Enhancing IoT Service Delivery: A Layered Fog Architecture Approach for Sensing and Actuating as a Service

A thesis submitted during 2021 to the University of Hyderabad in partial fulfillment of the award of a Ph.D. degree in the School of Computer and Information Sciences

by

ABDULSALAM ABDO MUSAED ALI ALAMMARI

SCHOOL OF COMPUTER AND INFORMATION SCIENCES

UNIVERSITY OF HYDERABAD

(P.O.) CENTRAL UNIVERSITY

HYDERABAD - 500 046, TELANGANA, INDIA

FEBRUARY, 2021

CERTIFICATE

This is to certify that the thesis entitled **"Enhancing IoT Service Delivery: A Layered Fog Architecture Approach for Sensing and Actuating as a Service"** submitted by **ABDULSALAM ABDO MUSAED ALI ALAMMARI** bearing registration number **14MCPC22** in partial fulfillment of the requirements for award of **Doctor of Philosophy** in the School of **Computer and Information Sciences** is a bonafide work carried out by him under my supervision and guidance.

The thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for award of any degree or diploma. The student has the following publication(s) before submission of the thesis/monograph for adjudication and has produced evidence for the same in the form of acceptance letter or the reprint in the relevant area of her research:

1. Abdulsalam Alammari, Salman Abdul Moiz, Atul Negi; Internet of Things Sensors and Actuators Layered Fog Service Delivery Model SALFSD, MIWAI 2019, Springer, pp 15-25
2. TAbdulsalam Alammari, Salman Abdul Moiz, Atul Negi: TOWARDS TRULY SMART CITY SERVICE PROVIDERS: A VIEW ON ON-DEMAND EVERYTHING AS A SERVICES, Journal of Critical Reviews, Vol 7, Issue 7, 2020, pp 428-436

Further, the student has passed the following courses towards fulfillment of coursework requirement for Ph.D:

Course Code	Name	Credits	Pass/Fail
CS 801	Data Structures and Algorithms	4	Pass
CS 802	Operating Systems and Programming	4	Pass
IT 810	Mobile Computing	4	Pass
AI 853	Data Mining	4	pass

Prof. Salman Abdul Moiz
Supervisor

Prof. Chakravarthy Bhagvati
Dean of SCIS

DECLARATION

I, **ABDULSALAM ABDO MUSAED ALI ALAMMARI**, hereby declare that this thesis entitled **"Enhancing IoT Service Delivery: A Layered Fog Architecture Approach for Sensing and Actuating as a Service"** submitted by me under the guidance and supervision of **Salman Abdul Moiz** is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma.

A report on plagiarism statistics from the University Library is enclosed.

Date:

Name: **ABDULSALAM ABDO MUSAED ALI ALAMMARI**

Signature of the Student:

Reg. No.: **14MCPC22**

Signature of the Supervisor:

To the soul of my father, to my mother, who sacrificed many comforts to get a good education for their children.

To my siblings, my wife, and my children. Without their support and encouragement, this would not have been possible.

Abstract

The increasing amount of objects connected to the internet using various preexisting technologies for communication, storage, ubiquitous and pervasive computing, wireless sensor networks etc, are considered as the Internet of Things (IoT). This emerging technology is defined by the interconnection of different types of smart objects Things provided with sensing, actuating and communication capabilities to interact with their environment and with each other. IoT is the enabler for connecting the physical devices to the digital world, making numerous physical devices connected to the internet and allowing remote control and heterogeneous devices collaboration. This made such devices to be an essential component for several ecosystem applications. Several works have been proposed in the literature for IoT. These works considered standards, reference models, architectures, and communication protocols, etc. In this thesis, the focus in on IoT architectures.

IoT has become the backbone towards realizing smart cities and their ecosystems. Academia and industry have gone a long way in this a path that it has become a reality. Smart cities require a different way of thinking to make several ecosystems, authorities, and ownerships collaborate and work together smoothly and automatically with less human interactions. This thesis includes a proposed vision for smart city service providers towards such a goal.

The cloud paradigm has been a revolutionary computing model in the last decade. It provides a highly virtualized infrastructure with powerful computing hardware and software resources in the form of services

accessed through the internet. Cloud paradigm is characterized by rapid elasticity and flexibility, which allows the users to scale in and out easily on demand. Even though most IoT applications are effectively handled by the cloud model, its centralized architecture and remote location cannot deal with time-sensitive applications. Fog Computing is recently proposed as a new computing paradigm to overcome cloud challenges. Fog computing consists of a widely distributed architecture which offers processing, storage, and networking services between the cloud and the network edge. In other words, fog services can be exploited to process part of tasks closer to data sources rather than entirely relying on the remote cloud datacenter. This significantly minimizes the communication in the network and consequently decrease the latency and response time. However, the design and management of such a distributed environment in combination with the cloud in IoT applications are still challenging.

Sensing and Actuating as a service (SAaaS) is a paradigm where several types of sensors and actuators are provided to the end users as per their desired composition upon which end users can build their application. Typically, SAaaS infrastructure is governed through cloud computing mechanism. SAaaS is mainly a complex paradigm. The system must operate under a diversified ecosystem and properly manages the observations and actuation requests. One of the most critical factors affecting QoS of IoT application is the wise deployment of the application tasks and entities in the network topology. To this end, we have proposed a layered fog architecture for sensing and actuating as a service paradigm. Observation monitoring and decision making are essential for such paradigm, hence are being taken care in close proximity of the IoT devices. Such monitoring can also be offloaded to upper layers in case fog nodes in lower layers are overloaded with processing. The proposed architecture is also equipped with fault resistance mechanism as fog nodes are well known to be potential points of failure in any fog system network. Further, the IoT sensors' observations are being filtered through all fog layers to get rid of corrupted once and avoid unnecessary bandwidth consumption. We have

formally verified the work and simulated it to evaluate the architecture and the features proposed. The evaluation conducted shows the enhancement in the metrics tested.

Acknowledgements

First and foremost, I thank Almighty Allah.

While writing is performed in isolation, it's never a solitary act. This thesis is the result of several years of continuous research and learning. It is the culmination of many years of the unending amount of wisdom, support, and advice from heroes, friends and colleagues. I am indebted in more ways than I can count to those who lent their wisdom and effort to elevate me to get the doctorate degree and finish this thesis.

I also would like to express my sincere gratitude to my family members for their true, unconditional love and continuous moral support. I will never forget my idol word's, my late father, *" You are my long term investment. If you can't raise my head, don't make it down. I am always prying for Allah to ease all difficulties in your path."* My mother's words during my stress times *" Pause research work and free your mind for a day or two, you will resume stronger. I am praying to Allah to temper stem hearts for you and grant you his and people acceptance."* And my sibling's words *" You are our ambassador, be a good sample of the family and make us proud of you."* I continue to be indebted to my wife and kids who had to bear the brunt of the stress of research and writing processes. My wife, you have my deepest love and endless gratitude. My kids, you are my life, my dream, and my goal.

I would like to extend my gratitude to my supervisor Prof. Salman Abdul Moiz who had trusted me for this PhD. position and had invested time and effort guiding me to be on the right track. I really think him for his invaluable guidance, endless patience and support, his insightful comments and suggestions, while always being humble that I never felt like a research scholar working under a professor. Prof. Salman, you were

the captain who granted me the required confidence during my entire PhD. journey, from the uncertain takeoff, to a safe and smooth landing.

My gratitude also goes to my doctoral committee members Prof. Atul Negi and Dr. R.P. Lal. for the very long discussions and their helpful suggestions. Prof. Atul, it has been an honour to work under your supervision in my MTech, and have you by my side till this stage.

I especially thank Prof. Rajeev Wankar and Prof. Hrushikesha Mohanty for the patience they have shown and the time and effort they spent in listening to my research work and for their constructive remarks. Prof. Mohanty, the amazing academic and nonacademic discussions we had will always remain in my mind; you are a unique kind of humble, friendly, and life-loving person.

I would also like to extend my gratitude to Prof. Appa Rao, Vice Chancellor, and Prof Chakravarthy Bhagvati dean of SCIS school for providing necessary facilities to carry out my research work.

Dr. Ammar Zahary -my bachelor supervisor- thank you for your continuous advice and support since then till now. You made me accept that ups and downs are common doctorate life phenomena, and cope with it. Thanks for always asking about my research progress.

A lot of beautiful memories and useful discussions with my friends will remain in my mind forever. In particular, Dr. Hassan Aldheleai, Dr. Abdo Mahyoub, Dr. Abobakr Alshamiri, Mr. Anwar Qaraah, Mr. Zaid Alhuda, Mr. Hassan Alshehari, Mr. Ahmed El-Shekeil, Dr. Fahim Baggash, Dr. Mahmoud Altarabin, Mr. Eyad Himdiat, Mr. Majid Alshwafi, Mr. Abdulbari Albarakani, Mr. Wahieb Albarakani, Mr. Essa Alduais, Mr. Hasan Salem. Mr. Mohammed Nasher, Mr. Saleem Abbas, Mr. Amhed Al-Haidari, Mr. Ahmed Alshwafi, Mr. Hamzah Alshamiri, Mr. Abdul Basit, Ms. Ayangleima, Mr. Omkarendra, Ms Melinda, Mr. Anil GR, Mr. Manoranjan GR, Mr. Abhaya Kumar, Dr. Vikas Pandey, Dr. Venkat Kagita, Dr. Venkatesh Pandiri, Mr. Rakaral Rajesh, and Mr. Raghu Ghanapuram. Thanks for many useful discussions.

I sincerely thank all my teachers for their constant support, suggestions, motivation and encouragement.

Thanks to all the great hearts around me, without you all, this would not have been existed. I hope you live life to the fullest, contribute as only you can, and be self-satisfied.

ABDULSALAM ABDO MUSAED ALI ALAMMARI

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing amount of objects connected to the internet using various preexisting technologies for communication, storage, ubiquitous and pervasive computing, wireless sensor networks etc, are now considered as the Internet of Things (IoT).

This emerging technology is defined by the interconnection of different types of smart objects "Things" provided with sensing, actuating and communication capabilities to interact with their environment and with each other to be an essential component for several ecosystem applications [75]. Hence, IoT refers to a new range of connecting devices/objects (other than the known networks of computers) watch, curtain, juicer, TV, car, kitchen appliances, health devices and much more. Figure 1.1 shows an overview of IoT .

IoT is defined in [20] as "*Interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless ubiquitous sensing, data analytics and information representation with Cloud computing as the unifying framework.*"

IoT paradigm is a result from the convergence of three different visions listed below [9], and also depicted in Figure 1.2.

- Things-oriented visions: Concentrates on outfitting objects with smartness to be

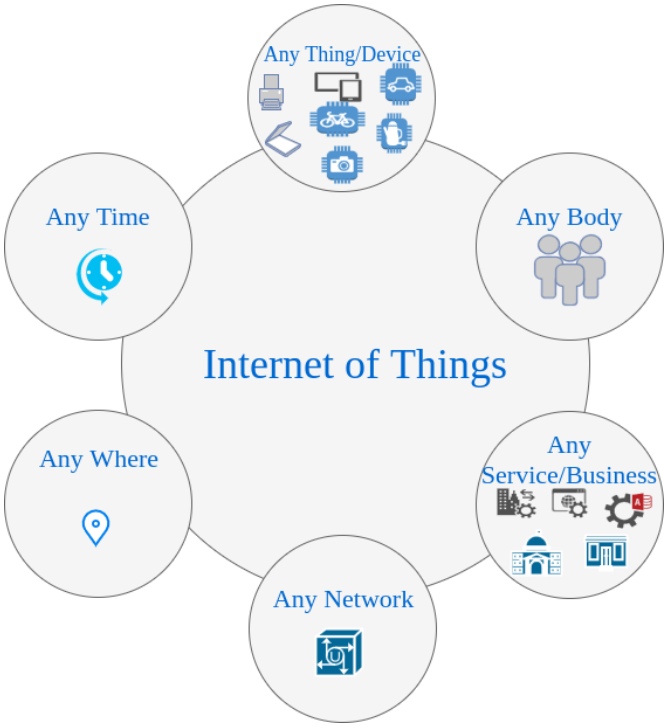

Figure 1.1: Internet of Things Overview

able, for instance, to sense their environment and actuate on it, to communicate among themselves and connect to the internet, etc. Must of which are based on microcontrollers that eventually link the real world with the digital world.

- Internet-oriented vision: Concentrates on the smart objects communication-related aspects like communication protocols and interoperability concepts through internet and web technologies.

- The unique addressing of the massive amount of connected smart objects, and the representation and storing the huge amount of data they generate turned to be the most challenging issue resulting in semantic-oriented visions or perspectives of IoT. Ontologies and Resource Description Framework (RDF) are supporting concepts to deal with such an issue. Ontologies are widely used to manage the object metadata; Web Ontology Language (OWL) is used to define such ontologies. Where RDF is for data representation.

IoT paradigm has witnessed a spectacular evolution during the last few years; it

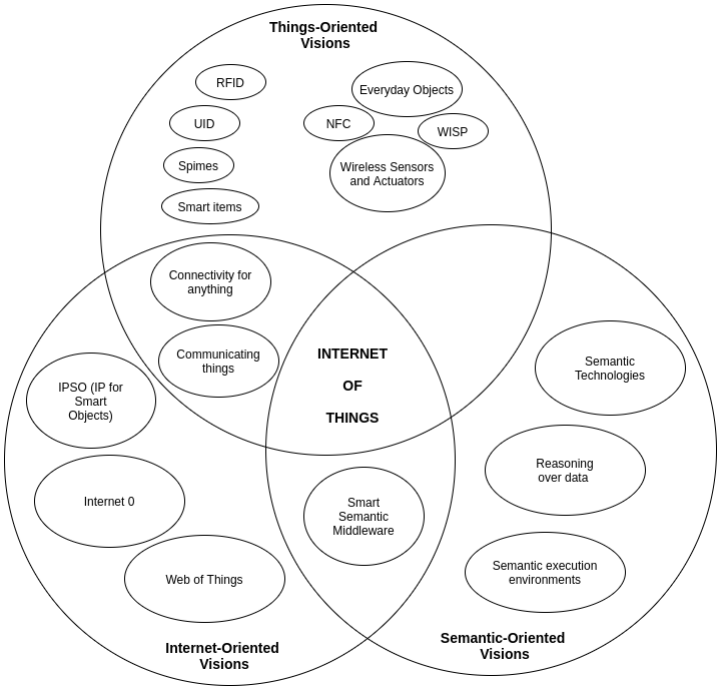

Figure 1.2: The convergence of different visions resulting in IoT Paradigm, Adapted from [9]

is also expected to open new fields of perspectives in research and industry in the next decade. As the Internet is spreading and the microchips and sensors/actuators are getting cheaper, more IoT applications come to the reality leading to more types and number of connected devices.

It was estimated that by 2020 more than 30 billion IoT devices will be connected worldwide. Hence, this massive IoT deployment will generate an enormous quantity of data as never before [65].

IoT enables such physical objects to communicate, coordinate and share their data by employing the above mentioned pre-existing technologies. This arrangement is more suitable for these physical objects and allows them to cope with their constraints and limitations to make them smart, rather than as traditional passive objects [5]. In other words, IoT is the enabler for connecting the physical devices to the digital world, making numerous physical devices to be connected to the internet and allows remote

control and heterogeneous devices collaboration.

IoT devices addressing is satisfied by IPv6, however, according to [4] the increasing of IPv6 addresses cant keep going hence the NDN (Named Data Networks) raised in 2009. IoT also requires dedicated scheduling models [53].

IoT generally uses the existing technologies and paradigms to serve or change the way services are provided to many application domains like smart home, smart grid, smart agriculture, smart forest, healthcare etc. and paves the way for new applications and business opportunities.

IoT aims at connecting people, places, data, and things anywhere anytime to create a smart world based on smart wearable/room/home/building/parking, wider to smart sensing and Actuating networks, transport, smart grid, and reaching smart cities.

IoT and pervasive computing are supposed to provide services to the user to meet their requirement with minimal or no user intervention. In some cases, the requirement of users may be fulfilled with an atomic service. However, in IoT world a lot of services and their sources (usually resource-constrained devices with digital services) are available, but one key challenge is allowing easy and dynamically composition of available distributed and heterogeneous services to collaborate and to perform an advance or complex task which may not be fulfilled using one atomic service. Later, the composite service may be used to fulfil other users requirements in different scenarios [39].

IoT devices have well-known limitations and constraints (e.g., limited storage, limited processing and communication capabilities) which are mainly being overcome with the help of cloud computing paradigm that acts as a backbone for remote processing and storage of IoT devices generated data. Cloud computing provides IoT with virtually unlimited resources [15]. Besides, sensors as essential components of IoT applications that generate a large amount of data, i.e. Big Data [56]. Big Data has its associated challenges like storage and communication that IoT constrained devices

cannot address without the great solutions provided by IoT integration with cloud computing. This integration supports storage as a Service [80], Software as a Service and Platform as a Service [81]. However, the more sensors are connected, more data is generated and sent to the cloud, leading to increasing transmission overhead, network congestion, and latency issues [35].

The cloud paradigm has been a revolutionary computing model in the last decade. It provides a highly virtualized infrastructure with powerful computing hardware and software resources in the form of services that are accessed through the internet. Cloud paradigm is characterized by rapid elasticity and flexibility, which allows the users to scale in and out easily on demand [23]. Even though most IoT applications are effectively handled by the cloud model, its centralized architecture and remote location cannot deal with time sensitive applications. Moreover, the spread of IoT devices in the coming years will expose the cloud to a high volume of data which may lead to network saturation, high workload and response time. This is because the cloud environments and network infrastructures can not resist the growing data processing load and communication required by these systems.

Fog Computing is recently proposed as a new computing paradigm to overcome the cloud challenges; it is consist of a widely distributed architecture which offers processing, storage, and networking services between the cloud and the network edge [38]. In other words, fog services can be exploited to process part of IoT and cloud tasks in a closer geographically to data sources rather than fully relying on the remote cloud datacenter. This significantly minimizes the communication in the network and consequently decrease the latency and response time.
However, the design and management of such a distributed environment in combination with the cloud is still challenging. Also, the heterogeneous data generated from different types of sensors adds another layer of complexity. Therefore, the data in the system need to be carefully designed, or unified communication language need to be developed to ensure the interoperability among different modules, services, and devices [105].

Sensing and Actuating as a service (SAaaS) is a paradigm where several types of sensors and actuators (Ss/As) -stand-alone or in wireless sensor networks- are provided to the end users as per their desired composition upon which end users can build their application. Typically, SAaaS infrastructure is governed through cloud computing mechanism; devices owners offer the functionality of their devices to the service provider either free of cost or on rent basis. While the cloud acts as an intermediate interface which delivers the systems resources as services for the registered users as agreed in the purchased offer. Thereby, the end users will be charged on the bases of a pay-as-you-go pricing model which permits them to scale the systems resources easily.

SAaaS is mainly a complex paradigm. Besides that, the system must be able to operate under diversified ecosystems that properly manages the observations and actuation requests. Also, the infrastructure needs to be highly virtualized to permit the exploitation of the same physical layers components by multiple users simultaneously, which can reduce the cost of the service [59].

1.1 Motivation

The contributions of this thesis are generally motivated by the state of the art development in service composition and features provided by cloud and fog computing: pay-as-you-use, federation, mobility, security, scalability, interoperability, etc.

Integrating cloud and fog computing has led to a breakthrough revolution in the way IoT applications are thought off and designed, the way devices connect to the internet and interact with it. This make it possible almost for any kind of devices to interact with the environment and modify their operational manner and behaviour, capture heterogeneous and complex information. The same has motivated to suggest architectures for collaboration among several service providers and for sensing and actuation as a service, as well as to develop a view of the future smart city.

1.2 Research Objectives

The research objectives are:

- To exploit the advancements of the federation, accounting, and composition concepts and suggest a new view of future IoT service providers suitable for the foreseen smart city visions.
- To enhance the present SAaaS architectures design by empowering it with robust layered fog able to cope with the sensitivity of IoT applications requirements.
- To suggest better deployment of SAaaS tasks in the network topology aiming to reduce the observation and actuation latency.

1.3 Contributions of Thesis

The thesis main contributions are listed below.

- Towards future smart city vision, an architecture is proposed which composes/aggregates different types of services that may belong to different ownerships, architectures, or application domains. This smart city vision will eventually make cloud service provider be as an integration of several services providers which encourages fast and economically inexpensive innovative IoT applications.
- As we believe that one of the most critical factors affecting QoS of IoT application is a good deployment of the application tasks and entities in the network topology, a layered fog architecture is proposed for sensing and actuating as a service paradigm. Observation monitoring and decision making are essential for such paradigm, hence are being taken care in close proximity of the IoT devices. Also, the architecture is equipped with fault resistance mechanism as fog nodes are well known to be potential points of failure in any fog system network.
- The proposed layered fog architecture for sensing and actuating as a service is extended, with enhanced features like observation monitoring offloading and a new observation filtering scheme that takes care of corrupted observations throughout the entire network topology. The work is simulated to evaluate the

architecture and the features proposed. The evaluation shows enhancement in the metrics tested.

1.4 Structure of the Thesis

Following is the organization of the thesis. Chapter 2 discusses IoT main building blocks, fundamental existing concepts and techniques for IoT and both cloud and fog computing. It also lists several evaluation metrics and parameters as well as the primary requirements for successful IoT ecosystems, cloud, and fog computing along with related work in such areas. Chapter 3 presents the vision of future smart city service providers as well as the proposed architecture for the same. In chapter 4, we present the theoretical details of layered fog for sensing and actuating as a service proposed along with a feature-wise comparison with some core works in the same area. The extension of layered fog proposal, details of entire SALFSD components and communication among them, details of added contributions, and formal verification are presented in chapter 5. This is followed by the experiment analysis and evaluation of the architecture in chapter 6. Finally, chapter 7 concludes the thesis contributions and highlights the future works.

1.5 Publications

1. Abdulsalam Alammari, Salman Abdul Moiz, Atul Negi; Internet of Things Sensors and Actuators Layered Fog Service Delivery Model SALFSD, Multi-disciplinary Trends in Artificial Intelligence, 13th International Conference, MIWAI 2019, Kuala Lumpur, Malaysia, November 1719, 2019, pp 15-25. (SCOPUS Indexed)

2. Abdulsalam Alammari, Salman Abdul Moiz, Atul Negi: TOWARDS TRULY SMART CITY SERVICE PROVIDERS: A VIEW ON ON-DEMAND EVERYTHING AS A SERVICES, Journal of Critical Reviews, Vol 7, Issue 7, 2020, pp 428-436. (SCOPUS Indexed)

3. Abdulsalam Alammari, Salman Abdul Moiz, Atul Negi: Enhanced Layered Fog Architecture for IoT Sensing and Actuation as a Service. Communicated to Journal of Wireless Networks.

Chapter 2

Fundamental Concepts and Literature Review

This chapter begins by briefly going through foundation concepts and several pre-existing technologies and paradigms on which sensing services and the IoT is built upon and depends on. These technologies and paradigms are used for communication, architecture, storage, sensing, arrangement and more. Existing literature that uses stand-alone sensors/actuators or mobile phone integrated sensors/actuators for some sensing and actuating applications with and without the IoT are listed. These are followed by showing the most famous and widely adopted integrations that is used to support and cover IoT limitation and increase its acceptance for building IoT-based applications; i.e. cloud computing and fog computing.

2.1 Fundamental Concepts

2.1.1 IoT Building Blocks

These are the main elements required by any IoT-based application to achieve its functionality.

- Identification: There are billions of IoT devices and services; hence IoT needs to identify them uniquely. There is a difference between object ID (that refer

to the object name or the services name like wind speed sensor or D1) and its unique address in the communication network [5]. Radio Frequency Identification (RFID) is one technology used for identification, microships attached to any object to support its automatic identification [35]. IoT devices addressing is satisfied by IPv6, however, according to aggarwal et. al. [4] the increasing of IPv6 addresses can not keep going, hence the Named Data Networks (NDN) raised in 2009.

- Sensing: Sensing means reading the environment data (observations) using the sensors devices and upload it for analysing or storage [5].
- Communication: There are several heterogeneous and resource constrained IoT devices in which they require low power for operation hence require light weight communication protocols that support such needs. Some of the communication protocols used in IoT are, Bluetooth, WiFi, IEEE802.15.4, LTE-Advanced, Z-wave, RFDI (Radio Frequency Identifier), Ultra-Wide Bandwidth (UWB) and Near Field Communication (NFC) [5].
- Computation: IoT applications require processing units (hardware and software) to run and provide the computational mean. Arduino, Raspberry PI, WiSense, Cubieboard, Z1, BeagleBone, Intel Galileo, FriendlyARM are some examples of hardware platforms commonly used for running IoT applications. Real-Time Operating Systems (RTOS) like Contiki RTOS with its simulator -Cooja- is used to devolve real-time IoT-based applications. TinyOS, LiteOS and Riot OS are also lightweight operating systems designed to suit for IoT environments [5].
- IoT Services: Services that enable interaction with the physical world. IoT Services can be stand-alone as a field on their own. Compared to Internet services, which run on powerful and rich resources (power and computation ability) computers, IoT services are deployed and run on resource-constrained devices [94].
- Semantics: Refers to the ability of IoT heterogeneous devices to extract knowledge smartly out of the sensed raw data. This takes great support of Semantic Web technologies like Resource Description Framework (RDF) and the Web Ontology Language (OWL) that help in describing and representing information in machine-understandable ways [5].

