Machine learning based computational offloading in the Fog Cloud Internet of Things ecosystems

by

Adam A. Alli

154701

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering

Islamic University Of Technology - Dhaka Bangladesh

May 22, 2023

Certificate

The thesis titled ”Machine learning based computational offloading in the Fog Cloud Internet of Things ecosystems ” submitted by Adam A. Alli, Student no. 154701 of Academic Year 2015-2016 has been found satisfactory and accepted as partial fulfillment of the requirement for Doctor of Philosophy in Computer Science and Engineering(CSE)

Declaration

It is hereby declared that this thesis report or any part of it has not been submitted elsewhere for the award of any Degree or Diploma.

Muhammad Mahbub Alam, PhD
Professor,
Computer Science and Engineering department,
Islamic University of Technology (IUT),
Date: May 22, 2023.

Adam A. Alli
Student No.: 154701,
Academic Year: 2021—22,
Date: May 22, 2023.

"in fact the reply of true believers is when they are called to Allah and his Messenger to judge between them is only that they say: we hear and we obey. and those are the successful ."

Quran 24:51

Abstract

Internet of Things (IoT) is the new prototype that is shaping Information and Communication Technologies (ICT) in a new direction. The rate of adaptability, ingenuity, and expansion is promising to bring connectivity to all things in homes, offices, vehicles, etc. This paradigm is causing researchers to find new ways to connect things efficiently, improve processing power of devices, create value for data collected from IoTs, and ensure the security of the data. Although numerous data-intensive applications (self-parking cars, trackers, domestic appliances, smartphones, etc.) have been developed to employ the use of smart devices they are small in size, battery-powered, and limited in processing, storage, and memory supplies. Their size and nature of applications act as a bottleneck to implement powerful applications on them. Moreover, the amount of data generated by IoT devices collectively surge the network. Therefore, it is necessary to utilize the Cloud of Things (CoT) infrastructure. A CoT provides extensive processing power and unlimited storage that permits fast processing, and bulk storage of data produced by the new cloud infrastructure. It works well in situations that are not delay-sensitive, and require no immediate responsiveness. However, the use of CoTs exclusively is not effective in applications that require immediate processing, high responsiveness, and real-time analysis of client's requests because of the distance between the CoTs and IoTs. To this purpose, fog computing has been designed. Fogging enables computational offloading, data aggregation, and storage. The offloading process enables smart objects to realize the full potential of the IoT-Fog-cloud infrastructure by sending part of its computation routines to remote sites for processing. This permit saving computational resources such as processors, storage, memory, and energy. Nonetheless, the question of when, what, and how to offload has not been fully resolved. Moreover, new challenges keep emerging such as finding the better offloading point given mobility, heterogeneity in IoT devices, and dynamic communication environment. Further, dealing with offloading that is constrained by location and latency-sensitive submissions remain hard to solve. This study is intended to solve an IoT-Fog infrastructure performance problem in which each of the IoT devices may contain computation-intensive, or security-sensitive, or delay-sensitive tasks to offload. If the IoT device finds that it cannot execute the tasks, an offload to an optimal Fog is initiated through a Smart

gateway (SG). The Fog either performs the tasks or sends it to the cloud. The intension of the study is to perform dynamic offloading while maintaining the user's sensitive tasks in the Fog during offloading. The study is expected to achieve high performance in terms of throughput, delay, energy consumption, resource utilization rate and response time. For this purpose, a computation offloading framework is proposed with the engagement of a four-tier cloud infrastructure using a pipeline of machine learning (ML) strategies. This pipeline consists of a set of ML models connected in series to facilitate the offloading efficiently. The study is intended to design a fast, efficient and robust algorithm that enables the selection of optimal fog and cloud when the need arises, while providing mobility when network conditions fluctuate.

Acknowledgements

All praise and gratitude be to Allah, the most beneficent, the most merciful. ”Recite: In the name of thy Lord who created man from a clot. Recite: And thy Lord is the Most Generous Who taught by the pen, taught man that which he knew not.” (Quran, 96:1-5), I pray that the knowledge I have gained during this uneasy years rises me to the heights of goodness and to a noble position that will enable me to serve Allah subuhanawa'tallah, associate me with sovereigns in this world, and make me attain the perfection of happiness in the next. I pray that Allah gifts me and my associates with Janna in the hear after.

I am deeply indebted to my supervisor Prof. Muhammad Mahbub Alam, PhD, without his guidance, encouragement, experience and practical advise this work would not have been possible. Secondly, I am grateful to the members of the Doctoral Committee that included Prof. Muhammad Mahbub Alam, Prof. Dr. Abu Raihan Mostofa Kamal, Prof. Dr. Md. Hasanul Kabir and my external supervisors Prof. Dr. Md. Saifu Islam of IICT, BUET, Dhaka and Prof. Dr. Mostafizur Rahman for all their guidance through the process of developing the this thesis and feedback which was absolutely valuable. Third, I am thankful for my family, friends, brothers, sisters, my professors at the department of Computers Science and Engineering IUT, Dhaka, the Uganda Community members at IUT and every body who in one way or the other contributed to my success during this work.

With all my heart I want to thank my wife Mrs. Nakalema Sauda for her patience during the time I was away pursuing my studies may the Almighty reward you abundantly. My mothers Ramulah Badaru and Wibabara Rehema nothing I have achieved that you have not blessed. I will forever be indebted to you.

In a special way I wish to acknowledge the efforts of Mr. Adugo Adam Ayub, Thanks for always having my back, Dad. I love you.

I will not forget to acknowledge my No.5, Mr. Adam Semugerere(Breaker) and your family you have been there for me as always. My sincere thanks also goes to Anewa Anita, Jameson Nimwesige, Babirye Amina and Sarah Namusisi. I will at this point take time to recognize my special buddie Babirye Zahara, My children will always love you for every things you do for them.

My boys Waleed, Rabah, Fattah, and Abdul Rahman and girls Moseka, Neva, Dhahabia and Nuswaira. you have been so patient. May Allah allow me to always be there for you.

To my other family, I want to thank My Father−in−law Haji Lubega Abdu Mude and Mother in law Hajati Halima, I can't thank you enough for your support all this time. My sisters and brothers at both ends may Allah reward you all. I have not forgotten every beat that you have done for me all these time.

Contents

List of Figures

List of Tables

To my wife Mrs. Nakalema Sauda, children, Family and Friends. . .

Chapter 1

Introduction

Internet of Things (IoT) is the new archetype that is shaping information and communication technologies in a new direction. The rate of adaptability, ingenuity, and expansion is promising to bring connectivity to all things in homes, offices, vehicles etc. [3–5]. This paradigm is causing companies and research bodies to find new ways to connect things efficiently, create value from data harvested from abundant sensors and actuators, and ensure security of the data harvested from the IoT devices. Further, the massive amounts of data harvested from the IoT devices require new ways of analytics to create insights from them. To realize reasonable insight from data harvested from IoT devices powerful mechanisms to process the data is required. For the aforementioned motive, many research areas have been developed in the areas of Internet of Things related systems [6, 7].

The challenges faced in IoT research are still enormous. The first is how to support billions of IoT devices through stable network infrastructure. The second challenge is how to ensure security and privacy of massive data generated by IoT devices. The third challenge is how can IoT devices perform complex operations within their small processing power, and still achieve limitless potential of IoT platforms. Lastly, how to incorporate the endless list of use cases that finds the applications of IoT devices suitable for their operation without disrupting the status quo. To this regard, issues related to improving system performance, standards and policies, increasing capacity of smart devices to process complex applications, designing methods to handle big data and its analysis creates new areas of research in IoT ecosystems [8].

Amidst all the above challenges, the use of smart devices (smart phones, tablets, smart locks, smart cars etc.) by public to realize multitude of applications is limitless in real life [9]. The use of smart applications has the biggest impact on how things are done. This is visible in our homes, smart cities, smart health industry, smart factories, and smart transport industry [5]. These applications may be relatively simple like sending an alerts to your mobile about the status of a smart home, or relatively complex such as synchronizing data coming from all IoT devices using data analytics to predict future events, or very complex such as in real time identification of tourist sites using augmented reality. IoT systems present unlimited use of sensors to accomplish endless use cases in artificial intelligence, medical engineering, crowd sourcing etc. that are complex.

In the past, complexity of application was handled by increasing the capabilities of hardware, but in recent past, complexity of applications is growing more faster than hardware. To catch up with application complexity and big data produced in computing environment, paradigms such as client-server computing, Cloud computing, Edge computing, Lambda etc. have been recognized. In the recent, cloud computing paradigm has enabled IoT users and enterprises to gain flexibility with re-provisioning. Re-provisioning allows users to expand technology infrastructure resources, perform complex applications, reduce on cost of processing, provide remote access without worrying of location of the processing machine, reduce maintenance costs, and improve performance of systems [3].

The cloud paradigm has made computing resources distributed, and no longer dependent on single infrastructure owned by a single organization [10]. It is becoming increasingly easy to migrate data, processing, and applications to high performance machines located a distance [11]. This process of migrating data, processes and applications to a more powerful infrastructure gives the users enhanced performance and Quality of Experience(QoE). So far, cloud computing procedures are adopting integration of the use of information supplying applications across the cloud and on-premise systems. This creates hybrid cloud that gives the ability to integrate big data from the Internet of Things(IoT), Machine learning(ML), container management platforms, Artificial Intelligence (AI), Blockchain, and cloud security.

In this thesis, we explore areas in which the cloud computing paradigm can be improved to offset its limitations. Particular to this study, we develop computational offloading to the Fog-Cloud of things framework using machine learning mechanisms. We continue to illustrate the use of machine learning at the IoT layer to select Fog devices to which they can offload their computation. At the network layer we implement and demonstrate how security can be achieved through neural fuzzy systems, and at the Fog layer we show how offloading is done to hybrid cloud. All the above is done in efforts to overwhelm the issues of performance and security that form major drawback in the cloud computing paradigm.

1.1 Motivation

Internet of Things (IoT) devices are continuously developing into powerful mechanisms to achieve smart applications in health, cities, utilities etc. Although IoT devices have found use in many applications, they are limited by their size in terms of processing power, memory and energy supplies. To overcome these challenges caused by their structural limitation, cloud computing has been employed. [4, 12–14].

Computational offloading to the cloud enable users to outsource computational intensive tasks to the cloud. Also, big data generated by IoT devices that cannot effectively be stored on user equipment is relocated to the cloud for cheap storage. It is generally understood that computational offloading to the cloud can improve performance of IoT systems, because computationally hungry part of a process is transferred to the cloud for processing, and results are sent back to the IoT device. This in one way enables the IoT devices to save energy. Some systems include energy harvesting mechanisms to mitigate battery consumption [12, 15, 16].

However, the biggest challenge facing IoT applications such as in tele—medicine and connected vehicles is how to reduce latency for time sensitive applications. In such kind of use cases the cloud ceases to be the right choice for computational offloading, because of loss of end-to-end control and visibility over the network. Secondly, in real-time applications such as Artificial intelligence, Data Analytics, Augmented Realities(AR), online gaming etc., there is increased demand for quality connectivity and bandwidth which the cloud may not provide [17]. Thirdly, issues of security and privacy raise a number of concerns ranging from loss of control, and trust. Lastly, network coverage in especially rural areas tend to be sparse and have less speed as compared to more urban places. Connectivity may be unstable, in addition, IoT devices are sometimes mobile. Therefore, leveraging new paradigms such as MEC, cloudlets, and the Fog have been found to improve performance, uplift security, privacy and latency issues. [9, 18, 19].

As of yet, more investigations are required to continuously improve performance of IoT-Fog-Cloud offloading ecosystems. Applying variety of machine learning strategies at all levels of continuum i.e. IoT layer, network layer, and the Fog layer may improve performance, and elevate weakness in offloading to better accommodate new use cases in the IoT ecosystem. With the above we find a motivation to further investigate secure computational offloading using machine learning models.

1.2 Problem Statement

The inability of IoT devices to execute complex application due to structural inefficiency creates a computational problem that has not been fully resolved yet. Efforts have been made to develop solutions to transfer services that can not be handled at the IoT devices to cloud, but more service application that require more resources have appeared. Hence a requirement to transfer traditional services that are provided by the traditional cloud closer to the user at the network edge has emerged.

A set of frameworks to achieve computational offloading at the edge, assess suitability of the fog paradigm, study the dynamic resource estimation and pricing, security management in ubiquitous environment have been proposed in the literature [17, 18, 20, 21]. The current limitation of most of the solutions proposed is that they assume that resources are often available and secure. In an ideal environment, resources are dynamic, flexible, and prone to attack. Neglecting the nature of applications, security, and fluctuating connectivity present a limitation to many of the existing frameworks.

The task at hand is to device a novel approach that is efficient, secure and dynamic to overcome the existing limitations of the existing approaches. At the same time, loosen the coherence with the local infrastructure during processing. Moving forward, given a network of IoT devices connected to the Internet and to the cloud infrastructure, if a given IoT has a task to execute, it can either perform the computation locally on the devices if it has enough resources; otherwise, it offloads the task to a remote device.

If a device chooses to offload the task for execution to remote device, then it must choose from a set of powerful devices in neighbourhood, where computational resources are in abundance. The question at hand is, which device within network range can accept to perform the workload at cheapest price in terms of latency, memory, storage etc. Examples of nearest neighbor devices include switches and routers, data centers and clouds.

The problem can be viewed in the following way where

i) ω represents the workload to be offloaded

ii) η represents the a set of devices on the network that accept offloadable workload

iii) $\psi(\mu\epsilon\eta, \omega)$ represents the function that evaluates the cost of executing the workload ω in either whole or part on a device μ

Then the formal definition is to Minimize $\psi(\mu\epsilon\eta, \omega)$ for all values $\mu\epsilon\eta$.

1.3 Objectives of the study

This study was aimed at achieving a computational offloading framework with engagement of four tier fog-cloud using a pipeline of ML strategies that include the particle swarm optimization at the lower tier, Neural fuzzy in the smart gateway and reinforcement learning at the fog tier; the thesis shall focus on the following objectives:

i) Design and develop an efficient offloading strategy that exploits a pipeline of machine learning strategies across the IoT-Fog-Cloud platform.

ii) Improve robustness of the system by implementing surrogate entities and Task classifiers in the Fog.

iii) Design and implement a Neuro-fuzzy model to isolate nodes that exhibit low trust.

1.4 Outcomes of the study

This study aimed at the following outcomes

(i) An efficient offloading algorithm that leverages the 4-tier IoT-SG-Fog-cloud framework.

(ii) A surrogate entity that enables stability by providing mobility in network fluctuations.

(iii) A classifier based on Load input ratio that categorizes data in terms of complexity and sensitivity in the Fog layer.

(iv) A robust procedure that mitigates the risk of flooding the Fog with unnecessary big data which may result in an attack.

1.5 Methodology

In this thesis, we propose two machine learning based secure offloading herein referred to as SecOFF-FCIoT and NiCO-FCIoT frameworks. Our proposals are a software solutions that approaches offloading computations to the Fog or the cloud using machine learning and nature inspired approaches. Our dynamic and secure offloading is hoped to reduce offloading latency and minimize energy consumption.

Machine learning based approaches have been chosen because, Machine learning has demonstrated latent power in solving complex problems in science, engineering, industrial and professional practices [22–26]. For example, in banking and finance machine learning can be used by executive managers to make informed decision. Machine learning can help banks spot potential business partners and their expenditure. Using Machine learning algorithm smart machines can be trained to monitor trends in the market and react in real time [27] .

In medicine, machine learning has been used in diagnosis of complicated diseases. Moreover, it has been used in analysis of clinical parameters and their combination of the prognosis [28]. It can help in integrating computer based systems in health care industry. Additionally, machine learning has demonstrated ability to assist in solving complex problems in aerospace engineering. In [29], machine learning has been used to recognize specific defects of aerospace structures, decrease approximation errors and compute the closest possible outcome. The limits due to the great amount of data and the complexity of data processing make machine learning as defect classifier in aerospace structures very useful [29, 30]. Machine learning has also been used to optimize parameters so as to find best options that maximize use of resources in logistic and supply management. Here, machine learning allows improved planning for unexpected events and help predict orders along the supply chain with at most accuracy [31].

In this thesis, therefore, we propose a pipeline of Machine learning based methodologies in Fog Cloud of thing environment as shown in figure 1.1, to improve performance and secure data coming from abundant IoT devices.

In general, The IoT devices generate workload that may have offloadable content. The IoT device select a most suitable Fog node that can process the request, and send back the results with in the required threshold. PSO is used at this level to choose an optimal Fog node. The offloadable content is forwarded through a smart gateway to the Fog. The smart gateway secures the data using Nuero-Fuzzy model. The Fog may offload the workload coming from the IoT to the cloud using reinforcement learning mechanism. After processing, the cloud sends back the results to the IoT through the Fog, and Smart gateway.

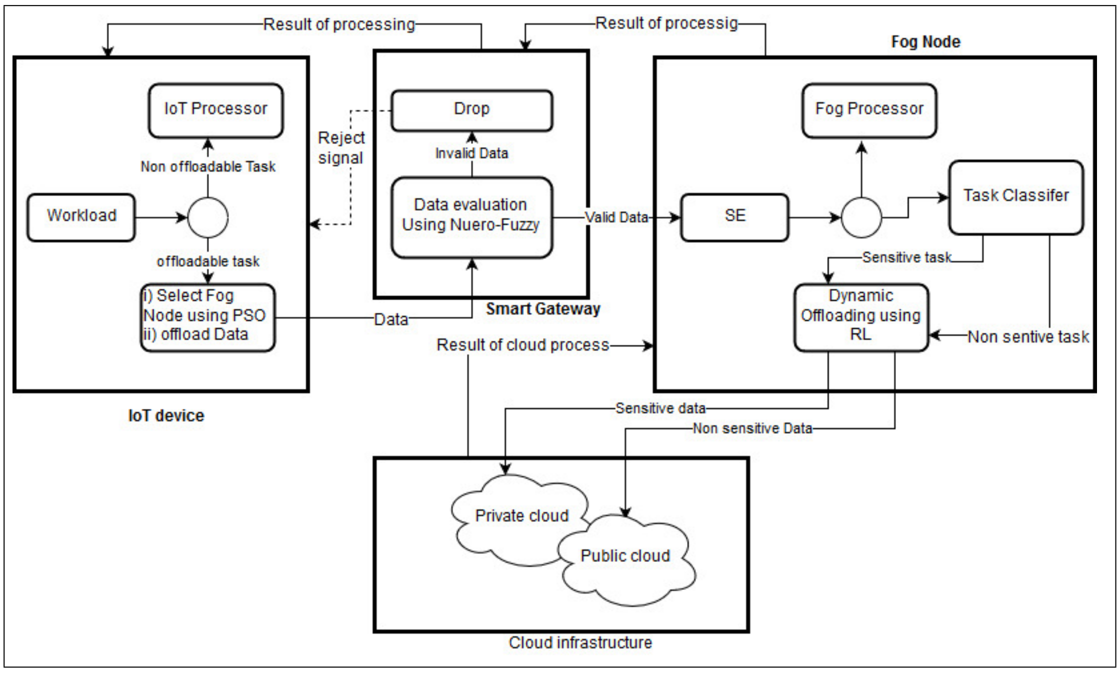

Figure 1.1: The General Pipeline of Proposed Secure Offloading Scheme

1.6 Thesis contribution

This thesis focused on solving the problem of secure offloading of sensitive data using nature-inspired algorithms, surrogate entities, and a Neuro-fuzzy model. These approaches improve latency and optimize resource utilization.

The thesis addresses several challenges related to secure offloading in the Fog Cloud ecosystem, making the following three main contributions:

- A secure offloading framework is proposed that utilizes Particle Swarm Optimization (PSO) and Reinforcement Learning (RL) for efficient offloading in the Fog-IoT environment. PSO is employed at the IoT level to select the optimal node for offloading, while RL is used at the fog level to select the appropriate cloud.
- The use of a Neuro-fuzzy model is introduced to identify IoT nodes attempting to congest the network by sending invalid data. If an IoT device is detected sending invalid data for offloading, the data is dropped. This prevents nodes with low trust from engaging in malicious transactions on the network.
- An improved version of SecOFF-FCIoT, named modified NiCO-FCIoT, is proposed in Chapter 4.

1.7 Thesis outline

Chapter 1 of this thesis serves as the foundation, covering various aspects necessary to understand the research. It begins with an introduction to the topic, followed by a discussion on the motivation behind the work in 1.1. The problem statement is clearly stated in 1.2, whereas objectives are in 1.3. The outcomes are stated in 1.4, and the general methodology is outlined in 1.5. The contributions of the thesis are highlighted in 1.6. Finally, subsection 1.7 summarises the chapter by presenting an overview of the thesis organization.

The remaining chapters of the thesis are structured as follows: In chapter 2, We delve into the fog—cloud of things concept, covering architectures, standards, tools, and applications. This literature review serves as a foundation for the technical solutions proposed in Chapter 3 and Chapter 4.

Chapter 3 introduces our proposed solution Secure offloading in Fog-cloud of Things (SecOFF-FCIoT), which leverages machine learning to enhance the offloading of IoT-Fog-cloud ecosystems. The chapter focuses on the computational offloading framework and the associated pipeline of machine learning mechanisms, all aimed at achieving improved performance.

In Chapter 4, NICO-FCIoT, an enhanced mechanism for initiating the offloading process at the smart gateway in a clustered IoT-Fog-cloud environment is presented. This proposal utilizes modified dynamic particle optimization. Our simulations demonstrate the resulting improved performance.

Chapter 5 serves as the conclusion, summarizing the research contribution and discussing future directions for the research.

Chapter 2

Literature Review

2.1 Introduction

Internet of things(IoT) is becoming a primary enabler of powerful technologies in our society that are poised to change our perception about devices and our environment. The intersection of the internet of things, Big data, Artificial intelligence, cloud computing, software-defined networks, and invention of more powerful and intelligent edge devices (Switches, routers, mobile terminals, etc.) have not only resulted into an explosion in the number of connected devices and data over the internet infrastructure, but links between machines and the human world.

According to IoT analytics [32], the number of connected devices that are in use worldwide in 2018 surpasses 17 billion of which 7 billion are IoT devices, if this trend continues the number of active connected IoT devices is likely to surpass 22 billion in 2025. However, compared to traditional devices such as laptops, and desktops, IoT devices are limited not only in terms of computing power, storage capacity and battery life [33], but also limited by their heterogeneous nature. The heterogeneity of IoT devices restricts the promotion of unifying one size fits all solution. Currently, there are hundreds of competing standards, chipsets, and backend products to choose from if one wants to develop an IoT application. Moreover, there are tens of viable standards before one starts thinking of thousands of tools which can be used to realize competing IoT invention. The sphere of the internet of things architectures, standards, and tools provide prospects to progress new kind of services, and application facilities in Cloud-of-Things environment[10, 11, 33, 34].

