 2017] X. Sun and N. Ansari. *Latency Aware Workload Offloading in the Cloudlet Network*. IEEE Communications Letters, vol. 21, no. 7, pages

1481–1484, July 2017. 49, 101, 120, 121, 124

[Suri *et al.* 2016] N. Suri, M. Tortonesi, J. Michaelis, P. Budulas, G. Benincasa, S. Russell, C. Stefanelli and R. Winkler. *Analyzing the applicability of Internet of Things to the battlefield environment*. In 2016 International Conference on Military Communications and Information Systems (ICMCIS), pages 1–8, May 2016. 1

[Taleb *et al.* 2017a] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal and H. Flinck. *Mobile Edge Computing Potential in Making Cities Smarter*. IEEE Communications Magazine, vol. 55, no. 3, pages 38–43, March 2017. 46

[Taleb *et al.* 2017b] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta and D. Sabella. *On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration*. IEEE Communications Surveys Tutorials, vol. 19, no. 3, pages 1657–1681, thirdquarter 2017. 46

[Tel] *Telosb-Datasheet*. http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf. Accessed: Apr 22, 2019. 65

[Tian 2017] Yu-Ping Tian. *Time synchronization in WSNs with random bounded communication delays*. IEEE Transactions on Automatic Control, vol. 62, no. 10, pages 5445–5450, 2017. 41, 82

[Ting *et al.* 2015] Wang Ting, Guo Di, Cai Chun-yang, Tang Xiao-ming and Wang Heng. *Clock Synchronization in Wireless Sensor Networks: Analysis and Design of Error Precision Based on Lossy Networked Control Perspective*. Mathematical Problems in Engineering, 2015. 17

[Truong-Huu *et al.* 2014] T. Truong-Huu, C. K. Tham and D. Niyato. *To Offload or to Wait: An Opportunistic Offloading Algorithm for Parallel Tasks in a Mobile Cloud*. In 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, pages 182–189, Dec 2014. 48

[Wang *et al.* 2017a] C. Wang, C. Liang, F. R. Yu, Q. Chen and L. Tang. *Computation Offloading and Resource Allocation in Wireless Cellular Networks With Mobile Edge Computing*. IEEE Transactions on Wireless Communications, vol. 16, no. 8, pages 4924–4938, Aug 2017. 46

[Wang *et al.* 2017b] C. Wang, F. R. Yu, C. Liang, Q. Chen and L. Tang. *Joint Computation Offloading and Interference Management in Wireless Cellular Networks*

with Mobile Edge Computing. IEEE Transactions on Vehicular Technology, vol. 66, no. 8, pages 7432–7445, Aug 2017. 46

[Wang *et al.* 2017c] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer and K. K. Leung. *Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs*. IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 4, pages 1002–1016, April 2017. 47

[Wang *et al.* 2017d] Zhaowei Wang, Peng Zeng, Mingtuo Zhou, Dong Li and Jintao Wang. *Cluster-Based Maximum Consensus Time Synchronization for Industrial Wireless Sensor Networks*. Sensors, vol. 17, no. 1, pages 141.1–141.16, 2017. 31, 32, 34, 35, 39, 53, 55, 71, 73, 75, 76

[Werner-Allen *et al.* 2005] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees and M. Welsh. *Monitoring volcanic eruptions with a wireless sensor network*. In Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005., pages 108–120, Jan 2005. 6

[Wu *et al.* 2011] Yik-Chung Wu, Qasim M. Chaudhari and Erchin Serpedin. *Clock Synchronization of Wireless Sensor Networks*. IEEE Signal Process. Mag., vol. 28, no. 1, pages 124–138, Jan. 2011. 82

[Wu *et al.* 2015] J. Wu, L. Zhang, Y. Bai and Y. Sun. *Cluster-Based Consensus Time Synchronization for Wireless Sensor Networks*. IEEE Sensors Journal, vol. 15, no. 3, pages 1404–1413, March 2015. 31, 32, 33, 34, 39, 53, 55, 56, 57, 71, 73, 75, 76

[Xie *et al.* 2013] L. Xie, Y. Shi, Y. T. Hou, W. Lou, H. D. Sherali and S. F. Midkiff. *Bundling mobile base station and wireless energy transfer: Modeling and optimization*. In 2013 Proceedings IEEE INFOCOM, pages 1636–1644, April 2013. 11

[Xiong *et al.* 2018] Y. Xiong, N. Wu, Y. Shen and M. Z. Win. *Cooperative Network Synchronization: Asymptotic Analysis*. IEEE Transactions on Signal Processing, vol. 66, no. 3, pages 757–772, Feb 2018. 85

[Yang *et al.* 2004] H. Yang, F. Ye and B. Sikdar. *A dynamic query-tree energy balancing protocol for sensor networks*. In 2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733), volume 3, pages 1715–1720 Vol.3, March 2004. 24

[Yedidia *et al.* 2005] Jonathan S. Yedidia, William T. Freeman and Yair Weiss. *Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms*. IEEE Trans. Inf. Theory, vol. 51, pages 2282–2312, July 2005. 43

[Yi *et al.* 2009] Gao Yi, Sun Guiling, Li Weixiang and Pan Yong. *Wireless sensor node design based on solar energy supply*. In 2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS), volume 3, pages 203–207, Dec 2009. 6

[You *et al.* 2017] C. You, K. Huang, H. Chae and B. H. Kim. *Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading*. IEEE Transactions on Wireless Communications, vol. 16.3, pages 1397–1411, March 2017. 47, 48