2.1.2 Fundamental Technologies and Paradigms

Some foundational technologies and paradigms shaped IoT and sensing applications are listed below:

- Cloud Computing: It is the provision of providing several types of services and resources through the internet. These services are like platform, software, computation, storage. It has become a common choice for individuals and enterprises as it saves development costs and increases security and performance [67].
- Infrastructure as a Service (IaaS): Is one form of cloud computing where infrastructure components like servers, networking hardware, storage are being hosted at the cloud of the service provider and made accessible to user over the internet [67].
- Platform as a service (PaaS): Is the form of cloud computing where the service provider offers hardware and software tools and make them accessible over the internet [67].
- Software as a Service (SaaS): Is the provision that allows users (individuals or enterprises) to connect to and use cloud-based applications over the Internet. In SaaS applications are hosted at the third party service provider and made accessible for user through the internet [67].
- Fog Computing: It is the paradigm that brings some of the cloud capabilities (computing, storage, and applications) closer to the data sources. The location of fog nodes is not necessarily fixed; they maybe anywhere between the data sources and cloud [64].
- Sensing as a Service (S^2aaS): Service providers introduce observations of particular sensors to end-users as a service either free of cost or on rent basis. The sensors may belong to the service provides themselves or being rented from device owners [78, 89].
- Sensing and Actuating as a Service (SAaaS): It is quite similar to SaaS adding to it that providers also offer applying forces to particular actuators as per end-users requirements [26].

2.2 Literature Review

In this section, we list the research issues that were mainly studied and hence fundamentally led to this thesis contributions. Issues related to the composition of IoT services with mobile sensors are listed. Followed by the issues related to the composition of sensors and actuators. The main focus is to show the types of integrations and compositions of several technologies and paradigms that resulted in a new view of sensing and actuating applications and architectural views. To name few, mobile phones, cloud computing, and fog computing.

Mobile phones are widespread, low in cost, integrated with many sensors like GPS, light, microphone, accelerometer, motion, camera etc. and able to connect to external sensors. The same made mobile phones good or first choice to serve as interfaces for many types of applications and provide them with the ability to improve the performance of IoT- based applications. This is also because such devices are also equipped with many communication models (Cellular, Bluetooth, NFC, and WiFi) through which they communicate with sensors and among themselves. Though, the mobility of mobile phone sensors results in new challenges in designing and managing sensing applications [76].

The domain area of mobile computing and IoT was defined Kamilaris et. al. [46] as the research area that involves case studies, prototypes, demonstrations, applications and business cases of the IoT/Web of Things, through mobile phones. In such domain, the user of the mobile phone interacts with cyber-physical things that are enabled to the Internet/Web through their phone device, exploiting at the same time the sensing capabilities of the phone.

2.2.1 Participatory and Opportunistic sensing

Two types of smart phone sensing were identified in [1, 55, 89]; Participatory sensing and Opportunistic sensing. In participatory sensing users are involved in the tasks of sensing (where, what and when to sense are determined manually by participants) and uploading the observation whereas opportunistic sensing it happens automatically

without the intervention of users, which was first intreduced by Goldman et. al. [29]. However, [35, 47] favoured smart opportunistic over participatory sensing for IoT applications.

For continuous roaming inside and outside the hospital, two models of data transmission to the server are used for location-aware patient monitoring system [19]. The first model uses the LAN internet if the body is within the LAN coverage. Otherwise, the CDMA network is used to transmit data to the server.

Eco-feedback sensing systems was developed by Jacucci et. al. [44] with mobile phone interface to enable monitoring the power consumption, and with the use of a mobile cameras (pointing it to the device) by Jahn et. al. [45].

Smart parking system, Park Here [83], take benefit of mobile devices sensors and short-range communication to find and spread the availability of parking spots. Park Here uses two modes of parking spot data spreading. Local, device-to-device, using WiFi Direct to disseminate data around the user locality. Global, to a remote server, via a 3G/4G network. The availability of spot is identified without geo-locating it as the system does not use GPS. ParkNet [66] used vehicles GPS to identify the parking spaces. While [54, 61, 71, 100] used smart phone to automatically identify whether the parking spot has been vacated.

AndWellnes system (personal data collection system) [40] uses mobile phones to collects data form participants to conduct surveys and display the participant's statistics in real-time for both participants as well as researchers. The participants are prompted in their mobile at a configured time intervals to answer surveys (time-based, location-based, or other contextual parameters) where the answers are sent wirelessly to the server. Besides the configured intervals, the participant's mobile sensors can be used to upload continuous data with user location tracking using GPS. Hence, AndWellnes provides participatory sensing as well as smart sensing

SenSay (a single sensor-based approach) [49] was developed to identify the particular mobile device context using only the embedded sensors in that device. It realises

the change of context of the mobile user (standing, sitting, walking or running). In SenSay, there is no cooperation between sensors that belongs to different handsets. Alternatively to the use of a single sensor, multiple simple sensors in the handset are integrated to identify the context cooperatively [31, 43].

In SPA [87] -mobile-based system- a collaboration between body area sensor network and environment sensors is suggested to continuously sense and upload the gathered data to a remote server to be analysed by professional for continuous health monitoring.

CONSORTS-S [84] designed a platform to provide mobile users with context-aware services by accessing wireless sensor networks in the surrounding. CONSORTS-S architecture consists of three main parts; 1) The mobile phone to be used as an interface. 2) For overcoming the mobile phone limitations, CONSORTS-S platform has attached mobile sensor router to the mobile device to help in the communication with the surrounding wireless sensors. 3) Middleware located in the remote server for management and sensed data analysis.

2.2.2 Urban Sensing

Urban sensing is the use of digital environmental sensors in urban landscapes like buildings, vehicles, healthcare etc. It was addressed in several studies. Some of such works were designed for specific applications. CodeBlue [63] introduced a wireless infrastructure for emergency medical care application. CodeBlue integrates low-power, wireless vital sign sensors, Personal Digital Assistants, and PC-class systems with the ability to scale to a network of dense devices.

A set of guidelines to establish ideal architecture and platform for industrial applications were established Kumar et. al. [52]. The work also developed a general architecture for industrial application to monitor and predict equipment failure. Wisden [101] designed a wireless sensor network system for acquisition of structural data from different locations for identifying damages in surfaces where they are deployed.

The system Identified the damages -if any- and there locations by monitoring changes in sensed accelerometer parameters.

Similarly, UrbanRadar [47] -location-based application- an application which discovers and communicates with environmental services; provided by Web-enabled urban sensors. In UrbanRadar, user can create urban mashups, which are satisfied only when several predefined conditions (from the mashup services) are met. As an example, a location-aware environment monitoring approach introduced in UrbanRadar to make users updated with the current real-time environmental conditions. Their novelty is that they reduce the direct citizen involvement by reading the data through mobile devices and share them over the web as a contribution towards real-time digital city vision supporting urban sensing. Their developed mobile application interacts with web-enabled sensors deployed in the environment around the user.

Finally, a general purpose framework, AnonySense [21], to allow participatory sensing to answer the application's query about the context. It leverages sensors on participants mobile phones taking the privacy of participants into account.

2.2.3 People-Centric Urban Sensing

It is a landscape of urban sensing where the importance is given to people's attributes, surroundings, and how they interact with their surroundings. People-CentricUrbanSensing [17] was an attempt to encourage the movement of sensors network from application-specific - mostly small scale- to the scalable and general purpose network for commercial mainstream which can support a variety of urban application (buildings, cities, enterprises). Moreover, People-CentricUrbanSensing is designed to extend the general purpose infrastructure to "new sensing edge for the internet." People-CentricUrbanSensing architecture favours the opportunistic sensing-based approach to support its scalability. Several issues like (the coordination among static sensors, people-centric mobile sensors and edge wireless access nodes) that aid opportunistic sensing, tasking and data collection are identified. Besides, challenges like wide coverage of sensing areas with numerous mobile sensors, responsibility hand of, sensors coordination, network

performance, privacy and security were also discussed in this work for the support of the new view.

MetroSense [16] is also based on people-centric sensing approach that follows three main stages (sense, learn and share). MetroSense sensing is based on three sources of sensing; mobile phones sensors, fixed sensors in civic infrastructure, and edge wireless access node which provides the internet gateway.

Participatory sensing for large scale sensing is the evolution as mobile crowd sensing using internet-enabled smart phones . It is an enabler for a broader range of urban applications like traffic planning, environment monitoring and public safety [48, 60].

2.2.4 Integrating with the Cloud

IoT is mainly characterized by numerous real word things with the following main limitations; low energy, limited storage and processing, and short-range communications [79, 82, 95]. Cloud infrastructures are ubiquitous and provide virtually unlimited, inexpensive and easily accessible (from anywhere) resources [15, 92]. Integration of the cloud and IoT is a solution to overcome IoT limitations leveraging the cloud performance and always available and scalable cloud resources [58, 62, 90]. Mobile phone sensing also has taken plenty of benefits when integrated with the cloud.

Besides, Taivalsaari et. al. [93] foresees that by the time IoT platforms reach maturity, the following topics (mostly depend on cloud computing) will take the forefront in such platforms: 1) Things management, for devices registration and virtualization. 2) Devices remote programming. 3) System-level programming and orchestration. 4) Edge clouds, local connectivity, and fog computing. 5) Security.

Using smartphones as a source of sensing services was first introduced and termed as Sensing as a Service (S^2aaS) by Sheng et. al. [89], and further elaborated and detailed by Sheng et. al. [88]. S^2aaS concept strongly depends on cloud computing as an essential computing model with many services offered which can be leveraged by

this concept. The work also details some fundamental requirements for any sensing as a service via smart phones:

1. Ability to support different mobile platforms serving several types of mobile phones sensing applications.
2. Having in mind that such devices are power constraint, it is also required that the system is power efficient.
3. Support for efficient interconnection mechanisms. Sensing request can be handled from different locations with the help of the cloud and the deployment of sensing server in different locations.

Besides, Sheng et. al. [89] stated the following functionalities to be supported by S^2aaS Cloud. Web interface to be provided for users accesses both from mobile devices and normal computers:

1. The ability to generate new sensing task (stander format including what data to be collected, sensors to be used and area of interest) for each new data collection request.
2. The ability to recruit participants to participate in sensing for each new request.
3. The ability to schedule the sensing activities between mobile participants.
4. The ability to deploy sensors managing applications on the participant mobiles such that the data sensing and sending them to the server are managed and under control.
5. Finally, the ability to save data for future use.

Also, Perera et.al. [78] explored the concept of sensing as a service model and how it can fit into IoT.

Spheres [33]: A web services framework for smartphone sensing as a service. Spheres introduced (SOA) services oriented architecture based framework integrating

mobile internet, wireless sensor network and cloud to provide smartphone crowdsensing to be shared for both public and private use. Spheres is based on [34] with the main difference is that in Spheres mobile phone sensors are not in fixed location. In [34] wireless sensor nodes were connected to a gateway and formed a wireless sensor network.

In S^2aaS [77], the word sensors refers to physical sensors as well as virtual sensors. In S^2aaS, any sources that generate data, social media accounts and weather APIs, for example, are considered as virtual sensors.

Mobile phones, as well as standalone sensors/actuators, are being used to actuate and control physical devices in several domains, and a number of models and architectures have been suggested in the literature for the same either for specific or general domain applications.

Internet technology along with IoT are foreseen to be the future standards, particularly for home automation and control [32, 51], providing interoperability between heterogeneous types of devices. Applications for monitoring and controlling home appliances were introduced in [10, 99]. A general mobile phone application that provides the ability of creating physical mashups of composition of different smart things was introduced by Guinard et.al. [36].

Sensing Cloud Infrastructure (SCI) was introduced by Yuriyama et. al. [106] to provide virtualized (on the cloud) set of physical sensors required by the user on demand. User has control and monitor over the virtual set of sensors on Information Technology infrastructure and out to destroy them once they are not in need. SCI contains five servers, mainly; portal server, service provider server, resource management server, monitoring server and virtual server.

SCI was extended to the concept of IoT by Madria et. al. [62] which comprises of three layers: client-centric, middleware, and sensor-centric.
SCI was optimized in SenseCloud [50], a cloud based solution with unified access to different sensor networks and a level of abstraction that hides the underlying com-

plexity. SenseCloud proposed a system which exploits the virtualization technology to manage and share the same IoT infrastructure between different users and allow dynamic provisioning; the strategy consists of creating virtual sensors for the users, which corresponds to the requested physical sensors. The management of sensing data is handled by the service provider through the cloud and shared with users either in real-time or historically. Moreover, the proposed system is provided with security, load balancing, multitenancy, failover, high availability and reliability mechanisms.

Abdelwahab et. al. cldassiremsensCARS2014 surveyed the use of cloud computing as to provide remote sensing services (Cloud-assisted remote sensing, CARS for short) as an enabler of Internet of Everything(IoE); hence, they believe it leads to prompting smart cloud services. Through real-world application, researchers explained the benefits of using cloud-based services to empower remote sensing. Besides, a four-layered architecture (CARS) is proposed in this work: 1) fog layer; 2) stratus layer; 3) alto-cumulus layer; and 4) cirrus layer.
CARS mainly provides three service models: Sensing and Actuating Infrastructure as a Service (SAIaaS), Sensing and Actuating Platform as a Service (SAPaaS), and Sensing Data and Analytics as a Service (SDAaaS), which are alike to cloud computing service models: 1) IaaS; 2) PaaS; and 3) SaaS. It worth mentioning that, in CARS architecture, sensing and actuating resources are encapsulated in the fog layer. Several essential design components for Cloud-assisted remote sensing were described in this work.

Cloud of Things (CoT) [3] introduced cloud architecture for sensing as a service. Authors of CoT hypothesis that taking benefit of IoT devices global sensing resources in cloud platform to support remote sensing is an efficacious way to achieve their sensing as a service foresight in their previous work, CARS [2].
In contrast to the traditional cloud platforms that collect data for later use, Cloud of Things processes the data in-network by IoT devices to directly provide meaningful information. Three main elements form CoT architecture; 1) IoT devices; sensors devices with the ability to serve specific as well as general purpose remote sensing applications. 2) First tier clouds; traditional cloud platforms that provide unified user access interface. 3) Cloud agents; resource reach, trusted, and well connected to the

traditional cloud platforms and to the internet located near the network edge. Cloud agents are basically edge nodes used for discovery, virtualization, and forming a cloud network of IoT devices.

Sensing and Actuating as a Service (SAaaS) provides various types of sensors and/or actuators (S/A) resources from sensor networks and personal mobile devices to the end user as services. The end user can then build their own application based on rented composed/aggregated types of resources of their desired locations, functionalities and time. SAaaS also provides the end user with the ability to trigger actuating commands as per their analysis of the scenarios/real/historical data etc. The idea of SAaaS was encouraged by the following factors that made robust base for SAaaS: 1) The globally distributed and wide spread of heterogeneous smarts devices. 2) Smart devices, sensors and actuators that can be categorised location wise or by their type/functionality. and, 3) The spread of Sensing/Infrastructure/Storage as a service.

In SAaaS owners (being individuals, enterprise, universities, etc.) of sensing/actuating devices may contribute their devices services for free or provides the same on rent bases. Device owners may contribute standalone Sensors/Actuators (S/A) devices or WSN in which they are called contributing nodes which (whether have single or group of sensors or actuators) must belong to the same administrative domain [26]. End users (the once will take the benefits of the sensors and actuators) might also be individuals, enterprises, universities, etc.

The SAaaS provider acts as the mediator between the two parties and manages all the aspects of the paradigm. Among SAaaS provider responsibilities are device owners and end user enrolment, Service Level Agreement creation, physical nodes selection, rent calculation, virtualizatin, etc. SAaaS virtalization has two types. First is the virtalization of the observation data in witch the provider receives the sensors observations and saves them to provide each end user with the required data at their prespecified time. Second is the actuator vitrualization where the provider facilitates the end user access (sending an actuator command) to the previously rented actuator. This concludes that the provider must create a virtual set of desired sensors and actuator for

each end used as per their requirement.

The main concepts and players of SAaaS were identified by Distefano et. al. [26]. The work also introduced an architecture for SAaaS in which they make a Hyervisor in the nodes to abstract the creation, management and virtualization of sensors and actuators as a virtual instances in the cloud. Even though the Hypervisor works at the device level, it enables direct communication with sensing and actuating devices through the Automatic Enforcer which is located between the VolunteerCloud Manager and the Hypervisor.

Distefano et. al. [28] used their hypervisor propesed in [26] to introduce more detailed architecture for SAaaS, giving more concentration for management, abstraction and virtualization of sensing resources. Their proposed architecture allows devices to provide their sensors as a virtual instance which makes a particular physical device handles concurrent requests. Besides, the architecture can compose/aggregate a network of resource instances out of simple individual instances. It is worth mentioning that, since this architecture is based on the hypeovisor, it is a device-oriented approach but still concurrent requests of the same physical device are ensured.

A further extension to their works in [26] and [28], authors proposed utility framework [27] for IoT SAaaS approach inside IoT-A reference architecture [72]. This work implemented the earlier proposed idea (implementation for Android mobile) in a real life IoT scenario to show its feasibility.

Stack4Things [59] adopted the OpenStack [73] (Infrastructure as a Service framework) to propose a framework for Sensing and Actuating as a Service (device-centric approach). Stack4Things shows the detailed subsystems for resources management and their observation data, and some use cases were demonstrated.

Sensing and Actuation as a Service Delivery Model (SAaaSDM) [85] is a novel model of system management and computing for SAaaS proposed based on cloud edge centric IoT model. The purpose of SAaaSDM to overcome the issues of centralized cloud architecture while taking advantage of computing and storing capacities

of both cloud and edge computing. Therefore, the model aims at providing robust management for the existing IoT sensors and actuators in Cloud Edge-Centric fashion and process their data in the edge node (cloud gateway). Compared with other works, SAaaSDM brings many benefits to sensing and actuation applications. In addition to exploiting virtualization and enabling collaboration between various type of IoT devices, SAaaSDM enhances the performance of the paradigm by decreasing both bandwidth usage and processing response time as well as improving the network lifespan.

The issue of heterogeneity and interoperability of IoT devices were tackled by Yun et. al. [105]. Authors designed an IoT platform (nCube) based on oneM2M standard, which unifies the communication language among IoT devices. To achieve such unified communication, they developed middleware programs (thing adaptation software TAS) which enable the translation of sensing value and actuation commands into a defined oneM2M resources accessible and manageable through a web application.

2.2.5 Integrating the Cloud with Fog Computing and IoT

Cloud computing has led to a breakthrough revolution in the way applications are thought off and designed, the way devices connect to the internet and interact with it. The same makes it possible almost for any kind of devices to interact with the environment and modify their operational manner and behaviour, capture heterogeneous and complex information etc.

Such revolution has been a milestone and enabler in the emerging of IoT paradigm and the development of several systems that continuously monitor and interconnect with the environment and take action automatically in response to certain conditions or as per users' command. Notwithstanding the indispensable benefits offered by the cloud, it is still subject to many challenges, including increased latency of both data and services, network congestion, and the increased bandwidth and communication concerns. This is because the cloud environments and network infrastructures can not

resist the growing data processing load and communication required by these systems.

Mostly, IoT applications have rigorous requirements. Plenty of them requires, to some extent, real time responsiveness, while the privacy and security, the quality of service, and the location awareness of the response are achieved. For such requirements, the (cloud-IoT architecture) can barely support the processing and communication of such a massive amount of devices and data. Besides, if cloud-IoT devices architecture could support such applications, this will be with limited response time, scalability, and quality of service [14].

In highly dynamic and real-time scenarios, data rapidly and continuously change, and the reciprocated data between IoT and the cloud might not reflect the real time values or be accurate; the reason is the high latency throughout interactions [12].

Several architectures and solutions are proposed attempting to support the requirements of IoT applications or/and processing such amount of data. New architectures that enable integrating the cloud with any type of lightweight sensors are described in [98, 107]. These architectures tried to enable the integration by getting over the inherited cloud issues: latency, the ability to support periodic events, handling continuous sensing, the lack of elasticity when data of numerous sensors is transmitted simultaneously.

The problems stemmed from continuous sensing was addressed by Lane et. al. [55] which propose that devices should collect data and upload them to the cloud. This occurs only sporadically. This delay-tolerant model of sensor sampling and processing critically bounds applications effectiveness and the system ability to be aware of its real time context, adapt and react to timely situations.

Therefore, some questions like, *what to process*, and *where to process*, or *where to take a decision based on realtime data streaming* are resolved in a more recent paradigm called Fog computing or mini-cloud. Fog computing is mainly about bringing the processing closer to the data sources. However, what processing is to be pulled

out from the cloud, and where to be placed in the network topology are the main important issues in defining exactly a fog node and its location [64].

OpenFog Consortium defined fog computing as *"A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum."* Hence providing the missing link in the cloud-to-things continuum. The consortium has also defined eight common pillars of OpenFog reference architecture [74]: security, scalability, openness, autonomy, RAS (reliability, availability and serviceability), agility, hierarchy, and programmability. Defining standards to ease interoperability and implementation of IoT applications is one of the main objective of OpenFog Computing consortium.

Fog computing is an extension of the traditional cloud computing where architecture implementations may be established in several layers in the topology of the network. Though, fog and cloud work together, and fog extensions must preserve the cloud benefits: virtualization, containerization, manageability, orchestration and efficiency. In other words, fog works as an interface between IoT and the cloud; supporting them to communicate. Consequently, fog gathers the best features of each technology, expanding the application range of cloud computing and growing the resource availability in IoT [12, 74].

While the cloud provides global centralization, fog provides localization hence reducing the latency and provides context-awareness. Both; the cloud centralization and fog localization may be required for many types of applications [14]. And both computing paradigms have their associated pros and cons hence can complete each other, particularly for supporting IoT applications. Fog provides support for fast response, mobility, and processing, while the cloud offers processing and storage for necessary tasks. In short, an effective IoT management system needs both cloud and fog support. The important factors are what services are to be deployed on the cloud and on the fog, mostly depending on the IoT application features and characteristics. It is even likely to replicate some services on both the fog and the cloud [8].

Also, Bonomi et. al. [13] discussed how fog could be a useful paradigm to support the cloud in terms of latency, real-time processing, large scale and location awareness, particularly for IoT applications. The discussion has also pointed out the aforementioned benefits for Connected Vehicles, Smart Grid, and Wireless Sensors and Actuators Networks applications with the IoT. Besides, the work listed out some applications were fog better fits than the cloud paradigm; applications require very low latency, geo-distributed applications, fast mobile applications, and large scale distributed control systems.

IoT service delay can also be minimized by service offloading between fog nodes or from fog nodes to the cloud. A similar concept is workload or tasks assignment to either fog nodes or to the cloud that is considered a static optimization problem associated with high complexity [6, 37]. Though, they did not take offloading from fog to cloud into consideration. However, fog services and tasks offloading has less complexity and more flexibility: the number of network layers and fog node, their organization, etc. do not have any restrictions on offloading mechanism [104].

Several works like [13, 91, 103] argues that due to its proximity, its mobility support and dense geographic coverage, for platforms can be utilized to operate IoT applications and services from the network edge and end devices (access points, road side units, set top boxs, Machine to machine (M2M) gateways.) Such utilization reduces the latency, improves QoS, and allows real time data analysis with actuation leading to consumer centric IoT products [22]. Besides, Botta et. al. [15] explains that the real time association among IoT tools gets the support of fog and cloud computing which consequently minimizes latency in data processing and analysis.

Satyanarayanan [86] highlighted that proximity helps in: 1) Masking cloud outages; due to cloud failure, network failure, or any attack like a denial-of-service the cloud services may become unavailable, in such case a fallback serves in the nearest fog may mask the failure untill the cloud services are restored to normal status. 2) Security and privacy enforcement [74, 86]; being the first point of contact in the system infrastructure for the sensed data, a fog node can enforce the security and privacy

policies prior to forwarding data to the cloud.

Tseng et. al. [96] worked in migrating oneM2M (a global IoT/M2M) to extend its high scalability from the cloud in their previous works [18, 24] to the fog. Investigations on scalability of IoT/M2M in the cloud are conducted in [18, 24]. According to the IoT traffic load, the resources in the cloud can scale up or down based on such investigations. Nevertheless, managing such traffic only in the cloud does not ultimately solve the scalability issues; the traffic must be dealt with before reaching the cloud. In other words, fog networks between the things and the cloud must offer the scalability by; 1) dynamically and flexibly scaling the serving instances in/out over fog nodes to harmonise with the incoming IoT traffic, and 2) devices may join or leave the network as needed.

Scalability of fog systems in IoT domains is considered among the essential evaluation criteria in several surveys and works like [68, 70, 74] and also one of the eight pillars defined by OpenFog reference architecture [74]. This is due to the fact that fog systems are foreseen to be connected to millions of users and IoT devices and may include a large number of applications, fog nodes and domains. Consequently, this will raise the amount of data to be generated, gathered, and processed.

Besides, Geodistribution and Big Data are fundamentally related characteristics; more dense and wide-range the system is, more data to be generated [97]. Therefore, Bellavista et. al. [12] highlighted the system characteristics to scale relatively with the quantity of the managed information. Geo-distribution scalability is an underlined property required of fog computing system so they can manage the distributed, even highly distributed, applications and devices, as compared to the typical centralized cloud. In highly distributed systems, fog must be able to efficiently manage a large number of nodes widespread in geographic areas, plus different degrees of density. Different distribution configuration and types of topologies are expected to be handled by fog with the ability to scale and adapt to the IoT applications and their rigorous and diverse requirements.

Internet of Thing applications are supposed to know their location and the external context at the place where they are deployed. Such requirement is considered as a core requirement in some standards and works, like geographical distribution requirement in [42] and the hierarchy pillar in OpenFog [74].
In fog for IoT solutions, location-awareness will produce systems with a greater consciousness degree, empower IoT applications, consequently, a greater degree of resilience to the outside world. Location-awareness helps in the knowledge of the sensed environment enhancement towards better system adaptability, response accuracy and, hence, an improvement on the system execution and higher application quality. The system is accurate as it knows the environment it works at and, hence, its responses are correct, precise, and applicable as compared with systems without information of their environment and location [12].