The cloud delivers exciting opportunities for nonprofit of all sizes. Many organizations using the cloud services are able to cut down Information communication operational cost, leverage cloud tools and infrastructure for their IT services , and enable new levels of collaboration between users of all kinds[10, 15]. The variety of services provided by Cloud Service Providers(CSP) are incredible, hence making its adoption for organizations a rational decision. However, the drawbacks associated with the use of cloud such as downtime due to overload caused by various clients, or the unavailability of internet connection may cause short-lived interruption. Moreover, security concerns, vendor deadlocks, and limited access make the exclusive use of cloud paradigm less attractive. [3–6].

Authors in [35], surveyed extended cloud technologies (fog and mobile Edge Computing (MEC)), in their study they compared the cloud to the extended cloud technologies in terms of latency, location of service, geo-distribution, mobility, location awareness, type of mile connectivity and distance from the clients. They pointed out that allowing computing to happen closer to the user's location provides opportunities to support applications that have constrained processing requirements. In addition, they reviewed security challenges and they filed the need of resilience as basic property that is required for availability of services, maintenance of the confidentiality and integrity of the information. In the face of new use cases new challenges continue to arise, among them is how to handle the flood of big data, security and privacy of sensitive data. Also how to reduce the increasing cost of implementing distributed fog systems.

In [36], a comprehensive survey of fundamental and recent advancement towards fog computing-enabled network architecture was given. In addition to issues mentioned in [35], they surveyed the F-RAN and related critical issues that include energy utilization, Spectral Efficiency, backhaul traffic management, user mode selection and service allocation and selection , and resource management challenges. Given the exponential growth of IoT devices and data across IoT-Fog-cloud ecosystems it is not unique for backhaul traffic management challenges to shift to the fog platform, also challenges of integrating new methodologies such as machine learning, artificial intelligence and blockchain remain to be new phenomena that need to be surveyed.

C. Puliafito et al. in [37], presented a survey on employment of fog cloud to support IoT devices and services. They highlight six application domains that might benefit from the Fog-cloud ecosystems, then they elaborated on the new challenges that arise as a result of the new fog-cloud formations along with software, hardware platforms and standardization effort. Authors in [38] performed a classification-based investigation into the requirements of fog infrastructure, platform, and applications mapped to current research. In addition, they discuss existing research works and gaps in resource allocation and scheduling, fault tolerance, simulation tools, and Fog-based microservices. They also present some open issues, which will determine the future research direction for the Fog computing paradigm. Furthermore, they described some simulation tools used to accomplish fog research, nonetheless a number of issues have come to light in the IoT-fog ecosystems that need to be filed. These include real-time analytics , resource management, methods of fog−to−fog communication, supporting crowdsourcing applications, new challenges in trust and security etc. To this end, we expand on surveys in [35–38] by providing more insight of research aspects filed , the state of art, future development trends and open issues.

In this chapter therefore, we provide extensive exploration on the fog cloud of things to give a foundation to solutions proposed in this thesis. We investigated the building blocks in IoT-Fog-cloud computing specifically the concepts, architectures, standards and tools. This survey enabled us to gain insight into the requirements for building fog−cloud solutions proposed in this thesis with a better understanding of computing infrastructure created by the IoT−Fog−ecosystems and the current trends of research filed.

Thus, this chapter is organized as follows: section 2.2 provides an overview of fog architecture, ecosystems, and fog standards (IEEE 1934 and ETSI 2016). In section 2.3, we present Emerging trends, issues, and challenges in fog cloud of things and ideal solutions. Section 2.4 presents a taxonomy based on emerging issues in the fog−cloud of things environment. In section 2.5, we present key applications driving fog computing paradigms. Section 2.6 presents the fog cloud of things tools including selection considerations and examples of tools commonly used in developing concepts in fog research domain. Further, the tools that can be used to model and develop proof of concepts, test and implement Fog experiments. In section 2.7, we present Open issues in fog−cloud of thing ecosystems, and we present a summary of the survey in section 2.8.

2.2 Overview of fog architecture, ecosystems, and standards

2.2.1 The Fog cloud of things architectures

In principle, the Fog computing framework is a N- tier hierarchical with distinct layers. At the bottom of the framework are the end devices, a collection of IoT devices, (sensors, actuators and smart devices such as smart phones). The next layer is Fog layer which forms the middle tier and the cloud layer which forms the upper layer. We provide discussion on the components of Fog framework based on Fig. 2.1.

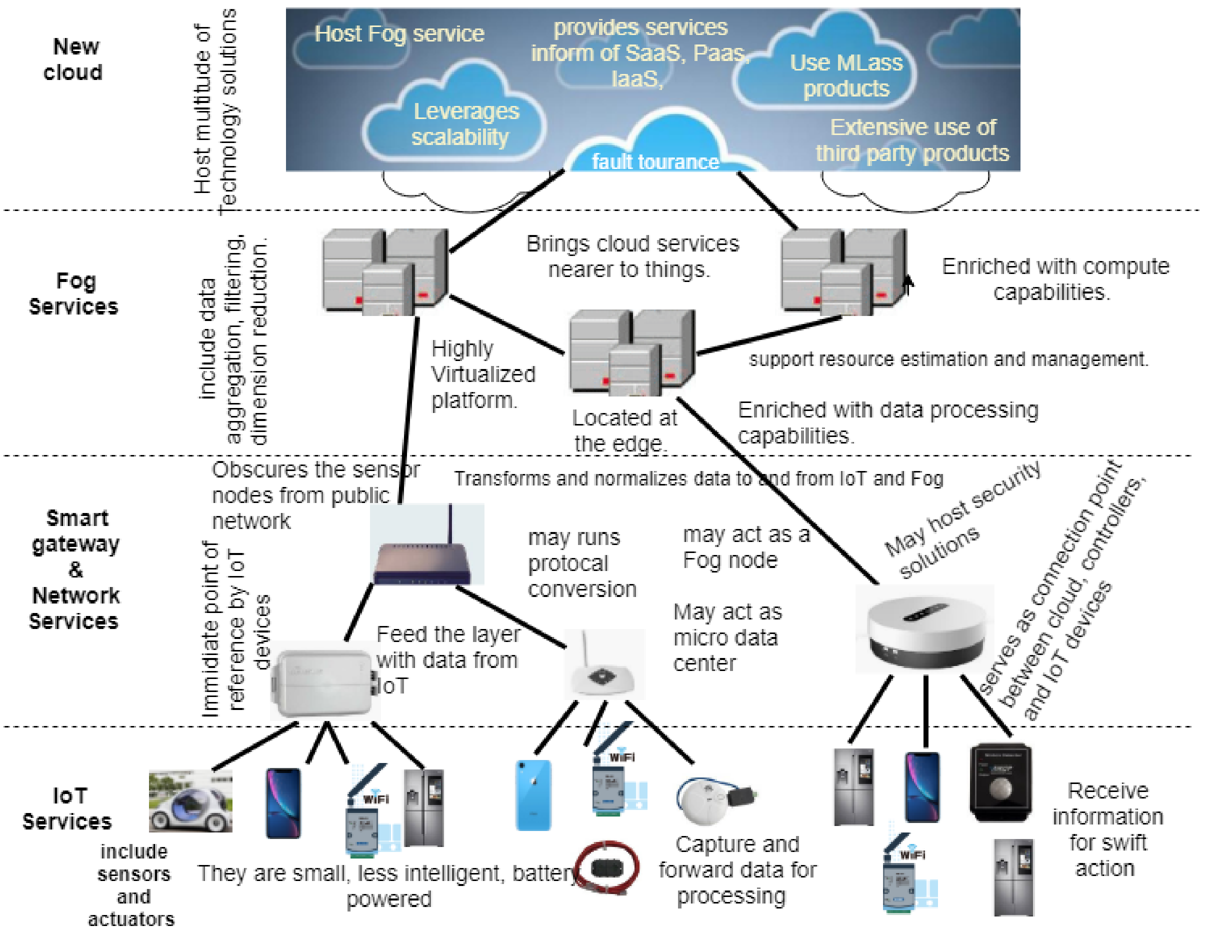
New cloud
Host multitude of Technology solutions
Host Fog service
provides services inform of SaaS, Paas, IaaS,
Use MLass products
Leverages scalability
fault tourance
Extensive use of third party products
Fog Services
include data aggregation, filtering, dimension reduction.
Brings cloud services nearer to things.
Enriched with compute capabilities.
Highly Virtualized platform.
support resource estimation and management.
Located at the edge.
Enriched with data processing capabilities.
Smart gateway & Network Services
Obscures the sensor nodes from public network
Transforms and normalizes data to and from IoT and Fog
May host security solutions
serves as connection point between cloud, controllers, and IoT devices
Immidiate point of reference by IoT devices
may runs protocal conversion
may act as a Fog node
May act as micro data center
Feed the layer with data from IoT
IoT Services
include sensors and actuators
They are small, less intelligent, battery powered
Capture and forward data for processing
Receive information for swift action

FIGURE 2.1: The hierarchical components of Fog Computing Paradigm and services at each layer.

i) **The IoT Nodes** : Depending on the size and purpose, IoT serve the purpose of sensing the environment, collect data inform of observation, transform and process data in real time when supported. Even though they have limited computational power, they can be used to accomplish processes that need real-time responsiveness. Moreover, they are used in transmitting data to the next level of the hierarchy. IoT devices consist of digital sensor component for sensing the the external conditions, digital converter, data storage unit used to store data, connectivity used to ensure network function and power unit [39]. IoT nodes are equipped with some security mechanism. Security is critical for helping the micro controller perform a secure boot, insuring that the core is running the code is meant to run [40]. Some IoT may be used to ensure security by tagging counterpart IoT's that exhibit suspicious behaviour. The IoT's make the front end of IoT-fog-cloud infrastructure.

ii) **The Fog Nodes**: The fog is positioned in between the cloud and IoT devices in the IoT−Fog−cloud hierarchy. The fog undertakes processing, storage, secures the IoT devices and supports the cloud. When supported they perform transactions analytics, which may include cleaning data, dimension reduction, and perform automatic handing over of handful of data for processing to the cloud (near end real−time data analysis). The fog supplies the infrastructure with intelligence that the IoT nodes cannot offer due to lack of sufficient compute power. They may adopt cognitive abilities that allow them to monitor and control IoT devices, perform traffic migration in situation where traffic concentrates on the server platform, and balance server workload. The fog make the middle tier of the IoT-fog-cloud infrastructure [41].

iii) **The cloud**: The cloud provides reliable services to the IoT-Fog-cloud infrastructure. These services can be accessed anywhere at anytime through the Internet. They provide support to both IoT and Fog node where necessary, specially for those tasks that need abundant resources that is not available at the lower levels of the infrastructure. The cloud avails services inform of infrastructure, platforms and software. examples of cloud services include Amazon EC2 for virtual IT, Google App Engine for Application hosting, Apple iCloud for Network storage, DigitalOcean for Servers hosting which involves Iaas and PaaS, etc. [42].

2.2.2 Fog related ecosystems

The Fog computing paradigm is poised to seamlessly lift the burden posed by IoT on the current Internet and data center infrastructure through increased agile extension of cloud services over distributed infrastructure that allow quick access and processing [43]. Extended cloud include the fog, and variation of edge computing (Multi-access Edge Computing (MEC), dew, and Mist computing) [36].

From a general perspective, the fog is super set of extended cloud technologies that is about spreading computing resources and services along the network continuum starting from the IoT nodes, the edge and the cloud [44]. Services and resources include computing, storage, control, and networking. The resources may be located within the organizational premises or in the cloud. The fog creates a middle layer between the IoT-Fog-cloud ecosystem that allow handling processing of data locally closer to the edge especially for complex and latency sensitive computing jobs. The fog also forwards process intensive tasks to the cloud and received the results before sending them to IoT device. Moreover, they may serve as gateways [41].

The Fog is defined around a network structure in which resources including data and application are strategically positioned between the data source and the cloud. Fog computing has been designed for applications that require at most latency, faster processing and quick response [6, 44, 45]. Examples of such applications are found in telemedicine, here response to patient requires at most response and available bandwidth for the medical works to keep in touch when solving emergency [46]. In health care, patients may require immediate attention from the doctor [47]. Other cases are found in machine to machine communication in industries (industrial Internet of Things (IIoT)), where reports on a component is required more often, since its failure may result in breakdown of the whole system [48].

Fog nodes are classified according to proximity, size of computing power, purpose of deployment, memory, and other computing resources. Some are deployed more closer to the IoT devices than others depending on the purpose of deployment, and some may be mobile [33]. For example, if the purpose of the fog is to combat latency issues, then the node must be deployed as close to IoT nodes as possible, in addition to possession of considerable compute property. If the fog node is meant for data storage then it should have sufficient storage property and may be deployed away from the IoT nodes. The fog Nodes are connected to the cloud and

IoT devices using communication technologies that define the topology and other communication requirements.

They are formed by at least a physical device on which some functionality of the cloud may be installed. They may include, switches, routers, min servers, desktops etc. Each of these fog nodes have capacity to provide compute, storage as well as a continuum to the cloud. The fog brings closer to the edge the services of the cloud, and provide physical services where fog computing is deployed.

The fog and edge computing paradigms are often based on similar architecture and standards. They provide distributed cloud environment and are location aware except edge computing focuses on executing computing processes at the edge of the network. Moreover, the edge may operate without a devoted cloud or fog. They are limited to a small number of peripherals through which intelligence, processing and communication is pushed to switches, routers, Radio Access network(RAN), or programmable automation controllers [49]. The edge simplifies internal communication and may use computational capabilities to make store and forward decisions to the cloud for further analysis, inopportunely they are less scaleable, built based on proprietary corporate networks with stringent conditions with less interoperability. Most often resource pooling and cloud awareness is not a design strategy in the edge.

The middle ground between the cloud, fog and the edge is the mist computing that powers light weight computing residing on network using microcontrollers and microchips. The Mist is capable of making local decision on the data and work with the cloud and the fog computing systems. Similar to edge the mist may not need to devote to the cloud infrastructure [50].

In table 2.1 we present a summary of relationships between the fog and other edge computing technologies. We consider Common edge computing technologies to include the MEC, Dew, and Cloudlets.

Table 2.1: Relationship among the fog and other edge technologies

Relationship parameters	The Fog	The Edge	Mist
Transfers processing of data closer to the user	Yes	Yes	Yes
Reduce the amount of data flooding the cloud	Yes	Yes	Yes
Decreases latency	Yes	Yes	Yes
Improves system response time	Yes	Yes	Yes
Improves security and Trust	Yes	Yes	Yes
Location awareness	Yes	Yes	Yes
Related to IoT and 5G	Yes	Yes	Yes
Architecture	Distributing resources along the network	Proprietary designed to serve base stations cell tower etc.	Design strategy is between the Fog and Edge

Relationship parameters	The Fog	The Edge	Mist
Configuration	Fairly complex, hierarchical	Simple one hop configuration	Moderate
Processing	Scaleable, handle both light weight & slightly heavy processing	Handles mostly computational processes	To larger extend handle light weight
Appropriate standards used	IEEE 1934 – IEEE Standard	ETSI GS MEC 003	—
Relevance	Evolution of local Area network, network is viewed to have multiple data points that feed the network	Evolution of telecommunication networking, devices work independently to avoid single point of failure	—
Intelligence	realized at the local area network	realized at edge gateway or appliance(devices e.g programmable automation controllers	—

2.2.3 The Fog cloud of things standards

Computing standards refer to mechanisms that enable vendors, developers and businesses resolve issues that concern homogeneity, uniformity in computing environment, interaction between heterogeneous platforms and performance. Standards are important because they permit transparency in the systems so as to support all forms of computations that happen in the ecosystem.

The fog cloud of things ecosystem supports three types of computation; a) Localized computation, b) Fog-edge computing and c) remote computing. Localized processing occurs at the local devices residing in the local area network or cellular network. They use resident resources to accomplish tasks. These resources include CPU of the IoT, memory, storage etc. Whereas, Edge, and remote computing occurs at the Edge or the cloud. The edge cloud consists of localized data center solutions that abstracts and virtualizes compute, storage and networking resources. Computational solutions at the edge must be light weight to fit the small infrastructure at central office, base stations, routers and switches. In addition, Edge solutions are required to preserve some functionality of the cloud, achieve low-latency, be simple, support automated delivery, and be relatively secure as the cloud. Examples of such solution is Contrail Edge Cloud [1] and Cisco Fog Data Services [2]. Edge computing is supported by wireless technologies such as WiFi as means of communication. Lastly, remote computation occurs in the remote data center or the cloud, remote computation is supported by powerful heavy weight systems that bare infinite resources. Communication between the cloud clients and cloud infrastructure is made possible through Internet connection [51].

The Fog cloud of things involves infrastructure setting that consist of devices and hardware from different vendors. To ensure computational efficiency and interoperability among devices and software developers, a set of standards must be established. Standards address a range of issues that include fueling compatibility, simplifying product development, and speeding up market adoption of products. They also make it easier to identify competing products in market place.

[1]https://www.juniper.net/uk/en/products-services/sdn/contrail/contrail-edge-cloud/

[2]https://www.cisco.com/c/en/us/products/cloud-systems-management/fog-data-services/index.html

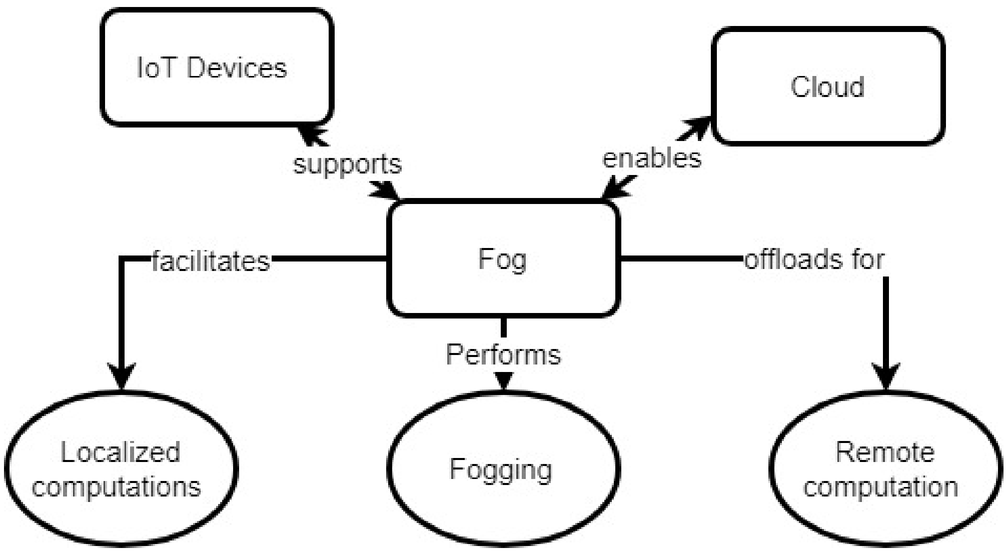

FIGURE 2.2: Types of computation that are supported in the fog cloud of things infrastructure.

In the following subsections we describe two competing standards a) the OpenFog Reference model based on IEEE 1934 [52], and b) the MEC reference Model based on ETSI 2016 [1]

The OpenFog Reference Model

IEEE 1934, is a newly approved IEEE standard that adopts the OpenFog technical framework to enable data-intensive requirement of Internet of Things, 5G and artificial intelligence applications. This standard is a blue print that will accelerate the development of new applications and business models in Fog computing environment. Moreover, the standard forms the building block for constructing solutions through consistent protocols that is understood across the board.

The OpenFog Reference model is intended to distribute compute, networking etc horizontally along the network continuum. In the same regards, it is used to define the means of extending the cloud services along multiple layers on the network topology. This yields hierarchical IoT-fog ecosystem that preserve important features of the cloud that include resource pooling, easy maintenance, availability, containerization, virtualization, orchestration, etc. These model are organized in three dimensions that describes the whole IoT-Fog-Cloud ecosystem. The dimensions are Pillars, perspectives and views. Pillars represent the theme upon which an IoT product that fulfills the demand of Vendors, users, and developers. Perspectives, represent the viewpoint of the architectures, whereas views deals with interpretation in terms of the systems, hardware and software. There are seven(7) pillars, described by the OpenFog standard, Five(5) perspectives and three(3) views. The pillars include Openness pillar, security pillar, Scalability, RAS, Agility pillar, Hierarchical pillar,and programability pillar. Similarly, the five perspective are performance, security, manageability, data analytics and control, and IT business and cross fog applications. In addition to the pillars and perspectives, the reference model includes a layered structure which is categorized into software view, system view and node view.

Using the themes described as pillars, the OpenFog reference model outlines mechanisms of resolving issues of taxonomy in terms of homogeneity, market place, and uniformity in computing environments. Moreover, standards determines matters of open API's that enables interactions between heterogeneous platforms. In particular;

The **Open pillar** enables all players i.e. the proprietors, single vendor solution providers and individuals technical groups alike to have similar access to technology. Through this pillar vendors can afford to build technology products at the same pace. This creates competitive advantage and cultivates a successful ubiquitous fog computing ecosystem for IoT platforms. The open pillar facilitates inclusion of all parties in an open form to ensure quality, reduced cost of production, and speeds up innovation in products formed in the IoT cycles. The open pillar is mainly concerned with matters of resource visibility and control, interoperability, white box decision making, data normalization etc.

Security pillar: This pillar is concerned with issues of cyber and physical security in the Fog ecosystem to deliver operational benefits, such as attestation, authentication and authorization. The ability of the Fog architecture to enforce trust, attestation and privacy is positioned in this pillar. Therefore, all means that ensure security of all the processes, data and means of transporting data are bundled under the security pillar.

The **Scalability pillar** deals with issues of localized command, control and processing, orchestration and analytics, etc. This pillar relates closely to system cost and performance. scalability allows implementations to adapt to business process.

The **RAS pillar**, deals with aspects of reliability, availability, and serviceability. This pillar embraces orchestration of existing and new resources in such a way that if a new object recognition models are trained for visual analytics, these inference engine models should be updated on near edge devices without impacting availability of the solution.