[Yu *et al.* 2018] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin and X. Yang. *A Survey on the Edge Computing for the Internet of Things*. IEEE Access, vol. 6, pages 6900–6919, 2018. 46

[Yuan *et al.* 2016] W. Yuan, N. Wu, B. Etzlinger, H. Wang and J. Kuang. *Cooperative Joint Localization and Clock Synchronization Based on Gaussian Message Passing in Asynchronous Wireless Networks*. IEEE Transactions on Vehicular Technology, vol. 65, no. 9, pages 7258–7273, Sep. 2016. 44

[Zanella *et al.* 2014] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi. *Internet of Things for Smart Cities*. IEEE Internet of Things Journal, vol. 1, no. 1, pages 22–32, Feb 2014. 3, 52, 63

[Zennaro *et al.* 2011] Davide Zennaro, Emiliano Dall'Anese, Tomaso Erseghe and Lorenzo Vangelista. *Fast clock synchronization in wireless sensor networks via ADMM-based consensus*. In Proc. 9th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw., pages 148–153, May 2011. 10, 31, 82

[Zennaro *et al.* 2013] D. Zennaro, A. Ahmad, L. Vangelista, E. Serpedin, H. Nounou and M. Nounou. *Network-Wide Clock Synchronization via Message Passing with Exponentially Distributed Link Delays*. IEEE Transactions on Communications, vol. 61, no. 5, pages 2012–2024, May 2013. 10, 20

[Zhang *et al.* 2014] Y. Zhang, D. Niyato, P. Wang and C. K. Tham. *Dynamic offloading algorithm in intermittently connected mobile cloudlet systems*. In 2014 IEEE International Conference on Communications (ICC), pages 4190–4195, June 2014. 48

[Zhang *et al.* 2016a] K. Zhang, Y. Mao, S. Leng, A. Vinel and Y. Zhang. *Delay constrained offloading for Mobile Edge Computing in cloud-enabled vehicular networks*. In 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM), pages 288–294, Sept 2016. 48

[Zhang *et al.* 2016b] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan and Y. Zhang. *Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks*. IEEE Access, vol. 4, pages 5896–5907, 2016. 47

Publications

Journal Papers

1. G. S. S. Chalapathi, Vinay Chamola and S. Gurunarayanan. *A Testbed Validated Simple Time Synchronization Protocol for Clustered Wireless Sensor Networks for IoT*, Journal Of Intelligent and Fuzzy Systems, vol. 36, no. 5, pages 4531-4543, May 2019.

2. G. S. S. Chalapathi, Bernhard Etzlinger, S. Gurunarayanan and Andreas Springer . *Integrated Cooperative Synchronization for Wireless Sensor Networks*, IEEE Wireless Communication Letter, vol. 8, no. 3, pages 701-704, June 2019.

3. G. S. S. Chalapathi, Vinay Chamola, S. Gurunarayanan and Biplab Sikdar. *E-SATS: An Efficient and Simple Time Synchronization Protocol for Cluster-based Wireless Sensor Networks*, IEEE Sensors, May 2019 (Accepted).

4. G. S. S. Chalapathi, Vinay Chamola, Chen-Khong Tham, S. Gurunarayanan and Nirwan Ansari. *An Optimal Delay Aware Task Assignment Scheme for Wireless SDN Networked Edge Cloudlets*, Future Generation Computing Systems , Jan 2019 (Under review).

Conference Papers/ Posters

1. G. S. S. Chalapathi, R. Manekar, V. Chamola, K. R. Anupama and S. Gurunarayanan. *Hardware validated efficient and simple Time Synchronization protocol for clustered WSN*. In 2016 IEEE Region 10 Conference (TENCON), pages 2162–2166, Nov 2016.

2. V. Chamola, C. K. Tham and G. S. S. Chalapathi. *Latency aware mobile task assignment and load balancing for edge cloudlets*. In 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pages 587–592, March 2017.

3. R. Manekar, G. S. S. Chalapathi, V. Chamola, K. R. Anupama and S. Gurunarayanan, *A Simple Time Synchronization Algorithm for WSNs in Smart Grid Applications* In IEEE Symposium on Emerging Topics in Smart and Sustainable Grids, Singapore, Sept. 2016. (unpublished poster).

Biographies

Brief Biography of the Candidate

G Sai Sesha Chalapathi, received his B.E. in Electrical and Electronics Engieering, M.Sc. (Hons.) in Physics and M.E. in Embedded Systems in 2009, 2009 and 2011 respectively from Birla Insititute of Technology and Science (BITS) Pilani, Pilani Campus. He has been working as an Assitant Professor-II (previously called Lecturer) in the Department of Electrical and Electronics Engineering, BITS Pilani since 2012. He has been a Visiting Researcher at Johannes Kepler University, Linz, Austria during the summers of 2016 and 2017. He has also visited National University of Singapore in 2014 for research collaboration. His research interests are Edge Computing, Wireless Sensor Networks, Internet of Things and Embedded Systems.

Brief Biography of the Supervisor

Prof. S. Gurunarayanan received his Ph.D. from BITS Pilani. He is currently a Professor, in the Department of Electrical and Electronics Engineering, BITS Pilani. He has about three decades of teaching experience at BITS-Pilani. His research interests are multi-core processor architectures, cache and memory architectures and embedded systems. He is also serving as the Dean (University-wide), Practice School Division, BITS Pilani.