Mobility is also highlighted as a core requirement for fog computing in several research works [11, 42, 70]. With respect to fog computing for IoT and its applications, most of such applications directly relate to mobility. Mobile Internet of Things (MIoT) extends the IoT concept with ubiquitous coverage and mobile support [12].
The growing number of mobile devices, their ubiquity, and the dominant role of wireless access elevate the need to introduce mobility support in fog computing. Therefore, for fog nodes to be efficient, they must be able to adapt themselves to handle devices with high mobility, even fog nodes may be mobile as well. Particularly in data-rich mobility applications, the ability to locate the correct data in the fog achieves better data models and local cashing as well as better overall performance. Besides, reliable handoff mechanisms are required in fog computing to support the possibility of mobile devices shifting among fog nodes authorities without interrupting the system operations [12].

IoT applications and their deplyment in real life secinaros highly require real-time responsiveness as a main enabler concept [42, 70].
Fog Computing is important to accomplish the low-latency requirement as direct Cloud-Edge interactions will not be satisfied for several reasons [12]:

1. It improves the temporal accuracy as the sensed data are processed and acted upon (actuation command or decisions making) at real time. Consequently, it continuously uses data reflecting the instant situation correctly.

2. Fog surpass distance concerns, reducing the number of network hops by moving computation near the data sources.

3. Sensors generate a tremendous amount of data, if the entire data is sent to the cloud, this may cause a network to slow down and, consequently, may knock-on effect throughout the whole system and leads to its slow down.

Fog should offer real time execution for types of tasks that do not require high resource consumption or long analysis. Also, some systems, the data must be pre-processed in the fog nodes before being uploaded to the cloud, consequently decreasing the load on the core network [12].

For real world IoT applications, data quality is an important and related requirement as they are not only made for sensing but also altering the scenarios; making, usually irreversible, modification to the physical world [12]. Giving more focus to fog for IoT, Bellavista et. al. [12] consider data quality as a fundamental requirement to permit the integration and use of heterogeneous IoT sensors. Also authors of [12] claim that general purpose fog computing surveys and standards do not express such requirement, though, in [42], it was explained as an element of heterogeneity management. Authors of [12] also believe that the more the system quality increased, more improvement throughout computation and actuation stages, hence leading to better quality for the entire system. Data quality stands on the meeting of several techniques, such as: data filtering, data aggregation, data normalization, etc. The real time proactive maintenance and anomaly detection is achieved by the combination of data filtering, data aggregation, data normalization and data analytics. Faulty data quality is among the most serious problems in ubiquitous environments as it is hard to be discovered and concerns both system reliability and performance. Therefore, fog nodes are supposed to assist data quality and drop the worthless data as close as possible to its source to reduce the amount of data to be processed or forwarded to the cloud [12].

Interoperability is also amongst the central issues in fog computing agreed upon in several related surveys and standards [42, 68, 74]. As of fog computing for IoT, interoperability issue becomes more serious as IoT itself is an extremely heterogeneous environment that works in real world scenarios, built upon a wide range of diverse devices that collect heterogeneous information from the environment [12]. Fog nodes are also heterogeneous and range from end devices like sensors, mobile phones, vehicles, etc. as well as access points, set-top boxes, edge routers, high-end servers. Furthermore, often, fog computing services and applications must be federated, be flexible to span with different levels in fog hierarchy as they need the cooperation of several providers and also to enable multi-vendor ecosystems [14, 74]. Besides, any data generated or collected by any particular fog node should be shareable with remaining fog nodes in the system hierarchy [74]. Without expensive computations on a cloud layer, fog environments make a proper position to: 1) Fulfil interoperability enabling processes. 2) Originate a unique data stream. 3) Present generic Application Programming Interfaces (APIs) to be used by diverse applications or an initially unique services federation [12].

Fog computing involves a significant number of devices operating different tasks and activities in a distributed manner, therefore, reliability is realised as an essential requirement. Some works consider reliability as a part of the quality of service management, usually falling under network or system specifications [68, 70]. In OpenFog, RAS (reliability, availability and serviceability) pillar is a wide umbrella that encompasses many different issues [74].
Concerning fog computing for IoT, reliability is a critical requirement that is supposed to be provided at the different system layers and also spans several different prospectives [12]:

1. Hardware must be reliable and work as expected ;for instance, a sensor must provide the expected readings with the frequency set.

2. Communications among all network elements must be reliable, with support for data transport and message exchange.

3. A reliable scheduling policy, data centre management, and power consumption model.

4. Fog nodes must generate the expected output ;data processing or distinguishing the action to be taken.

Fog for IoT must be reliable, regarding several types of failure; failure of any fog node(s), the failure of the service platform, the entire network failure, the failure of the users interface to system etc. Achieving such reliability requires using different types of techniques [12].

The number of IoT devices is increasing as well as their generated data. If such data is processed on the cloud, this may result in privacy concerns, network delay and congestion [41]. As a solution, [41] leveraged the fog computing concept integrating heterogeneous fog devices, including cloud servers, edge network devices, and end devices. Such utilization of heterogeneous fog devices benefits: 1) location independent, 2) time dependent, 3) massive scale, and 4) latency sensitive applications, which are suitable to run in the fog.
Besides, Basir et. al. [11] argues that the Industrial Internet of Things (IIoT) requirement can be satisfied by fog computing architectures. Also, they believe that the integration of fog with the existing communication technology will reshape all sectors that involve IIoT applications.

Fog Computing, particularly when integrated with cloud computing, has already presented valuable support for distributed systems, entertainment, and advertising as well as several other sectors. The concept of fog and Cloud Computing has a great and recognizable influence on both private and public organizations and has raised their performance and increased business opportunities. Involvement, collaboration, and support of communities, industries and standards organizations are required for the prosperity of such a computing paradigm [74, 86]. Besides, the development and growth of long-awaited new revenue-generating applications and services will be earlier realised with the support of fog computing [7].

Fog computing has certainly added several benefits to the cloud paradigm and also solves some of its challenges. Nevertheless, it still meets several technical and non-technical challenges. In contrast with the centralised infrastructure of cloud computing that has low management cost, with fog computing and/or fog with cloud, the

complexity of management is considerably increasing. It is a research priority for fog computing to develop solutions to address and reduce such complexity. Security in fog computing is also weaker compared with the cloud; hence, mechanisms to empower fog computing security are highly required [86]. Though, Firdhous et. al. [30] believes that the integration of fog and cloud offers the business with a more secure environment as the risk of data theft or manipulation is reduced because data travels less in the network.

Also,as fog computing brings some of the cloud-based applications services and analytics as close as possible to the data source (if not to the edge of the network), the entire system performance will improve. Fog computing will also allow the system to adapt better to the traffic patterns changes; performance enhancements happen faster. However, the massive amount of data created by the connected devices will result in network congestion and performance challenges at the edge of the infrastructure [74].

Moreover, as mentioned earlier, fog proximity and integrating it with cloud lead to reduced latency, the location of application components in the system hierarchy makes latency varies. Application components placement is not an easy task, and optimal placement for better latency needs complicated placement algorithms [102].

From the literature review conducted, we realized that there are several different ways to design architectures for IoT sensing as well as sensing and actuation systems. The emergence of cloud computing and its compensation with fog computing have improved systems quality in many ways resulting in several creative designs. However, we believe that fog in combination with the cloud should be better organized to support better tasks deployment; particularly for SAaaS. This requires a different layering of the architectures than the once followed in most literary works. Besides, services providers are encouraged for more collaborations towards realizing the true smart cities visions. In the remaining chapters, we mainly contribute to these directions.

2.3 Summary

In this chapter, we have presented the foundation concepts and several pre-existing technologies and paradigms on which sensing services and IoT are built upon and depend on for communication, architecture, storage, sensing, arrangement and more. Work in literature of the following sensing services types and applications is listed: participatory sensing, opportunistic sensing, urban sensing, and People-Centric Urban Sensing, etc.

This was followed by showing how integrating such services and concepts with the cloud computing provided remarkable solutions which facilitated and enriched sensing applications, particularly for the IoT with its sophisticated requirements.

Finally, fog computing integration with the cloud and the IoT, with the tackled limitations and benefits added were listed. Cloud computing, fog computing as well as their integration advantages, requirements, evaluation factors, and open challenges were also explained with reference to the IoT and its applications. The explained benefits and features of fog computing were the primary motivation for our contribution.

Chapter 3

Towards Truly Smart City Service Providers: A View on On-demand Everything as a Service

In this chapter, we present a view of the future smart city service providers or the way the future service composition to be performed in IoT and other applications. We also discuss this with an example scenario of composing services from different providers and application domains and present a simple example architecture to illustrate the same.

3.1 A Vision on Future Smart City Service Providers

The future smart city requires a rethinking of the current techniques and come out with plans for faster service composition to provide ready to use services. Besides, consumers -that may subscribe to one service provider only- may always be in a fixed location or mobile and may require new services that in turn may require recomposing new components. The components of service needed to be composed may belong to different owners, be in different locations, or are managed through a different management authorities or service provides (i.e. WSN, IaaS, or S^2aaS, SAaaS).

We emphasise on the services and components that belong to different service providers that require proper plans to allow a smooth and flexible composition.

Any of the services providers can manage the composition; collaborates with other service providers to use their services along with his own once. Or the composition can be handled through an independent third party. The vision is depicted in figure 3.1, and conceptually detailed below:

- There are three connected service providers. The connections show on-demand collaboration.
- Each service provider is supposed to operate fully in isolation. This indicates that each service provider must have:
  - Its own complete architecture of the cloud and IoT devices; architecture design may vary from service provider to another.
  - Virtualization mechanism for its linked IoT devices.
  - Users registration, authentication, and accounting systems.
  - All required database, etc.
- Each particular service provider (say SP1) can offer its registered end users with their requires sensors/actuators by virtualizing its own architecture IoT physical devices.
- SP1 can also request any partner service provider (say SP2 or/and SP2, or both) for any service/IoT device functionality that SP1 does not have in its own architecture.
- A virtual service of service offered by a partner service provider will be then added to the virtual set created by SP1 for each end user requesting it.
- End users nither involved with the communications or any arrangements among partners service providers, nor are they aware of such background details. End users only request services (sensors/actuators) from the service provider they are registered with.

- The IoT physical devices that are the sources of virtual services created for each end user may not always be the same. Assume that a particular end user registered with SP1 has among their requested devices a device that is around their current location with a certain diameter. In such a scenario, selecting the physical IoT device to provide the service must be dynamic.
- The source IoT device must be changed as and when the end user changes their location. The new end user location may be in an area of a partner service provider. This location change results in recomposing the end user's services based on the new target IoT devices.
- End user payment for services is always made to the service provider they are registered with.

The state-of-the-art in the fundamentals technologies like federation, multitenancy, interoperability, etc. is paving the way toward such vision. Yet there will be several challenges like security, privacy and other IoT and networking related issues.

We believe that such collaboration among service providers will eventually lead to widening the area that each independent service provider can cover alone. The same is particularly useful in IoT SAaaS where, for instance, an environmental specialist may be interested in a wide area that is not covered by a single SAaaS provider.

An example architecture is given for composing services from different service providers; i.e. SAaaS provider and mobile applications service provider. This is a simple example to assist in proving the vision explained above.

3.2 Example Architecture

In this example, we aim at making an intermediate architecture between IoT services in the locality and mobile applications services such that the integrating happens temporally on-the-go, as a form of advice -search for what can be useful at this time in this location being IoT services that user can access and use or mobile application services

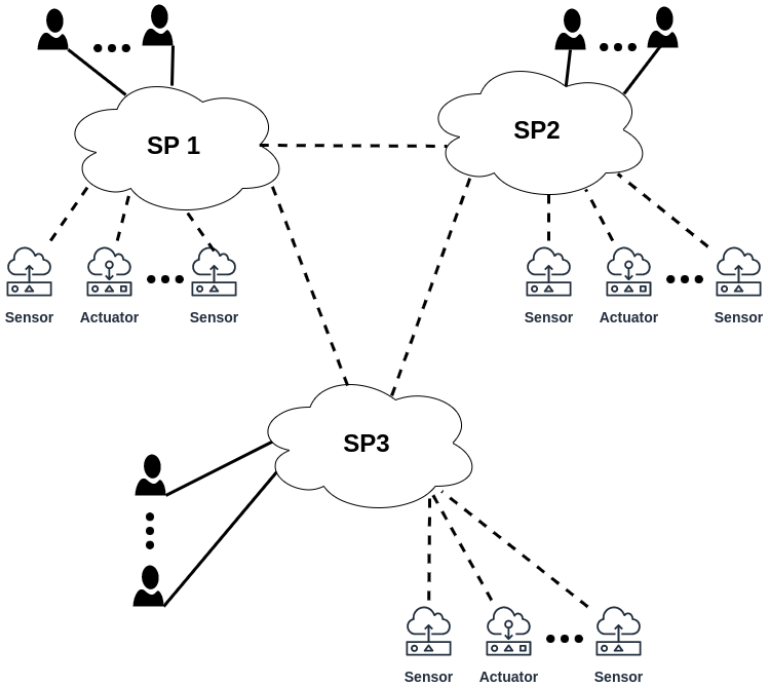

Figure 3.1: Future Smart City Service Providers

or coupon/deal etc- and give advice to the user.

For the purpose of simplicity, we adopt the concept of location-aware recommenders for getting mobile applications services in the locality. we did not follow any specific scheme like publish/subscribe as we will list all available mobile applications services in the locality besides the IoT services in the locality.

Some context parameters like location, date and time, will be taken into consideration to improve the accuracy of list of advised services.

3.2.1 Example Scenario

For the sake of simplicity, we have selected the Shopping Mall scenario as an example to illustrate our proposal throughout this paper. In this scenario, a user Sam enters into a shopping mall, upon his arraival, Sam's location will be determined using his smartphone GPS. A search will be conducted to identify IoT services in the mall like parking navigation system, or availability of kids tracking devices, etc. And also a search in

the commercial applications (like the popular applications in India; Paytm, Little, and nearby,PhonePe, etc.) for any deals to be used in shops, salons, cafe, or restaurants, etc. at this mall. Sam will get a list of all services from which he may use or buy a coupon to grab a deal. This may attract the customer to use the services as they are in physical proximity.

There are many other scenarios in which the idea of composing IoT with mobile applications services can be implemented. However, in this work, we selected a basic scenario for the sake of simplicity of delivering and illustrating the idea.

3.2.2 Example Architectural Design

We hereafter explain the example architecture based on the example scenario. The high-level example architecture is depicted in figure 3.2. And the operational procedure is depicted in figure 3.3.

The location manager determines the customers location using their device GPS and sends it to the Service List Generator.

Service List Generator, in turn, searches the Location IoT Services Data Base and Stores List Data Base and searches the mobile apps to find any deals -in stores in this particular mall- which can be used at the current time.

An advised list of services and deals coupons/stores providing deals will be generated and displayed in the user mobile. The user then may think about what services he likes or coupons to buy.

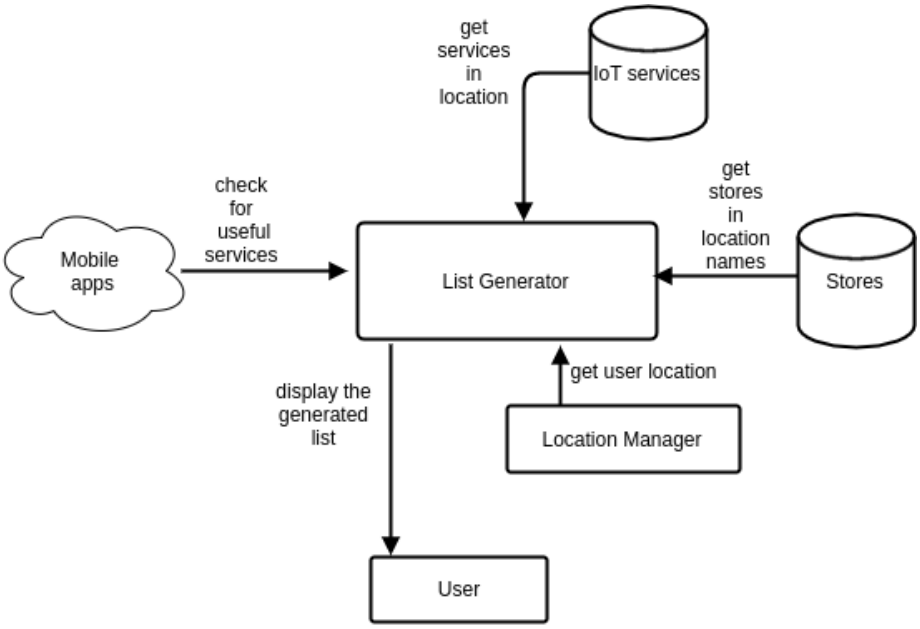

Figure 3.2: High-level Design of the Example Architecture

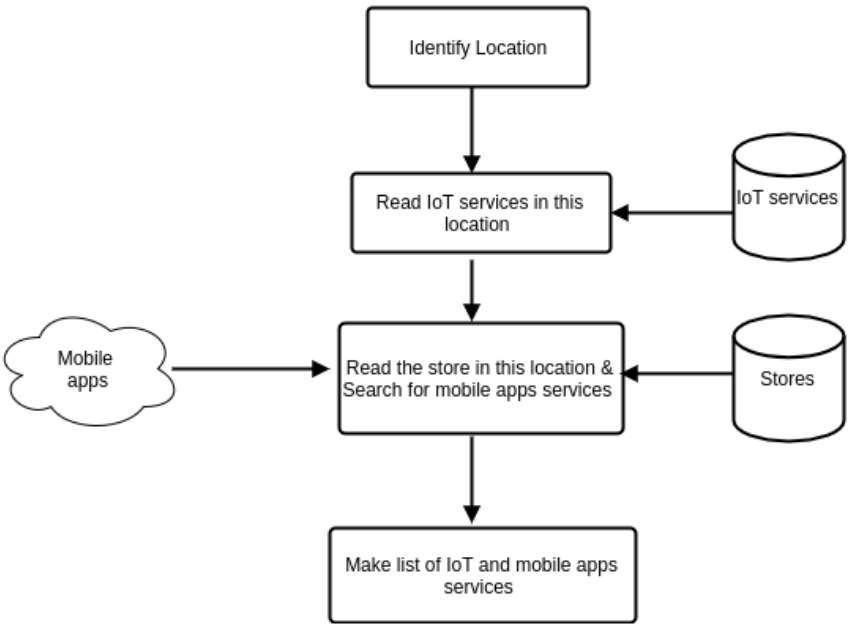

Figure 3.3: The Operational Procedure of the Example Architecture

3.3 Summary

Supported by cloud and fog integration features, IoT service providers -particularly sensing and actuating- are advised to be enhanced. This enhancement such a way to provides service delivery models which composes several types of services from different application domains/service providers. That eventually will make service provider be viewed as an integration of several services provides. This is feasible if such integration is designed carefully following the requirements of the IoT application, as well as cloud fog integration requirement advised by several researchers and standards towards better IoT support: federation, mobility, security, scalability, interoperability, etc.

This is also possible with the state-of-the-art accounting concepts like pay-as-you-use that can govern the relationship among such several service providers as well as end users. This view is more suitable for fast growth of IoT applications and new start-ups which don't have to invest for their own infrastructure as there will be many competitors service providers offering A to Z hardware and software requirements.
This is a vision towards ready made, on demand, and heterogeneous smart city infrastructures wish encourages fast and economically sound innovative applications.

Besides the vision of future smart city service providers, an architecture for integrating IoT services with mobile applications services is proposed in this chapter as an example. The architecture makes the integration as a form of suggestion/recommendation of available and attracted services without the involvement of real composing to create new composite services.

Each service provider in the integrated architecture proposed in this chapter is seen as SAaaS provider. Throughout the remaining of the chapters we discuss our proposed architecture design of such service provider, suggest a suitable tasks deployment in the topology, and simulate the same to clearly show the advantage of such design over the literature towards the vision of the future smart city service providers proposed.

The integration/collaboration of service providers proposed in this chapter can be done as follow:

Chapter 5 details the way a single service provider (say SP1) composes the services belong to its own architecture. Assuming that SP1 is collaborating with other services providers (say SP2), SP1 may request a service from SP2 and create a virtual service and add it to the virtual set created for the end user requesting it.

Chapter 4

Internet of Things Sensors and Actuators Layered Fog Service Delivery Model SALFSD

In this chapter, we propose adopting layered fog architecture to enhance sensing and actuating as a service delivery model. The main goal is to take the benefits of fog computing architecture with built-in failure plan and reduced response time.

4.1 Proposed Architecture (SALFSD)

In this architecture, we follow the layered topology for Sensing and Actuating as Service Delivery Model SAaaSDM which was introduced in [85] adopting Cloud Edge-centric style. We, however, aim at enhancing the framework by adopting dumb gateway nodes besides having layered fog that will enhance the performance, reduce the latency, and provides more composition flexibility and options. Before explaining our proposed architecture in details we explain the same scenario used in [85] with some changes that show the need for our suggested modifications along with a top level description of our approach.

It is worth to mention here that dumb gateways can be considered as an independent service provider that is collaborating with the service provider owning the architecture,

as suggested in chapter 3, or as the managing authority of the IoT devices in case the service provider is working independently.

4.1.1 Top Level Description

The environment specialist (end user) requires observations of many sensors (perhaps of same type or different types of sensors) as well as a the ability to trigger/actuate on actuator(s). To fulfil this request, we may have to hire those devices from different owners (same type of device may belong to different owners or come under different networks/gateways) or an average of observation values is required for some location which require readings from multiple sensors. The Fog Node Manager in the cloud is in charge of selecting the fog node(s) and the gateway(s) associated with the requited Sensor(s)/Actuator(s) which may be in different locations. Figure 4.1 shows the network layers and figure 4.2 shows the top level diagram of the architecture along with the connections among layers.

The following subsections present a bottom-up layerwise illustrating of the architecture Followed by general explanation regarding reducing response time and failure plan taken care in the proposed architecture.

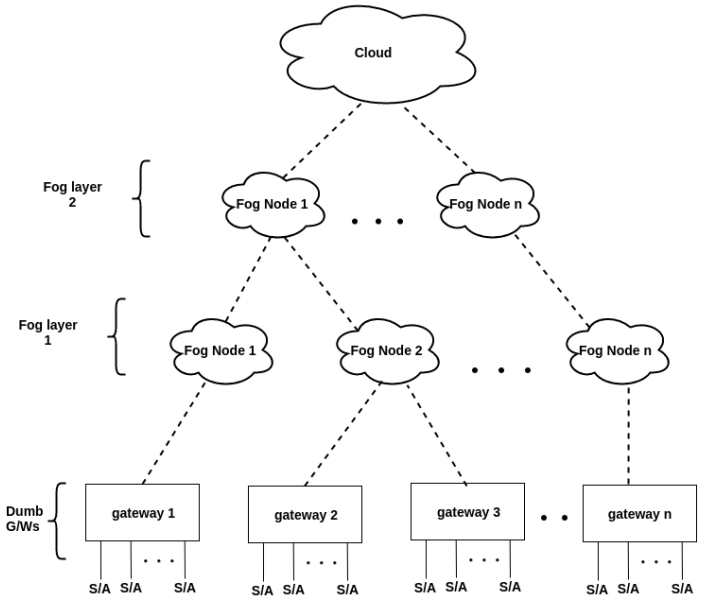

Figure 4.1: Network Layers of the Proposed Architecture

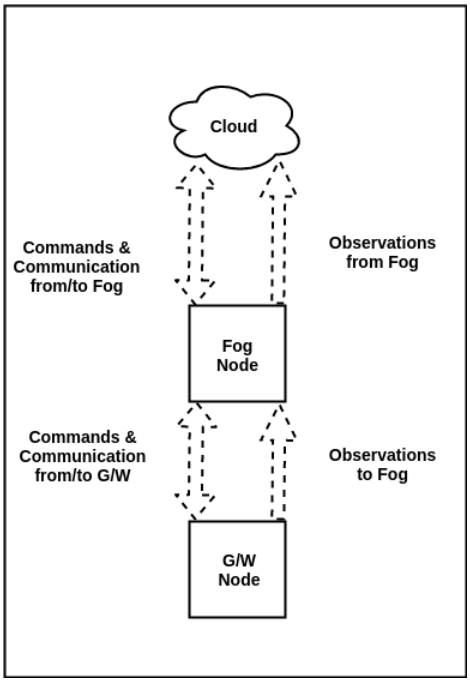

Figure 4.2: Top Level Diagram

4.1.2 Things and Gateway layer

Figure 4.3 shows the gateway architecture, sensors and actuators are connected to dumb gateways. The gateways do not do any processing. The Actuator Selector receives the commands from the fog node (via MQTT message translator that makes it clear it is an actuating command and also extract the target actuator from the command) and applies the force on the targeted actuator. Similarly, Sensors and Observation Selector is responsible for sending each sensors observation values to the fog node via dedicated MQTT channels. The board pins are to connect the board with the physical sensors and actuators for observation receiving and applying commands respectively. The reason of having MQTT translator here is that MQTT does not have any fixed message forms hence the translator will take care of avoiding any clashes in the sensors observation, their channels, sensors IDs, time-tamp etc. the same which are agreed with the fog node.