The Agility pillar is about tactical and strategic decision making, and data to wisdom processing.

The **Hierarchy pillar** deals with incorporating the fully functional cloud computations at all levels though at different scale at the lower of the hierarchy, real-time computations/analytics, in the middle of the hierarchy transactions analytics and at the top business analytics. in addition to computation, it caters for autonomy at all levels.

The **Programmability pillar** deals with programmable hardware and software, virtualization and multi-tenant and application fluidity. Visual analytics is utilized to facilitate this scenario.

The perspective of the reference model is organized in five areas as seen enumerated below

i) **Performance** It is a cross cutting concern because it has system and deployment scenario impacts time critical computing, time sensitive networking, network time protocols.

ii) **Security:** Data integrity is a special aspect of security for devices that currently lack adequate security. End-to-end security is critical to the success of all fog computing deployment scenarios.

iii) **Manageability:** Managing all aspects of fog deployments, which include RAS, DevOps, etc. Managebility is a critical aspect across all layers of a fog computing hierarchy.

iv) **Data Analytics and Control:** The ability for fog nodes to be autonomous requires localized data analytics coupled with control. The actuation/control needs to occur at the correct tier or location in the hierarchy as dictated by the given scenario. It is not always at the physical edge, but may be at a higher tier. The data analytics component is important for processing big data, Big data is the trend that is defining characteristics of many networks built today.

v) **IT Business and Cross Fog Applications**: In a multi-vendor ecosystem applications need the ability to migrate and properly operate at any level of a fog deployment's hierarchy. Applications should also have the ability to span all levels of a deployment to maximize their value. This pillar is important in bringing together businesses, and IoT-fog products.

In another dimension reference model provides three views listed below

i) **Software View**: The software view consists of the first three top layers. The Application Services, Application Support, and Node Management (IB) and Software Back plane.

ii) **System view:** forms the middle layers in architecture description, that include Hardware Virtualization, open fog Node management, open node security, Fog Node platform(network, accelerators, compute, storage etc), and Hardware Platform Infrastructure.

iii) **Node view:** is represented in the bottom two layers shown in the reference descriptor. This is mainly applicable in the IoT device the layers includes the Protocol Abstraction Layer and Sensors, Actuators, and Control.

Multi-Edge Computing reference Architecture

Mobile Edge Computing (MEC) is a reference architecture that brings computational capabilities and services closer to mobile network users by deploying computing resources at the edge of the mobile network. The MEC system level refers to the overall architecture and components involved in delivering MEC services.

MEC computing reference Architecture uses a similar framework like that used in the OpenFog. The IoT devices, the Edge devices and the cloud are organized in a hierarchy. The IoT devices form the lower layer, the MEC devices form the middle layer and the cloud forms the upper layer. Unlike the OpenFog Standard that is structured according to pillars, perspectives and views, the MEC is arranged into two levels (the Mobile Edge system level and the Mobile Edge host level). Each of the levels is composed of components that communicate with each other using reference points [53].

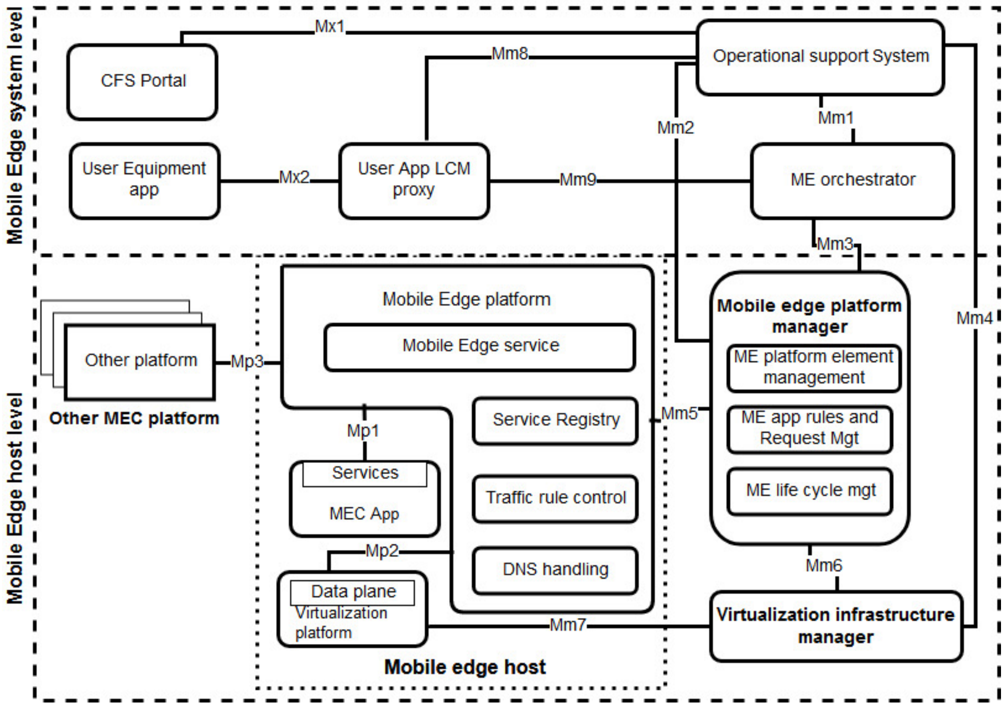

FIGURE 2.3: The MEC System architecture, adopted from [1] [2]

The Mobile Edge system level

This level consists of five (5) components; Customer Facing Services(CFS) portal, operational support system(OSS), Mobile Edge(ME) orchestrator, User App LCM proxy, and User Equipment App.

a) The CFS portal

The CFS portal is an entry point for third parties. Third parties include developers, enterprises and business clients. Developers may use this component to make applications available in the mobile operator, whereas enterprise may use the same portal to select applications of interest to them and give instructions of use of some selected applications. In addition, this portal is used to relay business related information which may include what services are available, billing, and service level agreements. Moreover, this portal is used by operations to manage provisioning, and other related services of the ME application. The CFS portal communicates to operations support system through reference point Mx1.

b) User Equipment Application (UE app)

User Equipment Application is an entity of MEC architecture at the mobile system level, UE app define applications in the User Equipment that have the capability to interact with the mobile edge system via a user application management proxy. The user application is a mobile edge application that is instantiated in the mobile edge system in response to a request of a user via an application running in the UE. UE connects to the user application lifecycle management proxy through reference point Mx2. Mx2 is used by a UE app to request the mobile edge system to run an application in the mobile edge system, or to move an application in or out of the mobile edge system. This reference point is only accessible within the mobile network. It is only available when supported by the mobile edge system.

c) The Operations Support System (OSS)

OSS is the highest management system entity of the MEC architecture that promote mobile system to run in a desired location on a network. OSS receives instructions to instantiate or terminate mobile services from both CFS portal and user equipment. Moreover, it acts as a bridge between the operator and external world, checks integrity and authenticity of application packages. It also authorizes and forwards request to ME orchestrator for further processing. The OSS receives requests via the CFS portal and from UE app using reference points Mx1 and Mm8 respectively. Reference Mx1 is used by the third-parties to request the mobile edge system to run applications in the mobile edge system and Mm8 is used to handle UE applications requests for running applications in the mobile edge system. OSS may have capacity to relocate applications between servers.

d) User Application Life cycle Management (LCM) proxy, The User Application Life Cycle Management (LCM) proxy facilitates UE applications in requesting on-boarding, instantiation, termination, and, if applicable, relocation of user applications within the mobile edge system. It also stores information about the status of user applications. This proxy authorizes requests from UE applications, interacts with the OSS and mobile edge orchestrator for processing, and is accessible exclusively within the mobile network, provided that the mobile edge system supports it.

e) Mobile Edge Orchestrator (MEO) MEO is an entity in the mobile system level which maintains an overall view of the mobile edge system based on deployed mobile edge hosts, available resources, available mobile edge services, and topology. It is responsible for on-boarding of application packages, checking the integrity and authenticity of the packages, and validating application rules. It may adjust requirements to comply with operator policies, and keeping a record of on-boarded packages. In addition, it prepares the virtualization infrastructure manager(s) to handle the applications. The MEO is tasked with the selection of appropriate mobile edge host(s) for application instantiation based on constraints, such as latency, available resources, and available services. Further, it triggers application instantiation and termination, application relocation as needed when supported. Reference point Mm1 connects the orchestrator to operation support system and Mm9 connects the orchestrator with user application life cycle management (LCM) proxy. Mm1 is used for triggering the instantiation and the termination of mobile edge applications in the mobile edge systems, whereas Mm9 is used to manage mobile edge applications requested by User Equipment app.

Mobile edge Host level

This level consists of three major components in addition to other MEC platforms. These components include the mobile edge host, mobile edge platform manager, and virtualization manager [54]. The mobile edge host is portioned into three units; the mobile Edge platform, MEC App, and virtualization platform, whereas the mobile edge platform manager is portioned in Mobile Edge platform element management unit, Mobile Edge applications and rules management, and Mobile Edge life cycle management Fig 2.3.

a) Mobile edge host Inside the Mobile edge host, the Mobile edge platform communicates with the MEC app, the virtualization infrastructure, and other MEC platforms through reference mp1, mp2, and mp3 respectively. At the same time, it communicates to ME platform manager through reference point mm5. The purpose of Mobile edge platform is to create an environment in which the mobile edge applications can discover, advertise, consume and offer mobile edge services. This services may include services available via other platforms. Normally the services are registered in the service registry. In addition, using the traffic rules received from the mobile edge platform manager the Mobile edge platform can control traffic using the traffic control module. When supported, this includes the translation of tokens representing UEs in the traffic rules into specific IP addresses. Lastly, the Mobile edge host receives DNS records from the mobile edge platform manager and configuring DNS proxy/servers. Moreover, it hosts services, provide access to persistent storage, and time of the day information using the mobile edge app service and the virtualization information platform.

The mobile edge app module is a service provider. The service provided are consumed by either the mobile edge platform, or the mobile edge applications. Applications may be registered in the list of services to the mobile edge platform over the Mp1 reference point. The MEC app services avails compute, storage, network resources and services in MEC ecosystem.

Virtualization infrastructure module; This module contains the data plane whose responsibility is to provide information about traffic, routing and forwarding protocols between the applications and services on the network.

b) The Mobile Edge platform manager(MEPM) The Mobile Edge platform manager(MEPM); this is an entity of MEC host level that is used to perform the management utilities. Using ME life cycle module, MEPM starts and terminates applications based on instantiation and termination procedures. Further, it provides indication to the orchestrator about the application events as well as perform authorization and system conflict resolution.

MEPM uses reference point Mm5 for policy and platform configuration so as to manage reallocation of application, perform traffic filtering and support application life cycle procedures. MEPM uses reference point Mm2 for performance management of MEC platform and other configurations that may include Fault recovery. Similarly, reference point Mm3 is used to maintain up-to-date information on available ME services, provision application related policies and life cycle supervision [1].

c) Virtualization Infrastructure Manager(VIM) Virtualization Infrastructure Manager(VIM) is an entity of MEC infrastructure at the mobile host level. The purpose of VIM is to manage the virtualized resources for the ME applications. This includes allocating and releasing virtualized computing, storage, and network resources provided by the virtualization infrastructure, in addition the VIM also prepares the virtualization infrastructure to run software images. The VIM is also used to sustain faults and monitor performance. Performance monitoring is completed by collecting and reporting information on virtualized resources and providing the information further to server and system level management entities. The VIM uses reference points Mm7 to manage the virtualized resources. Reference point Mm4 is used for management of the application images and the virtualized resources as well as for monitoring the availability of the resources.

Reference point Mm6 is used to manage the virtualized resources for the ME applications during their life cycle.

In general the framework presented in fig. 2.3 describes both the hardware and software components of multi-access edge computing facility. Each of the components are connected to each other through reference points. Reference points Mp1-Mp3 are related to mobile edge platform, whereas reference Mm1-Mm9 are related to mobile Edge management, while Mx1 and Mx2 are related to external entities.

2.3 Emerging trends, issues, and challenges in fog cloud of things and ideal solutions

Increasing digitization and government initiatives ranging from low income to high income countries is likely to increase the use of fog computing in their smart innovations, health care, crowdsourcing, smart energy, smart utility and environmental monitoring significantly. Smart cities are potential candidates for fog computing because they enable control, monitoring, security and surveillance systems [55]. Whereas, In health industry adoption of fog computing and related health applications using IoT is becoming acceptable as means of improving the overall performance of the health industry. Subsequently fogging encourages better operation efficiency, reduce operational cost and optimize power consumption [56]. On the other hand, Crowdsourcing is a visible future prospect of fog computing given that crowdsourcing involves individuals using their smart devices to participate in tasks related to some locations in the physical world [57, 58].

In this section we discuss the emerging trends, issues, challenges and ideal solutions in the fog cloud of things ecosystems. After careful review of papers related to fog computing domain, 20 papers were selected to position literature in relation to novelty of the current fog models. We categorized the papers in three categories. Seven in the smart city innovation category, seven in the smart healthcare , and the remaining papers were categorized in the crowdsourcing category.

In the smart city domain, there is increased need to manage the vast information communication resources (IoT, and Fog devices), perform real-time analytics of big data closer to users, and ensuring security of both the infrastructure and information generated by the smart city systems. Issues of node failure due to slow convergence or malicious attack, reliability and availability of services due to traffic congestion, processing massive amounts of data in real-time at the edge of network so as to improve quality of services, performance of smart city infrastructure in terms of reliability, robustness and endurance are yet to be completely resolved.

In [59] a multi-tier fog computing model based analytics service for smart city applications was proposed. In the study, the authors explain how their multi-tier fog computing paradigm provide a fertile platform for analytics of both light weight and heavy workloads. From their experimental results it was concluded that heavy workloads required dedicated computing resource. Moreover, it was noted that large scale analytics can be achieved with multi-tier fog architecture. In a similar study [34] the author highlighted the fact that different IoT applications need different levels of intelligence and efficiency in processing data. They presented a case study SLAM(Simultaneous localization and Mapping) which involves delay sensitive processing of large amounts of data from heterogeneous sensors in real time. Such future internet of things could be deployed widely in many places in the real world. This therefore, calls for multi-tier framework which are not only hierarchical but can support fog-to-fog communication across the fog ecosystem that are applicable not only to smart city applications but also to health, and other related use cases.

Authors in [33] presented machine learning based computational offloading framework involving neural-fuzzy model to improve robustness of fog system and achieve offloading by incorporating PSO and reinforcement learning. From their results better performance and robustness in fog system can be achieved using machine learning at all levels of the fog hierarchy.

Authors in [60] proposed multi-level system of fog computing services for end-to-end support of IoT applications framework, their work is motivated by services that need a vast range of end devices and the tremendous amount of data. This framework is visualized to wrap all fog resources(hardware, software and application functionalities) as micro services. Results from their experiment show that FA2ST responds dynamically to end users' demands based on available services. Models such as FA2ST are seen to be applicable to smart city innovation to improve quality of service and experience.

It is challenging in smart city innovation to obtain and batch process data with multiple dimension from heterogeneous sources, which may need different levels of intelligence[61]. In addition, social media is becoming more acceptable means of acquiring data to analyse citizen engagement, behavior, and sentiments about services provided by city authorities [62].

To improve performance of smart city fog related infrastructure, applications of machine learning, artificial intelligence, software defined networking etc. in the fog cloud ecosystem is indispensable. Zhang et al. in [63] proposed fog based cognitive computing framework which derives its intelligence from collaborative sensing, cognitive services, and applications. Similar works is found in [64], in their work they introduce the overall framework for intelligence in smart cities consisting of three levels of intelligence: smart city and IoT infrastructure, fog computing, and cloud computing. From their framework they illustrate the overall position of machine learning approaches within the hierarchy of smart city infrastructure. Machine learning, AI, and cognitive models are likely to bring efficient computing, storage and reliability and robustness in the fog cloud ecosystem especially in smart city innovations.

In health industry adoption of fog computing and related health applications using IoT is becoming acceptable as means of improving the overall performance of the industry. Fog and IoT systems encourages better operation efficiency, reduce operational cost and optimize power consumption [56]. The nature of health care applications and data in terms of privacy, delay, sensitivity, real time analytics, and permitting immediate action are drivers that encourage increased use of fog computing. Studies in [56] show that substantial solutions for health purposes can be developed to use the fog. Consequently the future is going to see complex solutions to improve health care services develop at the fog. For example authors in [65] developed a framework for health and wellness applications. In their work they considered a three layer fog architecture with the lower layer representing hospitals and clinical organisations, smart homes and individual patients equipped with sensors and actuators. The second layer consists of the fog and the components that collect process, and secure data. Whereas, the top layer consist of the cloud and end user service providers (medical institutions, insurance companies etc.). The proposed framework enables data outsourcing, querying, searching and extraction of responses etc. From their results better performance was observed. Moreover, the framework provides a platform that may handle complex security issues.

To achieve precise processing of health data and make real-time decision across wide geographical coverage without compromising the sensitivity of the data for future health applications is difficult. It is challenging to efficiently deliver sensitive data over the network infrastructure, difficulty in managing system failure, security, multiple system configurations, and heterogeneous systems with multiple dimension. Authors in [66] proposed a framework to improve throughput and latency for efficient analysis and transmission of geo-health data, they laid an analysis of energy saving and computational cost of their proposed architecture. Their result demonstrate autonomous operation even in absence of Internet connectivity, easy to deploy and distributed fog nodes are seen to improve performance, throughput utilization, manageability of the system with less expertise. Utilization and load balancing have been considered in [67], while energy efficiency in health system has been explored in [68] and performance issues have been deliberated in [69]. From these works there is a considerable effort in improving fog computing in health industry so that sensitive and bulky data from clients and service providers can be managed effectively. It is feasible for the future fog health systems to solicit the use of crowd sources and social media data to predict wellness of patients in real-time.

Crowdsourcing is another visible future prospects of fog computing since it involves individuals to participate in tasks related to some locations in the physical world. Although there is still a challenge in allocating resources for crowd sourcing with at most precision there has been tremendous acceptance of crowd sourced services amongst users. In addition to the vision of autonomous vehicles and drones, the future use of fog in refining crowd sourcing applications cannot be under estimated. In [58] crowdsourcing-based disaster management (DMFC) was presented. Their model is hoped to help authorities to plan efficient rescue protocol for disaster struck places and takes advantage of crowdsourced data by performing real-time analytics at the fog so as to relay immediate feedback to save lives in disaster struck locations. Their model is seen to minimize latency. Moreover, they propose blockchain mechanism to offload disaster data. In similar study [70] a framework known as SAFER that concentrates on instances of data classification and analysis to find emergency instances was proposed. They used Software Defined Network(SDN) technology to build efficient and dynamic network. Crowdsourcing based fog of things is likely to dominate future smart cities. smart health, smart agriculture etc.

In fog based crowdsourcing applications it is challenging to rely on the data collected from individual persons than sensed data. In the same vein, issues related to social mining at the fog and quality of the overall information collected arises [71]. The quality of information can be improved through application of approaches that encourage autonomous learning based on context learning, reinforcement learning, artificial intelligence, knowledge discovery, self healing, etc.

Generally, overtime there is steady increase in the research interest in discovering the use of fog computing in smart energy systems, smart farming and agriculture, transportation and autonomous vehicles, environmental monitoring, etc. The many challenges fog inherits from its distributed nature, which include difficulty in managing system failure, security, multiple system configurations, and heterogeneous systems with multiple hardware and software requirements still hold fog computing one step from maturity.

2.4 Taxonomy for fog cloud of things

Table 2.2 presents our taxonomy for fog cloud of things based on emerging novel issues of the existing works. We highlight the five major aspects of fog-cloud of things that allows us to view holistically fog computing in terms of their applications, management, security and privacy, collaboration and communication, and architectures and systems as follows:

i) Applications

   A considerable number of applications that require fog as a service in terms of fog platforms, fog infrastructure, or services have become common. This requirement has grown exponentially because of increase in the number of applications that require high latency, storage, and privacy requirements. As a result new issues which require intelligence in balancing and scheduling services, data distribution, content awareness enable applications of Fog-cloud to be taxonomized in computational offloading, data analytics at the edge, content aware delivery Fog systems, and intelligent crowdsourcing systems.

ii) Management

   The distributed nature of fog-cloud of things require sophisticated management of resources, and data so as to minimize long term resource utilization such as power consumption, memory and storage. In addition, data control and service placement is becoming complicated as the number of devices and service requirements for IoT increase. This enables us to recognise fog-cloud of things management classes based on resource management, data management and affinity management.

iii) Security and privacy The flexibility in deployment of fog-cloud of things, coupled with massive amounts of data generated and mobility characteristics of thing results into new security and privacy concerns that are not effectively handled by the existing detection and trust mechanisms. For this reason, the need to improve security control, and efficient monitoring of devices behavior have risen in the new IoT ecosystem. In addition, extending the new authentication mechanism to all thing across the the fog-cloud ecosystems is important. The new breed of security and privacy concerns have enable us to identify classes in the fog-cloud of things security falling into intelligent detection, trust management and adoption of new mechanisms that may use Fog-blockchain concept in achieving security in fog systems.

iv) Communication and collaboration

Most of the fog-cloud of things architectures have not resolved the issues of communication and collaboration between Fog2Fog and Fog2Other IoT devices completely. As a result many algorithms that are based on optimizing resources for communication need to be improved especially in the context of resource awareness and collaborative utilization of resources, therefore two classes have been identified i.e Fog2fog, and Fog2cloud.

v) Architectures and systems

Architectures of Fog cloud of things illustrate the system components, and explain their roles in the ecosystem. This enables experts to describe how devices are capable to communicate amongst themselves. Through this architectures the responsibility of Fog nodes and cloud in ensuring services, service placements, service delegation between components of the ecosystems are explained. Based on the purpose of the architecture two common classes of architectures are identified i.e. single-tier and multi-tier architectures.