Assuming the case of collaboration with any other service provider say SPx, then the above details of the gateway do not hold; SPx is supposed to take care of all man-

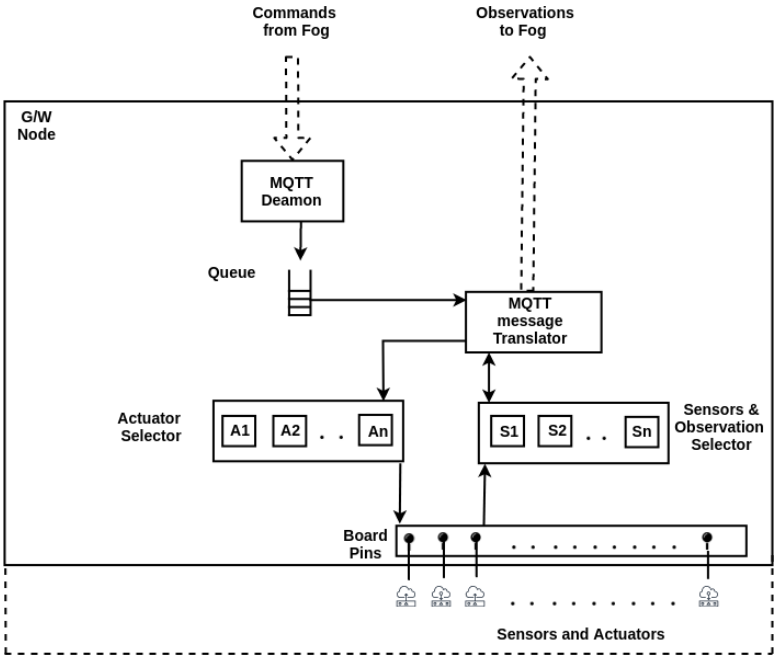

Figure 4.3: Gateway Architecture

agement aspects and communicate the observations and actuation commands directly to the cloud of service provider owning this architecture.

4.1.3 Fog layer

Figure 4.4 shows the fog node architecture. Each fog node is responsible for managing the gateway(s), sensors and actuators assigned to it by the cloud assignment submodule. Fog node receives observations, convert them into MongoDB and store them in the observation data base. Figure 4.4 depicts Fog Node architecture. End users may have some cases with predetermined reaction to be taken, hence the monitoring and the decision will be done at the Specified Cases Manager in the fog node in which the sensors and actuators associated with the case are assigned to. For example (in figure 4.1), an end user has rented sensors and actuators belongs to gateway 1, then the specified case monitoring will be assigned to fog node 1 in fog layer 1. If the sensors and actuators of end user interest are in gateway 2 or gateway 3, or some in both gateways, then the monitoring must be in fog node 2 in fog layer 1. Assuming

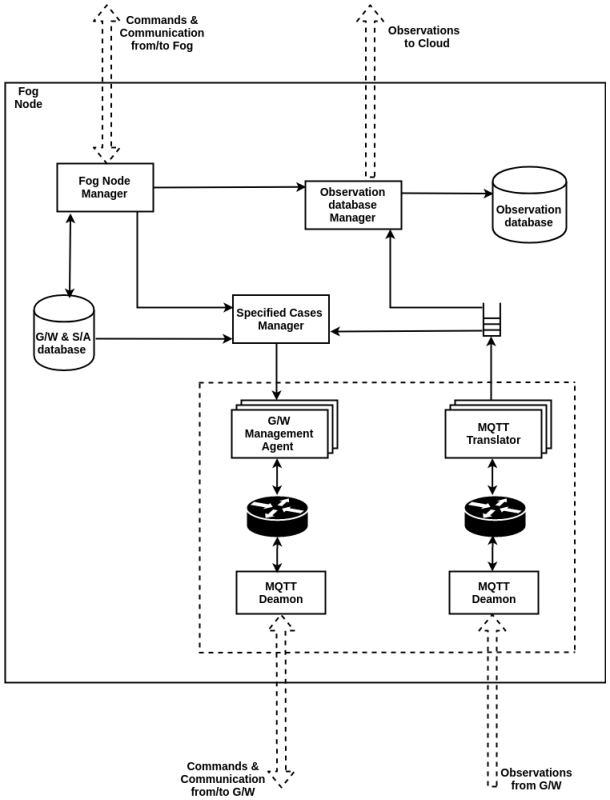

Figure 4.4: Fog Node Architecture

the sensors and actuators of end user interest belong to gateway 1 and gateway 3, then the monitoring must be in fog node 1 in fog layer 2. This monitoring plan holds while going up in the topology; in case there layer 2 is the last fog layer before the cloud, then cloud itself will monitor cases of sensors and actuator belong to gateways 1 and n.

Once Cloud assigns gateways, and sensors and actuators (G/W and S/A for short, respectively) to any fog node, their information is stored in the G/W and S/A database through the Fog Node Manager which is responsible for communication with the Cloud. The dashed line around G/W Management Agent and MQTT Translator in the fog node in figure 4.4 shows the logical grouping that is to be reassigned to the nearest fog node in case of failure of current fog node managing the gateway; this is the main idea of failure plan that will be discussed in a later subsection. Observations sent from G/W(s) will be received by MQTT translator and enter into the queue in which the Observation Database Manager is responsible of converting them into Mon-

goDB and storing them in the observation database. Whereas G/W Manager Agent receives the commands of applying force on a target actuator from Specified Cases Manager and sends it to the gateway in which the target actuator is connected.

4.1.4 Cloud layer

In cloud side, as showing in figure 4.5, the Cloud G/W is the gateway between end users and the service provider. It receives the end user requests and delivers the virtual set of devices. Cloud G/W is also the gateway for devices owner to register their devices details. Core management send the request details to the Physical S/A Selection module which in turn is responsible for selecting the optimal devices as per the Service Level Agreement and previous customers review. We also take the availability of the device into consideration for selection; for example the end user request for specific type of sensor (i.e. from 2AM to 11 AM) can not be fulfilled by hiring one physical device as it is not available for the required duration (available only from 2AM to 5Am) hence this submodule will select one or more devices (which are available for the remaining time required) of the same type at the same required location and compose them for the required duration.

The selected physical devices will be then virtualized to create a virtual set for the end user, here the virtualization is on data level with respect to sensors observation which is the responsibility of Virtualization Manager. Fog G/W assignment is responsible for assigning the management of G/W to the nearest fog node and reassignment to any other node in case of fog node fails. The Specified Cases Manager monitor the cases that are not assigned to any fog node as per the network topology.

4.2 Reducing Response Time and Failure Plan

The use of fog layer between the cloud and deploying the specified cases monitoring in fog nodes lead to starting the checking process of the observation and triggering the desired actuation command on actuators (if any is required as per the specified cases) faster than having it at the cloud side. This is the main way we reduce the response time for getting the end user's actuating command done faster for the prespecified cases; in

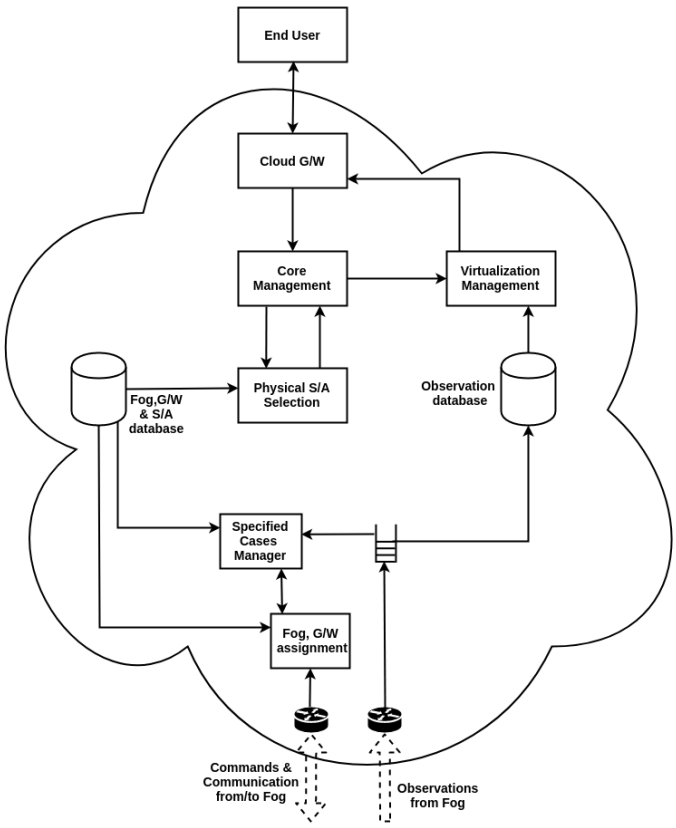

Figure 4.5: Cloud-side Architecture

other words we move the decision to lower level of architecture.

Failure Plan: The main goal of the failure plan is to cope with any fog node failure situation as fog nodes are known to be potential points of failure. For example, in figure 4.1, if Fog Node 2 fails, then Fog G/W assignment manager in the cloud will reassign the management of gateway 2 to Fog Node 1. That explains the dashed line around G/W Management Agent and MQTT translator in fog nodes in figure 4.4 that shows the units that are supposed to be migrated to the new management fog node.

4.3 Comparison with Related Work

In this section we present feature-wise comparison for our proposed work against the aforementioned related work with respect to points listed in table 4.1.

Table 4.1: Comparison of Proposed Work with Related Work

Reference	[26]	[28]	[27]	[59]	[85]	SALFSD
Topology	Node-Cloud	Node-Cloud	Device-Cloud	Node-Cloud	Cloud Edge-Centric	G/W-Layered Fog-Cloud
Processing at	No *	Mote, End User	Mote, End User	Cloud	Edge, Cloud	Fog, Cloud
Virtualization	Device-level	Device-level	Device-level	At Cloud (Open-Stack based)	At Cloud	At Cloud
Failure Plan	No**	No**	No**	No**	No	Yes
SCM	No**	No**	No**	No**	No	Yes
Monitoring Offloading	No**	No**	No**	No**	No	Yes
Observation Filtering	No**	No**	No**	No**	No	Yes
Working Model/Simulation	No	No	Only For Android	No	No	Simulated by YAFS [57]

Regarding processing in [26] (*), the work did not state any thing about data flow or processing. Also with respect to failure plan (**), works [26, 27, 28, 59] do not adopt

any concept like Fog or Edge nodes which is where SALFSD has the failure plan, specified cases monitoring, monitoring offloading, and observation Filtering.
This comparison is based on features (listed in the table) which highlights the main differences among works involved in the comparison.

4.4 Summary

The most important factor in Sensing and Actuating as a Services, from our point of view, is best utilizing the network topology in a way that observation processing, physical nodes selection, virtualization, and realtime decision making etc. to be made in the correct layer for better enhancement of performance. In Fog computing, some tasks are offloaded from the cloud to the fog node that significantly reduces the processing time and communication overhead. This results in better performance than having all tasks executed centrally at the cloud, such performance enhancement is significant in domains like healthcare and for SAaaS in general for faster triggering the actuation commands.

To this end, and to provide multiple sensors and actuators composition options (of sensors and actuators belonging to different gateways/locations of the end used area of interest), with reduced response time and in built fault resistance plans, we in this chapter proposed a layered architecture (Things and gateway layer, Fog layer, and Cloud layer) SALFSD. The architecture is designing for Sensing and Actuating as a Service Delivery for Internet of Things. Having fog between the gateway and the cloud will be a good addition for such architectures as it will have several advantages: 1) Will enhance the performance. 2) Reduce the latency, and 3) Provide more options for composing the requested types of sensors and actuators belonging to different gateways/locations under the area(s) of end user interest.

The architecture discussed in this chapter is further extended in chapter 5. The extension details more features of the architecture, elaborates in the communication among nodes in the different layers, and details the communication among both the cloud and fog node internal modules. While chapter 6 details the simulation configuration and results analysis.

Chapter 5

Enhanced Layered fog Architecture for IoT Sensing and Actuation as a Service

This chapter presents an extension of architecture introduced in our previous work in chapter 4 where we theoretically presented a high-level design of SALFSD discussed our expectations of SAaaA performance enhancement considering SALFSD design. Besides, this work details entire SALFSD components and communication among them, and adds some improvements to it by considering the key performance metrics of SAaaS from network and architecture points of view. Also, formal verification of architecture correctness is presented.

5.1 Proposed Architecture

SALFSD aims at enhancing the performance of the typical SAaaS paradigm with several layers of fog nodes between the cloud and the sensing and actuation layer (IoT layer).

SALFSD is designed to contain several layers of fog nodes; the number of layers and their deployment depend on the organisation of the geographical area covered by the service provider and has no technical impact on the design. For instance, it can be as

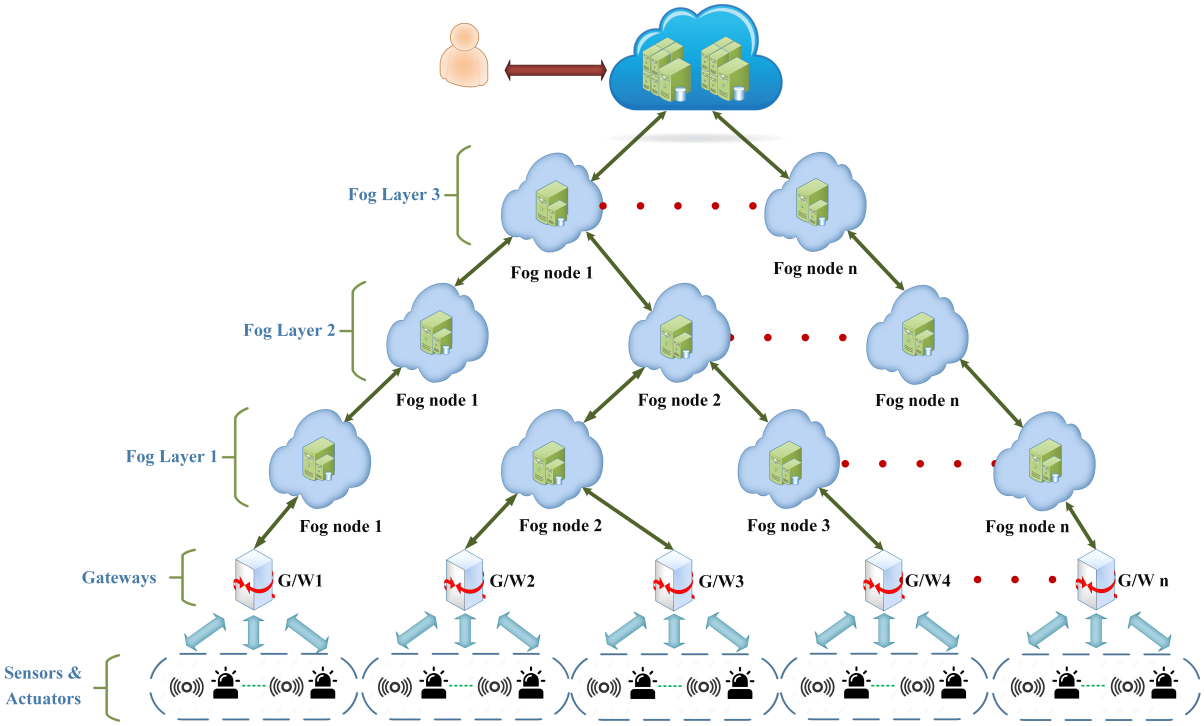

Figure 5.1: SALFSD Topology

organisations, cities, countries and so on as depicted in figure 5.1. IoT devices are connected to the gateways connected to fog nodes in the lowermost fog layer, which on turn are connected to the upper fog layer. The uppermost fog layer is connected to the cloud. There is no collaboration or connection among the fog nodes in the same layer; all go through the cloud.

To this end, the performance metrics governed SALFSD design are:

1. Scalability and availability of SAaaS provider network.
2. Avoiding observations and actuation commands loos due to fog node(s) failure while maintaining reduced observation and actuation latency.

This section starts by listing the main contributions, then explaining a general scenario of SALFSD architecture. Finally it explains SALFSD layers, their components and communication among them with details.

5.1.1 Contributions

To the best of our knowledge, this work is the first attempt to have multiple layers of fog nodes in between the cloud and IoT devices in SAaaS with the main contributions listed below:

1. Failure Plan: As every fog node is known to be a point of failure in the network, we have proposed a dynamic reactive failure plan to maintain the connectivity by reassigning the tasks of the failed fog node to the nearest connected fog node in the same layer or to the parent of the failed node.

2. Specified Cases Monitoring (SCM): Typically, in SAaaS, end user receives the observations and decides upon the desired actuation to be triggered. However, end user may have pre-specified actions to be triggered in response to some types of sensor(s) observations thresholds (i.e. acting in certain situations). We deploy the monitoring of such cases in the fog nodes as well as in the cloud.

3. Monitoring Offloading: The default fog node responsible for monitoring the user specified cases is the one closer to the gateway(s) connected to sensors and actuators rented by the end user. However, as the specified cases pre-specified by end user needs actions to be taken fast in response to a certain condition(s), we may dynamically handle the monitoring task to the parent of the fog node in case it is overloaded to reduce the actuation latency.

4. Observations Filtering: Fog nodes also filter the observations before forwarding to parents or to the cloud to drop the corrupted messages - if any- to avoid unnecessary consumption of the bandwidth.

5. Sensors and Actuators Selection: We have implemented the Nondominated Sorting Genetic Algorithm II (NSGA-II) [25] to select the optimal sensors and actuators of end user interest types at their chosen location. The selection is bi-objective based on cost and feedback.

5.1.2 Example Scenario

Assume that SALFSD service provider has set up its infrastructure (cloud and layered fog nodes) which covers a particular geographical area; city(s), country(s),

or region(s). The service provider has agreements with many sensor/actuator devices owners (mobile devices, wireless sensor networks, or even standalone devices) that are linked to gateways. The gateways are connected to fog nodes in the lowermost fog layer in SALFSD infrastructure. As per the agreement between the services provider and devices owners, sensor devices must send their observation to the services provider frequently/continuously as per the time agreed. Also, actuation devices owners must allow the service provider to send actuation commands to be executed.

Now assume end user (environment specialist, agriculture monitoring agency, forest monitoring, etc.) is interested in getting observations of type(s) of sensors either coming under one or several network/area. According to the analysis of the received observations, end user may decide to send actuation command(s) to one or many actuators. Again, actuators may be under one or several network/area. As explained in 5.1.1, SALFSD deploys SCM of each end user -if any- in fog nodes as well as in the cloud where the said node will trigger the actuation command on behalf of the end user.

5.2 SALFSD Cloud

This section details the components of SALFSD cloud and their responsibilities as depicted in figure 5.2. The communication among the cloud, fog and gateway layers is depicted in figure 5.12.

5.2.1 Cloud Gateway

Cloud gateway is the interface with the end user for registration, receiving requests (required sensors and actuators and other details), sending observations to end user, and receiving end user's actuation request. End user's requests are forwarded to Core Management.

5.2.2 Core Management

Core management extracts the types and numbers of S/A requested by end user and budget, sends them to Physical S/A Selection model to get the optimal S/A devices of types requested by end user in his area of interest. Virtualization Management model is informed about those selected devices.

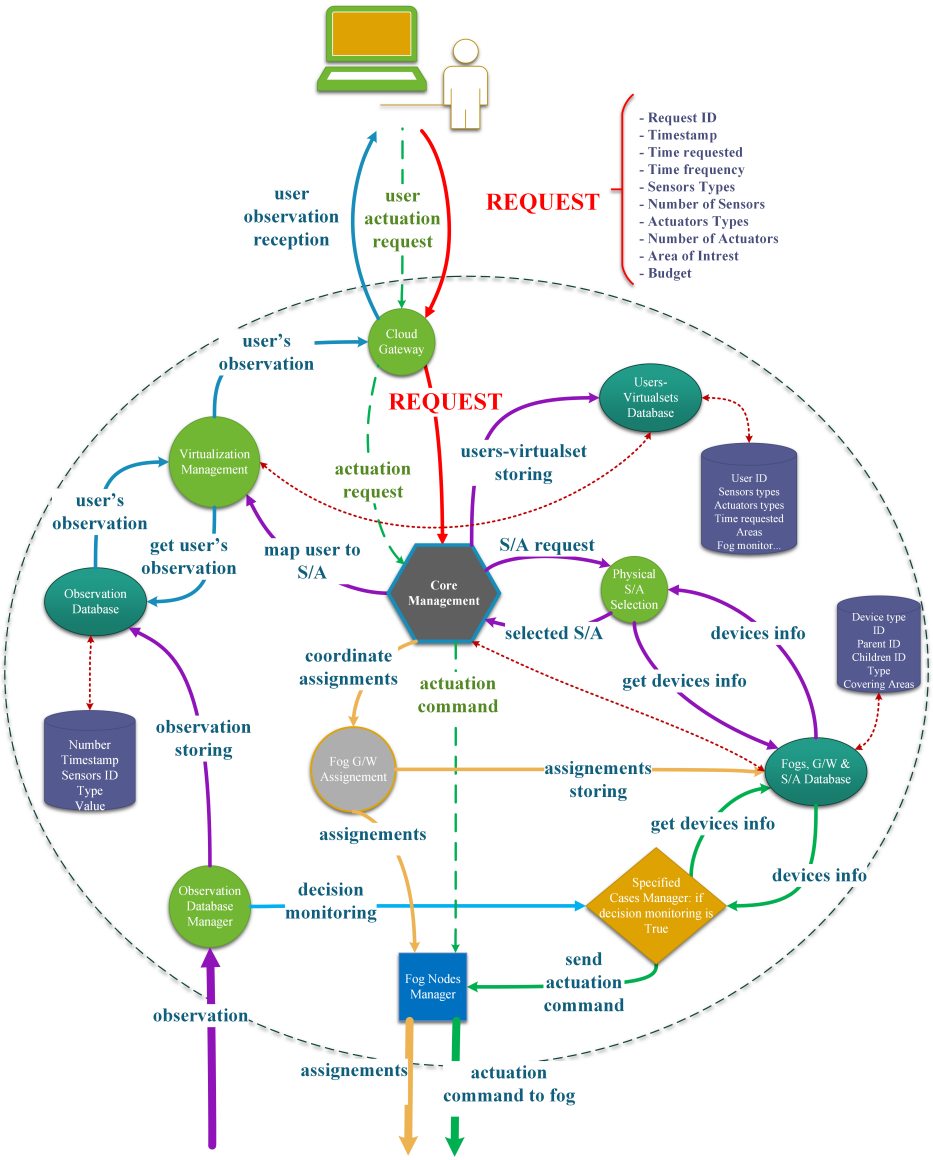

Figure 5.2: Cloud Design with all its Components and Types of Internal Messages Among its Modules

5.2.3 Physical S/A Selection

Physical S/A selection is responsible for selecting the optimal sensors and actuators requested by end user in their area of interest, the selected sensors/actuators are passed to virtualization management to create virtual set for each end user.

End user may request a set of sensors/actuators which may be available with different

feedback ratings and different associated costs. For effective based service, there is a need for bi-optimization of sensors/actuators cost and feedback as below:

$$Minimize(F_{Cost}) \tag{5.1}$$

$$Maximize(F_{Feedback}) \tag{5.2}$$

NSGA-II is a well known multi-objective optimization algorithm and has solved several optimization problems in IoT and many other different areas.

SALFSD strategy adapted NSGA-II method where every task is defined as vector to be optimized. Comparison of cost and feedback values produces an optimal sensors/actuators list. Algorithm 1 shows SALFSD NSGA-II based algorithm. The solution list, generation number and sensors/actuators list are initialized (lines 2-4). Then the list of active sensors/actuators are aggregated and assigned to solution list(line 5). Each solution is equipped with fitness_1 and fitness_2 functions (line 8). In order to equate each population sample values with the other solution from the list, NSGA-II uses a rapid $non\_dominated\_sort()$ to filter the individual solution list into some kind of different dominant types (line 8). Then crowding distance is calculated to find nearby values (line 10). $Crossover\_Mutation()$ is used to produce the new population (line 12). As per the non-dominated strategy, every entity governs another, even if the dominant term is utilized to break the N-list into a number of fronts (lines 13-15). SALFSD utilizes $crowding\_distance()$ to choosing a subset of solutions of similar domination rank by removing the fronts(line 17). Such phases go on until the end criterion for completion or stopping. Then the optimal top-N suggestions for specific solutions are considered. The maximal optimum S/A_id will therefore be selected (lines 19-22).

$$CD_{im} = \frac{f_m(x_{i+1}) - f_m(x_{i-1})}{f_m(x_{max}) - f_m(x_{min})}, i = 2..., (l-1) \tag{5.3}$$

$$CD_i = \sum_{m=1}^{M} CD_{im} \tag{5.4}$$

The crowding distance is calculated as per equations 5.3-5.5. In equation 5.3, in ascent order of fm, sort and compute all the 'l' solution in a pareto front and in equation 5.4, reiterate it for every objective and identify the crowding distance of i solution. In equation 5.5, owing to two i and j solutions, the solution i would prefer to solution j as it is with lower (better) rank than solution j or both may be with the same rank but solution i is in less crowded region.

$$Ri < Rj \; or \; (Ri = Rj \; and \; CDi > CDj) \tag{5.5}$$

Algorithm 2 selects the optimal sensors/actuators Id(s) based on 3 cases of end user budget inputs: 1) If there are sensors/actuators in S/A equal to the given budget, then the specific optimal sensors/actuators Id(s) will be selected line 5. Otherwise, 2) nearest cost to end user budget in the list of Ss/As with a certain threshold value are considered for the optimal $S/A\_id(s)$ selection line 6. 3) If budget is not taken into account, all available sensors/actuators are considered for the optimal selection line 7.

5.2.4 Virtualization Management

Once the optimal S/A are selected, the virtualization manager is in charge of creating virtual sets for each user as per their selected sensors/actuators and handling sensors observations to the cloud gateway to be sent to the end user. Figure 5.3 shows sensors and actuators virtualization.