Table 2.2: Taxonomy for fog cloud of things

Articles	Domain	Sub domain	Emerging issues	Category of Problem
[33, 72–75]	I) Application	A) computational offloading, B) data analytics C) Content Delivery Network D) crowdsourcing	a) Load balancing and scheduling, b) Intelligent task and data distribution algorithms c) Distributed data processing at the edge e) Content aware protocols at the edge f) Content localization	i) Resource (CPU, memory and bandwidth) utilization ii) Overall minimization of response time, latency, energy etc. iii) Performance improvement in terms of fairness. iv) Throughput and application delay v) Localized content delivery services, vi) Content aware Optimized Link State Routing
[76–79]	II) Management	A) Resource mgt B) Data mgt C) Affinity mgt	a) Intelligent decision aimed at minimizing long term power consumption b) Resolving resource coupling problems c) Dynamic data plane control d) Service placement to maximize efficiency	i) Designing more efficient minimum cost scheduler and maximum resource utilization algorithms ii) Using machine learning to improve resource coupling problem iii) Efficient data plane control mechanism iv) Service selection and placement in the fog v) Continuous improvement of systems resources management to attain better latency, resource utilisation

Articles	Domain	Sub domain	Emerging issues	Category of Problem
[80–83]	III) Security and Privacy	A) Attack detection B) Trust management C) Blockchain driven secure mechanisms	a) Cybersecurity control in increased surface attack b) Behaviour monitoring d) Blockchain enabled user authentication and secure management	i) Improving machine learning attack detection mechanism ii) Detection of malicious node iii) recovering misjudgement based on trust score iv) Evaluation strategies based on service records v) blockchain as a service
[84–86]	IV) Communication and collaboration	A) Fog to fog B) Fog2Cloud	a) Efficient algorithms to permit fog2fog communication b) Information Centric Fog to Fog communication	i) Communication technologies that allow fog2fog ii) minimizing the overall end to end latency iii) the use of Information-Centric Network specific protocols over the MAC layer iv) performance of adaptive low-latency Medium Access Control (MAC) layer scheduling among IoT devices.
[82, 87–91, 91, 92]	V) Architectures and systems	A) Single Tier ii) N-Tier	a) Reflection on architectures and functional components, b) applications, c) security and privacy	i) Connecting things that seamless compute, route, and communicate through diverse ecosystems ii) Reflection on functional components and applications, security and privacy concerns.

2.5 Applications

The key applications driving fog computing paradigms include computational offloading, Big data analytics, distributed content delivery and caching, improved web performance, smart city applications, crowd sourcing, etc.

2.5.1 Computational offloading

Computational offloading is a technique used by resource-constrained IoT devices to reserve computational processing, battery life, storage, etc. Offloading is achieved by partially or fully pushing the intensive task to resource sufficient environment. The cloud has been the most significant environment to achieve offloading until the upcoming of use cases that require better performance and efficiency in terms of immediate reaction to the data generated by the sensed environment.

Computational offloading is mostly encouraged to reduce latency, save energy, prolong battery life at the same time improve user quality of experience. The inability of IoT devices to execute heavy applications call for edge devices to become computing resource power horse to enable execution of heavy tasks. They run sophisticated applications offloaded by the IoT devices [93].

Computational offloading has attracted a series of research interest to solve challenges related to offloading. These challenges include (i) resolving issues of diversity and heterogeneity in both applications and platforms (ii) finding factors that impact offloading decisions and processes especially for new use cases (iii) design strategies that are lightweight and can run on edge devices without being limited by processing power and power supplies. All the above form new challenges that require improvements in the current solutions available [33].

Authors in [4, 16, 33], have proposed offloading methods based on machine learning strategies to further improve performance, energy harvesting, resource utilization, etc. [94] explores dependence problem between individual parts of a task that should be offloaded by developing a code profiler and decision engine responsible for offloading, authors in [95] look at the optimization of energy and delay in fog computing. Lastly, authors in [49] performed a comprehensive review on offloading frameworks, analyze computation offloading technique and provides open issues in computational offloading in the MCC environments.

2.5.2 Big data analytics

A new paradigm shift from a data consuming network to data producing network has been created recently, thanks to IoT, Machine learning, AI and social media platforms. The internet generates a massive amount of data in different dimensions and having different formats causing us to enter a data-driven epoch [96]. IoT is a significant enabler of big data, for example on average 500 tweets, 300 billion emails, and close to 65 billion WhatsApp messages are sent every day using smartphones. If consideration is given to connected things, internet searches, activity logs, surveillance and guiding tools over the internet, the amount of data whose magnitude is very high is produced. In addition, the biggest portion of data produced every day is unstructured. The remaining small portion is structured, analyzed for decision making. Stack in [97] mentions that only 1% of structured data produced over the internet is analyzed and used. IoT infrastructure has made it possible to collect data from toys, fridges, cookers, security cameras, etc., making reliance on the cloud, data centers and in-house data processing and storage alone almost impossible.

The IoT view of connected devices to improve quality of life by consuming the result generated from intelligent analytics on the data produced by devices require that intelligence be spread along with the network. Real-time analytics can be realized in the device, transactional analytics at the Fog and business intelligence at the cloud. Since most IoT devices are resource-constrained, it is quite challenging to realize real-time analytics for a long time. The Fog creates an intermediary form of analytics known as transaction analytics. This service can speed up processing and data delivery requiring less network overhead as compared to the cloud [98]. Similarly, data filtering can occur at the edge to reduce data transaction burden at the core.

Fogging introduces gateway servers, cloudlets, fog nodes and micro data centers that act as negotiation points for data points along the IoT-Fog-cloud continuum. Spreading data along multiple points on the network may create multiple challenges in creating analytics model in one location and executed in another location or multiple locations. These challenges will see a rise in new Fog products such as

Function as a service (FaaS), where Software developers can leverage serverless to deploy an individual "function", action, or piece of business logic pushed to the edge [99]. FaaS may enable fast transaction on a high volume of data at the edge, provide for Dynamic or burstable workloads in such a way that users can pay only for services rendered at a specific instance of the function and enable codes to be scheduled at a particular time. Further, Backend as a service (MBaaS) [100], may be made available to facilitate IoT user authentication, data persistence, file storage, messaging and custom business logic that is good enough to perform analytics at the edge without worrying about server infrastructure. Generally, as edge computing matures the influence of IoT on Big Data will impact on the existing framework of analytics. EveryThing-as-a-Service (ETaaS) may be pushed to the edge and IoT devices at the leaf end.

2.5.3 Distributed content delivery and caching:

Millions of video content is transmitted over the internet per second in form of Internet videos, IP videos on demand, video-stream gaming, online education video content generated for education and entertainment, and video conferences etc. These contents generated contain new file formats that vary in size, encoding etc. The assorted nature of these file present needs to handle storage, caching, and delivery of content generated using technologies that will enable distributed designs and processing closer to the source of data.

The biggest part of content over the internet is video. Internet video delivered over mobile video and television has taken a precedence [101]. This is mainly due to Over-The-Top (OTT) applications. OTT enables users to deliver audio, video and other media over the internet at reduced cost, with less skills, independent of technology affiliations and permission from telecommunication operators. Cisco systems in [7] predicted that by 2021, the total Internet traffic had approximated 35GB/person per month and as of 2018, In 2020, approximately 2.5 quintillion bytes of data is generated every day, which is was estimated to be around 44 zettabytes. By 2025, the number will reach 463 exabytes globally.

Content Delivery Networks enable contents of any format to be stored in central database, transcoded from the source format before final delivery. They are used to distribute content through several streaming servers located on different locations in diverse networks. Though enough efforts have been dedicated to improve the distribution of video content, users still experience service interruption caused by network delays, buffering and jitters caused by absence of content in close proximity to users.

A number of research efforts have confirmed that using the Fog to Extend the Content Delivery Network (CDN) services closer to the users is one way of improving the user QoE, relieve the back haul and core network from heavy use. Particularly, work in [102], proposed considerable flavors of architectures for accommodating distributed parallel edges capable of performing video caching and streaming to increase QoE for content delivery. Similarly, caching content may result in reducing the capacity requirements by up to 35% [103]. For example, Media Cloud[3], proposed an elastic virtual content delivery network at the edge pre-caching at the edge popular content according to statistics and user/service forecasting intelligence. Providers can take full advantage of this to provide services build upon OTT applications. A context-aware network with edge cloud caching capabilities based-on big data analytics is considered in work in [104], content placement and delivery approach based on the prediction of content popularity and the characteristics of the wireless environment has been looked at in[105].

2.5.4 Improved web performance.

Web performance is mainly centered on three concepts; (a) content optimization, (b) augmented browsing, and (c)web acceleration [106]. Web performance is a trademark for web sites because it is about retaining users in a sea of content, improving conversions and maintaining quality of experience. Every individual who builds a website (personal blog or business sites) hopes that people visit it. Every venture that is posted online requires to be visited. Slow sites are likely to put off user resulting in a loss of viewership and hence the business. Heavy websites with unnecessarily and not customized content attract costs that users are not ready to bare. Using the Edge content localization and customization it is possible to selectively send content to individuals who find them beneficial. The Fog brings web performance utilities closer to the user. Web performance is viewed in the following three dimensions

[3](https://mediacloud.org/)

i) **Content Optimization:** is an important component of web performance as it allows a web page and its content to be useful, attractive, and actionable. Web optimization includes fixes and improvements on technical performance but for web page speed. It provides improved performance and better ranking. With the current drive of 10x content, content optimization become a very competitive subject. Hosting content optimizer at the edge provides a success path for both users and organizations by ensuring page-level audits based on user experience, analyze what customers are doing for each query/page based on geographical location, re-optimizing the web and other assets

ii) **Augmented Browsing:** A typical web fetch cycle involve a request being sent to the server, processing the information, before filtering and downloading the response to the device. Hardware-based augmented browsing can be achieved by installing GPU at the edge closer to the user to help the IoT devices in speeding the fetch cycles [107]. Solutions such as accelerated web browsing proposed in [108] are more useful to achieve augmented browsing.

iii) **Web acceleration:** It is not uncommon today for websites to contain large files, ranging from text, photos, video, and multimedia. The web usage can be personal and fragmented. The size of content, the processing power of the devices in addition challenged network environments greatly affect the loading time of web content [109]. Content define network based at the server provide a better alternative to achieve faster content loading. Web acceleration at the Fog may be achieved through catching particular content [106]. Once the content has been downloaded on the Fog after an initial request, all the subsequent request are browsed using store and forward mechanism.

2.5.5 Smart city applications

Cities are traditional places constructed around business, industries and other commercial activities. Because of the services and infrastructure they become powerful places where governments stage important activities that determine the well-being of societies. There unique characteristics make it easy for them to adapt and adopt changes very quickly. Recently, increase in city populations, change in life style driven by the use of technologies such as smart phones, arrival of big data analytics and internet of things has resulted into concept of smart cities.

The concept of smart cities is hinged on adopting IoT applications and other technologies in efforts to increase efficient use of city infrastructure, improve reliability, responsiveness, cut down cost, and increase innovation in city environment. In the sphere of smart cities, many applications have appeared cutting across infrastructure, services, and communication technologies. Some of examples of applications in smart cities are smart city lighting, smart waste management, smart infrastructure monitoring, smart water management, and smart energy management among others [110].

Key drivers of smart city applications are is due to the integration of ICT's infrastructures into a city to improve quality of life of citizens. Increase in the population that calls for efficiency and optimization of city resources, availability of big data that gives insight to many activities that are happening in cities, etc. [33].

2.5.6 Farm applications

Like any other aspects of life, The Internet of things and connected things have found many applications in agriculture. These applications have provided a significant face-lift on how farmers conduct their activities on farms. Among these include achieving a better control over produce, maintain standards of farm production, expand the market place and monitoring of crop and life stock health in real time. Use cases in this area include monitoring and predicting climate conditions, crop management, and life stock management among other applications [41, 111].

2.6 The Fog cloud of things tools

Research and development in the IoT-Fog-cloud is required to produce tangible products that leverages and exploits products manufacturer, industries and service providers to create smart environment. Ideally, IoT-Fog-cloud research process begins with development of ideas and ends with products deployment. Deployed products contains both virtual and physical components. The virtual component of IoT products is achieved through proof of concept realized through simulation, whereas physical components of the product is achieved through prototyping realized by testbed experiments [112].

In this section we provide an overview of existing research tools that support proof of concept and prototyping phases in the IoT-Fog-cloud. This is done by examining some of the simulators, emulators and large scale open test beds in the IoT-fog-cloud system.

2.6.1 Simulation tools

Simulation tools are used in Proof of Concept(PoC). During the PoC, assumptions are tested so as to confirm that the idea is feasible, viable, and applicable in real world. PoC helps to avoid possible technical problems in the future, and allow developers obtain valuable feedback at an early stage of the development cycle, thus reducing unnecessary risks.

Simulation tools should be able to handle heterogeneous situations, scalability, computational efficiency, reiteration of experiment and reliability. There three (3) kinds of simulation tools in IoT-Fog-cloud ecosystems a) those that simulate the whole infrastructure in response to emerging situations and provide end-to-end support. These simulators predict the behavior of a computer network and provide an accurate understanding of system behavior in form of statistics, b) the second form are those that are used to visualize the data generated from the infrastructure and perform analytics, and c) and those that emulate the network. Simulations offer two important advantages to researchers, the first is that researchers can proof that their idea is really worth making the effort to implement, second is that they can save money by avoiding to spend a budget on what is not viable. To achieve successful simulation, researchers should consider the tools ease to use and approachabilty, support, accuracy of simulation results, its robustness, and cost. Below we discus examples of simulation tools used in simulating IoT-fog-cloud ecosystems and present consideration for selecting simulation tools in table 2.3.

iFogSim

iFogSim[4] is an open source simulation tool for fog, edge, and IoT modeling and simulation. It integrates resource management techniques customizable based on

[4]https://github.com/Cloudslab/iFogSim.

research areas. Its connotation with cloudsim enables it to simulate cloud scenarios. It is easy to install and use. iFogSim provides classes written in Java that can simulate devices(Fogs, Sensors, actuators , etc), applications, tuples, resource management, and monitor the edge [113].

MobIoTSim

MobIoTSim[5] is an Android application for simulating an Internet of Things environment. Using MobIoTSim researchers can create and configure devices that are simulated. This is done by setting the cloud and other devices along IoT-Fog-cloud hierarchy. This simulation tool is easy to use, and has ability to simulate hundreds of customizable devices, the cloud gateway and achieve services used to gather real-world sensor data. It enables low level configurations, collect statistics based on elapsed time during the execution of some functions. MobIoTSim supports protocols such as MQTT for sending and receiving data [114].

ns−3 Network Simulator

ns-3 [6] is a popular discrete-event network simulation tool publically available for network research and development in the computer network domain. It provides simulation core that are well documented and easy to use. ns−3 avails the entire simulation work flow starting with configuration to trace collection and analysis. ns−3 supports both IP and non-IP network simulation which are sufficiently realistic involving models for wired and wireless networks systems at layer 1 and 2, moreover ns−3 can be used as real−time scheduler for interacting with real systems. It may serve as an interconnecting framework that link between virtual machines. Availability of vast amount of tools allow developers to create solutions in C/C++, java and other programming platforms that simulate fully functional IoT-fog-cloud environment. Although there exists no explicit ns−3 IoT-Fog-cloud community as of yet, support can be solicited from all over other ns−3 communities in collaborative manner [115].

In general, there are myriads of simulation tools available that can be used to simulate IoT-Fog-cloud systems. The choice of what tool to use in any research depends on simplicity, reliability and reputability of simulation environment. The researchers level of confidence, expertise, comfort and ingenuity in the use of simulation tool play an important role in the success of PoC.

[5]https://github.com/sed-szeged/MobIoTSim.
[6]https://www.nsnam.org/

Table 2.3: Factors considered when selecting simulation tool

Considerations	iFogsim	MobIoT	ns-3
Installation and approachabilty	Open source and easy to install	Android based	Open sources discrete event
Online support and tutorial	Comprehensive	Available	Very comprehensive
Support for design analysis	All except Fog_to_fog	support many customisable	Handle multitudes of design analysis
Accuracy of results	accurate results	Accurate	Accurate results
Cost	Free	Free	Free
Support for well documented models	supported	Well surported	

2.6.2 Emulators

The primary goal of network emulation is to create an environment whereby users can connect the devices, applications, products and/or services being tested in order to validate their performance, stability, or functionality against real-world network scenario.

There are two categories of network emulators that are used in exploring Fog-cloud ecosystems (a) hardware based emulators, and (b) software based emulators [7]. Each category of emulators differ from the other based on performance and precision [116]. In table 2.4 we shows factors considered when making a choice of an emulator to use in research endeavours .

Hardware emulators are built from scratch to include field-programmable gate array (FPGA) responsible for processing traffic. FPGAs isolate the CPU from handling network packets. Including FPGAs in the hardware emulator guarantees performance. Network processes in this situation is not affected by decrease in size of packets and/or the increase in the number of peripherals attached to the hardware. They allow researchers to perform experiments with required precision and

[7]in other literature referred to as appliance based emulators

can allow reiterating experiments under the same condition to reach a conclusive stand [117].

On the other hand, software based emulators are built based on personal computers with emulation software installed on them. unlike the hardware based simulators, in software based emulation environment the CPU is shared across all other ports attached including those that are not meant for networking processes. Since the CPU is responsible for managing all activities such as managing HW, interrupts etc. the priorities of the emulator software may be differed causing deficiencies in the results. In the same vein, it is difficult to separate defects caused by CPU from those caused by Network activities. Moreover, repeating the experiment under the same condition may be difficult to achieve. Without repeating the experiment under a similar environment makes it difficult to troubleshoot and validate test results [116].

Hardware based emulators may be expensive and may require more explicit knowledge, but provide better experimental conditions with advanced features that address issues of scalability, packet corruption, modification, loss of packets etc. in addition they support multiple network protocols, provide real−time statistics and detailed analysis and allow the researcher to set the same experiment under the same condition. Hardware based emulators allow precise measurement of network performance matrices such as throughput, delay, energy consumption, resource utilization and response time.

Table 2.4: Factors considered when selecting emulation tool

Considerations	Hardware emulators	Software emulators
Processing network traffic	Uses FGPA	CPU
Accuracy of results	Guaranted	Estimated
Repeating experiments	Occurs in same conditions	may not in same conditions
Cost	Expensive	Cheap
Robustness	Restricted by Hardware	Robust

2.6.3 Test beds

Simulations provide an excellent environment to start thinking of a product, however the ideas thought over in proof of concept must be verified. To this purpose test bed experiment must be implemented. Testbeds provide opportunities and platform for innovation of new products in the IoT research. Through these facilities ideas can be nurtured and tested. The results of the testbeds are used as feedback to guide and inform invention of better products in the market.

Some examples of open testbeds are (a) Asset efficiency testbed [8]; this testbed is supported by Bosch, IBM, Intel, etc. and led by Information technology consulting company (Infosys). The testbed is formed to support HighTech industrial in manufacturing. It is aimed at collecting assets information efficiently and run analytics for rational decision making. (b) Connected care testbed [9] is another testbed supported by PTC computer software and services company and led by Infosys and other partners. It concentrates on open IoT ecosystem for clinical and remote medical devices. It is aimed at IoT products that bring patients data into a single data management platform. These two examples are drawn from industrial consortium, which list a number of testbeds in the IIoT.

Universities in different parts of the world have created testbeds that are open for use by researchers to further their ideas, examples of such testbeds include National Chiao university Testbed and Shangai Tech Testbed described in [118].

2.6.4 Resource Hubs

Resource hubs provide libraries of knowledge, tools and good practice, which developers use to plan, develop, and manage their own IoT projects. They consist of collaborative resources that have been tested and developed by both academia and industry. These resources may be accessed through white papers, insight reports, and models. Some resource hubs include methods for creating IoT-fog projects, and modelling tools. Moreover, some resource hubs include directories of leading experts in the IoT domain such as engineers, project managers, programmers, and product managers from whom help can be solicited during development of

[8]https://www.iiconsortium.org/asset-efficiency.htm
[9]https://www.iiconsortium.org/connected-care.htm

ideas in IoT-fog research. Lastly, resource hubs may contain codes, tutorials, frequently asked questions, and videos used as guidance for beginners. An example of resource hub is the ICC resource hub [10]

2.7 Open issues in fog cloud of thing ecosystems

As the use of fog cloud of things ecosystems is gaining substantial interest a number of frameworks have been developed and tested in the research community. There is a clear roadmap to improve performance, robustness and security and trust in many ways. In the same way efforts are being positioned to solve methods of Fog2Fog communication, boosting Fog based intelligence using Artificial intelligence and machine learning, improving resource management among the fog nodes, bringing real−time analytics of data including analytics of imbalanced data at the edge, enhancing fog based learning, and handling trust and security using mechanisms such as blockchain at all levels of the fog cloud ecosystem.

Most studies conducted so far have placed considerable efforts to solve IoT2Fog and fog2cloud communication, examples of such studies are found in [119–121]. In considering multi-tier fog paradigms it is necessary to discover efficient mechanisms for Fog2Fog communication. Authors in [14] and [84] have presented analytical insights into fog to fog scenarios. From their work, round trip delays are minimized. As a result of the frameworks and formalising fog2fog communication, tuning the frameworks such that more parameters such as Fog2Fog selection, task prediction from historical data, Fog2fog resource utilization, Fog2fog scalability, robustness and mobility of the fog nodes remains open issues. Moreover, validation of the models through simulations and real life implementation of these F2F models is requirement.

Achieving intelligence in in the IoT infrastructure that mimics cognitive mechanisms is another open area of research. In this respect issues that deal with obtaining data that is effective and valuable for heuristic algorithms such as Particle swarm optimization, colony optimization, reinforcement learning, and genetic algorithms remain open issues. Other upcoming open issues include support for social internet of things in the fog-cloud infrastructure, more specifically performing

[10]https://www.iiconsortium.org/resource-hub.htm

real time analytics on crowdsourced data to realize sentiments, eliminate inconsistencies and dealing with imbalanced data at the fog for critical applications are still open.

Other issues that have been explored but require further understanding include performance of fog ecosystem, robustness, storage, energy conservation, handling distributed fog through a combination of device to device (D2D) and cellular network, mobility and node selection, trust and security issues using new technologies at all level of fog paradigm are concerns to be handled at length by the researchers. As researchers are looking forward to kickoff 6G to revolutionize the latest wireless innovation, adoption of the fog cloud ecosystem using a wide range of Artificial intelligence, machine learning, blockchain, heuristic and genetic algorithm is a necessary trend in the IoT research.

2.8 Summary

In this article, we provide a survey on IoT Fog cloud ecosystems. We review concepts, standards and tools that are used in exploring IoT and its infrastructure setting. We continue to discuss the emerging trends, issues, challenges and ideal solutions in the fog cloud of things ecosystems, and include a taxonomy based on emerging issues. Besides, we present and described some applications that find fogging attractive. From this survey, we conclude that fogging provides a richer platform to provide services among resource-constrained and latency-sensitive connected systems. Although the question of coexistence in standards still exists; it is apparent that standards harmonize the development of IoT products and accelerates development in the fog computing environment. The use of machine learning and artificial intelligence in the fog will see the fog ecosystems perform better. Thus, this survey positions our proposals in chapter 3. by presenting the gaps, choice of tools to develop the proposal, and applications to which the proposal are deemed fit. This chapter contributes a new taxonomy based on the fog standards that is presented herein.