5.2.5 Fog G/W Assignment

This model takes care of assigning end user specified cases monitoring to the nearest fog node as per their geographical distribution. For example, in figure 5.1, suppose that end user1 has hired sensors/actuators in G/W1, then fog node1 in fog layer1 will be assigned the monitoring of end user1 specified cases. If end user has hired sensors/actuators from G/W2 and G/W4, then fog node2 in fog layer2 will be assigned the monitoring. In case end user sensors/actuators are in G/W1 and G/W4, then fog node1 in layer fog3 will be assigned the monitoring. Cloud as well may monitor the end user specified cases in case there is no any fog in the topology that has access to (parent of) all G/Ws the end user has hired S/A from, G/W1 and G/Wn for instance. In addition, and in case of fog node(s) failure, Fog G/W Assignment is in charge of

Algorithm 1: NSGA-II based $SALFSD$-1 (S/A_list)

Initialize population
$N_{size} \leftarrow size(S/A\_list)$
$G_n \leftarrow 0$ ▷ Generation_number
$S_l \leftarrow 0$ ▷ Sensor_list
for $(S\ in\ S/A\_list)$ **do**
    $S_l \leftarrow (S, sensor/actuator)$ ▷ List of active sensors/actuators
$S_n\_list \leftarrow S_l$ ▷ Solution_list
Initialize population for fitness function
while *(*$G_n < max\_gen$*)* **do**
    $F_p \leftarrow Computefitness\_1, S_n\_list$ ▷ price Finess function
    $F_f \leftarrow Computefitness\_2, S_n\_list$ ▷ feedback Finess function
    $S_{ND} \leftarrow non\_dominated\_sort(F_p, F_f)$ ▷ NSGA-II fast non dominated sort
$C_D = [\,]$
for *each* S_{ND} **do**
    $C_D.append(crowding\_distance(F_p, F_f, S_{ND}))$ ▷ Find crowding distance
$S_n\_list1 \leftarrow S_n\_list$
for *each* S_n_list1 **do**
    $S_n\_list2 \leftarrow Compute\_Crossover\_Mutation(S_n\_list1)$ ▷ Compute Crossover Mutation
$F_{p_1} \leftarrow Computefitness\_1, S_n\_list2$
$F_{f_1} \leftarrow Computefitness\_2, S_n\_list2$
$S_{ND_1} \leftarrow non\_dominated\_sort(F_{p_1}, F_{f_1})$
$C_{D_1} \leftarrow [\,]$
for *each* S_{ND_1} **do**
    $C_{D_1}.append(crowding\_distance(F_{p_1}, F_{f_1}, S_{ND_1}))$ ▷ Find crowding distance
$S_n\_list2 \leftarrow [\,]$
for *each* S_{ND_1} **do**
    $front = Sort(S_{ND_1}, C_{D_1})$ ▷ Sorting the front values
    for *each* $val\ in\ front$ **do**
        $S_n\_list2.append(val)$
        if $(size(S_n\_list2) == N_{size}$ **then**
            break
$S_n\_list = S_n\_list2$ ▷ Restore the solution
$G_n = G_n + 1$

Algorithm 2: $SALFSD$-2 Optimaization Selection

1: $S/A\_list \leftarrow$ update(S/A_id) ▷ Update S/A_list
2: $Optimal\_S/A\_id = 0$ ▷ Initialize S/A_id
3: S/A_list=[ ] ▷ Initialize S/A_list
4: *Budget* $\leftarrow User\_input$ ▷ Cost Budget of end user
5: **if** *price*(S/A_id) == *Budget* **then**
    $Optimal\_S/A\_id \leftarrow Optimal\_feedback(S/A\_id)$ ▷ Optimal sensors/actuators Id(s) equal to budget
6: **else if** $Cost(S/A\_id) >$ *Budget* **then**
    $New\_Cost(S/A\_id) = Cost(S/A\_id)$ + threshold
    $S/A\_list.append(new\_cost(S/A\_ids)$
    $Optimal\_S/A\_id \leftarrow SALFSD-1(S/A_list)$ ▷ Optimal sensors/actuators Id(s) with budget + threshold value
7: **else**
    $S/A\_list.append(S/A\_ids)$
    $Optimal\_S/A\_id \leftarrow SALFSD-1(S/A_list)$ ▷ Optimal sensors/actuators Id(s) from all available once
8: return $Optimal\_S/A\_id$

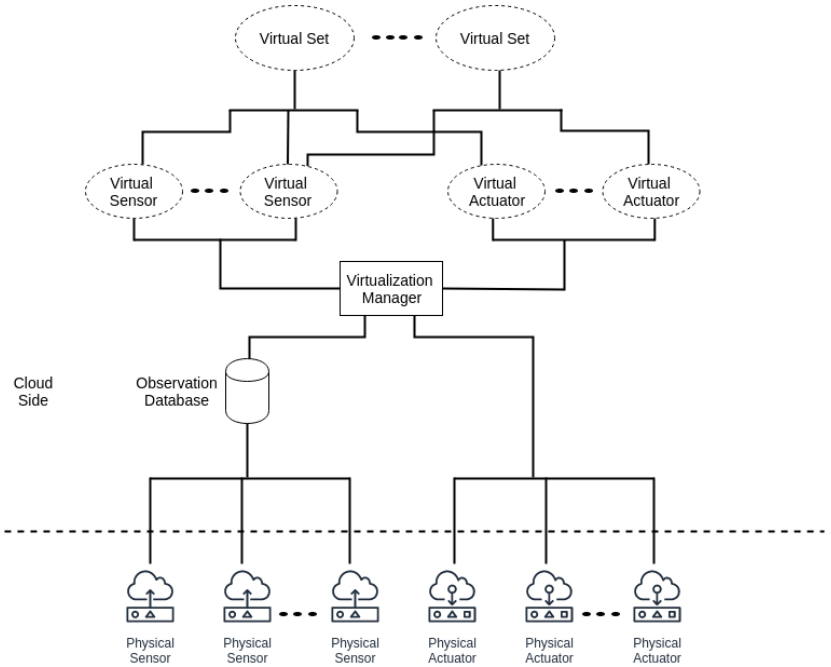

Figure 5.3: Virtualizing Sensors and Actuators

reassigning children fog nodes and/or G/Ws.

The failure plan algorithm is given in algorithm 3. If a failure occurred, the failed node id and ids of its children are retrieved, line 2. Line 3 finds the nearest fog node in the same layer. If there is no such node or it is far from the geographical location of the failed node, its parent node will be selected; line 4. The reassignment is done for all children; line 5. If previously failed node reconnected, the original branch path will be restored by reassignment of children nodes to the reconnected fog node, lines 6-7.

Algorithm 3: Failure Plan

```
1: topologystate ← topologyMonitoring()
2: if topologystate = failureOccurred then
       failedNode ← getFailedNode()
       // Read Fogs, G/W & S/A Databases
       childrenNodes ← listChildrenNodes(failedNode)
3: for each fog node in topology do
       reassignedNode ←
     getNearestNodeInSameLayer(failedNode)
4: if reassignedNode = Null then
       reassignedNode ← getParentNode(failedNode)
5: for each childNode in childrenNodes do
       newAssignement(childNode, reassignedNode)
       break

6: else if topologystate = reconnection then
       reconnectedNode ← getreconnectedNode()
       childrenNodes ← listChildrenNodes(reconnectedNode)
7: for each childNode in childrenNodes do
       newAssignement(childNode, reconnectedNode)
```

5.2.6 Specified Cases Manager

Specified cases manager in the cloud is to monitor the specified cases for end users in such cases that the cloud is the only parent of all branches of G/Ws where the end user sensors/actuators are connected to; for instance G/W1 and G/Wn in figure 5.1.

5.2.7 Fog Node Manager

All control messages to be sent from the cloud to fog nodes (e.g. assignments and actuation commands either form generated by SCM or received from end users) are sent through Fog Node Manager.

5.2.8 Fogs, G/W & S/A Database

Being in the cloud, this database stores information about all devices in the entire network; such information is used for S/A physical selection, specified case monitoring, and assignment.

5.3 SALFSD Layered Fog

This section explains a fog node components, their responsibilities, and communication among them as depicted in figure 5.4.

5.3.1 Fog Node Manager

Fog node manager receives control messages from the cloud or from parent fog node. Two types of control messages are received:

1. Assignment: Either to be forwarded to child fog node or to the node itself. The former is passed to "Fog, G/W Management Agent" and the later is stored locally.

2. Actuation Request: Either to the current node itself or to its child node, both are passed to "Fog, G/W Management Agent."

5.3.2 Specified Cases Manager

Specified cases manager in each fog node is to monitor the specified cased for end users whom there hired Sensors/Actuators come under it. For example, fog node 2 in fog layer 2 monitors of end users hired Sensors/Actuators in G/W 2 and G/W3, fog node 2 in fog layer 2 monitors for end user of either G/Ws2,3, and G/W4, fog node 1

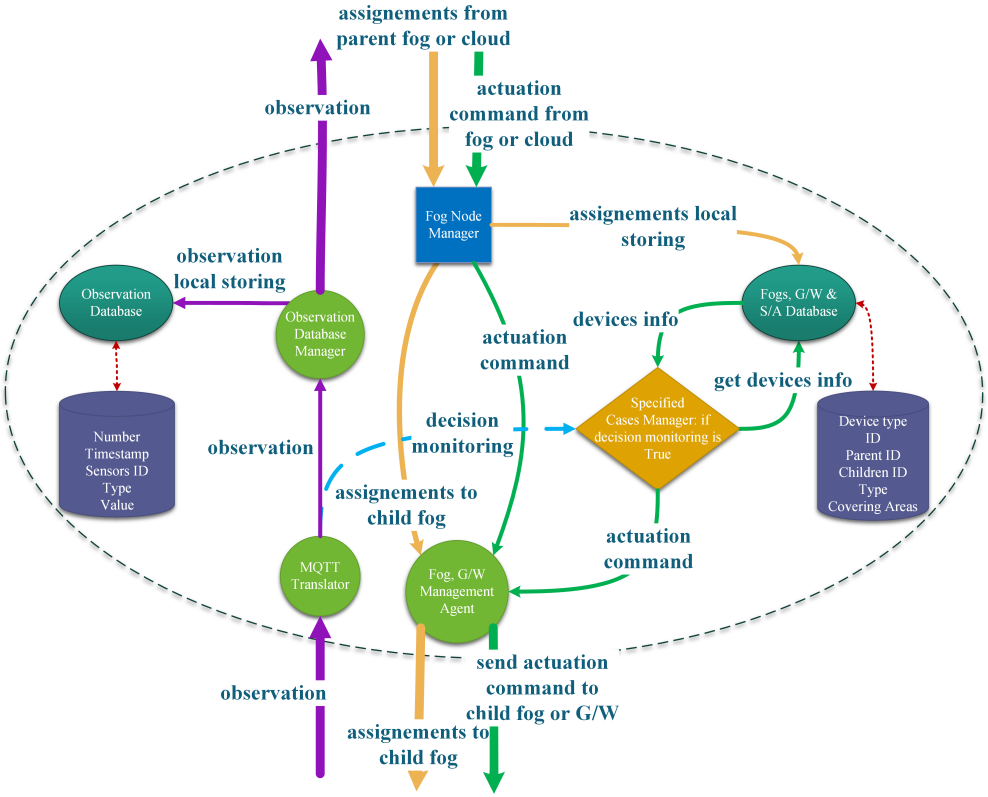

Figure 5.4: Fog Node Design with all its Components and Types of Internal Messages Among its Modules

in fog layer 3 monitors for end users of G/W1 and any of G/Ws2,3, and 4.

The above explained monitoring is the default setup by the cloud as per SALFSD infrastructure and end users hired Sensors/Actuator. However, to avoid increased actuation latency, if a particular fog node in any layer is overloaded, it handles the monitoring to its parent node as shown in algorithm 4. For each observation, monitoring manager checks if the node has previously offloaded the monitoring to its parent but now it is not overloaded; hence it will retrieve the monitoring from its parent; lines 1-2.

If there was no previous offloading, the fog node makes sure that it has to monitor the observation either being the default monitoring node or its child node has offloaded the monitoring task to it; line 3. If so, it checks its monitoring load if it is overloaded, it offloads the monitoring to its parent node; line 4. Otherwise, it will go ahead with

checking the SCM; line 5. In case SCM generates an actuation command, it is passed to (Fog, G/W Management Agent.)

Algorithm 4: SCM Monitoring

```
1: for each Observation do
2: if (Self is defaultMonitoringNode) and (currentMonitoringNode is not Self)
   and (monitroingLoad < threshold) then
       // Previously offloaded monitoring, but now node is not overloaded.
       currentMonitoringNode = Self
       chechSCM(Observation)
       break
3: if Self is defaultMonitoringNode or childOffloadedMonitoring then
       monitroingLoad ← checkMonitroingLoad(Self)
   4: if monitroingLoad >= threshold then
           parentNode ← getParentNode(Self)
           currentMonitoringNode = parentNode
           offloadMOnitoringToParent(parentNode)
   5: else if monitroingLoad < threshold then
           chechSCM(Observation)
```

5.3.3 Fog, G/W Management Agent

Fog, G/W Management Agent receives two types of messages:

1. Actuation commands either from Fog Node Manager or from Specified Cases Manager. In any case, if the current fog node is in the lower fog layer, the command is passed to the designated G/W where the targeted actuator is connected to. Otherwise, the actuation command is forwarded to the designated child fog node.

2. Assignment from Fog Node manager. In such a case, the current fog node is clearly not in lower fog layer; hence it forwards the assignment to the designated child fog node.

5.3.4 MQTT Translator

MQTT (Message Queuing Telemetry Transport)[69] is a light weight messaging transport protocol. MQTT is suitable for IoT devices and machine to machine communication as it was designed with such devices constraints and requirement in mind; low power and bandwidth consumption, small and easy code implementation in devices, low latency, and continuous session awareness. Due to its many-to-many communication nature, MQTT supports sensor data in real time.
MQTT does not have designated message form; hence SALFSD uses a translator for avoiding any mixups in the sensors' observations, sensors' channels, ids, time-tamp etc.
MQTT translator is the fog node entry for incoming observations for gateways. Upon receiving each message from gateways, it first checks the observation if it is corrupted it will drop it; algorithm 5. lines 1-2. Otherwise, only observation value and required information in the message will be extracted, and a copy is sent to both specified cases manager and observation database manager line 3.

Algorithm 5: Observation Filtering

```
1: observationStatus ← checkObservationStatus()
2: if observationStatus is corrupted then
       dropObservation(observation)
3: else
       forwardObservation(observation)
```

5.3.5 Observation Database Manager

Observation Database Manager forwards all observations received from MQTT Translator to the parent (fog node or the cloud- replication is guaranteed at a higher level) and stores a copy locally.

5.3.6 Fogs, G/W & S/A Database

Being in fog node, this database stores information about all devices that comes under it in the network topology; such information is used for S/A physical selection and

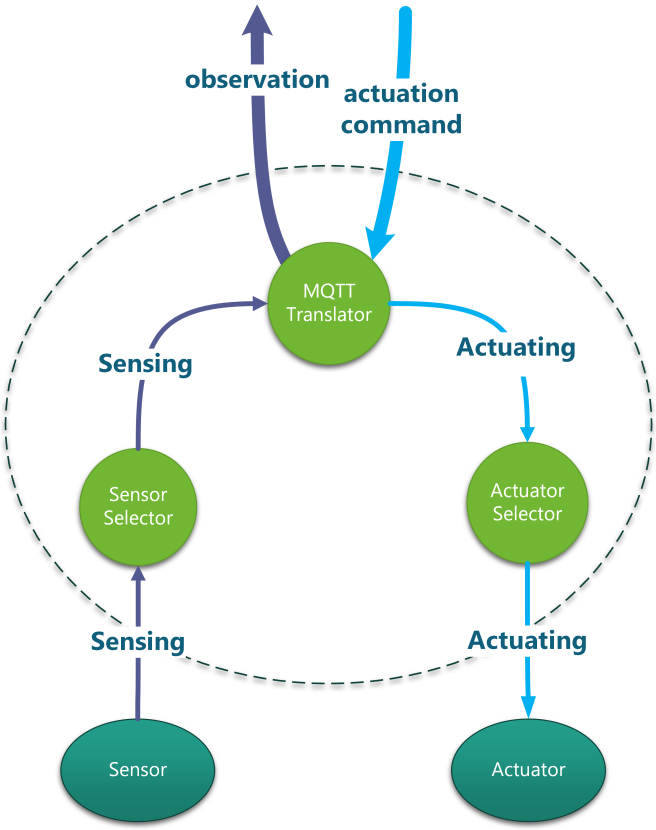

Figure 5.5: Gateway Node Design with all its Components and Types of Internal Messages Among its Modules

assignment.

5.4 Gateway in SALFSD

Gateways in SALFSD are considered dumb gateways as they do not do any form of data processing; only forward observations from sensors to fog node and apply the actuation commands received from fog nodes on the sensors. Below subsections explain SALFSD gateway components and their responsibilities as depicted in figure 5.5.

5.4.1 MQTT Translator

MQTT Translator is responsible for communication between G/W and fog node.

- Once it receives actuation command messages, it passes only the necessary in-

formation to the Actuator Selector; Actuator cloud id, actuation period (alarm actuator for instance.)

- Observations received from Sensors Selector are sent to the fog node via the dedicated MQTT channel.

5.4.2 Actuator Selector

Actuator Selector in the gateway receives actuation commands from MQTT Translator and sets the commands in the designated channel of the target physical actuator.

5.4.3 Sensor Selector

Sensor Selector in the gateway receives sensing from the physical sensor, bends them with the correct sensor cloud id and pass it to MQTT Translator.

5.5 Formal Verification of Architecture Correctness

This section proves the correctness of the failure plan and topology connectivity monitoring, SCM offloading, and observation filtering. First, the related architecture invariants and properties are listed below.

5.5.1 Architecture Invariants and Properties:

(i) Architecture Invariants:

1. The architecture is fully connected if all branches are connected.
2. Failure of any fog node in a branch leads to the branch being disconnected, hence the architecture is partially disconnected.
3. If a branch is disconnected, the dependent entities (Children fog nodes of the failed fog node, Gateways, and IoT devices) are unreachable.

(ii) Final Objective:

4. Reassignment of dependent entities of the failed node to any other node reconnects the branch and make the dependent entities reachable again, and hence the architect is fully connected again.

(iii) Functional properties:

5. When failure occurs, all the dependent entities are eventually reachable either through a neighbouring node of the failed fog node or through its parent node.
6. Any failed fog node may not always reconnect, hence the original branch path may not always be restored.
7. When a failure occurs, sending messages to dependent entities of the failed fog node is paused until an alternative path is established by the reassignment.
8. The reassignment can always end correctly either to a neighbouring node or to the parent node.

5.5.2 Connectivity Monitoring and Failure Plan Proof

This subsection proves the correctness of the failure plan and topology connectivity monitoring.
Let T represents the topology.
FN Represents the set of *N* number of fog nodes in the topology.
fxi is a particular fog node *i* such that
fxi $\in \{fx1, fx2, fx3, .., fxN\}$

Fully connected (Fc) Represents the topology when all its branches are connected.
Partially disconnected (Pc) Represent the topology when any fog node fails resulting on a branch disconnection.
Virtually partially disconnected (Vpd) Represent the topology when any previously failed node reconnects, this is a temporally state being made to pause sending messages to dependent entities and do the reassignment to the connected fog node so that the original path of the branch is restored.

The following scenarios are realised for the failure plan:

- Scenario (1): A fog node fails, and there is no any neighbouring fog node in the same layer and same location.
  - The system topology T is working in the normal state; Fc. Then a fog node *fxi* fails. The connectivity monitoring discovers such node failure

and starts the failure processing. At this moment, the topology has lost one branch; hence the topology is in Pd due to *fxi* failure. The dependent entities (children fog nodes, gateways, and sensors and actuators) on *fxi* are identified, then sending messages to such entities is temporally paused. Then topology will be searched for the nearest neighbouring fog node in the same layer and same location of *fxi*; there is no such node. Hence the dependent entities will be assigned to the parent node of *fxi*. The topology is then updated to reflect the new assignment and reconnect the branch, changing the topology states to Fc. The paused messages are now resumed.

- Scenario (2): A fog node fails, and there is a neighbouring fog node in the same layer and same location.
  - The system topology T is working in the normal state; Fc. Then a fog node *fxj* fails. The connectivity monitoring discovers such node failure and starts the failure processing. At this moment, the topology has lost one branch; hence the topology is in Pd due to *fxj* failure. The dependent entities on *fxj* are identified, then sending messages to such entities is temporally paused. Then topology will be searched for the nearest neighbouring fog node in the same layer and same location of *fxj*; such node is found say *fxj+1*. Hence the dependent entities will be assigned to *fxj+1*. The topology is then updated to reflect the new assignment and reconnect the branch, changing the topology states to Fc. The paused messages are now resumed.

- Scenario (3): A previously failed fog node reconnected.
  - The system topology T is working in the normal state; Fc. Then a previously failed fog node *fxi* reconnects. The connectivity monitoring discovers such node reconnection and starts restoring the original path of *fxi* branch. *fxi* depending entities are identified and sending messages to such entities is temporally paused, making the topology to be in Vpd while the original path is restored. The topology is then updated to reflect the new assignment and reconnect the branch, changing the topology states to Fc. The paused messages are now resumed.

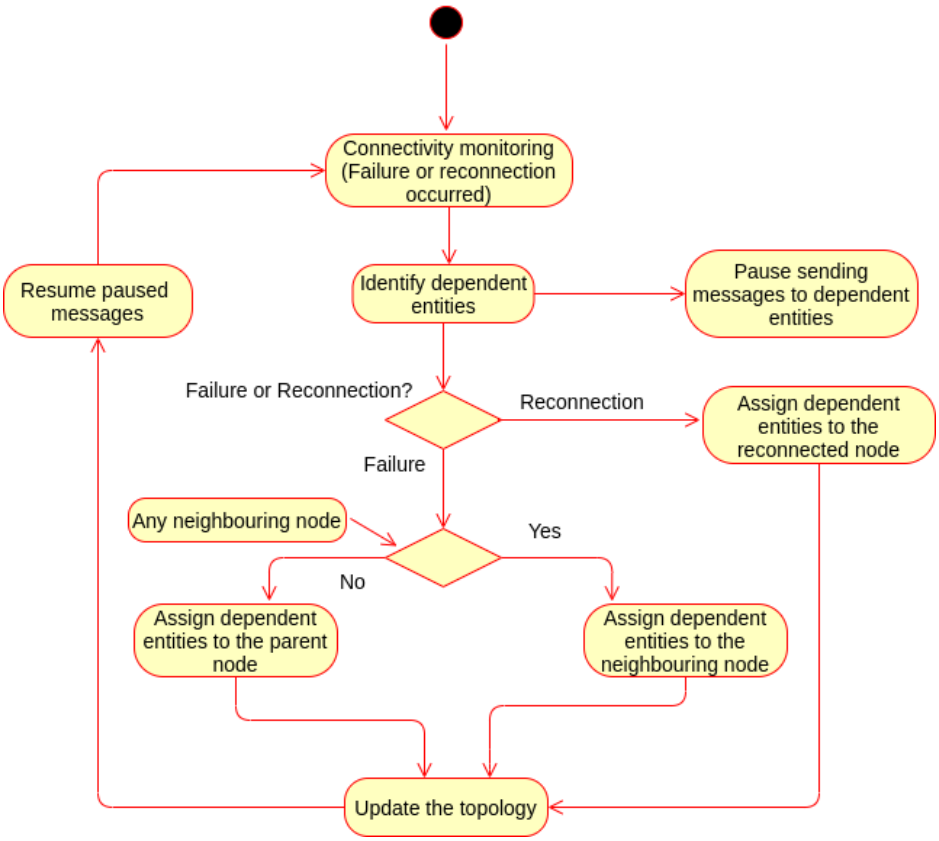

Figure 5.6: Behaviour of Connectivity Monitoring and Failure Plan

The above scenarios show that failure plan always succeeds in reconnecting the affected branch due to fog node failure, and the final objective is always reached. Figure 5.6 describes the behaviour of the proposed failure plan.

State Transition

The state of the topology is said to be fully connected if all its branches are connected, a normal state. Failure of any fog node makes the topology enters the partially disconnected state. Reconnection of any previously disconnected fog node makes the topology temporally enters virtually partially disconnected state. Figure 5.7 shows the state machine of the proposed failure plan.

The state machine M of the topology is represented as a pentuple
M = (Q, Σ, δ ,q_0, F)

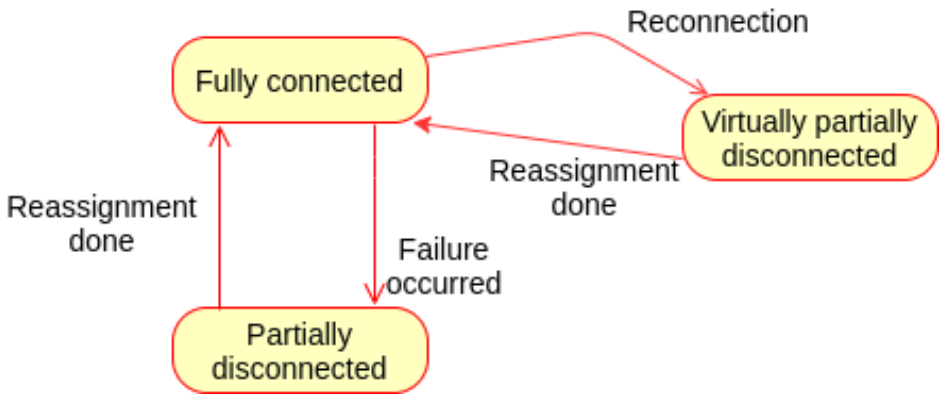

Figure 5.7: State Machine of Connectivity Monitoring and Failure Plan

Where Q Represents set of sates.
Σ represents the set of inputs needed for transitions.
δ represents the transition function.
q_0 represents the initial state.
F represents the final state.

Q = { Fully connected, Partially disconnected, Virtually partially disconnected}.
Σ = {Failure occurred (Fo), Reconnection (R), Reassignment done (Rd)}.
q_0 = {Fully connected}.
F = {Fully connected}.

The transitions of the state machines are defined as:
a. δ (Fully connected, Fo) = Partially disconnected.
b. δ (Partially disconnected, Rd) = Fully connected.
c. δ (Fully connected, R) = Virtually partially disconnected.
d. δ (Virtually partially disconnected, Rd) = Fully connected.

The following cases describe the correctness of the connectivity monitoring and failure plan of the architecture topology.

- Case (1): A fog node fails and dependent entities will be assigned to other fog node.

- δ (Fully connected, Fo) = Partially disconnected.
- δ (Partially disconnected, Rd) = Fully connected $\in$ F, hence accepted.

The topology T state is changed from Fully connected to Partially disconnected (Failure occurred). When reassignment is successfully done (Rd), it enters into the Fully connected state which is the accepted final state in the state machine M. The state transitions are same whether the assignment will be for a neighbouring node or to the parent node of the failed one; hence this case is applied for both scenarios.

- Case (2): A previously failed node reconnected.
  - δ (Fully connected, R) = Virtually partially disconnected.
  - δ (Virtually partially disconnected, Rd) = Fully connected $\in$ F, hence accepted.

The topology T state is changed from Fully connected to Virtually partially disconnected (Reconnection). When reassignment is successfully done (Rd) to restore the original branch path, it enters into the Fully connected state which is again the accepted final state in the state machine M.

5.5.3 SCM Monitoring and Offloading Proof

As explained earlier, SCM is assigned by the cloud to the lowermost fog node connected to the gateway(s) from which the end user Ss/As are connected to. However, this monitoring node can offload the monitoring to its parent if it is overloaded to reduce the actuation commands latency. Later, if the fog node is not overloaded, it can retrieve the monitoring offloaded to its parent.
The following scenarios are realised for the SCM monitoring and offloading, as depicted in figure 5.8:

- Scenario (1): A fog node is not overload; hence it will monitor the SCM locally as assigned by the cloud.

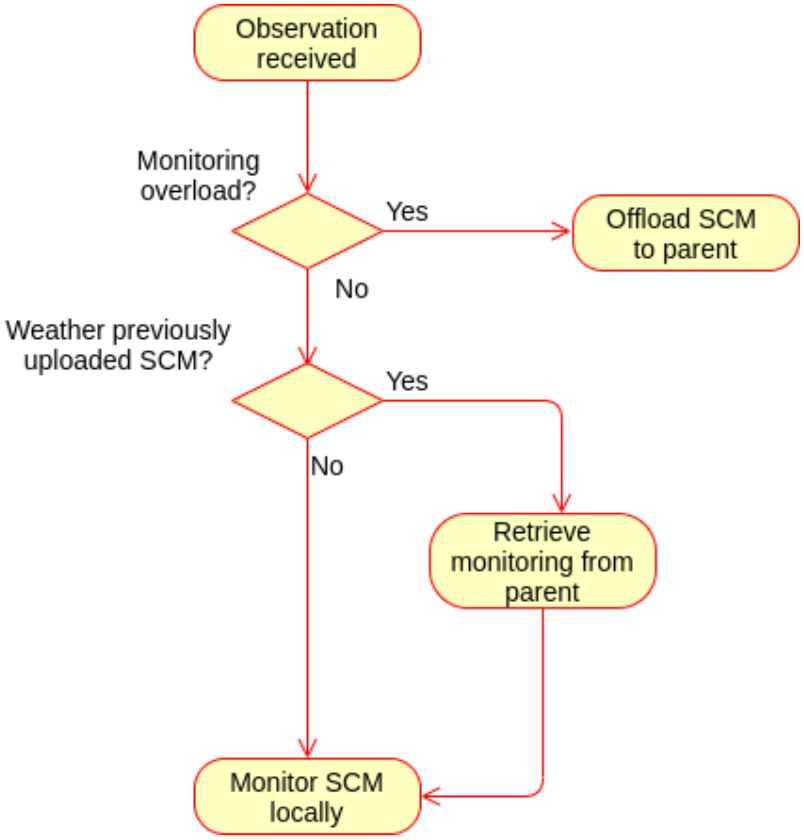

Figure 5.8: Behaviour of SCM Monitoring and Offloading

  - For any observation sent to SCM, the specified cases manager will check if the node monitoring load is equal to or more than the threshold. If not, the node will continue monitoring the specified cases assigned to it locally. Also, if the node is not overloaded, it checks whether it has previously offloaded the monitoring to its parent. If so, it will retrieve monitoring from parent and continue monitoring locally.

Note that the threshold may be different from node to node depending on the node processing capability.

- Scenario (2): Fog node is overloaded.
  - If the specified cases manager receives observation during which the node is overloaded (node monitoring load is equal to or more than the

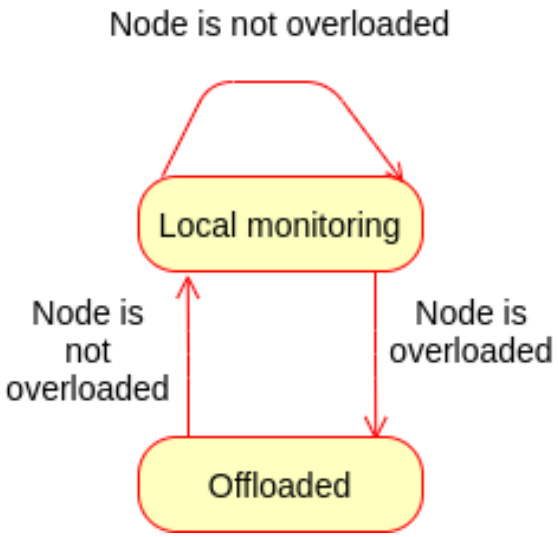

Figure 5.9: State Machine of SCM Monitoring and Offloading

threshold), it will offload the monitoring to the parent fog node.

State Transition

The state of SCM monitoring can be either local monitoring or offloaded; this is governed by the threshold of the node monitoring load. Figure 5.9 shows the state machine of the proposed SCM monitoring and offloading.

The state machine M of SCM monitoring is represented as a pentuple
M = (Q, Σ, δ ,q_0, F)

Where Q Represents set of sates.
Σ represents the set of inputs needed for transitions.
δ represents the transition function.
q_0 represents the initial state.
F represents the final state.

Q = { Local monitoring, Offloaded}.
Σ = {Node is not overloaded (Nno), Node is overloaded (No)}.
q_0 = {Local monitoring}.

F = {Local monitoring, Offloaded}.

The transitions of the state machines are defined as:
a. δ (Local monitoring, Nno) = Local monitoring.
b. δ ((Local monitoring, No) = Offloaded.
c. δ (Offloaded, Nno) = Local monitoring.

The following cases describe the correctness of the specified cases manager for monitoring and offloading.

- Case (1): The node monitoring load is less than the threshold; the node is not overloaded.
  - δ (Local monitoring, Nno) = Local monitoring.

The state of monitoring will continue to Local monitoring as long as the monitoring load is less that the threshold. This is an accepted state; $\in$ F.

- Case (2): The monitoring load goes equal or more that the threshold. Therefore the manager will offload the monitoring to the parent node.
  - δ (Local monitoring, No) = Offloaded.

As the node is overloaded, the monitoring will enter an Offloaded state which goes on as long as the node is load overloaded. This state is also accepted; $\in$ F. This way the monitoring will continue even at the parent node.

- Case(3): The manager has previously offloaded monitoring, but now the node is not overload.
  - δ (Offloaded, Nno) = Local monitoring.

Once the node is not overloaded, the manager will retrieve the monitoring from the parent node as assigned by the cloud. This is again the same accepted state; $\in$ F.

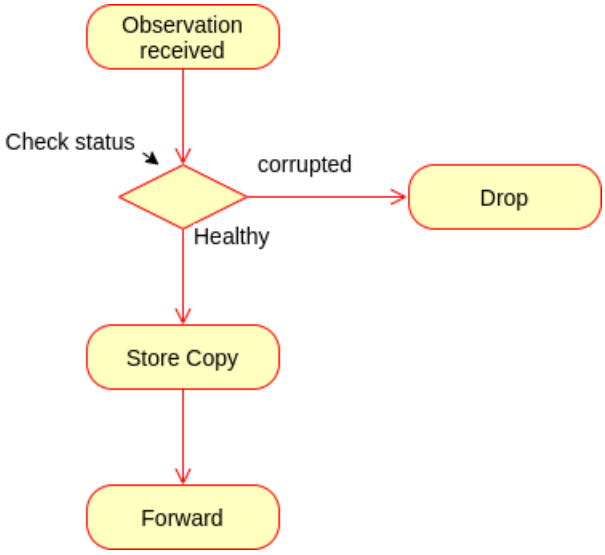

Figure 5.10: Behaviour of Observation Filtering

5.5.4 Observation Filtering Proof

The observation received to MQTT will be filtered, and only the healthy once will be farther forwarded upward. Figure 5.10 describes the behaviour of the proposed observation filtering.
The following scenarios are realised for the observation filtering.

- Scenario (1): Received observation is corrupted.
  - Upon receiving any observation, MQTT translator checks its state. If it is corrupted, it will be dropped.
- Scenario (2): If the received observation is not corrupted, a copy will be stored in the fog observation database, and a copy will be forwarded upward.

State Transition
The state of observation and the transitions of states are depicted in figure 5.11.
The state machine M of observation filtering is represented as a pentuple
M = (Q, Σ, δ ,q_0, F)

Where Q Represents set of sates.
Σ represents the set of inputs needed for transitions.

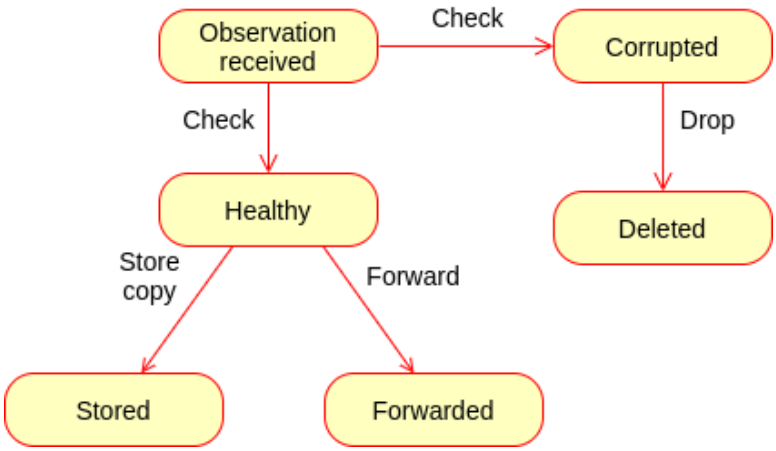

Figure 5.11: State Machine of Observation Filtering

δ represents the transition function.
q_0 represents the initial state.
F represents the final state.

Q = { Local monitoring, Offloaded}.
Σ = {Observation received , Corrupted, Healthy, Stored, Forwarded, Deleted}.
q_0 = {Observation received}.
F = {Stored, Forwarded, Deleted}.

The transitions of the state machines are defined as:
a. δ (Observation received, Ck) = Healthy.
b. δ (Observation received, Ck) = Corrupted.
c. δ (Healthy, Sc) = Stored.
d. δ (Healthy, F) = Forwarded.
e. δ (Corrupted, D) = Deleted.

The following cases describe the correctness of observation filtering .

- Case (1): The received observation is corrupted.
  - δ (Observation received, Ck) = Corrupted.
  - δ (Corrupted, D) = Deleted.

If the received observation is found to be corrupted, it moves into a Corrupted state result of checking. Then it ends in the Deleted state as a final accepted state; $\in$ F.

- Case (2): The received observation is healthy.
  - δ (Observation received, Ck) = Healthy.
  - δ (Healthy, Sc) = Stored.
  - δ (Healthy, F) = Forwarded.

Upon receiving, the observation enters into received sate, as a result of checking it enters into Healthy sate. The Healthy observation must be forwarded to the parent node, and a copy must be stored in the local observation database. Both Stored (locally) and Forwarded (upward) states are accepted as final states of the healthy observation; $\in$ F.

5.6 Summary

The entire proposed architecture is detailed in this chapter. The topology and components of each layer, the communication among nodes, and internal communication among nodes submodules are thoroughly discussed. Besides, the performance metrics governed the architecture design, the main contributions and the way they are expected to enhance the performance of SAaaS are also detailed. Finally, formal verification of the architecture correctness is presented. Experiments methodology and the simulation results are discussed in the following chapter.

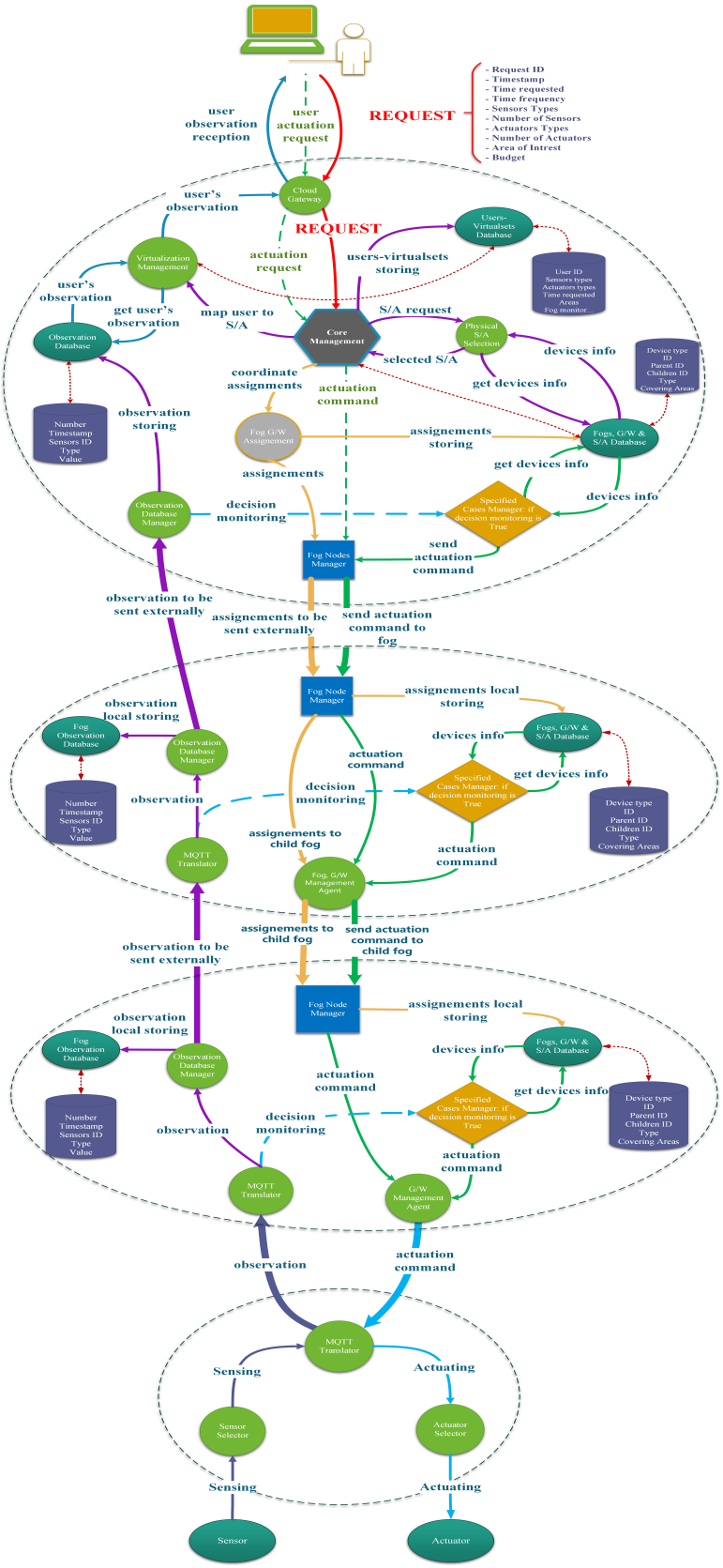

Figure 5.12: Communication among the Cloud, Parent Fog, Child Fog, and Gateway

Chapter 6

Experiment Results and Discussion

To evaluate the contributions in SALFSD, the architecture was simulated using YAFS (Yet Another Fog Simulator)[57]. Several simulations types are conducted to highlight the different contributions like failure plan, observation filtering, monitoring end users' specified cases, and offloading specified cases. This chapter explains each simulation type, modes of each type, compares and discusses the results.

6.1 YAFS

YAFS is a discrete event simulator very similar to iFogSim yet more flexible. It provides powerful tools to easily design fog computing applications, implementing routing strategies, and also permit a dynamic resources allocation and topology management, which are very useful to implement failure plan scenarios. YAFS is built using two main Python libraries. 1) Simpy for discrete event simulation, it represents the core of the simulator which control the different process of the simulation such as the tasks generation, messages transmission and execution. 2) NetworkX is the graph theory library integrated into YAFS to define system architecture in which nodes represent the topology device such as sensors, actuators, fog devices, and the cloud, along with edges which represent the links among system devices.

6.2 Experiment

Before explaining the experiments, the general simulation setup parameters for all experiments are specified in table 6.1, and the equations used to calculate latencies with their abbreviations are listed in table 6.2.

Table 6.1: General Simulation Setup Parameters

Parameter	**Value**
Cloud CPU	16 Ghz
Fog CPU	2 to 6 Ghz
Observation/Actuation message instructions	1 Million Instructions
Observation/Actuation message size	10 Kbyte
Bandwidth	2 to 6 Mbyte

Table 6.2: Abbreviations Used in the Equations

Abbreviation	**Definition**
m	Message
n	Computing node
LCP	Computing latency
LCN	Communication latency

Computing latency (LCP):

$$LCP_m = \frac{instructions_m}{CPU_n} \tag{6.1}$$

LCP actuation messages:

$$LCP_{actuation} = \sum_{0}^{i} LCP_m(actuation) \tag{6.2}$$

LCP observation messages:

$$LCP_{observation} = \sum_{0}^{i} LCP_m(observation) \tag{6.3}$$

Communication latency (LCN):

$$LCN_m = \frac{size_m}{bandwidth} \tag{6.4}$$

LCN actuation messages:

$$LCN_{actuation} = \sum_{0}^{i} LCN_m(actuation) \tag{6.5}$$

LCN observation messages:

$$LCN_{observation} = \sum_{0}^{i} LCN_m(observation) \tag{6.6}$$

The different types of experiments, their configuration and results are discussed below.

6.2.1 Failure Plan

As every fog node is known to be a point of failure, failure plan in SALFSD is to monitor the failure of any fog node and reassign its tasks and responsibilities to the nearest fog node in the same level or its parent fog node. Here we evaluate the benefit of having reassignment by comparing average sent and received observation and actuation commands for the following modes of tests with regards to failure plan:

1. Mode 1: Simulation runs without failing any fog node.
2. Mode 2: Ten fog nodes are randomly failed during simulation without reassignment; reassignment is disabled.
3. Mode 3: Ten fog nodes are randomly failed during simulation and their tasks are reassigned to other fog nodes.

The simulation configuration parameters of failure plan are specified in table 6.3.

Table 6.3: Configuration Parameters for Failure Plan Modes

Parameter	**Value**
General Parameters for All Tests	
Number of Fog Layers	3
Number of Fog Nodes	39
Simulation Time	500
Failure Mode	1,2, and 3
Number of Failed Fog Nodes	10, for modes 2 and 3 only
Actuation Mode	Both End User and SCM
Monitoring Mode	No Offloading
Corrupted Messages Mode	No Corrupted Messages
Number of Areas Per User	Random
Parameters for 100 Users Tests	
Number of Gateways	80
Number of Sensors	237
Number of Actuators	240
Parameters for 200 Users Tests	
Number of Gateways	91
Number of Sensors	265
Number of Actuators	251
Parameters for 300 Users Tests	
Number of Gateways	74
Number of Sensors	234
Number of Actuators	224
Parameters for 400 Users Tests	
Number of Gateways	79
Number of Sensors	243
Number of Actuators	230
Parameters for 1000 Users Tests	
Number of Gateways	73
Number of Sensors	201
Number of Actuators	218

Figure 6.1 presents the results of average observation generated by sensors, average observation successfully received by cloud, average actuation requests generated, and

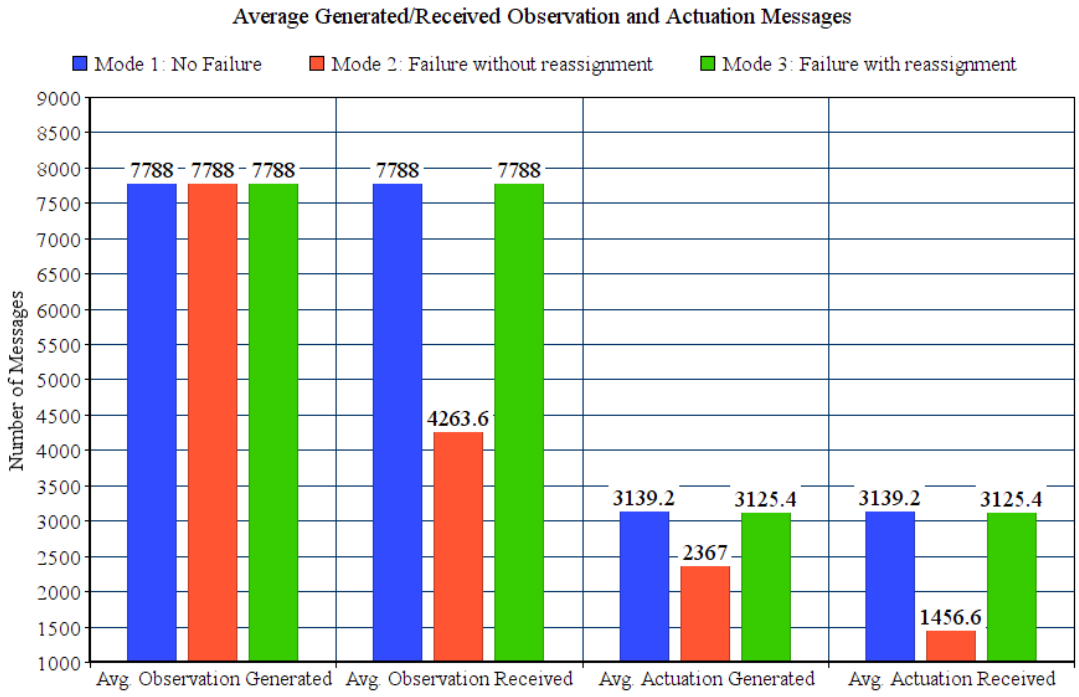

Figure 6.1: Packets loss due to Failed Fog Nodes

average actuation successfully executed; the average is for five tests (i.e. 100, 200, 300, 400, and 1000 users). Results show that mode 3 -reassignment- avoids loss of observations and actuation requests comparing with mode 2 - no reassignment. In case of no reassignment, the network topology remains disconnected; could have many disconnection points depending on the number of failed fog nodes hence leading to a vast amount of missing messages.

6.2.2 Actuation Mode

In SAaaS, typically end user gets the observations, analyses them and then decide the proper actuation; if any. However, as explained in 5.1.1, SALFSD deploys users' SCM in fog nodes as well as the cloud. Here we evaluate the benefit of having SCM by comparing actuation communication latencies for the following modes of tests:

1. Mode 1: Actuation request is generated by end users only.

2. Mode2: Actuation request is generated by SCM only.

3. Mode3: Both end users and SCM generate actuation requests.

The simulation configuration parameters of actuation modes are specified in table 6.4.

Table 6.4: Configuration Parameters for Actuation Modes

Parameter	**Value**
General Parameters for All Tests	
Number of Fog Layers	3
Number of Fog Nodes	39
Simulation Time	500
Failure Mode	Failure With Reassignment
Number of Failed Fog Nodes	10
Actuation Mode	1, 2 and 3
Monitoring Mode	No Offloading
Corrupted Messages Mode	No Corrupted Messages
Number of Areas Per User	Random
Parameters for 100 Users Tests	
Number of Gateways	79
Number of Sensors	395
Number of Actuators	226
Parameters for 200 Users Tests	
Number of Gateways	70
Number of Sensors	350
Number of Actuators	195
Parameters for 300 Users Tests	
Number of Gateways	90
Number of Sensors	450
Number of Actuators	257
Parameters for 400 Users Tests	
Number of Gateways	70
Number of Sensors	350
Number of Actuators	217
Parameters for 1000 Users Tests	
Number of Gateways	73
Number of Sensors	750
Number of Actuators	237

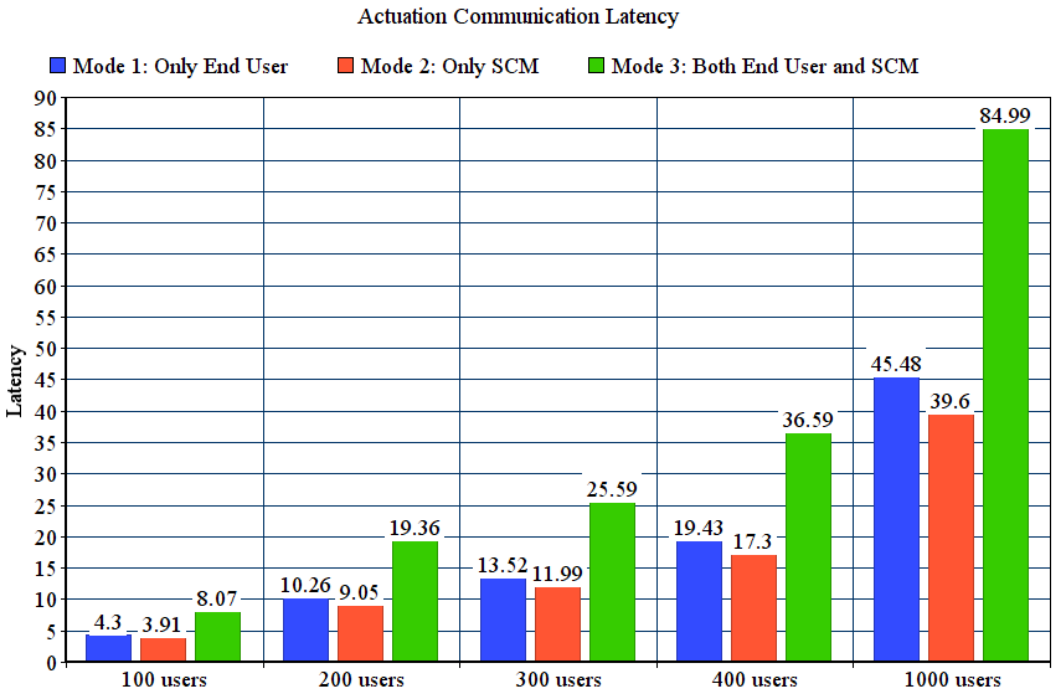

Figure 6.2: Actuation Communication Latency for Actuation Modes

Figure 6.2 shows actuation communication latency. In mode 1 and 3, the observations have to travel to the cloud then to the end user. Then the end user will analyse them and decide the proper actuation to be taken and sends back the actuation command; this takes longer time compared to mode 2. Mode 2 always results in less communication latency, which makes actuation commands generated by SCM in several network points fits time-sensitive applications.