Chapter 3

Machine Learning Based Secure Offloading in Fog-Cloud of Things

3.1 Introduction

The emergency of ubiquitous and pervasive things have resulted in production of very extensive amount of data; as a result, data processing requirements in IoT ecosystems is growing much more faster than processing power, memory, cache, and battery life of the devices [122]. Cisco Global Cloud Index estimates that nearly 85.0 ZB was generated and consumed by all people, machines, and things by 2018 up from 22.0 ZB generated in 2016; of the 85.0ZB expected only 10% was useful and the rest will be ephemeral in nature. Again, useful data will be four times greater than the existing capacity of data centers of the time [7]. The trends of data cited show 77.5 exabytes of data will be produced per month by IoT devices in 2025, all the above present opportunities for the use of Edge and Fog computing.

Fog computing environment enables computational offloading, data aggregation and storage. This allows Internet of Things (IoT) devices to provide users with satisfactory quality of service and quality of experience [16, 123, 124]. Numerous data intensive applications have been developed to employ the use of smart devices. Some examples of applications include self-parking cars, wearable devices, trackers and domestic appliances [125]. Most of these devices are small in size, they are battery powered, and house limited processing, storage, and memory supplies.

Their size act as a bottleneck to implement and run powerful applications on them. We also note that the amount of data generated by IoT devices is increasing gradually, thereby increasing the network burden in terms of network congestion. Hence, it is necessary to utilize on-demand resourceful cloud paradigm that enables to process the data generated from IoT devices. This paradigm is often called Cloud of Things (CoT) [126, 127] .

Cloud computing paradigm provides extensive processing power and infinite storage that permits fast processing and bulk storage. It has been found useful in many applications that are not delay sensitive and do not require immediate responsiveness. However, the use of cloud computing paradigm exclusively may not be attractive in applications that require immediate processing, high responsiveness, and real-time analysis of IoT clients requests [128]. To this purpose, fog computing has been designed to mitigate issues of latency, fast responsiveness, provide real-time transactions, and bridge the network unreliability concerns.

Cloud computing extends the concept of fog computing for better utilization and minimum energy consumption [7, 124]. Combining Cloud of Things to Fog Computing minimizes the service delay for IoT applications. In the [129], a novel service called Offload as a Service (OaaS) has been presented. OaaS provide the capability to extend limitations of mobile resources such as GPU, CPU, storage etc. However, the open issues in fog computing environment such as dynamic offloading, scalability, security , the use of minimum number of fog nodes to achieve efficiency and effectiveness still remain open issues [130, 131].

To overwhelm all the aforesaid issues in the Cloud Integrated Fog-IoT paradigm, machine learning based approaches have been proposed in [16, 132–134]. In our study, we choose a pipeline of machine learning strategies to achieve better performance. Machine learning has demonstrated latent power in solving complex problems in science, engineering, industrial and professional practices [22–26]. For example, in banking and finance machine learning can be used by executive managers to make informed decision. Machine learning can help banks spot potential business partners and their expenditure. Using Machine learning algorithm smart machines can be trained to monitor trends in the market and react in real time [27].

In medicine, machine learning has been used in diagnosis of complicated diseases. Moreover, it has been used in analysis of clinical parameters and their combination of the prognosis [28]. It can help in integrating computer based systems in health care industry. Additionally, machine learning has demonstrated ability to assist in solving complex problems in aerospace engineering. In [29], machine learning has been used to recognize specific defects of aerospace structures, decrease approximation errors and compute the closest possible outcome. The limits due to the great amount of data and the complexity of data processing make machine learning as defect classifier in aerospace structures very useful [29, 30]. Machine learning has also been used to optimize parameters so as to find best options that maximize use of resources in logistic and supply management. Here, machine learning allows improved planning for unexpected events and help predict orders along the supply chain with at most accuracy [31].

In this chapter, we propose a machine learning based secure offloading framework herein referred to as Secure Offloading in Fog-Cloud in Internet of Things framework. Our proposal is a software solution that approaches offloading data to the Fog or the cloud using machine learning approaches. Our dynamic and secure offloading reduces offloading latency and minimizes energy consumption.

Thus the rest of this chapter is organized as follows; preliminary studies related to computation offloading and fog computing is presented in section 3.2. Other related studies are reviewed in section 3.3. We present our proposed system in section 3.4., in 3.5 a comparative study of our ML based framework and other related work is presented and a summary of the chapter is presented in section 3.6

3.2 Preliminaries

3.2.1 Computational Offloading Mechanism

Secure Offloading is technically a challenging problem in Edge/cloud enabled IoT environment, especially when dealing with wireless connected systems in which resources required for communication are highly dynamic.

In an IoT related environment, a device (smart phones, surveillance cameras, robot, smart-meter etc.) use applications installed on then to execute tasks. When the resources used by the device get below threshold the task is dropped. For instance, in a video chart, a call is placed through video application. During the call process the device continues to consume battery, memory, and storage space. When any of the resources (battery level, space or processing power) to support the call gets low, it is dropped. In this way the device is saved from dying out.

Instead of letting the device drop the process, a portion of data in the storage space may be relocated to a remote device to create more space on the device, or a part of the process which over utilizes the processor causing battery drainage can be migrated to remote systems. This helps the device to save battery as opposed to dropping the task. The above described process is known as computational offloading.

Computational offloading mechanism can be viewed as either fine grained or coarse grained procedure [135]. In coarse grain[1] computational offloading, the task is migrated to the cloud in whole and there is no need to estimate resources overhead. The decision needed in this coarse grain is either to execute the whole workload on the mobile device or it is sent to be executed at the cloud, while in fine grain computational offloading[2] a portion of the task is executed on the mobile device and the other portion is executed on the server (edge or cloud device).

In the fine grained offloading, little code is dynamically transmitted and only the computational hungry part of the code is offloaded to the remote device. Isolating processes in devices according to their processing needs can reduce unnecessary transmission overhead, improve performance and energy utilization [93].

The decision taken by an IoT device to offload part of its workload to remote device depends on the time taken to process the workload. Assume, there exists some workload (ω) at the user equipment(UE) side say a mobile device(m), whose processing speed is s_m. This workload can be partitioned into two; one part that will always be executed at the the user equipment side e.g. user interface and code that manages peripheral (camera, temperature sensors, proximity sensor, accelerometer, etc.), and the other part that may be offloaded. If the offloadable workload is locally executed on the user equipment then equation 3.1 represents the time (τ_m) required to execute the workload;

$$\tau_m = \frac{\omega}{s_m} \tag{3.1}$$

[1]Coarse grain offloading strategy is synonymous to static computational offloading
[2]Fine grain computational offloading is synonymous to dynamic offloading

otherwise, if the workload (ω) is executed on the server [3] whose processing speed is s_{se} then the time required to complete the workload (τ_{se}) is;

$$\tau_{se} = \frac{\omega}{s_{se}} \tag{3.2}$$

to execute the workload on the server equipment, it is necessary to transmit the workload over a network channel. To transmit the workload to remote it takes transmission time (τ_c). Transmission time can be computed as a ratio of data shared between the two equipment(UE and the Server) (d_s) and the bandwidth (B_w) between them .

$$\tau_c = \frac{d_s}{B_w} \tag{3.3}$$

Moreover, the energy required in completing the workload at the UE(En_m),server (En_{se}), and transmitting shared data to and from UE and server (En_c) can be computed in terms of τ_m, τ_s, τ_c as follows;
Let p_m be the processing power of UE and p_{se} be the processing power of the server equipment and p_c be the transmission power of the transmitting equipment for both the uplink and downlink[4], then

$$En_m = p_m \times \frac{\omega}{s_m} \tag{3.4}$$

$$En_{se} = p_{se} \times \frac{\omega}{s_{se}} \tag{3.5}$$

$$En_c = p_c \times \frac{d_s}{B_w} \tag{3.6}$$

The total energy consumed during offloading process is the sum of energy required to process the portion of workload at user equipment, energy required to maintain the offloading process, and energy required to transmit the remainder of workload to the server.
Ignoring other conditions such as complexity of the workload, time taken during the initial setup, the size of program, offloading to be beneficial then equation 3.7 must hold.

$$\frac{\omega}{s_m} \geq \frac{\omega}{s_{se}} + \frac{d_s}{B_w} \tag{3.7}$$

[3]used synonymous to imply Fog node or cloud server
[4]uplink and downlink power may differ from time to time depending on channel conditions

If the server is infinitely fast then equation 3.7 reduces to

$$\frac{\omega}{s_m} \geq \frac{d_s}{B_w}. \tag{3.8}$$

It is important to note that no matter how fast the server processor is, as long as the transmission time is greater than the time required to process the workload at the user equipment offloading will not improve performance. Considering workload that requires heavy computation, and light data shared is in the center of fine-grained computation offloading. Identifying parts of the task that require heavy computation with light data sharing is seen to improve performance [93]. Secondly, performance improvement is dependant on the offloading tasks that require heavy computation with light data sharing to minimize transmission energy. Moreover, dynamic offloading adapt to different run-time conditions e.g. fluctuating network bandwidths and mobility of devices in the network. Lastly, dynamic offloading make the use of prediction mechanisms such as machine learning, stochastic Bayesian methods, heuristic algorithms and other optimization algorithms attractive for decision making [93, 135].

During Dynamic offloading, the workload may be split into smaller portions so that part of it is processed locally and the other part is offloaded to many remote devices [134]. A number of approaches used to split tasks have been explained in [93]. The most commonly used method is the graph method[5] . The two factors latency and energy consumption play a considerable role in determining the quality of offloading via fog nodes.

3.2.2 Fog Computing

The concept of fog computing stretches from the outer edge where data is created to eventually where it is processed and/or stored. This computing and storage location could be within the organizational data center, the edge device(Router, switch etc.) or the cloud. Fog computing forms another layer of a distributed network environment in between the cloud computing and the Internet of Things (IoT) that provides a continuum to bridge the missing link for data that needs to be handled locally closer to the edge or pushed to the cloud [41]. Fog computing

[5]Graph method is method used to represent tasks as a directed acyclic graph (DAG). Vertices represent the computational components and the edges represent the communication between them. Splitting the task is done by partitioning the graph

paradigm contains Fog Nodes distributed within limited geographical area where certain IoT mobile devices computation tasks are executed.

Among other computational objectives, fog computing aims at reducing the data amount that requires to be forwarded to the cloud for processing and storage. This activity improves system efficiency when massive data processing, storage and analysis are required in real-time. Each IoT device performs its computational tasks or offloads them to the Fog [44, 136]. At the architectural point of view, the Fog computing provides a horizontal system architecture that distributes resources and services of computing, storage, control, and networking anywhere along the continuum from the cloud to Internet of Things.

With the applications or data that may need to be processed quickly for example in use case such as manufacturing, connected machines may require to respond to an event at most immediately. Fog computing provide appropriate choice as means of responding to such event. Secondly, the Fog is required in situations that may arise due to no bandwidth to send the data for processing to the cloud, or where amount of bandwidth required to send the data to either the organizational data center or the cloud is very expensive [132, 137, 138]. An additional benefit is that, the fog can be used to secure the data from segmented network.

3.3 Review of Related Work

Several works have been done related to computation offloading as well as the combination of cloud with IoT, but a few contributed to security in computation offloading. Our work adds to the body of knowledge related to secure computational offloading in trusted system. In this section, we present related work to computational offloading and security in fog environment as follows;

3.3.1 Review on computational Offloading in Internet of Things

An offloading architecture called AutoScaler was proposed in [139]. In this paper, provisioning offloading as a service was explored. They proposed large-scale offloading in IoT environment. An offloading system consisting of front-end, back-end and load simulator which generates dynamic offloading workload of multiple devices was designed. The AutoScaler front-end component introduces extra time overhead of ≈150 milliseconds in the overall response time of a client request. Adjusting a trade off between the utilization price and computational capabilities of the surrogate servers at the backend, it was observed that the total time of code invocation is reduced. Furthermore, the proposed architecture was easily deployable on large scale. Introducing surrogates entities in our Fog environment is likely to improve performance.

Xiao Ma et al. [140] used a game theoretical approach to analyze the decision problem of IoT devices. Their work based on a definition of potential game. Nash equilibrium was derived and algorithm known as computational offload decision(COD) was proposed. This algorithm, portray significant reduction in the system cost. Cost was viewed in terms of processing delay and energy consumption. COD is energy-aware computation offloading in cloudlet-based mobile edge computing. The computation offloading decision (COD) algorithm is based on decentralized computation offloading strategy for IoT devices. The energy consumption of IoT devices in computation offloading to cloud through base station was found to be significant. Another mobile cloud IoT (MCIoT) paradigm was proposed in [141] which uses a new nested game model for computation offloading. Firstly, each mobile device would determine the portion of remote offloading computation using Rubinstein game theory approach. Secondly, a computation resource in the cloud was assigned dynamically for the requested computation offloading. The nested game theory approach provides an optimal solution for the computation offloading in the MCIoT paradigm. However, game theory principle consumes more time for computation offloading.

Minghui et al. [16] proposed learning based computational offloading for devices with energy harvesting. They constructed a hotbooting Q-learning dynamic process that chooses a portion of data to offload to mobile edge computing devices(MEC) according to the system state (bandwidth, amount of energy harvested, and battery level) to determine an offload policy. Moreover, the system was formulated as a Markov Decision Process (MDP) in which the Q-learning technique was used to attain optimal policy. However, the hotbooting mechanism take significant length of time to converge. To improve further the system performance they proposed a Fast DQN computational offloading. The proposed system use the concept of machine learning in computational offloading though it implementation doesn't consider important issues such as mobility, cost of use of MEC and security in their utility function. These factors left out of the scope of the study could have an impact on the utility function and therefore impact on the policy. In addition, dynamic changes in the network parameters such as bandwidth and mobility was not considered. In [134], a deep reinforcement learning based on computational offloading has been proposed. Similar to [16], they have designed an offloading algorithm for optimal decision making subject to dynamics of the system in terms of user and cloudlet behavior. They concentrated on the composite behavior of the cloud queue and the distance between the cloudlet and the user equipment. Unlike in [16] where offload is achieved on one MEC at an instance, [134] solve the problem of offloading to multiple cloudlets.

Jie Zhang at el. [142] proposed a hybrid computation offloading algorithm that combines cloudlet with public clouds, to provide a more energy-efficient offloading strategy for home automation applications. Particle Swarm Optimization (PSO) based heuristic algorithm is implemented to schedule mobile services. The task scheduling mechanism uses queuing model based on First Come First Served (FCFS). In the same study, the waiting time in the cloudlet is modeled as a $M/M/m/\infty$ queue while PSO is used to select unscheduled mobile services in the work flow. Scheduling tasks using FCFS is likely to be slower as compared to other scheduling mechanism, whereas using the PSO for selecting the Fog node may improve system performance. PSO is suitable for this kind of problem since IoT devices exhibits characteristics of a swarm. Dynamic Task Offloading (DyTO), has been proposed in [143], in their work they introduced the concept of surrogate object in a computational offloading environment. A surrogate object installed on the mobile cloud takes care of information that tracks the mobile host thereby separating cohesion of resources to the mobile host as it traverses or looses connection. The surrogate object maintains the information to ensure proper delivery of data especially in situation where the network is unstable and characterized by disconnection. The proposed model is observed to save energy. Execution is faster because interruptions in processing caused by disconnection, or variations in network resources caused by mobility is avoided. Applying similar solutions in secure offloading appears feasible

3.3.2 Review on Security and Privacy in Internet of Things

Privacy, trust, and security are important factors to consider in IoT ecosystem. Huge amount of data outsourced from a global network of things connected to data centers, Fogs and cloud make IoT networks more vulnerable to cyber attacks [144]. Authors in [145] have proposed Smart Trust management method to detect On-Off attacks caused by nodes that may perform bad behaviour randomly in order to avoid low in trust ratings. Moreover, suspicious resources may behave differently with different neighbours. They proposed a method to collaborate with IoT devices to identify On-Off attacks and broken nodes. Interaction among IoT devices are evaluated using meta data attributes. A machine learning classifier is used to classify the data. From the trust score generated by the classifier, a decision is taken to either trust or mistrust the resource. Their proposed security method detect On-Off attack with 97% precision and 96% in real record data set. The method is 95% faster. This recent work is one of those that uses machine learning generated score in trust management of resources.

In [146], privacy preservation with IoT oriented offloading. This is a method for solving data transmission security problem. It was implemented in the WMANs (Wireless Metropolitan Area Networks). In their proposal, data is offloaded to the cloudlets through access points. To deal with shortest routing problem, the Dijkstra algorithm was used to establish shortest path between access points. Also NSDE (Non-dominated Sorting Differential Evolution Algorithm) was considered to resolve multi-objective optimization problem. Yinghui et al. [142] have presented privacy-preserving data aggregation from hybrid IoT devices in fog computing. The proposed scheme ensures data integrity by guaranteeing that injection data is received from legitimate IoT devices. The proposed scheme reduces communication overhead, but increases computation overhead when searching for appropriate fog device. Thus, it increases processing delay.

In [147], a practical evaluation of a high security energy-efficient gateway was considered in IoT fog computing applications. The primary contribution of this paper is increased security levels for sensed data in resource-constrained environment using Rivest–Shamir–Adleman (RSA) and Elliptic Curve Cryptography (ECC) . Furthermore, the proposal improves throughput and energy efficiency for IoT devices. In practical evaluation, ECC outperforms than RSA, but ECC needed further improvement to organize massive sensed data for real-time scenarios. Belem et al.

[148] have proposed a device-based security called PROTeCt (Privacy aRchitecture for IntegratiOn of Internet of Things and Cloud computing), which improves user privacy. In the proposed architecture, user privacy is given by cryptography –based scheme in which only the interested users can access their data, which is stored on cloud in encrypted form to protect IoT network from insecure access. In this works, gateway was authenticated. Users required to register and accepted to join the network time and again which increases communication cost.

To this end, our proposed work is one of the kind that is intended to tackle deficits in previous computation offloading works. To the best of our knowledge, we have presented a secure and dynamic offloading scheme which minimizes energy consumption and latency to obtain more efficient offloading in Fog-IoT environment.

3.4 Proposed System

3.4.1 Problem Statement

We consider a network of Γ IoT nodes such that $\Gamma = \{1, 2, 3 \ldots n\}$. Each of IoT devices in this network may contain computation-intensive, or delay-sensitive computation task. These IoT devices are deployed in a network which are connected through a smart gateway to the Fog nodes and the cloud respectively creating a hierarchical network. The fog nodes form a network continuum to the cloud. Given a task, the IoT evaluates the task to see if it can execute the task locally using resident resources or not. If the IoT finds that it cannot execute the task, it offloads the task to the Fog. The Fog either performs the tasks or sends it to the cloud.
Our intention is to perform dynamic offloading while maintaining user's sensitive tasks in the Fog during task offloading at the same time achieve high performance in terms of throughput, delay, energy consumption, resource utilization rate and response time.

3.4.2 System Overview

The proposed system architecture shown in Figure 3.1. consists of IoT mobile devices at the IoT layer, network devices at network layer, fog devices at the Fog layer, and cloud infrastructure at the cloud layer.

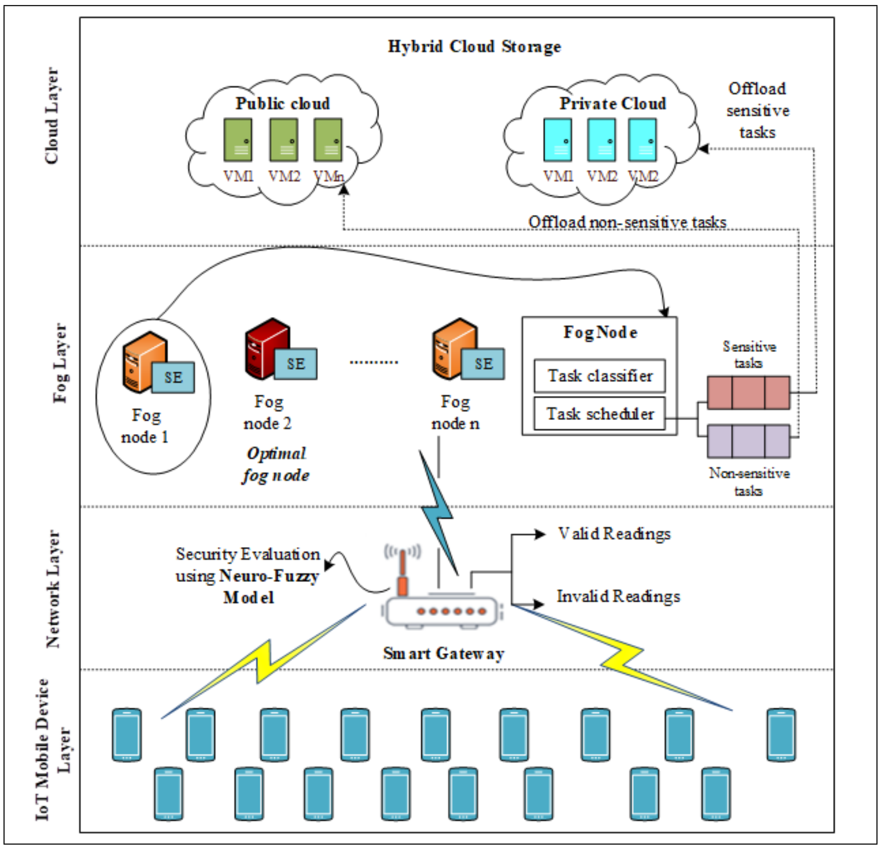

FIGURE 3.1: Proposed system

IoT devices

The IoT devices acquire, monitor and measure data. In addition, they send, receive data to and from the Fog. The infrastructure environments created by the IoT devices allow them to monitor and filter data from environment. IoT devices are characterized by low computational power, constrained by battery life and their small form factor. They have considerably low memory and powered by small battery cells [5, 13, 149].

Network layer

The network layer consists of network devices such as switches, routers and gateways. They may adopt the functionality of a fog on a small scale [5, 13]. To mention in this framework, a smart gateway is adopted to secure the network through evaluation of data coming from IoT device. Using Neuro-Fuzzy model, we are able to predict if an IoT device is a candidate for malicious attack through reading obtained from the devices. Based on the readings, the node can be trusted.