6.2.3 Monitoring Offloading

As explained in 5.3.2, SCM may be offloaded to parent fog node in case of the fog node responsible for monitoring is overloaded. To evaluate the benefit of offloading SCM we conducted and compared actuation computing latencies for two test modes:

1. Mode 1: No offloading; Fog node will keep monitoring even if it is overloaded.
2. Mode 2: Overloaded fog node offloads SCM to its parent.

Note that the threshold may be different from node to node depending on the node processing capability, hence threshold has been set randomly to simulate such differences among fog nodes.

The simulation configuration parameters of monitoring offloading modes are specified in table 6.5.

Table 6.5: Configuration Parameters for SCM Monitoring Offloading Modes

Parameter	**Value**
General Parameters for All Tests	
Number of Fog Layers	3
Number of Fog Nodes	39
Simulation Time	500
Failure Mode	Failure With Reassignment
Number of Failed Fog Nodes	10
Actuation Mode	Both End User and SCM
Monitoring Mode	No Offloading
Corrupted Messages Mode	No Corrupted Messages
Number of Users	100
Number of Areas Per User	Random
Parameters for 395 Sensors Tests	
Number of Gateways	79
Number of Actuators	226
Parameters for 648 Sensors Tests	
Number of Gateways	81
Number of Actuators	254
Parameters for 783 Sensors Tests	
Number of Gateways	77
Number of Actuators	230
Parameters for 875 Sensors Tests	
Number of Gateways	87
Number of Actuators	255

Figure 6.2 shows that offloading reduces the actuation computing latency in all conducted tests. Without offloading, monitoring observations is delayed as the node is

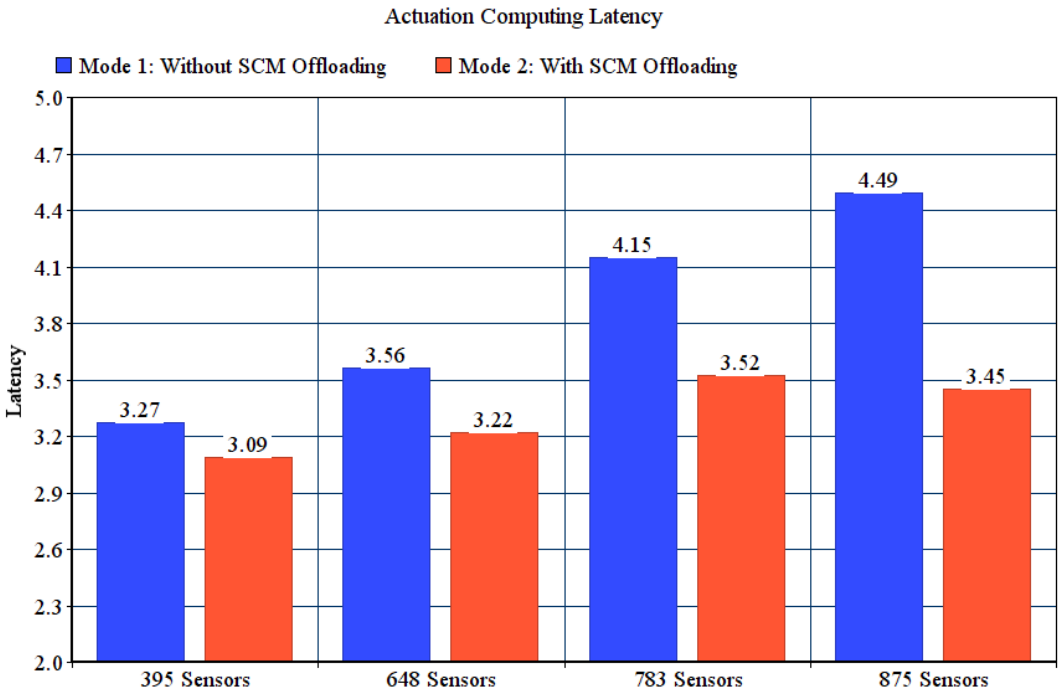

Figure 6.3: Actuation Computing Latency for SCM Monitoring Offloading Modes

overloaded, and hence the generation of actuation command.

6.2.4 Corrupted Observation Filtering

As explained in 5.3.4, MQTT translator filters all incoming observations and drops the corrupted once; if any. To evaluate the benefit of such Filtering, we have compared total observation latencies (communication latency + computing latency) for three test modes:

1. Mode 1: No corrupted observations are generated.
2. Mode 2: Corrupted observations are generated and forwarded; No filtering and all incoming observations are forwarded.
3. Mode 3: MQTT translator filters and drops the corrupted observations. Only

uncorrupted observations are forwarded.

The simulation configuration parameters of observation filtering modes are listed in table 6.6.

Table 6.6: Configuration Parameters for Observation Filtering Modes

Parameter	**Value**
General Parameters for All Tests	
Number of Fog Layers	3
Number of Fog Nodes	39
Simulation Time	500
Failure Mode	Failure With Reassignment
Number of Gateways	81
Number of Failed Fog Nodes	10
Actuation Mode	Both End User and SCM
Monitoring Mode	No Offloading
Corrupted Messages Mode	1, 2 and 3
Number of Areas Per User	Random
Parameters for 100 Users Tests	
Number of Sensors	243
Number of Actuators	234
Parameters for 200 Users Tests	
Number of Sensors	246
Number of Actuators	243
Parameters for 300 Users Tests	
Number of Sensors	255
Number of Actuators	254
Parameters for 400 Users Tests	
Number of Sensors	242
Number of Actuators	254
Parameters for 1000 Users Tests	
Number of Sensors	222
Number of Actuators	242

Figure 6.4 shows that corrupted observation filtering in mode 3 results in the same total observation latency as mode 1 -where corrupted messages are not generated- in

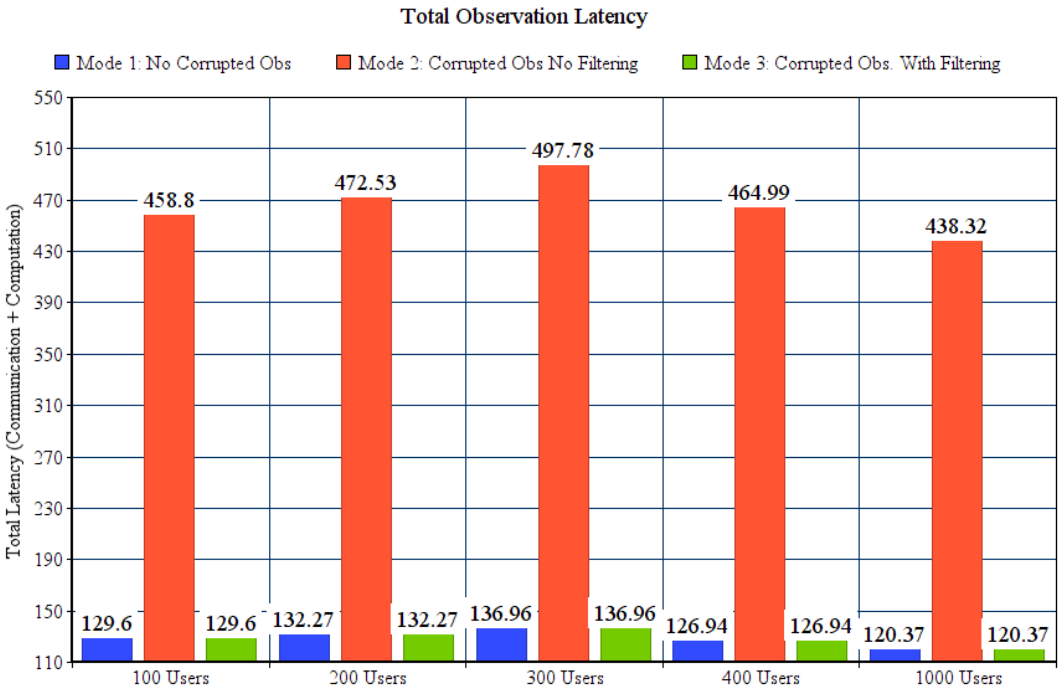

Figure 6.4: Total Observation Latency Corrupted Messages Modes

all conducted tests. The result also shows the increased total observation latency due to forwarding the corrupted observations in mode 2.

6.2.5 Comparing SAaaS with and without Layered Fog

To present the value added benefits of layered fogs in SAaaS combined with the cloud, we have compared total observation latency (communication latency + computing latency), actuation computation, and communication latencies for two test modes:

1. Mode1: SALFSD with layered fogs. Here only SCM is activated and remaining FDLFSD contributions are disabled; No offloading, no corrupted observations are generated, and no failed fog nodes.

2. Mode 2: SALFSD without layered fogs; Only the cloud and gateways with S/A. In addition, to keep the same distance between the gateways and the cloud, we

have placed routers nodes (instead of fog nodes for the same number of layers in mode 1; i.e. 3 layers.)

The simulation configuration parameters are listed in table 6.7.

Table 6.7: Configuration Parameters for Layered Fogs Modes

Parameter	**Value**
General Parameters for All Tests	
Number of Fog Layers	3
Number of Fog Nodes	39
Simulation Time	500
Failure Mode	No Failure
Actuation Mode	Both End User and SCM
Monitoring Mode	No Offloading
Corrupted Messages Mode	No Corrupted Messages
Number of Areas Per User	Random
Parameters for 100 Users Tests	
Number of Gateways	80
Number of Sensors	237
Number of Actuators	240
Parameters for 200 Users Tests	
Number of Gateways	91
Number of Sensors	265
Number of Actuators	251
Parameters for 300 Users Tests	
Number of Gateways	74
Number of Sensors	234
Number of Actuators	224
Parameters for 400 Users Tests	
Number of Gateways	79
Number of Sensors	243
Number of Actuators	230
Parameters for 1000 Users Tests	
Number of Gateways	73
Number of Sensors	201
Number of Actuators	218

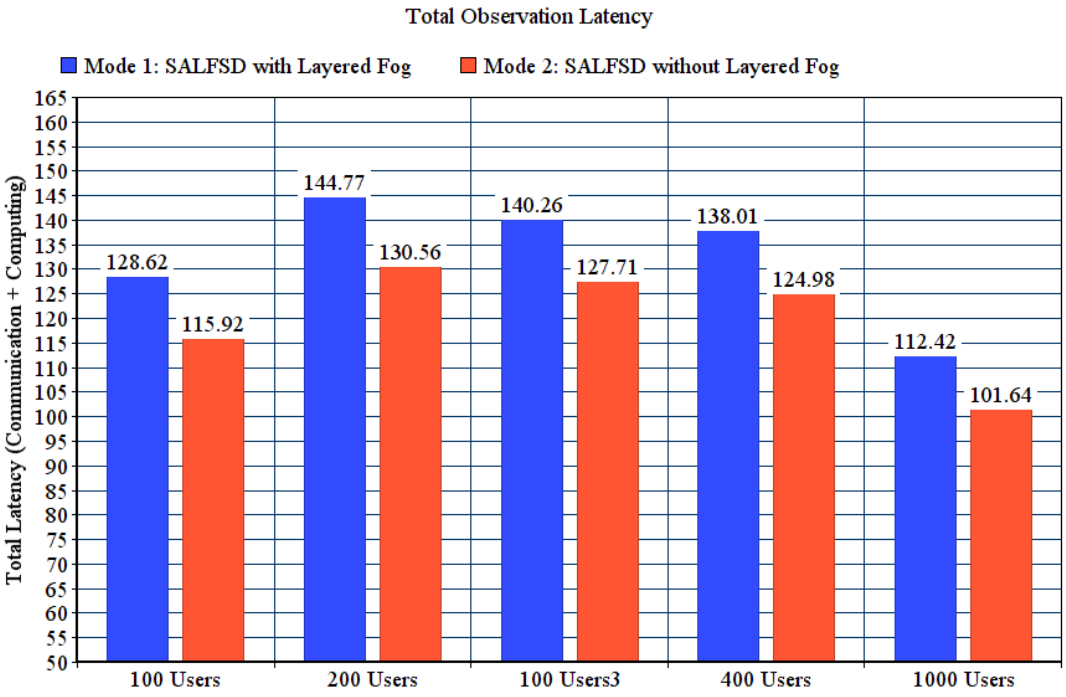

Figure 6.5: Total Observation Latency, SAaaS with and without Layered Fog

Figure 6.5 shows total observation latency for both modes. Despite having the same distance between the cloud and gateways witch makes the same communication latency, the reason that makes mode 1 results in increased total observation latency is having more computing in the layered fog; Fog nodes process the observation for the sake of SCM - consequently adding more observation computing latency. The same is also applied to actuation computing latency presented in figure 6.6, which is considered weak point in SALFSD. However, the trade-off is that mode 1 results in less actuation communication latency, and hence actuation commands can be triggered faster than having to be from end user side; as presented figure 6.7.

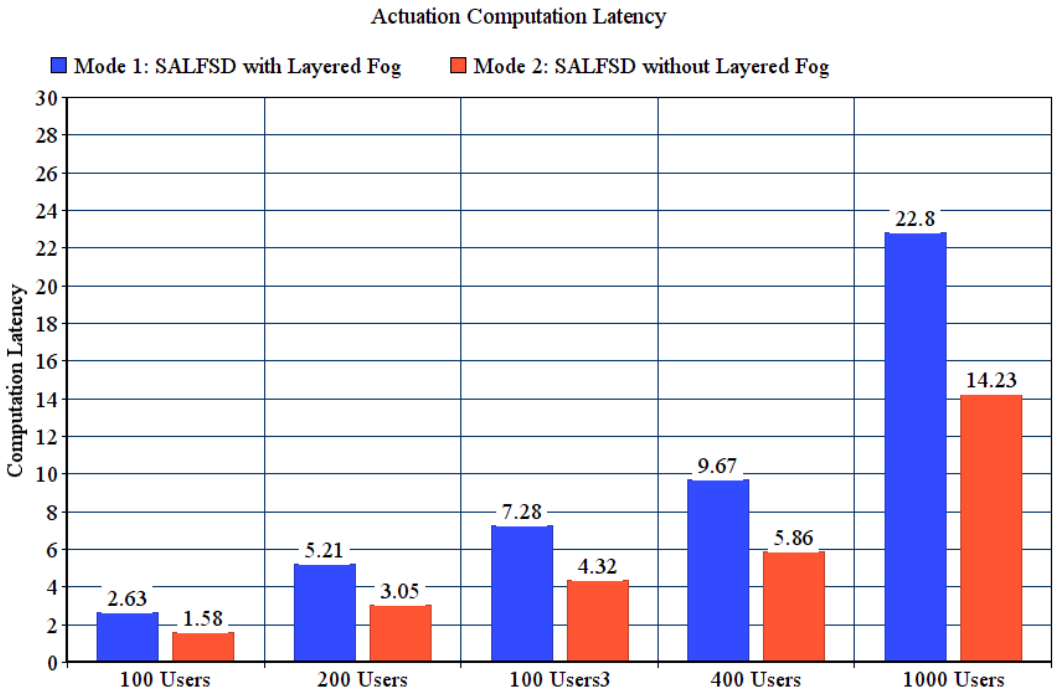

Figure 6.6: Actuation Computation Latency, SAaaS with and without Layered Fog

6.3 General Discussion on the Results

The added layered fog to SAaaS basic design and the proposed features implemented inside fog nodes and the cloud has been simulated and evaluated in terms of the following metrics: 1) total delivered messages, 2) computing and communication latencies for observation and actuation messages. The simulation was conducted in the same environment with different scenarios regarding the number of end users, number of gateways, number of sensors and actuators in IoT layer, and number of areas per end user from which their sensors and actuators are selected.

The overall study of the results shows that besides keeping the infrastructure connected, failure reassignment prevents loss of observation and actuation messages, consequently resulting in 100 per cent messages delivery.

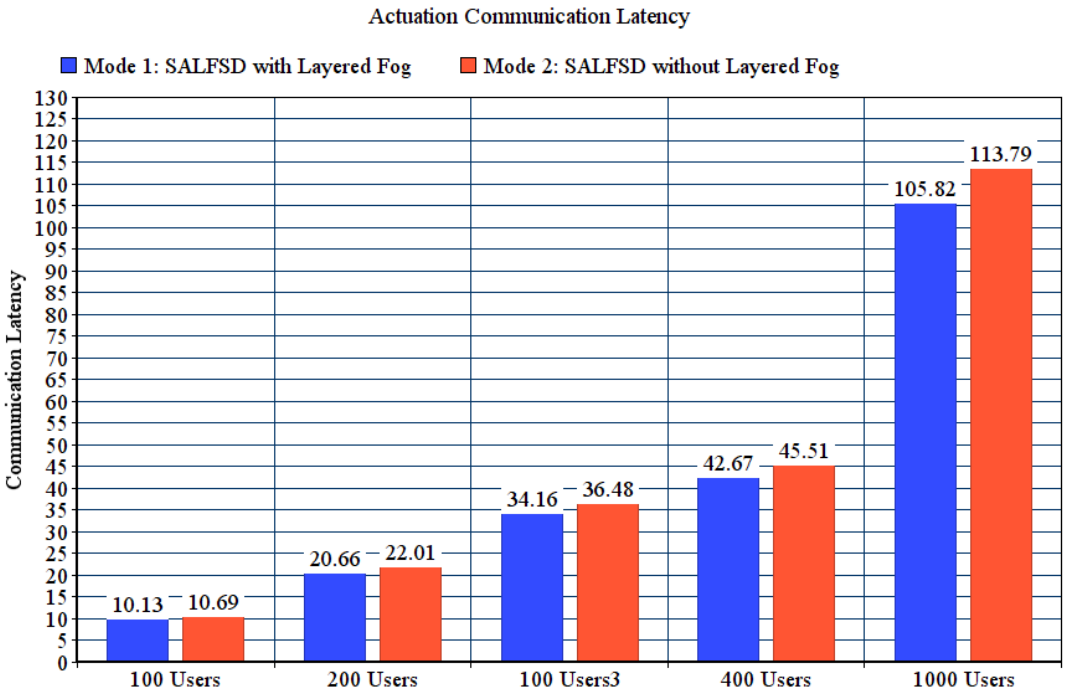

Figure 6.7: Actuation Communication Latency, SAaaS with and without Layered Fog

Also, results show that deploying end users' SCM in the cloud and fog nodes generally decreases the actuation latency as the decision of triggering the actuation command is being taken closer to IoT layer. Furthermore, SCM offloading also prevents increased actuation latency in case fog nodes are overloaded due to receiving a vast amount of observation to be monitored (an IoT associated issue; i.e. Big Data [56]). Such a case is expected with the increased number of IoT devices in SAaaS paradigm. This is particularly significant in time-sensitive applications.

In addition, dropping the corrupted observations reduces the unnecessary consumption of network bandwidth. Corrupted observation may be sent from sensors, or being effected while forwarding by fog node. The former is dropped by the fog node connected to the gateway in the lower fog layer. The later is dropped by parent fog node in upper fog layers.

Comparing SAaaS with and without layered fog presented less actuation communication latency with layered fog. Adding this point to the benefits of features implemented inside fog nodes, totally shows the performance enhancement provided by SALFSD to the basic SAaaS architecture.

6.4 Summary

Adding scalable and fault resistant layered fog architecture in SAaaS and wisely selecting tasks to be handled by fog nodes has improved the performance of SAaaS. Also, the failure plan suggested in SALFSD provides a persistent availability of the services provides network and hence to its promised services. SALFSD has proved itself in the evaluation results to be a piece of valuable advice for designing SAaaS architectures for IoT applications.

Chapter 7

Conclusions and Future Work

The primary and broad focus of this thesis is to attempt to come out with architectures for services composition in IoT toward a vision of the future smart city. These architectures are mainly concerned with sensing and actuating as a service. To this end, chapter 2 of the thesis presents a related work of several types of sensing systems and paradigms were discussed: participatory sensing, opportunistic sensing, urban sensing, and People-Centric Urban Sensing, etc. The same was preceded with briefly detailing the primary building blocks of IoT, the existing technologies in which such systems strongly depend on. Chapter 2 also presents the essential requirements of IoT systems that any architecture aiming for IoT, mainly based on cloud and fog computing and their integration, must take into consideration during the design and implementation stages. Such design and implementation requirements emerge from IoT systems sophisticated and rigorous requirements. Besides, chapter 2 detailed the indispensable benefits added to IoT towards overcoming its limitations by cloud computing, fog computing, and their integration, as well as their challenges.

Chapter 3 details our vision of the future smart city service providers, along with a high level architecture proposal for integrating several independent providers that may offer heterogeneous services.

In chapter 4, we present our proposed Internet of Things Sensors and Actuators Layered Fog Service Delivery Model SALFSD. The architecture is designed with inbuilt failure plan to maintain the network topology connection against any fog

node(s) failure. Also, several features like end user prespecified cases monitoring, corrupted observation filtering, monitoring offloading are provided in SALFSD. A feature wise comparison is conducted against related work in literature for sensing and actuating as a service with respect to the use of topology, failure plan, virtualization, etc.

Chapter 5 extends the work in chapter 4, detailed with the added contributions. The entire SALFSD components and communication among them are explained thoroughly along with formal verification of the correctness of the proposed architecture. The added layered fog to SAaaS basic architecture design (cloud and IoT layers), and the proposed features implemented inside fog nodes and the cloud has been simulated and evaluated in chapter 6 in terms of the following metrics: 1) total delivered messages, 2) computing and communication latencies for observation and actuation messages. The simulation is conducted for each feature in SALFSD and also a comparison of the architecture, including and excluding the layered fog is conducted. The results depict the latency reduction with the presence of layered fog in the architecture as well as while including each feature. Besides, results show that the failure plan prevents network disconnection and packets drop. Further, we have implemented a bio-optimization algorithm in SALFSD for sensors and actuators selection base on minimized cost and maximized feedback.

As future work, we are planning to enhance the current work by implementing the reassignment in fog nodes also beside the cloud to make a distributed failure plan; meaning that any parent fog node can issue the reassignment to any child node without having to wait for the same to be received from the cloud. Presently in SALFSD there is no communication among fog nodes in the same layer, this is also planned for future work to allow collaboration in the same layer. Besides, more work is required to improve the current architecture and implement many techniques for authentication, authorization, and both end users and devices management.