Fog layer

The fog layer consists of Fog nodes. Fog nodes are high-performance distributed systems. They report results of processing to both Cloud and IoT layer along the continuum. Unlike the IoT devices, fog nodes are more powerful and have considerable amount of storage. They can provide localized services when need arise. They can be equipped with ability to support data analytics at transactions level [5, 13, 149]. In our system ,we equip the Fog nodes with surrogate entity, task classifier, and task scheduler. Fog devices may be installed on moving objects such as in cars, buses or trains(mobile).

Surrogate Entity

Surrogate entities are software defined objects installed on the Fog Node. Their function is to collect and store information about the IoT devices which are in service. IoT device may roam from one network to another, or may face network unreliability situation which may lead to breakdown in services. As a result, the device is required to seek the service again when network becomes available. To avoid such situations, the surrogate entity [143], serves as information holder. If the IoT devices goes out of service, the service at the fog remains active. After the service is completed the fog node uploads the results to the IoT device. The entities help decouple IoT devices during service time hence improving performance, allowing for mobility and uncertainties in the network connectivity. Information in the Surrogate entity remains unchanged during service time, though some IoT device information may change from time to time due to mobility. They generate compact data sets which are less expensive to maintain.

Cloud layer

The cloud consists of very powerful, state of art centralized infrastructure, which has infinite capacity to realize secure and heavy computation. Cloud infrastructure may be private, public or hybrid. Private cloud allow organizations to run their cloud services based on proprietary architecture, within their own data centers. They are created and maintained by an individual organization. Public clouds allow organizations to run their cloud services based on third-party architecture. With public cloud, vendor aught not to set their own infrastructure. Public clouds are based on multi-tenant architecture and pay-as-you-go pricing model. Users only pay for what is required for their use [149, 150].

3.4.3 Optimal fog node selection using PSO

In IoT network, connectivity is achieved through wireless means. In such an environment resources can be dynamic. They are affected by unprecedented factors which include fluctuating bandwidth due to mobility, weather, and physical obstacles along communication path. This makes the workload at the Fog Node change frequently. Thus, an algorithm to update network information more often is required.

As the fog node current workload changes, we run the PSO algorithm [151] to update information used for selecting the optimal Fog node. The criterion for selecting the fog node is used to reduce the total processing delay between IoT mobile devices to the Fog. Optimal fog node is chosen by looking at two metrics : i) Available Processing Capacity (APC) and ii) Remaining Node Energy (RNE). Every node will calculate its fitness using these two metrics. When the request is made by the user device, the fog node with high APC and RNE will be chosen.

The PSO algorithm is meta heuristic algorithm which derives its intelligence from swarm. PSO has been used in many applications that exhibit characteristics of swarm similar to behavior of social creatures like birds and fish [152] . We find this algorithm suitable in this study since IoT devices behave the same way a swarm does. The mechanism used in PSO is simple, but powerful. PSO is used in many applications in science and engineering. Examples are found in studies that involve prediction of events [153], network planning [154], alignment optimization [155] etc.

PSO uses two basic principles that is communication and learning. A particle hereafter known as an agent who is a member of a swarm creates an intelligent behaviour which is absolutely unreachable by other agents in the swarm. Thereafter, communicates this information to all agents of the swarm. From information tendered in by each agent, all agents learn the best course of action to take. The agents create a single global optimal knowledge that determines the course of action for the whole swarm. Leveraging this simple cooperation results in a level of intelligence that can be reached by all agents of the swarm [152] .

The objective of PSO algorithm is to find the optimal solution by information sharing and cooperation among individuals in a group. Assume that one population comprised of n particles and D dimensional searching space. Each particle changes its position x at time t by

$$x(t+1) = x(t) + v(t+1) \tag{3.9}$$

$$\begin{aligned} v(t+1) = \omega v(t) + C_1 R(0,1) * (x_{pbest} - x(t)) + \\ C_2 R(0,1) * (x_{gbest} - x(t)) \end{aligned} \tag{3.10}$$

where v(t) is the velocity of the particle at time t, x(t) is the particle position at time t, ω is the inertia weight, C_1 and C_2 are the learning coefficient and acceleration coefficient respectively, R is the random variable range between 0 and 1, x_{pbest} is the particle best position, and x_{gbest} is the global best position. For each particle x_i of the $task_i$, we compute fitness value by using APC, and RNE.
Fitness function:

$$f(x_i) = w_1 \times APC(x_i) + w_2 \times RNE \tag{3.11}$$

where w_1 and w_2 are the two weight factors representing the importance of APC and RNE, respectively. $w_1 \in [0.1, 0.9]$ and $w_2 = 1 - w_1$.

Moving forward, let us consider, sz_i, cx_i, μ_i be the characteristics of $task_i$; where sz_i, cx_i denote the size complexity of the task i, and mean latency respectively. And let bs and F be the buffer the current buffer size and CPU frequency of the

Fog node, then we can compute $L(x_i)$, the latency of task x_i as follows;

$$L(x_i) = \frac{sz_i \times cx_i \times \mu_i + bs(x_i)}{F(x_i)} \tag{3.12}$$

After the optimal node is chosen by user, the request is sent to the fog node. User request consists of information about the tasks to be offloaded to fog nodes shown in Table 3.1.

Table 3.1: User request Task Information

Name	Description
ID_{Task}	Task unique identifier
$ApplicationType$	The type of the application
Task constraints	Specific constraints of tasks such as latency
$Fognode_{id}$	Optimal fog node within the user communication range

At the IoT level, the following proposed scheme is used to choose an optimal fog node to which offloading can be made.

```
Method OffloadingScheme-IoT():
    for each IoT device do
        if offloadable workload exists then
            select an optimal Fog Node using PSO
            send data through Smart Gateway
        else
            perform local execution on the IoT device
        end
    end
End IoT-Scheme
```

Algorithm 1: offloading scheme at IoT Node

3.4.3.1 An outline of design strategy of PSO in IoT Ecosystems

In this study, we suppose there exist a cluster that presents the fog-cloud of things ecosystem. Each cluster contains devices that are categorised as IoT devices, smart gateways, fog devices and cloud devices. All IoT nodes, fogs and clouds reside in a single cluster.

The strategy is to follow the candidate solutions in the ecosystem which presents the best fog to which an offload will be done until it is not capable anymore. Each candidate solution presents a computed fitness Value that determine the suitability for selection. At any point in time, a cluster is initialized before a search for optimal solution is initiated. For each iteration, the best solution achieved locally so far(fitness) known as local best. Amongst the local best, the candidate solution that presents the best fitness is considered as global solution.

Based on the above design strategy, algorithm 2 for selecting the best fog in the cluster is presented.

```
Method input:
    $N \leftarrow swarmSize$
    $D \leftarrow problemDimension$
    $maxit \leftarrow maxIterations$
    $searchSpace \leftarrow [lowerB, UpperB]$
Method outPut:
    $globalBest \leftarrow TheBestSolutionFound$
Method IoT-PSO():
    initialize the swarm randomly
    initialize the velocity vector v(t) initialize the particle position x(t)
    for it=0 to maxit do
        for each particle do
            compute fitness value using $f(x_i) = w_1 \times APC(x_i) + w_2 \times RNE$
            if current fitness is better then
                replace the existing fitness value
            else
                continue
            end
            Select the FogNode with the best fitness value of all the particles
            for each particle do
                compute velocity using $v(t+1) = \omega v(t) + C_1 R(0,1) * (x_{pbest} - x(t)) + C_2 R(0,1) * (x_{gbest} - x(t))$
                update the particle position using $x(t+1) = x(t) + v(t+1)$;
            end
        end
    end
End IoT-PSO
```

Algorithm 2: Selection scheme at IoT Node

3.4.4 Neuro-Fuzzy model for security evaluation

In IoT-Fog architecture, IoT devices communicate to the upper layers through the gateways. Gateways are responsible for bridging between IoT devices, the Fog, the cloud, and user equipment (smart phone, cyber-physical devices etc.). They provide a communication link, real-time control over the IoT devices, and provide offline services. Gateways can be used to secure data that is being transported to and from the upper layers. Security is achieved by isolating resources that exhibit abnormal behavior. In many occasions, security solutions require considerable amount of processing power which are expensive in terms of energy consumption. Implementing them on IoT devices is infeasible. In our study, we secure the network at using Neuro-Fuzzy network at the smart gateway because the smart gateway houses considerable processing power, and storage as compared to IoT devices. In [156], neuro-fuzzy model has been used in similar setting for purposes of routing to achieve better energy utilization.

Nuero-Fuzzy systems are suitable when we intend to manage parameterized components of a Fuzzy system. They are used to produce systems that deal with parameters that can be tuned through training. Nuero-Fuzzy systems are tools used for prediction and classification problems. They combine characteristics of both Neural networks and fuzzy techniques. Neural network tools bring useful traits of learning, generalization, optimization, and adaptation. Whereas, Fuzzy models bring human like intelligence using IF . . . THEN rules, Expert knowledge, simplicity in terms of linguistic variable with no mathematical expression to the system [157].

Nuero-Fuzzy systems are broadly categorized into; Neural Fuzzy systems, Fuzzy Neural systems and Hybrid Nuero-Fuzzy systems. In Neural Fuzzy systems, Neural network are used to determine functions and mapping between fuzzy sets that are used as fuzzy rules. They change weights during training so as to minimize mean square error between the actual output and target. In these systems, fuzzification functions, fuzzy word membership, functions and fuzzy rule confidences are represented as weights in neural network, whereas in Fuzzy neural network inputs are non-fuzzy, operations are replaced by membership functions, and aggregation operations such as max, min, t-norm and t-coform are used. Lastly, in hybrid Nuero-Fuzzy systems, each technique is used independently to accomplish a task. Fuzzy rules are interpreted in the of context neural networks while fuzzy sets are interpreted as weights [158, 159].

In this work, we employ Nuero-Fuzzy model at the smart gateway to evaluate the data coming from IoT devices. Two factors are considered for security evaluation i.e. Sensor value(Sv), Time (Ts). From these two values predicted value(Pv) is derived. If the predicted value is greater than 1.00, we consider the resource has valid reading otherwise the reading is invalid; therefore the resource is isolated from transaction. We assume that Neuro-fuzzy model consists of N devices. (d_1,

d_2 ,...,d_N). Each input has two parameters that is Sv, and Ts and outputs values, which are either valid and invalid.

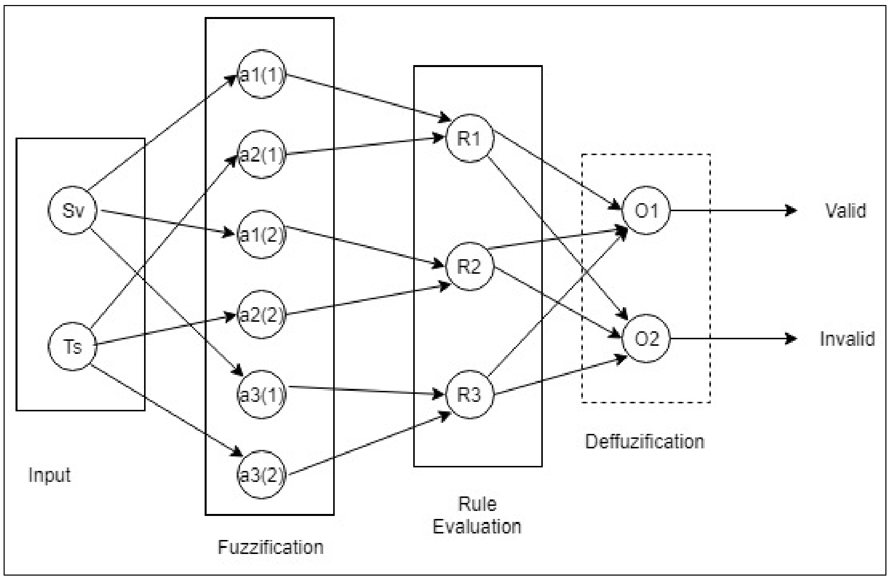

FIGURE 3.2: The Proposed Neural-Fuzzy model

The sensor value(Sv) can be Small, Medium or Large. The corresponding time value can be low, moderate, or high. In our model, we consider sensor value (Sv) to be small if data size is less than 100bits, Moderate if the data size is between 100 to 350 bits, and large if the data size is more than 350bits. The corresponding time is considered Low if the Time (Ts) is below 100ms, Moderate if the Time (Ts) is between 100 and 1000 ms and high if Time Ts is greater than 1000ms. For each episode, Sv, Ts, predicted values(Pv), and output are generated. These output values are stored in the knowledge base which acts as bases of experience. Using the knowledge base the Neuro Fuzzy network is trained to adapt to incoming data from the IoT devices. Fuzzy rules are constructed and these are based on experience/knowledge in the domain as follows;
IF Sv is Small AND Ts is High THEN Pv is Invalid
IF Sv is Small AND Ts is moderate THEN Pv is valid
IF Sv is Small AND Ts is Low THEN Pv is valid
IF $Sv$is medium AND Ts is High THEN Pv is valid
IF Sv is medium AND Ts is moderate THEN Pv is valid
IF Sv is medium AND Ts is low THEN Pv is valid
IF Sv is Large AND Ts is High THEN Pv is valid
IF Sv is Large AND Ts is moderate THEN Pv is valid
IF Sv is Large AND Ts is low THEN Pv is invalid

IoT devices are sensed to have invalid data due to incorrect sensor values and incorrect delay (ms). From the obtained results such as valid and invalid predicted values, the data form trusted devices are retained. Figure 3.2 , illustrates the proposed Neuro-Fuzzy model, and Table 3.2 shows sample of knowledge base of Neuro-fuzzy model.

TABLE 3.2: Sample of Knowledge base for Security Evaluation

D_{ID}	SV	T_S	Prediction	output
107	75	4137ms	0.00027	invalid
108	251	1564ms	1.00002	valid
116	0	951ms	0.00030	invalid
115	230	230ms	1.00458	valid
114	245	1117ms	1.00001	valid

At the gateway, the following proposed scheme is used to secure data coming from IoT device;

```
Method validateScheme(ioTScheme-data):
    for data received do
        validate data using Neuro-fuzzy model
        if data is not valid then
            drop the data
            inform IoT device to resent
        else
            forward data for processing to selected Fog Node
        end
    end
End validateScheme
```

Algorithm 3: Data validating scheme at Smart Gateway

.

3.4.5 Dynamic Task Offloading using Task Scheduling

When the fog node cannot process all received tasks within the latency constraints, the fog offload tasks to cloud server for further processing. For this action, a dynamic task offloading is proposed. Dynamic offloading is based on reinforcement learning scheme called Q-learning.
Q-learning is a model-free reinforcement learning mechanism from which the agent learns the best course of action through experiencing the consequences of an action, without necessarily building a map of the domain [160, 161]. Learning is achieved through trial an error until set of action taken yields optimal policy. This mechanism is realized by the agent trying an action at a particular state. The agent evaluates the consequences of the action in terms of an immediate scalar reward received and it's estimate of future rewards with respect to the state and action which it has taken. By trying all the states repeatedly, the best state of action is learnt. Q-learning is a naive way of learning, but, as such, it forms basis of complicated operations in intelligent systems. It has found many applications in gaming, computing, science, and industrial application[162]. Examples of real world application are found in studies such as, in gaming [163], performance analysis[164], and robotics [165].

Q-learning concept includes state space, action space and reward function. Each state s and action-pair Q(s,a) has a Q-value. If action is selected by the agent located in state s, the Q-value for state-action pair is updated according to the obtained scalar reward using equation 3.14. When choosing an action, the highest Q-value for the subsequent state s' is considered (ϵ-greedy strategy).

At the fog level, given $task_i$, an action a_i means "choose Virtual Machine (VM_i) from all the existing VM" that meets requirements of $task_i$ for offloading. The task requirements include the type of the server (private or public), The VM that may be used to perform the task within the task constraints (amount of CPU , Memory, latency and Priority attached to the task). Action space a consists of action $a = \{a_1, a_2 \dots a_i\}$. In addition, available VM in the cloud server define the state space. The state space $S_m = \{VM_1, VM_2 \dots VM_m\}$, each VM is characterized by the amount of CPU and memory($VM[CPU, MEM]$). The action pair is represented as follows;

$$s = \begin{bmatrix} VM_1, a_1 & ... & VM_1, a_i \\ ... & ... & ... \\ VM_m, a_1 & ... & VM_m, a_i \end{bmatrix} \in (S, a) \tag{3.13}$$

A task is assigned to any virtual machine that meets the latency and resource constraints. To determine optimal action on the current observation of both the server and task requirement, the Fog chooses appropriate cloud based on the current state and reward received from the environment. The goal of the system is to maximize rewards received and minimize latency.

Dynamic task offloading by task scheduling problem is viewed as Markov Decision Process(MDP). Herein the action space is described by a binary vector for each $task_i$. When a current $task_i$ is received by the available VM's, it is represented by 1 otherwise it is represented by 0, then the reward function is computed for state-action pair. The rewards obtained denotes the cloud servers current state (running, waiting, busy etc.). The state action pair rule is shown in 3.14

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \tag{3.14}$$

where $\alpha \in (0,1)$ is the learning rate, r is the reward received on taking action a while in state s, and $\gamma \in [0.1, 0.9]$ discount factor . The VM to where offloading is to happen is chosen $\epsilon-$greedily with a small chance of acting randomly, even if the update is done based on highest Q-Value.

The incoming tasks may be sensitive or non sensitive. The sensitive tasks are offloaded to the private cloud servers, non-sensitive tasks are offloaded to the public cloud. Since the type of tasks differ and data processing platform is not the same, a task classifier is employed.

The purpose of the classifier is to classify the in coming tasks in to sensitive S_i and non-sensitive tasks NS_i. Sensitivity of the task is determined according to its size, complexity and latency. During offloading, the task characteristics don't change.

In this study, the task size is scaled between 1 and 100 Kbits, and task complexity is scaled between 10 and 200 cycles per bit. Before offloading the computation tasks to cloud servers, tasks priority are assigned accordingly. Following the above proposed scheme dynamic offloading to either the private or public cloud is accomplished. The scheme is outlined in algorithm 4

3.4.6 Training, Validation and Testing the models

In this study we train, validated and tested both the Neuro-Fuzzy Model(NFM) and the reinforcement learning(RL) model.

The NFM model was trained to evaluate the suitability of data for offloading, using a set of simulated offloading scenarios based on the guidelines in section 3.4.4. The scenarios included patterns of delivering large, medium, and small data against small, medium, and large time intervals. During convergence, data was generated at small intervals, resulting in a training data set of slightly above 400 records. Around 287 instances (70%) of the data was randomly selected for training the model, while 63 instances (15%) was reserved for validation, and the remaining 62 instances of data was used for testing. The rules were fine-tuned through autonomous adjustment based on the training rules arise from the fuzzy implementation.

The NFM model demonstrated an accuracy of 85.5%, while its precision was found to be 97.3%. Additionally, recall scored 83.1% and F1 scores were measured to be 89.6%. The neuro-fuzzy model showcases a favorable equilibrium between precision and recall. Our model was effective in performing on simulated data during the initial training phase.

Moreover, the RL model is trained, evaluated, and tested based on the knowledge base obtained from the NFM, which includes sensitive information as computed in the load input ratio for each instance. To ensure the reliability of the model, 500 instances of data are simulated at system initiation for validation and testing. The data is partitioned such that 70% or 350 instances are reserved for training, while the remaining 30% or 150 instances are equally divided for validation and testing. It is crucial to keep the validation and testing data separate from the training data to prevent over fitting. The underlying patterns of the model are continuously updated during its operation(autonomous adjustment).

The RL model demonstrated an accuracy of 84%, while its precision was found to be 92%. Additionally, recall is 97% and F1 score is 94%. The RL model showcases a favorable equilibrium between precision and recall. Our model was effective in performing on simulated data during the initial generation phase because it is performing well on the data set, the harmonic mean of precision and recall, is also high, indicating that the model is well balanced in terms of its ability to identify both offloading cases.

```
Method dynaOffloadingScheme($Task_i$):
    for each $task_i$ do
        Compute execution time
        Classify tasks into $S_i$ and $NS_i$
        Assign Task priority tasks($P_i$) according to task size($sz_i$),
          characteristics($cx_i$), and mean latency ($\mu_i$)
        if $P_i > ththresthold$ then
            offload $S_i$ to private server
            offload $NS_i$ to public server
        else
            Perform $S_i$ on Fog
            offload $NS_i$ to public server
        end
        receive results from Cloud
    end
End dynaOffloadingScheme
```

Algorithm 4: Dynamic offloading scheme

3.5 Evaluation of Machine Learning Based computational offloading with comparative Study

3.5.1 Experimental Setting with Comparative Study

The performance evaluation analysis and comparative study is presented in this section. First, we present the experiment settings and then provide experiment results. We evaluated the performance in terms of throughput, delay, energy consumption, resource utilization, and response time. Finally, we show our proposed offloading scheme is secure and scaleable.

3.5.2 Experiment Environment and Use Case

Experiment Environment

We create a Fog-Cloud IoT network which, consists of 1 smart gateway, 5-10 IoT mobile devices, 5 fog nodes and 1 hybrid cloud server. We used network simulator (NS3.26) and Java programming in our experiment environment.

The purpose of this configuration is to link the fog-cloud paradigms to the IoT mobile devices. NS-3 is an open source network simulator that supports C++ and/or python packages. Additionally, NS-3 contains Integrated Graphical User Interface (GUI) that is used to visualize performance of simulated network. Further , it includes modules that can be used to simulate network performance metrics such as throughput, energy, delay etc. It also contains modules that can be used to simulate most state of art networks based on 5G, 802.11ah, and 802.11ax standard that support fast and reliable data transmission.

In our experiments, all the simulation parameters have been set to follow uniform distribution. Each device is powered by CPU whose clock frequency ranges from 1 GHz to 1.5 GHz. The clock frequencies are set randomly. We also set available bandwidth between mobile devices to range between 100 Kb/s to 1000 Kb/s. The computing offloading require CPU cycles and tasks to be offloaded in bits. Computational tasks are categorized into complex and non complex. In order to characterize the offloading task complexity, we used load-input data ratio (LDR)[6]. When LDR is high, the task is classified to be computationally-intensive otherwise the task is not. Non-computationally intensive tasks can be executed at local device or edge. The system configuration of this experiment is shown in Table 3.3.