References

[1] MAZLAN ABBAS. **Sensing-as-a-Service: The New Internet of Things (IOT) Business Model**. In *MIMOS Berhad*, 2014. (12)

[2] S. ABDELWAHAB, B. HAMDAOUI, M. GUIZANI, AND A. RAYES. **Enabling Smart Cloud Services Through Remote Sensing: An Internet of Everything Enabler**. *IEEE Internet of Things Journal*, **1**(3):276–288, June 2014. (19)

[3] S. ABDELWAHAB, B. HAMDAOUI, M. GUIZANI, AND T. ZNATI. **Cloud of Things for Sensing-as-a-Service: Architecture, Algorithms, and Use Case**. *IEEE Internet of Things Journal*, **3**(6):1099–1112, Dec 2016. (19)

[4] MAYANK AGGARWAL, KUMAR NILAY, AND KULDEEP YADAV. **Survey of named data networks: future of internet**. *International Journal of Information Technology*, **9**(2):261–266, Jun 2017. (4, 10)

[5] ALA AL-FUQAHA, MOHSEN GUIZANI, MEHDI MOHAMMADI, MOHAMMED ALEDHARI, AND MOUSSA AYYASH. **Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications**. *IEEE Communications Surveys & Tutorials*, **17**(4):2347–2376, 2015. (3, 10)

[6] MOHAMMED AL-KHAFAJIY, THAR BAKER, ATIF WARAICH, DHIYA AL-JUMEILY, AND ABIR HUSSAIN. **IoT-Fog Optimal Workload via Fog Offloading**. *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 359–364, 2018. (25)

[7] M. AL YAMI AND D. SCHAEFER. **Fog Computing as a Complementary Approach to Cloud Computing**. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–5, 2019. (30)

[8] H. ALSHAREEF, M. ALMASRI, A. ALBESHER, AND D. GRIGORAS. **Towards an Effective Management of IoT by Integrating Cloud and Fog Computing**. In *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 197–204, 2019. (24)

[9] LUIGI ATZORI, ANTONIO IERA, AND GIACOMO MORABITO. **The Internet of Things: A Survey**. *Computer Networks*, pages 2787–2805, 10 2010. (xii, 1, 3)

[10] LYN BARTRAM, JOHNNY RODGERS, AND ROB WOODBURY. **Smart Homes or Smart Occupants? Supporting Aware Living in the Home**. In *Human-Computer Interaction – INTERACT 2011*, pages 52–64, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. (18)

[11] RABEEA BASIR, SAAD QAISAR, MUDASSAR ALI, MONTHER ALDWAIRI, MUHAMMAD ASHRAF, AAMIR MAHMOOD, AND MIKAEL GIDLUND. **Fog Computing Enabling Industrial Internet of Things: State-of-the-Art and Research Challenges**. *Sensors*, **19**:4807, 11 2019. (27, 30)

[12] PAOLO BELLAVISTA, JAVIER BERROCAL, ANTONIO CORRADI, SAJAL DAS, LUCA FOSCHINI, AND ALESSANDRO ZANNI. **A Survey on fog computing for the Internet of Things**. *Pervasive and Mobile Computing*, **52**, 12 2018. (23, 24, 26, 27, 28, 29, 30)

[13] F. BONOMI, R. MILITO, P. NATARAJAN, AND JIANG ZHU. **Fog Computing: A Platform for Internet of Things and Analytics**. In *Big Data and Internet of Things*, 2014. (25)

[14] 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, page 1316, New York, NY, USA, 2012. Association for Computing Machinery. (23, 24, 29)

[15] ALESSIO BOTTA, WALTER DE DONATO, VALERIO PERSICO, AND ANTONIO PESCAPÉ. **Integration of Cloud computing and Internet of Things: A survey**. *Future Gener. Comput. Syst.*, **56**(C):684–700, March 2016. (4, 16, 25)

[16] A. T. CAMPBELL, S. B. EISENMAN, N. D. LANE, E. MILUZZO, R. A. PETERSON, H. LU, X. ZHENG, M. MUSOLESI, K. FODOR, AND G. AHN. **The Rise of People-Centric Sensing**. *IEEE Internet Computing*, **12**(4):12–21, July 2008. (16)

[17] ANDREW T. CAMPBELL, SHANE B. EISENMAN, NICHOLAS D. LANE, EMILIANO MILUZZO, AND RONALD A. PETERSON. **People-Centric Urban Sensing**. In *Proceedings of the 2nd Annual International Workshop on Wireless Internet*, WICON 06, page 18es, New York, NY, USA, 2006. Association for Computing Machinery. (15)

[18] E. CERRITOS, F. J. LIN, AND D. DE LA BASTIDA. **High scalability for cloud-based IoT/M2M systems**. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016. (26)

[19] W. CHUNG, CHIEW-LIAN YAU, KWANG-SIG SHIN, AND RAILI MYLLYLA. **A Cell Phone Based Health Monitoring System with Self Analysis Processor using Wireless Sensor Network Technology**. *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3705–3708, 2007. (13)

[20] ANDREI CIORTEA, OLIVIER BOISSIER, ANTOINE ZIMMERMANN, AND ADINA MAGDA FLOREA. **Responsive Decentralized Composition of Service Mashups for the Internet of Things**. In *Proceedings of the 6th International Conference on the Internet of Things*, IoT'16, pages 53–61, New York, NY, USA, 2016. ACM. (1)

[21] CORY CORNELIUS, APU KAPADIA, DAVID KOTZ, DAN PEEBLES, MINHO SHIN, AND NIKOS TRIANDOPOULOS. **Anonysense: Privacy-Aware People-Centric Sensing**. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys 08, page 211224, New York, NY, USA, 2008. Association for Computing Machinery. (15)

[22] S. K. DATTA, C. BONNET, AND J. HAERRI. **Fog Computing architecture to enable consumer centric Internet of Things services**. In *2015 International Symposium on Consumer Electronics (ISCE)*, pages 1–2, 2015. (25)

[23] M. DE DONNO, K. TANGE, AND N. DRAGONI. **Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog**. *IEEE Access*, **7**:150936–150948, 2019. (5)

[24] D. DE LA BASTIDA AND F. J. LIN. **OpenStack-Based Highly Scalable IoT/M2M Platforms**. In *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 711–718, 2017. (26)

[25] KALYANMOY DEB, AMRIT PRATAP, SAMEER AGARWAL, AND TAMT MEYARIVAN. **A fast and elitist multiobjective genetic algorithm: NSGA-II**. *IEEE transactions on evolutionary computation*, **6**(2):182–197, 2002. (52)

[26] S. DISTEFANO, G. MERLINO, AND A. PULIAFITO. **Sensing and Actuation as a Service: A New Development for Clouds**. In *2012 IEEE 11th International Symposium on Network Computing and Applications*, pages 272–275, Aug 2012. (11, 20, 21, 48)

[27] SALVATORE DISTEFANO, GIOVANNI MERLINO, AND ANTONIO PULIAFITO. **A utility paradigm for IoT: The sensing Cloud**. *Pervasive and Mobile Computing*, **20**:127–144, 2015. (21, 48)

[28] SALVATORE DISTEFANO, GIOVANNI MERLINO, ANTONIO PULIAFITO, AND ALESSIO VECCHIO. **A hypervisor for infrastructure-enabled sensing Clouds**. *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 1362–1366, 2013. (21, 48)

[29] EFFREY GOLDMAN ET AL. **Participatory Sensing: A citizen-powered approach to illuminating the patterns that shape our world**. In *White Paper, Woodrow Wilson Int. Center of Scholars*, Washington, DC, USA, May 2008. JAG. (13)

[30] MOHAMED FIRDHOUS, OSMAN GHAZALI, AND SUHAIDI HASSAN. **Fog Computing: Will it be the Future of Cloud Computing?** In *Proceedings of the Third International Conference on Informatics & Applications*, 10 2014. (31)

[31] HANS W. GELLERSEN, ALBRECHT SCHMIDT, AND MICHAEL BEIGL. **Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts**. *Mobile Networks and Applications*, **7**(5):341–351, Oct 2002. (14)

[32] C. GOMEZ AND J. PARADELLS. **Wireless home automation networks: A survey of architectures and technologies**. *IEEE Communications Magazine*, **48**(6):92–101, June 2010. (18)

[33] M. A. GRAY. **Spheres: A Web Services Framework for Smartphone Sensing as a Service**. In *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, pages 19–28, Sep. 2015. (17)

[34] MARK ALLEN GRAY AND PHILIP NEWSAM SCHERER. **Web Services Framework for Wireless Sensor Networks**. *The Sixth International Conferences on Advanced Service Computing*, **20**:15–23, 2014. (18)

[35] JAYAVARDHANA GUBBI, RAJKUMAR BUYYA, SLAVEN MARUSIC, AND MARIMUTHU PALANISWAMI. **Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions**. *Future Generation Computer Systems*, **29**, 07 2012. (5, 10, 13)

[36] DOMINIQUE GUINARD. **Mashing Up Your Web-Enabled Home**. In FLORIAN DANIEL AND FEDERICO MICHELE FACCA, editors, *Current Trends in Web Engineering*, pages 442–446, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. (18)

[37] X. GUO, R. SINGH, T. ZHAO, AND Z. NIU. **An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems**. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7, 2016. (25)

[38] P. HABIBI, M. FARHOUDI, S. KAZEMIAN, S. KHORSANDI, AND A. LEON-GARCIA. **Fog Computing: A Comprehensive Architectural Survey**. *IEEE Access*, **8**:69105–69133, 2020. (5)

[39] SON N. HAN, IMRAN KHAN, GYU MYOUNG LEE, NOEL CRESPI, AND ROCH H. GLITHO. **Service composition for IP smart object using realtime**

Web protocols: Concept and research challenges. *Computer Standards & Interfaces*, **43**:79 – 90, 2016. (4)

[40] JOHN HICKS, NITHYA RAMANATHAN, DONNIE KIM, MOHAMAD MONIBI, JOSHUA SELSKY, MARK HANSEN, AND DEBORAH ESTRIN. **AndWellness: An Open Mobile System for Activity and Experience Sampling**. In *Wireless Health 2010*, New York, NY, USA, 2010. Association for Computing Machinery. (13)

[41] HUA-JUN HONG. **From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices**. *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 331–334, 2017. (30)

[42] PENGFEI HU, SAHRAOUI DHELIM, HUANSHENG NING, AND TIE QIU. **Survey on Fog Computing: Architecture, Key Technologies, Applications and Open Issues**. *Journal of Network and Computer Applications*, **98**, 09 2017. (27, 28, 29)

[43] TOSHIKI. ISO, NORIHIRO. KAWASAKI, AND SHOJI. KURAKAKE. **Personal Context Extractor with Multiple Sensor on a Cell Phone**. In *International Conference on Mobile and Wireless Communications Networks*, September 2005. (14)

[44] GIULIO JACUCCI, ANNA SPAGNOLLI, LUCIANO GAMBERINI, ALESSANDRO CHALAMBALAKIS, CHRISTOFFER BJÖRKSKOG, MASSIMO BERTONCINI, CARIN TORSTENSSON, AND PASQUALE MONTI. **Designing Effective Feedback of Electricity Consumption for Mobile User Interfaces**. *PsychNology Journal*, **7**:265–289, 2009. (13)

[45] M. JAHN, M. JENTSCH, C. R. PRAUSE, F. PRAMUDIANTO, A. AL-AKKAD, AND R. REINERS. **The Energy Aware Smart Home**. In *2010 5th International Conference on Future Information Technology*, pages 1–8, May 2010. (13)

[46] A. KAMILARIS AND A. PITSILLIDES. **Mobile Phone Computing and the Internet of Things: A Survey**. *IEEE Internet of Things Journal*, **3**(6):885–898, Dec 2016. (12)

[47] ANDREAS KAMILARIS, NICOLAS IANNARILLI, VLAD TRIFA, AND ANDREAS PITSILLIDES. **Bridging the Mobile Web and the Web of Things in Urban**

Environments. In *In Urban Internet of Things Workshop, at IOT 2010*, 2010. (13, 15)

[48] S. S. KANHERE. **Participatory Sensing: Crowdsourcing Data from Mobile Smartphones in Urban Spaces**. In *2011 IEEE 12th International Conference on Mobile Data Management*, **2**, pages 3–6, June 2011. (16)

[49] Y. KAWAHARA, H. KURASAWA, AND H. MORIKAWA. **Recognizing User Context Using Mobile Handsets with Acceleration Sensors**. In *2007 IEEE International Conference on Portable Information Devices*, pages 1–5, May 2007. (13)

[50] MIHUI KIM, MIHIR ASTHANA, SIDDHARTHA BHARGAVA, KARTIK KRISHNAN IYYER, ROHAN TANGADPALLIWAR, AND JERRY GAO. **Developing an On-Demand Cloud-Based Sensing-as-a-Service System for Internet of Things.** *Journal Comp. Netw. and Communic.*, **2016**:3292783:1–3292783:17, 2016. (18)

[51] M. KOVATSCH, M. WEISS, AND D. GUINARD. **Embedding internet technology for home automation**. In *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pages 1–8, Sep. 2010. (18)

[52] LAKSHMAN KRISHNAMURTHY, ROBERT ADLER, PHILIP BUONADONNA, JASMEET CHHABRA, MICK FLANIGAN, NANDAKISHORE KUSHALNAGAR, LAMA NACHMAN, AND MARK D. YARVIS. **Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea**. In *SenSys '05*, 2005. (14)

[53] SUMIT KUMAR AND ZAHID RAZA. **Using clustering approaches for response time aware job scheduling model for internet of things (IoT)**. *International Journal of Information Technology*, **9**(2):177–195, 6 2017. (4)

[54] KUN-CHAN LAN AND WEN-YUAH SHIH. **An intelligent driver location system for smart parking**. *Expert Syst. Appl.*, **41**:2443–2456, 2014. (13)

[55] N. D. LANE, E. MILUZZO, H. LU, D. PEEBLES, T. CHOUDHURY, AND A. T. CAMPBELL. **A survey of mobile phone sensing**. *IEEE Communications Magazine*, **48**(9):140–150, Sep. 2010. (12, 23)

[56] JAE-GIL LEE AND KANG MINSEO. **Geospatial Big Data: Challenges and Opportunities**. *Big Data Research*, **2**, 02 2015. (4, 92)

[57] I. LERA, C. GUERRERO, AND C. JUIZ. **YAFS: A Simulator for IoT Scenarios in Fog Computing**. *IEEE Access*, **7**:91745–91758, 2019. (48, 78)

[58] F. LI, M. VOEGLER, M. CLAESSENS, AND S. DUSTDAR. **Efficient and Scalable IoT Service Delivery on Cloud**. In *2013 IEEE Sixth International Conference on Cloud Computing*, pages 740–747, 2013. (16)

[59] FRANCESCO LONGO, DARIO BRUNEO, SALVATORE DISTEFANO, GIOVANNI MERLINO, AND ANTONIO PULIAFITO. **Stack4Things: a sensing-and-actuation-as-a-service framework for IoT and cloud integration**. *Annals of Telecommunications*, **72**:53–70, 2016. (6, 21, 48)

[60] H. MA, D. ZHAO, AND P. YUAN. **Opportunities in mobile crowd sensing**. *IEEE Communications Magazine*, **52**(8):29–35, Aug 2014. (16)

[61] SHUO MA, OURI WOLFSON, AND BO XU. **UPDetector: Sensing Parking/Unparking Activities Using Smartphones**. In *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, IWCTS 14, page 7685, New York, NY, USA, 2014. Association for Computing Machinery. (13)

[62] S. MADRIA, V. KUMAR, AND R. DALVI. **Sensor Cloud: A Cloud of Virtual Sensors**. *IEEE Software*, **31**(2):70–77, Mar 2014. (16, 18)

[63] DAVID MALAN, THADDEUS FULFORD-JONES, MATT WELSH, AND STEVE MOULTON. **CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care**. In *MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004)*, June 2004. (14)

[64] EVA MARÍN-TORDERA, XAVIER MASIP-BRUIN, JORDI GARCIA ALMIÑANA, ADMELA JUKAN, GUANG-JIE REN, JIAFENG ZHU, AND JOSEP FARRE. **What is a Fog Node A Tutorial on Current Concepts towards a Common Definition**. *CoRR*, **abs/1611.09193**, 2016. (11, 24)

[65] GEORGE MASTORAKIS, CONSTANDINOS MAVROMOUSTAKIS, JORDI BATALLA, AND EVANGELOS PALLIS. *Convergence of Artificial Intelligence and the Internet of Things*. Springer, 01 2020. (3)

[66] SUHAS MATHUR, TONG JIN, NIKHIL KASTURIRANGAN, JANANI CHANDRASEKARAN, WENZHI XUE, MARCO GRUTESER, AND WADE TRAPPE. **ParkNet: Drive-by Sensing of Road-Side Parking Statistics**. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys 10, page 123136, New York, NY, USA, 2010. Association for Computing Machinery. (13)

[67] PETER MELL AND TIMOTHY GRANCE. **The NIST Definition of Cloud Computing**. Technical Report 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD, September 2011. (11)

[68] CARLA MOURADIAN, DIALA NABOULSI, SAMI YANGUI, ROCH GLITHO, MONIQUE MORROW, AND PAUL POLAKOS. **A Comprehensive Survey on Fog Computing: State-of-the-art and Research Challenges**. *IEEE Communications Surveys & Tutorials*, **PP**, 10 2017. (26, 29)

[69] **Message Queuing Telemetry Transport (MQTT)**. `http://mqtt.org/`. (63)

[70] R. K. NAHA, S. GARG, D. GEORGAKOPOULOS, P. P. JAYARAMAN, L. GAO, Y. XIANG, AND R. RANJAN. **Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions**. *IEEE Access*, **6**:47980–48009, 2018. (26, 27, 29)

[71] ANANDATIRTHA NANDUGUDI, TAEYEON KI, CARL NUESSLE, AND GEOFFREY CHALLEN. **PocketParker: Pocketsourcing Parking Lot Availability**. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp 14, page 963973, New York, NY, USA, 2014. Association for Computing Machinery. (13)

[72] MARTIN BAUER NEC, MATHIEU BOUSSARD ALBLF, NICOLA BUI CFR, FRANCOIS CARREZ UNIS, CHRISTINE JARDAK SIEMENS, D. LOOF, CARSTEN MAGERKURTH SAP, STEFAN MEISSNER UNIS, ANDREAS NETTSTRÄTER IML, ALEXIS OLIVEREAU CEA, MATTHIAS THOMA SAP, WALEWSKI, JULINDA STEFA SUNI, AND ALEXANDER SALINAS UNIWUE. **Internet of Things Architecture IoT-A Deliverable D1.5 Final architectural reference model for the IoT v3.0**. In *IoT-A2013*, 2013. (21)

[73] **OpenStack**. https://docs.openstack.org/stein/. (21)

[74] **OpenFog Reference Architecture for Fog Computing, 2017.** https://www.iiconsortium.org/index.htm. (24, 25, 26, 27, 29, 30, 31)

[75] SHENG-LUNG PENG, SOUVIK PAL, AND LIANFEN HUANG. *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*. Springer, 01 2020. (1)

[76] P. P. PEREIRA, J. ELIASSON, R. KYUSAKOV, J. DELSING, A. RAAYATINEZHAD, AND M. JOHANSSON. **Enabling Cloud Connectivity for Mobile Internet of Things Applications**. In *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, pages 518–526, 2013. (12)

[77] CHARITH PERERA. *Sensing as a Service for Internet of Things: A Roadmap*. Leanpub Publishers, 2017. (18)

[78] CHARITH PERERA, ARKADY ZASLAVSKY, PETER CHRISTEN, AND DIMITRIOS GEORGAKOPOULOS. **Sensing As a Service Model for Smart Cities Supported by Internet of Things**. *Trans. Emerg. Telecommun. Technol.*, **25**(1):81–93, January 2014. (11, 17)

[79] JAVAD POURQASEM. **Cloud-based IoT: integration cloud computing with internet of things**. *International Journal of Research in Industrial Engineering*, **7**(4):482–494, 2018. (16)

[80] B. B. P. RAO, P. SALUIA, NISHA SHARMA NISHA SHARMA, ARPAN MITTAL, AND S. V. SHARMA. **Cloud computing for Internet of Things & sensing based applications**. *2012 Sixth International Conference on Sensing Technology (ICST)*, pages 374–380, 2012. (5)

[81] BHASKAR PRASAD RIMAL, EUNMI CHOI, AND IAN LUMB. **A Taxonomy and Survey of Cloud Computing Systems**. In *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, NCM '09, pages 44–51, Washington, DC, USA, 2009. IEEE Computer Society. (5)

[82] BILJANA RISTESKA STOJKOSKA AND KIRE TRIVODALIEV. **A review of Internet of Things for smart home: Challenges and solutions**. *Journal of Cleaner Production*, **140**:14541464, 01 2017. (16)

[83] ROSARIO SALPIETRO, LUCA BEDOGNI, MARCO DI FELICE, AND LUCIANO BONONI. **Park Here! a smart parking system based on smartphones' embedded sensors and short range Communication Technologies**. *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 18–23, 2015. (13)

[84] AKIO SASHIMA, YASUO INOUE, TOMOHIKO IKEDA, TAKAJI YAMASHITA, AND KEISUKE KURUMATANI. **CONSORTS-S: A mobile sensing platform for context-aware services**. *2008 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 417–422, 2008. (14)

[85] SUCHISMITA SATPATHY, BIBHUDATTA SAHOO, AND ASHOK KUMAR TURUK. **Sensing and Actuation as a Service Delivery Model in Cloud Edge centric Internet of Things**. *Future Generation Computer Systems*, **86**:281 – 296, 2018. (21, 41, 48)

[86] M. SATYANARAYANAN. **The Emergence of Edge Computing**. *Computer*, **50**(1):30–39, 2017. (25, 30, 31)

[87] KEWEI SHA, GUOXING ZHAN, WEISONG SHI, MARK A. LUMLEY, CLAIRY WIHOLM, AND BENGT B. ARNETZ. **SPA: a smart phone assisted chronic illness self-management system with participatory sensing**. In *HealthNet '08*, 2008. (14)

[88] X. SHENG, J. TANG, X. XIAO, AND G. XUE. **Sensing as a Service: Challenges, Solutions and Future Directions**. *IEEE Sensors Journal*, **13**(10):3733–3741, Oct 2013. (16)

[89] X. SHENG, X. XIAO, J. TANG, AND G. XUE. **Sensing as a service: A cloud computing system for mobile phone sensing**. In *SENSORS, 2012 IEEE*, pages 1–4, Oct 2012. (11, 12, 16, 17)

[90] JOHN SOLDATOS, NIKOS KEFALAKIS, MANFRED HAUSWIRTH, MARTIN SERRANO, JEAN-PAUL CALBIMONTE, MEHDI RIAHI, KARL ABERER, PREM PRAKASH JAYARAMAN, ARKADY ZASLAVSKY, IVANA ZARKO, LEA SKORIN-KAPOV, AND REINHARD HERZOG. **OpenIoT: Open Source Internet-of-Things in the Cloud**. *Lecture Notes in Computer Science*, **9001**:13–25, 03 2015. (16)

[91] 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, 2014. (25)

[92] G. SUCIU, O. FRATU, S. HALUNGA, C. G. CERNAT, V. POENARU, AND V. SUCIU. **Cloud consulting: ERP and communication application integration in open source cloud systems**. In *2011 19thTelecommunications Forum (TELFOR) Proceedings of Papers*, pages 578–581, 2011. (16)

[93] A. TAIVALSAARI AND T. MIKKONEN. **Cloud Technologies for the Internet of Things: Defining a Research Agenda Beyond the Expected Topics**. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pages 484–488, 2015. (16)

[94] M. THOMA, S. MEYER, K. SPERNER, S. MEISSNER, AND T. BRAUN. **On IoT-services: Survey, Classification and Enterprise Integration**. In *2012 IEEE International Conference on Green Computing and Communications*, pages 257–260, Nov 2012. (10)

[95] PASCAL THUBERT, TIM WINTER, ANDERS BRANDT, JONATHAN HUI, RICHARD KELSEY, PHIL LEVIS, KRISTOFER PISTER, RENE STRUIK, JP VASSEUR, AND ROGER ALEXANDER. **RPL: IPv6 Routing Protocol for Low power and Lossy Networks**. *IETF*, **RFC 6550**, 03 2012. (16)

[96] C. TSENG AND F. J. LIN. **Extending scalability of IoT/M2M platforms with Fog computing**. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 825–830, 2018. (26)

[97] KOROSH VATANPARVAR AND MOHAMMAD ABDULLAH AL FARUQUE. *Control-as-a-Service in Cyber-Physical Energy Systems over Fog Computing*, pages 123–144. Springer, 05 2018. (26)

[98] W. WANG, K. LEE, AND D. MURRAY. **Integrating sensors with the cloud using dynamic proxies**. In *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, pages 1466–1471, 2012. (23)

[99] MARKUS WEISS, FRIEDEMANN MATTERN, TOBIAS GRAML, THORSTEN STAAKE, AND ELGAR FLEISCH. **Handy Feedback: Connecting Smart Meters with Mobile Phones**. In *Proceedings of the 8th International Conference on Mobile and Ubiquitous Multimedia*, MUM 09, New York, NY, USA, 2009. Association for Computing Machinery. (18)

[100] B. XU, O. WOLFSON, J. YANG, L. STENNETH, P. S. YU, AND P. C. NELSON. **Real-Time Street Parking Availability Estimation**. In *2013 IEEE 14th International Conference on Mobile Data Management*, **1**, pages 16–25, June 2013. (13)

[101] NING XU, SUMIT RANGWALA, KRISHNA KANT CHINTALAPUDI, DEEPAK GANESAN, ALAN BROAD, RAMESH GOVINDAN, AND DEBORAH ESTRIN. **A Wireless Sensor Network For Structural Monitoring**. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys 04, page 1324, New York, NY, USA, 2004. Association for Computing Machinery. (14)

[102] S. YANGUI, P. RAVINDRAN, O. BIBANI, R. H. GLITHO, N. BEN HADJ-ALOUANE, M. J. MORROW, AND P. A. POLAKOS. **A platform as-a-service for hybrid cloud/fog environments**. In *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 1–7, 2016. (31)

[103] M. YANNUZZI, R. MILITO, R. SERRAL-GRACI, D. MONTERO, AND M. NEMIROVSKY. **Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing**. In *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 325–329, 2014. (25)

[104] A. YOUSEFPOUR, G. ISHIGAKI, R. GOUR, AND J. P. JUE. **On Reducing IoT Service Delay via Fog Offloading**. *IEEE Internet of Things Journal*, **5**(2):998–1010, 2018. (25)

[105] JAESEOK YUN, IL-YEUP AHN, JAESEUNG SONG, AND JAEHO KIM. **Implementation of Sensing and Actuation Capabilities for IoT Devices Using oneM2M Platforms**. *Sensors*, **19**:4567, 10 2019. (5, 22)

[106] M. YURIYAMA AND T. KUSHIDA. **Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing**. In *2010 13th International Conference on Network-Based Information Systems*, pages 1–8, Sep. 2010. (18)

[107] IVANA ZARKO, ALEKSANDAR ANTONI, AND KREIMIR PRIPUI. **Publish/subscribe middleware for energy-efficient mobile crowdsensing**. In *UbiComp 2013 Adjunct - Adjunct Publication of the 2013 ACM Conference on Ubiquitous Computing*, pages 1099–1110, 09 2013. (23)

Enhancing IoT Service Delivery: A Layered Fog Architecture Approach for Sensing and Actuating as a Service

by Abdulsalam Abdo Musaed Ali Alammari

Enhancing IoT Service Delivery: A Layered Fog Architecture Approach for Sensing and Actuating as a Service

ORIGINALITY REPORT

15%	2%	14%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Overall Similarity Index: 15 - 10 = 05%

22/2/2021

1	Abdulsalam Alammari, Salman Abdul Moiz, Atul Negi. "Chapter 2 Internet of Things Sensors and Actuators Layered Fog Service Delivery Model SALFSD", Springer Science and Business Media LLC, 2019 Publication	10%
2	Paolo Bellavista, Javier Berrocal, Antonio Corradi, Sajal K. Das, Luca Foschini, Alessandro Zanni. "A survey on fog computing for the Internet of Things", Pervasive and Mobile Computing, 2019 Publication	2%
3	"Multi-disciplinary Trends in Artificial Intelligence", Springer Science and Business Media LLC, 2019 Publication	<1%
4	Submitted to Arizona State University Student Paper	<1%
5	www.openfogconsortium.org Internet Source	

This publication is by the Student

Professor
School of CIS
Prof. C.R. Rao Road,
Central University
Hyderabad-46. (India)

22/2/2021