Use Case

The proposed dynamic offloading framework with security evaluation has been simulated for realistic application scenario in smart city. The smart city concept shows the need of quality of service and experience, improved connectivity and performance to realize several urban services. With the use of Information and Communication Technologies (ICT), Smart City application explores cloud-based

[6]LDR is the ratio of input computational size and computational workload. Computational workload is measured in terms of CPU cycles required to complete a task on a device

Table 3.3: System configuration

Name	Description
Simulation tool	ns-3.38
Development toolkit	JDK-17
Operating System	Ubuntu 20.04 LTS
Development platform	Netbeans 17
Processor	Pentium (R) Dual-Core CPU E5700@3.00 GHz
Installed memory	16 GB RAM

and IoT based services for users in real-world through smart phones. Today, cloud technology is and will continue to be the backbone infrastructure for smart cities around intelligent transportation, public safety, public health and air quality monitoring programs. Cloud offers a broad set of services including storage, processing, compute, analytics, databases, networking, etc. Users can use these cloud based services to construct secure, agile and cost-effective solutions. Broadly speaking, smart city application can be categorized into: Smart Infrastructure Monitoring, Smart water monitoring and management, Smart Building and Property, Smart city Services,Smart Energy Management, and Smart Industrial Environment [166], [167].

3.5.3 Performance Metrics

In this work, the following performance metrics have been used for comparative analysis

(i) Throughput (R_T): It is defined by the number of tasks offloaded per unit of time T.

$$R_T = \frac{\# - taskOffloaded}{T} \tag{3.15}$$

(ii) Delay (T_d): Time duration for a task of the application is submitted and its results are obtained. It is also computed as follows:

$$T_d = T_{pro} + T_q + T_t + T_p \tag{3.16}$$

Table 3.4: Simulation Measurements

Simulation Parameters	**Values**
Number of nodes	10 IoT devices
Number of fog nodes	5
Number of cloud server	1 (Hybrid cloud)
Number of simulated tasks	”10, 20,30,40,50”
Number of Smart Gateway	1
Simulation area	1000m x1000m
Task arrival rate	”[0-5]”
Simulation time	100seconds
Initial energy of a node	5J
Traffic type	CBR
Packet interval	0.1s
Learning rate	0.2

where T_{pro} denotes processing delay, T_q denotes queuing delay, T_t represents transmission delay, and T_p denotes propagation delay.

(iii) Energy consumption (E_t): It is the amount of energy consumed by IoT mobile devices to perform given task.

$$E_t = E_p + E_t \tag{3.17}$$

where E_p is energy consumed during processing offloading a task, E_t is energy consumed during transmission and receiving the result of the task.

(iv) Resource utilization rate (RU): is defined as the total amount of resources consumed as compared against the amount of resources estimated. RU is expressed as the percentage of time mobile device utilize resources in 24 hours.

$$RU = \left(\frac{N(i)}{24}\right) \times 100 \tag{3.18}$$

(v) Response time (T_R): Time interval between a user request and the reception of an action.

3.5.4 Comparison Analysis

In this subsection, we present comparative analysis of our offloading schemes to other existing solutions in [142, 143, 168, 169] listed in Table 3.5. Our proposed offloading provide scaleable and flexible solution for IoT users. In the following subsections we illustrate our results.

TABLE 3.5: The Benchmarks Used in this Study

Reference	Target device	Major contribution	Drawbacks	Application for IoT
Zhang et al. [142]	Mobile devices	Computation offloading using PSO (FCFS) scheduling.	High latency and hybrid computation offloading increases network overhead	Smart home automation.
Gnana Jeevan et al. [143]	Mobile devices,	Dynamic task offloading using surrogate object model (DTO-SO).	Consumes more energy and lack of security and privacy.	Mobile applications.
Tongxiang Wang et al. [168]	Mobile devices ,	Cooperative multi-tasks scheduling based on Anti Colony Optimisation(CMS-ACO).	Static scheduling poor resource utilization rate and not suitable for sensitive tasks	Mobile applications.
Yucen Nan et al. [169]	General IoT devices,	Lyapunov optimization for application offloading based on time and energy cost model(LOTEC).	High energy consumption and high response time.	Green energy consumption application.

Impact on Throughput

Figure 3.3 shows the impact on throughput for proposed compared to DTO-SO in [143] with respect to number of requests. We set latency requirements for sensitive tasks to be 1 seconds and 1.5 seconds for non-sensitive tasks. We set number of tasks $n \in \{10, 20, 30, 40, 50\}$. We observe throughput as the number of requests increase.

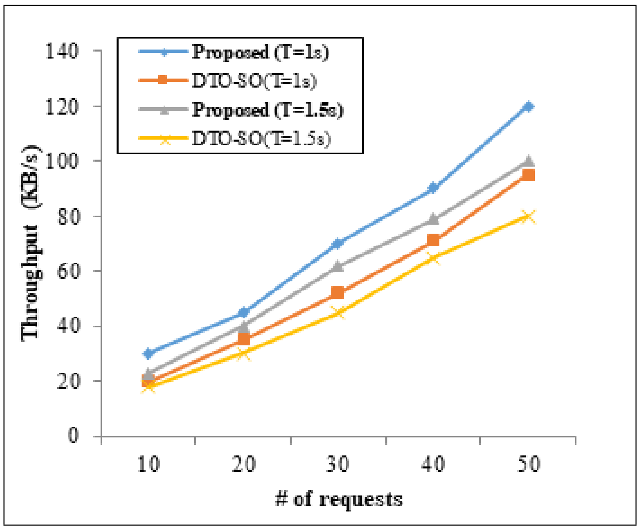

FIGURE 3.3: Impact on Throughput

Throughput increases with respect to the number of tasks transmitted by IoT device. At $n = 10$, throughput attained by our proposal is 30 KB/s for sensitive tasks, and 23KB/s for non-sensitive task. DTO-SO attains 18KB/s for both sensitive and non-sensitive for the same number of tasks transmitted. At $n = 50$, throughput attained by our proposal is 120 KB/s for sensitive tasks and 100KB/s for non-sensitive task whereas, DTO-SO attains 95KB/s for sensitive, and 80 KB/s for non-sensitive. Our proposed secure offloading scheme improves throughput by 23.2% as compared to DTO-SO. The increase of throughput is due to pipeline of machine learning offloading strategies that include PSO at the IoT nodes and dynamic offloading using Reinforcement learning at Fog node. Hence, the proposed offloading strategy is suitable for both complex and simple task computation offloading.

Impact on Delay

Figure 3.4 shows the delay per user requests for both sensitive and non-sensitive tasks.

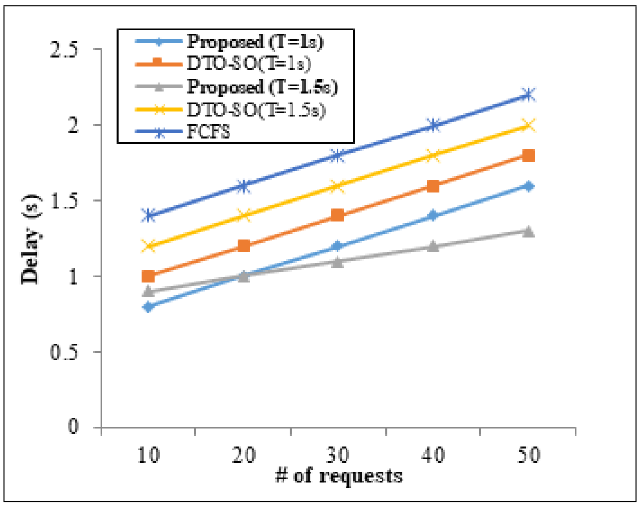

FIGURE 3.4: Impact on Delay

The delay of total offloading increases linearly with respect to number of requests. Delay is reduced by 20% and 60% as compared with DTO-SO and FCFS offloading approaches. In particular, considering sensitive tasks, The delay registered in our proposed scheme is up to 1.6s when $n = 50$. Whereas, the delay in DTO-SO is 1.8s and FCFS is 2.2s for the same amount of requests. Our proposed offloading scheme reduces delay because of optimal selection of fog node using PSO, and security evaluation at smart gateway does not increase latency. This in turn reduces time required to respond to offloading. Furthermore, task scheduling for sensitive (complex) tasks and non-sensitive tasks at the Fog is seen to reduce delay.

Impact on Energy Consumption

Energy utilization is one of the primary constraints in IoT environment. Executing complex tasks attracts massive usage of battery thereby compromising the device life. Migrating computationally intensive tasks and complex tasks by offloading such a task to the Fog and cloud server save energy and improve device lifetime. Figure 3.5 shows energy consumption of the proposed secure offloading vs. previous approaches LOTEC [169] and DTO-SO [143] with respect to number of requests. The energy consumption of total offloading tasks at mobile device is

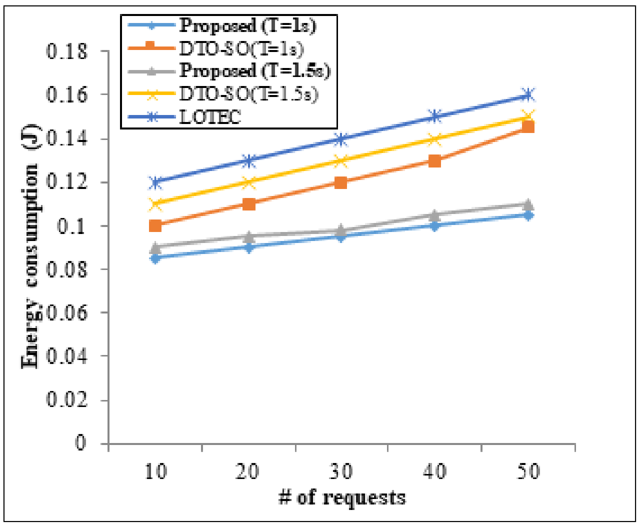

FIGURE 3.5: Impact on Energy Consumption

linear. We observe that the energy consumption during processing non-sensitive tasks is low (0.11J), which is 15% less than DTO-SO and 35% less than LOTEC. In previous approaches, the energy consumption rate increases gradually from $n = 10$ to $n = 50$ with higher gradient. This is because the delay sensitive and complex tasks consume more energy than non-sensitive tasks. Our proposed offloading scheme reduces energy consumption by the use of surrogate entity which decouples IoT devices from the burden of staying connected during processing of the offloaded work. Secondly, surrogate entity assigns cloud resources to device quickly thus reducing the energy required to maintaining computational offloading.

Impact on Resource Utilization Rate

Figure 3.6 shows resource utilization rate for proposed vs. previous approaches CMS-ACO [168], and DTO-SO [143]. Heterogeneous IoT devices request processing increases overhead and thus resource utilization rate is decreased. The proposed offloading scheme can handle real-life data, and monitoring of smart IoT devices. Thus it delivers highly scaleable sensing information and also security is evaluated before offloading. When $n = 10$, the resource utilization rate for proposed offloading scheme is to 92% at T =1s while the previous approaches utilizes 89%, and 87% for DTO-SO and CMS-ACO, respectively. In previous approaches, offloading tasks of certain resource-restrained devices may not increase resource utilization rate. This reveals the trade-off between energy consumption and delay

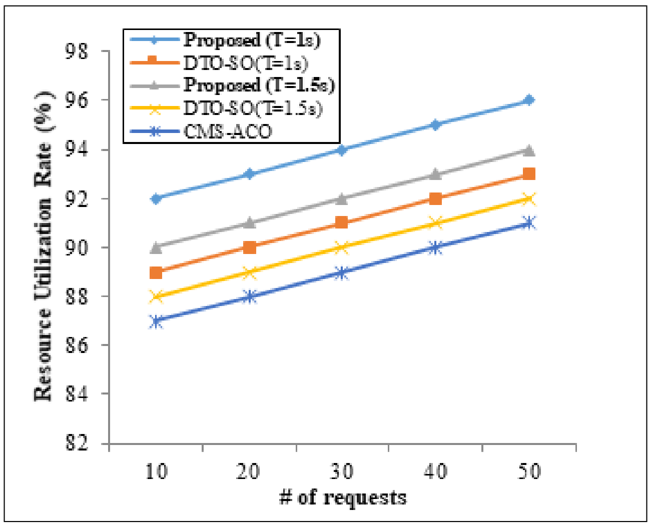

FIGURE 3.6: Impact on Resource Utilization Rate

in fog-cloud IoT environment. Our proposed offloading scheme is feasible in any resource-constrained environment

Impact on Response Time

Figure 3.7 shows the result of response time for proposed offloading scheme as compared with two previous approaches LOTEC[169] and DTO-SO [143]. The response time grows gradually In LOTEC and DTO-SO. Our proposed offloading takes 0.5s to complete a task as compared to 0.9s in LOTEC, and 0.7s in DTO-SO. Therefore, our proposed scheme is faster in terms of response.

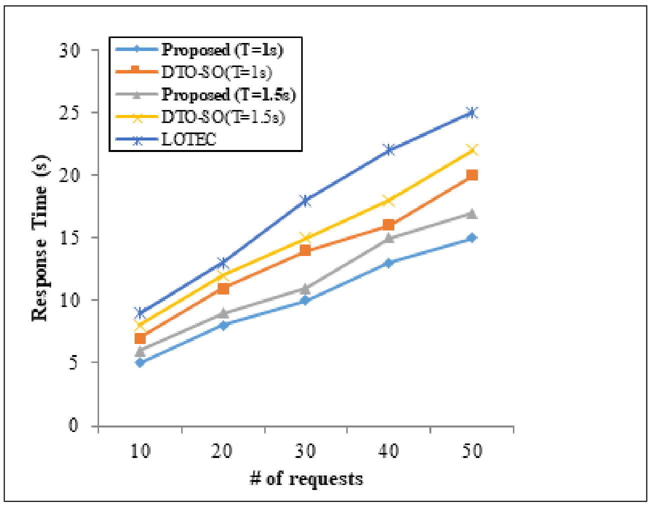

FIGURE 3.7: Impact on Response Time

Finally, we present average response time (a_{RT}) and average execution time (a_{ET}) of all the tasks for secure offloading and conventional offloading schemes presented in Table 3.6.

TABLE 3.6: Average Response Time and Average Execution Time

Metrics	State-of-the-art on offloading approaches				
	DTO-SO	CMS-ACO	LOTEC	FCFS	Proposed
$a_{RT}(ms)$	285.64	295.45	305.78	312.46	184.56
$a_{ET}(ms)$	1670.46	1700.23	1750.56	1790.9	956.23

The results show that our proposed system achieves response time of 184.54 (ms) as compared to 285.64(ms) for DTO-SO, 295.45(ms) for CMS-ACO, 305.78 (ms) for LOTEC and 312.46 (ms) for FCFS. At the same time execution time registered for our proposal is 956.23(ms) as compared to 1670.46(ms), 1700.23(ms), 1750.56(ms) and 1790.9(ms) for DTO-SO, CMS-ACO, LOTEC and FCFS. All experiments are done under similar environment.

The performance of various metrics vary depending on user request (task size, delay constraint and complexity). In all the figures (3.3, 3.4, 3.5, 3.6 3.7), graphical results were presented. Our results confirm that the proposed offloading scheme reduces delay and energy consumption than DTO-SO, FCFS, LOTEC, and CMS-ACO.

From the experiments conducted in this study, we conclude that the proposed mechanism improves performance of the IoT-fog-cloud ecosystem. It has been shown that offloading strategy minimize latency and energy consumption. Secondly, the implementation of security features such as euro-fuzzy model at layers of Fog-cloud of thing continuum guarantees security. Lastly, a well structured pipeline of procedures along the fog cloud of things improves the performance of the whole infrastructure by minimizing delay and realize negligible energy consumption. This makes the proposed approach robust. In a bid to further explore application of nature inspired mechanism we present an improved version of the current PSO to further improve the performance of the Fog cloud of things infrastructure in chapter 5.

3.6 Summary

This chapter forms the gist of this thesis. We illustrate the working process of our solution as follows; The IoT devices generate workload that may have offloadable content. The IoT device select a most suitable Fog node that can process the request and send back the results with in required threshold. PSO is used at this level as described in section 3.4.3. The offloadable content is forwarded through a smart gateway to the Fog. The smart gateway secures the data as described in section 3.4.4. The Fog may offload the workload coming from the IoT to the cloud using reinforcement learning mechanism described in section 3.4.5. After processing the cloud sends back the results to the IoT through the Fog, and Smart gateway. Thus forward, we present the evaluation of the mechanism presented in section 3.5

Chapter 4

Modified nature inspired computational offloading in Fog-cloud of things ecosystem

4.1 Introduction

The Internet of Things (IoT) industry has quickly expanded and revealed more opportunities to improve Quality of Service (QoS) in industries, governments and businesses through a connection paradigm that allows anything, anywhere to exploit connectivity to attain desired services [170]. The IoT ecosystems provide powerful platforms that influence intelligent systems in the automation of factories, education, military, medical care, surveillance, transportation etc. Embedding internet of things alongside data analytics has enabled development of high-performance systems that have upgraded the future prospects in cities, manufacturing, exploration of environment and space. Even if there is tremendous performance improvement in many systems developed today as a result of adopting IoT, performance of most of the systems developed are affected by; i) the nature of connectivity, ii) the characteristics of application that run on them, iii) the features of the platform that host the applications, and iv) the configurations of the IoT systems. Undistinguishably, the limitations such as uncertainty of IoT devices behavior, nature of operations as influenced by changing physical world, finding optimal solutions that map applications to the optimal remote platform for processing are largely unresolved [33, 39].

IoT systems performance can be improved by extending service of latency and security-sensitive applications to a remote location through fogging [171]. To effectively exploit fogging the challenge of selecting optimal location in the network to which tasks may be mapped in a constantly dynamic environment to achieve minimized latency, improve inter fog node communication and achieve better load balancing is still a challenge, moreover, minimized resource utilization in fog-cloud of things ecosystems is another important problem to study [134, 172].

Among the important problems to be resolved in computational offloading solutions is finding an appropriate offsite infrastructure to which resource-constrained user terminals in the fog-cloud of things ecosystems may offload complex applications so as to improve the general performance of the IoT ecosystems. Offloading process improves performance by reducing program turnaround and increasing system throughput [173]. This is done by minimizing communication overheads and maximizing resources utilization across the fog nodes. Inopportunely, there exist a trade-off in achieving an appropriate offsite infrastructure with sufficient resources to offload task(s) and minimizing intra-infrastructure communication that achieves sufficient turnaround required by IoT applications. This trade-off makes this problem NP-complete [174, 175]. NP-complete problems have no optimal solution in polynomial time, therefore, may yield better solutions when heuristic techniques are used.

From Solutions proposed in the literature, a number of optimisation techniques have been used to accomplish computational offloading in the IoT-fog ecosystems. For instance, Zhou in [176] explored contract-matching approach to task assignment and resource allocation in Vehicular Fog Computing (VFC). In their study, they proposed an efficient incentive mechanism based on contracts. They further transformed the task assignment into a two-sided matching between vehicles and user equipment. From their numerical result, the matching algorithm proposed was observed to improve performance. De Jong [177] proposed a deterministic delay constrained task partitioning as a mechanism to solve offloading decision. In their study, they highlighted the previous studies in which algorithms based on integer linear programming and stochastic analysis formulation were seen to under perform. They observe these mechanisms didn't guarantee polynomial-time convergence. To improve the performance of the system, they proposed a "deterministic approach", a proposal that guarantees polynomial convergence.

Recently, a number of heuristic mechanisms have been proposed for example ant colony mechanism was proposed in [168], they formulate multi-tasks scheduling as an optimization . Their optimization objective was to maximize profit and constraints. Their proposals performed better than previous explored mechanisms based on deterministic methods. In our previous work in [33] we proposed a pipeline of machine learning mechanisms to perform computational offloading. Amongst the mechanisms that was used in the pipeline to facilitate selection of suitable fog node for computational offloading was Particle Swarm Optimization (PSO). We assumed during the offloading process the network conditions and resources utilization remained unchanged. This study considered a Simple PSO(SiPSO) mechanism in which for every offloading scenario in a cluster there arises one peak fog that provides maximum processing power and other resources. This peak node is the one considered the most suitable candidate to execute an offloading process. The results of this study enumerated considerable improvement in resource, throughput, and energy utilization. Further there was considerable improvement in response time.

In the same study, dynamic nature of IoT environment that resulted in many IoT devices that may initiate offloading at the same time was considered, but clustered nature of Fog-cloud of things which could result in multiple topology, different underlying topological functions and batch offloading at diverse points in the network was not considered. Again, for simplicity we did not consider multiple fogs that may arise due to resource fluctuation in the network. Naturally, multiple offloading and varying availability of task processing resources at the offloading points on fog-cloud of things network may provide numerous optimum fog that may offer better offloading performance, or make the offloading positions actively change. This is a shifting fog-peak. The shifting fog-peak results in the peak fogs to gain or lose resources required to process offloading dynamically, this activity makes the algorithm fail to converge, hence requiring the algorithm to diverge and re-converge so as to find the optimum fog. And, most often when using traditional SiPSO the personal best and global best may change resulting in memory loss. Looking at the application of PSO proposed in our previous studies in [33], we intent to extend the proposed mechanism to include a dynamic selection considering multiple clusters. Our proposal takes care of changing dynamic that may offer moving maxima fogs during the offloading processes. This study continues the effort to explore PSO mechanisms and its variations for different offloading conditions in dynamic IoT-fog environment.

The chapter therefore, aimed at designing and developing a modified dynamic particle swarm optimisation algorithm for computational offloading in the fog-cloud of things ecosystem. During the study an evaluation of our algorithm in terms of latency, network utilization and energy consumption was done. Secondly, a comparison of the results obtained through simulation of mDyPSO, SiPSO and related benchmark is completed and summary provided. To achieve the aim of the study we

(i) performed a classification of computational offloading strategies. This enabled us to observe how experts have applied them to solve offloading and other related solutions,

(ii) developed and designed modified Dynamic Particle Swarm Optimisation (mDyPSO) algorithm for computational offloading in a clustered Fog of things ecosystem,

(iii) performed an evaluation of the developed mDyPSO algorithm and tested its performance against Simple Particle Swarm Optimisation and related benchmark.

The remainder of the chapter is organized as follows: Section 4.2 presents review of related literature in terms of offloading technologies, application of nature inspired algorithm for computational offloading and classes of computational offloading strategies as they appear in literature. Section 4.3 and 4.4, presents the proposed system, the network model, formulation of the optimisation framework and modified Dynamic Particle Swarm Optimisation offloading Mechanism. In section 4.5, the simulation results are presented and compared to results of SiPSO. a summary of the chapter is provided in section 4.6.

4.2 Review of related work

4.2.1 Offloading in the fog computing ecosystem

Fog computing paradigm provides data, compute, application and services to the user at the edge [41]. Fog computing is promoted by scenario that require fast and reliable computing closer to the source of data such as in autonomous systems, smart city applications, smart health care, smart infrastructure and disaster management systems. Fog computing is characterised by low latency computing closer to the edge of the network. In addition, they use reliable protocols, provide easy and affordable installations. Also they consist of slightly powerful devices that contain programmable function to allow accommodation of multiple applications. The Fog-cloud of things infrastructure that supports offloading processes run in versatile operating environment that calls for processing big data using powerful artificial intelligence applications. This phenomena may result in the device at the edge to fail handling the application. Besides, IoT devices at the edge are designed to handle very powerful application but come with small battery capacity, this therefore, calls for a mechanism that should allow devices at the edge to conserve energy. Fog technologies are often designed to conserve resources(processing power, energy and memory) by developing mechanisms that accept the edge devices to process active tasks generated by IoT and provide both trust and security.

Fog-cloud of things paradigm provides effective solutions to eliminate latency caused by physical distance between the cloud and devices in request of service coupled with huge volumes of data along the service path. Additionally, they enable organisations to save bandwidth and mitigate network congestion given that essential storage and computing can be provided along the edge [178–180]. Though the fog is viewed as Superior technology that will increase the efficiency of a network through mechanism such as computational offloading, they are deterred by their complexity and expensiveness due to their distributed nature. Their implementation calls for well defined scope in addition to equipment, applications and resources to meet the objective of adoption [181, 182]. Along with their location at the edge, their mobility make them vulnerable to security concerns. Lastly, their processing capacity may require data reduction which may result in partial data processing. Partial data processing at the edge limits their capacity to function in a similar way such as the cloud infrastructure in an intensive big data environment.

To achieve computational offloading at the fog the middle ware system should partition the offloadable tasks, present the task for processing at the host fog device and make an offloading decision. The computational offloading is based on if the ecosystem; (i) support offloading (ii) benefits from the offloading process, and (iii) if the resources are not available to process the application tasks locally [183].

Authors in [139] presented three classes of archetypes that defines computational offloading strategies in IoT-cloud ecosystem. They include homogeneous, heterogeneous and neutral models. To allow for offloading using homogeneous model, the run-time environment must be implemented on both the IoT device and the Fog/cloud infrastructure. This configuration enables the IoT to execute its task independent of network connectivity and Fog/cloud in situations when offloading is not necessary. The results of execution of tasks in this model are not compromised Offloading only happens when it is suitable and unavoidable. On the other hand, heterogeneous offloading models require that the run-time environment implementation is simpler and lighter for the IoT devices and complete for the fog/cloud environment. This enables the IoT devices to execute their own task but the result may not be as good as the one produced by the fog/cloud run-time environment. During offloading, input data is transmitted to the server and result of the computation received. Whereas, the neutral models does not require the run-time environment to be installed on the IoT-device during task outsourcing. Therefore, the IoT device must always consult with the fog/cloud to execute offloading. In this model IoT can not execute offloading independent of network connectivity and fog/cloud connection. Generally speaking implementation of computational offloading solution take one of the above three forms.

Offloading decision

The ability of the IoT devices or smart gateway to initiate a decision is regarded as an offloading decision. The decision occurs after an evaluation of application needs to warrant offloading. The evaluation is done in terms of data type, data size, power of devices, status of activities at the initiation point, intermediate nodes, and the end node. Furthermore, the need to improve performance of the IoT ecosystem is at the heart of offloading decision strategy. An offloading decision is passed subject to whether the application can benefit from offloading in terms of the overall performance of the system. Also, an evaluation to determine if data, its code or its application require to be offloaded. for every offloading strategy the destination to where an offloading request shall be processed is critical for a rational offload decision [183]. sometimes it is important to consider the portion of data that should be offloaded. Finally, offloading strategy used to perform offloading is critical for offloading. Most often offloading decisions are taken by middleware installed over the top of smart device.

The decision is held based on whether the application at hand i) needs extra computational resources in excess of the hosting device, ii) if it is latency sensitive, iii) if it is security or privacy sensitive, iv) does it require its data to be stored on storage space in excess to what the hosting device has, v) if the application at hand is demanding such that it's execution may degrade the performance of the system during operation, and, vi) does offloading improve the general quality of service of the whole IoT system. Other means through which an offloading decision may be necessary is if offloading is supposed to improve infrastructure utilization as observed in load balancing, parallel processing and distributed computing.

Application task partitioning

Application task partitioning is another important activity performed in preparation for offloading. it involves dividing an application tasks into subsequent chunks that can be executed either on a client device or the server. In a number of studies authors have shown that a good design strategy towards an optimum partitioning solution can affect resources utilization at run time, further levels of granularity affect compatibility, offloading, and performance in general [184]. Partitioning mechanisms can be static, dynamic or hybrid. Hybrid mechanism bridges the gap between adaptability to offloading conditions and balancing cost of performance.

Li et al. in [185] presented a partition scheme at a procedure call level. Their solution is based on a cost graph to model the behaviour of the task assignment during offloading. They further explored the computational expensive branch and bound mechanism for task assignment and improving it by implementing pruned heuristic component that improved its performance considerably. Gao at al. [186] proposed a layered computational strategy that performed partitioning of tasks based on deep neural networks. in their study joint optimisation design that minimised latency by optimising task allocation and offloading was realized. Another study by Jianhui et al. [187] illustrated that when portioning is done at both the mobile device and/or on the remote/edge device to facilitate further processing latency is minimised. These studies confirms that partitioning of task during or before the offloading process is an important factor. In our study, we consider the first partitioning occurs at the IoT device. Here the task is partitioned such that a portion that is processed on the local device is considered for offloading and further partitioning may occur at the smart gateway so that the task may be processed by multiple Fogs.

Preparation

During the preparation stage actions that are necessary to support successful optimal offloading performance are finalized. Three events are important here i.e i) selection of destination offsite location where offloading will occur, ii) Transfer and installation of code, and, iii) transfer of data to and from the offload site [188]. Preparation is done in an effort to initialize the offloaded process on the offsite environment. [49]. Mitsis et al. in [189] showed the importance of well designed selection mechanism. In their simulation study they proposed a two component data offloading and MEC server selection algorithm based on stochastic learning automata. Their mechanism scored sufficient performance improvement in realizing optimal offloading and pricing. Therefore,in our study we opt to develop mDyPSO in a Fog-cloud of things environment to improve performance by using PSO.

4.2.2 Nature inspired algorithm for computational offloading and related problems

The dynamic nature of applications, computational and data transactions that are supported by IoT-Fog-cloud ecosystems inspire the use of evolutionary algorithms based on either genetic algorithms or swarm intelligence. Yang et al. in [190] performed an analysis of nature inspired algorithms and their applications. In their study, they elaborated the two broad categories of nature inspired algorithms falling in procedure based and equation based. They further provided examples of such algorithms to include Deferential Evolutionary(DE), Particle Swarm Optimisation(PSO), fire-fry algorithm, Bat algorithm, Cuckoo search algorithm etc.

Also studies in [191] performed a comparative study of nature inspired algorithm on travelling salesman problem to show how they provide platform to solving combinatorial optimization. These algorithms provide diverse opportunities to solving many problems that arise in the IoT-Fog ecosystems except for lack of well-formed frameworks to enable a proper consideration of their efficiency, effectiveness and their robustness in the new computing environment. In this work we are in efforts to explore PSO for offloading mechanism in IoT environment.

PSO is principally based on social behavior of animals [192]. In a PSO system, multiple solutions co-exist and collaborate by iterative changing their current position until an optimum solution is achieved. Each set of candidate solutions are known as particles. In an bid to find an optimal solution in a search space of D dimension, the particles fly around adjusting their position in accordance with is personal experience (P_b) and neighbour particles experience (P_g). Each particle preserves a memory of its own experience and best experience of the neighbourhood. PSO combines particle dynamics and information sharing to derive a powerful heuristic optimization tool. The canonical PSO achieves optimisation through the following two equation 4.1 and 4.2

$$V_{ij}(t+1) = \omega V_{ij}(t) + C_1 \times r_1 \times (P_{b(i,j)} - X_{ij}(t)) + C_2 \times r_2 \times (P_g - X_{ij}(t)) \quad (4.1)$$

$$X_{ij}(t+1) = X_{ij}(t)) + V_{ij}(t+1) \quad (4.2)$$

where $V_{ij}(t+1)$ and $V_{ij}(t)$ are current and previous velocity respectively, whereas $X_{ij}(t+1)$ and $X_{ij}(t)$ are current and the previous particle positions, c_1 and c_2 are cognitive and acceleration coefficients, r_1 and r_2 are random numbers between 0 and 1, ω is the initial coefficients, t are the number of iterations [193] and lastly, $P_{b(i,j)}$ and P_g are personal and global best respectively [194].

Originally, PSO has been thought to solve optimization problems that were continuous in nature. Recently, PSO has attracted interest in solving both discrete and combinatorial problems with small modification, these problems include finding complex solutions computer aided design as illustrated by [195] in their work. Resendo et al. in [196] presented PSO with path rethinking for combinatorial optimization problems. In their work, they presented a PSO algorithm in which the particle was viewed to be guided by three components. These components include i) the component that is guided by it's own way k_1, ii) the other component that allows it to get to its previous best solution k_2 and iii) the component that allows it to align with the global solution of the whole swarm k_3. Component k_1 facilitates the local search, where as k_2 and k_3 helps move the particle to the new position. Grouping the components into functional partitions allows them to create the local search and the path rethinking dimension of the velocity equation in 4.1. From their formulation the PSO equations can be formed as in eqn. 4.3 and 4.4 as follows ;

$$V_{ij}(t+1) = \omega k_1(t) + C_1 \times r_1 \times (K_2(t)) + C_2 \times r_2 \times (k_3(t)) \tag{4.3}$$

$$X_{ij}(t+1) = X_{ij}(t)) + V_{ij}(t+1) \tag{4.4}$$

Referring to eqn. 4.3 $\omega k_1(t)$ represents the local search and the remaining part of the equation form the path rethinking. During their study, they Performed experiments to show how their results compared well with other solution presented in literature. This study gives evidence that hybridization is one of the ways of attaining a competitive discrete and dynamic PSO. Gupta et al. [197], proposed a hybrid Genetic Algorithm-Particle swarm optimization (GA-PSO) to solve travelling salesman problem. In their algorithm they take advantage of fast convergence rate of PSO and robustness of GA. Their result show that hybrid GA-PSO achieves better computational average mean time and low mean error, thus attaining superior performance. Mohammed et al. [198] investigated the application of PSO to solve the shortest path problem. They experimented their work of different network topology. In there study good success of discovering the shortest path was realized as compared to GA.

Authors in [199] and [200] presented PSO for transport problem and assignment problem respectively. In both studies they presented tractable solutions to the problems in questions. [199] noted in there study that the PSO traditional updating rule does not hold for the constrains that result from formation of transport problem. Therefore, they presented an alternative updating rule that suites transport problem as stated in equation 4.5 and equation 4.6

$$V_{ij}(t+1) = \begin{cases} r_1 \times (P_b(t) - X(t)) + r_2 \times (P_g(t) - X(t)), t = 0 \\ r_3 \times Vij(t) + r_4(P_b(t) - X_{ij}(t)) + r_2 \times (P_g(t) - X(t)), t > 0 \end{cases} \tag{4.5}$$

$$X_{ij}(t+1) = Vij_(t+1) + X_{ij}(t) \tag{4.6}$$

They considered the following conditions to overcome the shortcoming of the traditional PSO to solve TP problems

(i) $X(t)$ and V_{ij} is viewed in $n \times m$ dimension

(ii) r_1, r_2, r_3, r_4 are random number between 0 and 1

(iii) r_1, r_2 $r_1 = U(0,1), r_2 = 1 - r_1$

(iv) r_3, r_4 $r_1 = U[0.8, 1), r_4 = 1 - r_3$

(v) if $P_b(t) = X(t)$ AND $P_g(t)! = X(t)$ then $r_1 = 1$

(vi) if $P_b(t)! = X(t)$ AND $P_g(t) = X(t)$ then $r_2 = 1$

(vii) if $P_b(t) = X(t)$ AND $P_g(t) = X(t)$ then $r_3 = 1$.

Our study is based on assignment of tasks to a fog that has capacity. This problem is viewed as version of TP problem.

Rafique et al. presented Bio-inspired hybrid strategy based on PSO to schedule tasks and Cat swarm optimisation (CSO) algorithm to manage resources. Results of their study show with slight modification of the PSO and CSO NBIHA [201] balances load well amongst Fog nodes hence presenting an efficient resource allocation strategy. we deduce from this study that further tuning of PSO would yield improved performance in the Fog-cloud of things environment.

Similarly ML based on Deep learning has been presented in Heidari at el. [202]. Their deep Q-Learning technique for offloading computation in blockchain-enabled green IoT-Edge scenarios, in their study, they proposes an approach to address the problem of IoT-edge offloading enabled blockchain using the Markov Decision Process. The proposed approach employs the Post Decision State mechanism in online mode, and integrates edge/cloud platforms into IoT blockchain-enabled networks to encourage the computational potential of IoT devices. The system can be used both online and offline while maintaining privacy and security, and dynamically chooses and changes the master controller, offloading decision, block size, and processing nodes to reduce device energy consumption and cost. The results of proposed method outperforms four benchmarks in terms of cost, computational overhead, energy use, task failure rate, and latency. this Study therefore, provides a good yardstick to compare the results of our proposal.

In another study [203] Sun et al. developed a cache strategy for mobile edge networks that utilizes deep reinforcement learning for computational offloading. The objective of their study was to improve network energy efficiency, and they proposed an intelligent caching strategy that combines a deep neural network (DNN) and the Q-learning algorithm. The DNN is used to approximate the action-state value function in the Q-learning solution, and the stochastic gradient descent method is employed to improve the parameter iteration strategies in the proposed DQN algorithm for faster convergence to the optimal solution. Their proposed intelligent DQN-based content cache strategy significantly improves the energy efficiency of mobile edge networks, and their results demonstrate that with sufficient training steps, the proposed strategy can optimize the network performance of the content caching policy. Additionally, this study provides a basis for comparing an offloading policy for further offloading between fog and cloud in our proposal. Drawing from the experiences of the above authors we build confidence that our proposal may provide better platform to solve offloading problem.

4.2.3 Classes of computational offloading strategies

In this section we present 5 classes of taxonomy of computaional offloading mechanisms as presented in Table 4.1. In the first class, the focus is on energy minimization, which involves jointly optimizing energy and task completion while considering the time and energy constraints of mobile users. The second class deals with task offloading using integer programming, aiming to address the challenge of representing the high dimensions of the distributed system environment in fog computing. The third class explores bio-inspired methods for task scheduling and resource allocation, utilizing optimization problems and bio-inspired algorithms to match resources for computation in fog computing. The fourth class discusses machine learning-based offloading approaches that employ supervised, unsupervised, and reinforcement learning methods, including deep learning, to maximize network performance and enhance system utility. However, these approaches have limitations such as heavy training requirements, lack of explainability, and complex debugging. The research domain encompasses fog, MEC, cloud, and mist. Lastly, the fifth class focuses on stochastic offloading, which takes into account randomness in task generation, processing, and communication among nodes. Stochastic models help in understanding offloading behavior and complexities but face convergence issues when dealing with larger, dynamic, and complex systems.

Table 4.1: The state of art strategies for computational offloading

Sno.	Proposed offloading strategy	Principle	Limitation	Edge technology	Reference
1	Offloading for Energy minimizing in mobile-edge cloud computing	Formulated as joint energy and Task completion minimization problem for mobile users. mobile users are constrained by both time and energy consumption. Formed as a convex optimisation problem	formation results in complex system that difficult to solve for distributed Fog ecosystem	mobile Edge, cloud	[204–206]
2	Task offloading based on integer programming	Formulated as mixed integer non-linear programming problem. the problem is often transformed sub problems to tune the results of the system	Difficulty in representing high dimensions of distributed system environment created by the fog system	software defined mobile edge	[205, 207, 208]
3	Bio-inspired task scheduling and resource allocation.	Formulated as optimisation problems to be solved using bio-inspired algorithms. Approaches include modification of PSO, CSO, Bee swarms, etc. These methods are used to schedule tasks, resources, and demands	difficulty in matching resources for computation	Fog computing	[209–211]

Sno.	Proposed offloading strategy	Principle	Limitation	Edge technology	Reference
4	Machine learning-based offloading	Formulated based on machine learning approaches. They include supervised, semi-supervised and supervised Approaches. Deep learning and reinforcement learning have been widely proposed in literature. These methods have shown that reliable offloading that maximizes network performance and improves system utility is possible	They suffer from heavy training requirements, lack of explainability, and intelligence. Often the underlying systems are complex to debug	Fog MEC, Cloud, Mist	[212–214]
5	Stochastic offloading	Formulated with association of randomness in the system that generates the tasks, or the systems that processes the offload. Communication between the nodes may also be random. The success of stochastic systems are mainly dependant on task uploads and download possibilities	Stochastic models help in learning the behaviour of offloading and complexities involves but, suffer from convergence issues when the systems under study become larger, dynamic and complex	Fog, Mec	[215–217]

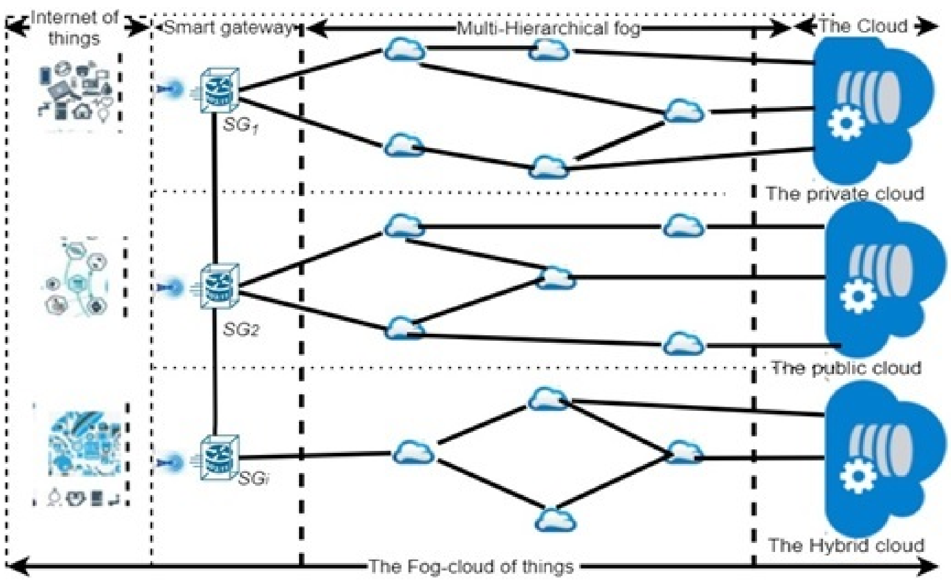

FIGURE 4.1: Multi clustered Network Model

4.3 Problem Formulation

4.3.1 Network model

Figure 4.1 shows the network model used in this study. We consider a moderately large IoT network of N devices. The devices in the network are grouped into smaller C clusters. C_k is the k^{th} cluster and k is a positive number. This model consists of p IoT devices at the lower layer, p Fog nodes organized in the multi-hierarchical middle layer and r clouds devices in the upper layer. In the model, the number of IoT devices, fog devices and the cloud devices are such that $p >> q > r$. All IoT devices in a single cluster are connected to the upper layers through the smart gateways. The fogs are in turn connected to the cloud infrastructure consisting of private clouds, public clouds or hybrid clouds systems. All packets shall enter or exit the cluster through the smart gateway (G_k). Each IoT device I belongs to some cluster C_k.

Application tasks($A\tau$) arrives to the IoT randomly and periodically. These tasks are associated with resource requirements. If the generated application tasks cannot be completed in the required sums of time slot (t) at I, an offload decision is initiated or the task is dropped [33]. The task generated at this point may be delay, resource constrained or demand driven. That means the tasks must be completed within specified time bound or the tasks demands processing requirements that are not available at I to complete the task. To this end, the offloading decision may be to execute the tasks in whole on a local device or offload the task(s) to fog device(s) in the next layer. Moreover, the offload decision might support further offloading to the cloud.

Let us consider $0 \leq \lambda \leq 1$ to be a fraction of the application tasks that shall be offloaded during offloading process. If $\lambda = 0$ then the task is executed in totality at point of generation that is the local device I else if $\lambda = 1$ then the task is offloaded in whole to the fog except for the none offloadable parts of the task, otherwise a portion λ of the task is offloaded to the fog and $1 - \lambda$ will be executed on the local device [33].

Each Task ($A\tau$) generated is characterized by input component (α), output component (β) and computational component (γ). Thus $A\tau$ is represented as a tuple $A\tau = \{\alpha, \beta, \gamma\}$. To offload $A\tau_i$ to a fog j a considerable delay bound (δ_i) is required. δ is the sum of transmission delays (δ_{tr}), queuing delay (δ_{qd}), fixed processing delay (δ_{fd}), packetization delay (δ_{pd}) and depacketization delays (δ_{dd}). Based on the the portion of data that may be offloaded a decision models may be chosen.

Let us denote f_i to be the i^{th} fog node to which an offload occurs at time slot (t_s); where, $i = \{1, 2, 3, \ldots, m\}$ and m, is the maximum number of fog nodes in the network organized in multi-hierarchy. Finally, the lower levels of the network are connected to the fog through smart gateway G_k. Fog nodes may be connected to the cloud directly or indirectly through multiple sub-layers of Fogs devices. The cloud has abundant resources but high latency requirement, therefore, if any task is offloaded to the cloud it shall be executed at negligible execution time. If a task is not executed by any fog, then the fog may perform additional offload to the cloud for complex part of the workload. In general, this framework adopts that there exists Fog(s) that are capable of solving the computational offload, therefore, offloading to the cloud may often be immaterial.

The computational offloading process happens in time slots. The time slots contain three (3) time cycles as follows a) The assignment cycle (T_{as}), the time through which the tasks shall be received by the smart gateway of the cluster, b) The operational cycle (T_{op}), the time cycle at which the PSO mechanism is run to select the optimal fog platform to execute offloaded task, c) The dispatch cycle (T_{di}), the time cycle through which the results shall be received by the IoT devices.

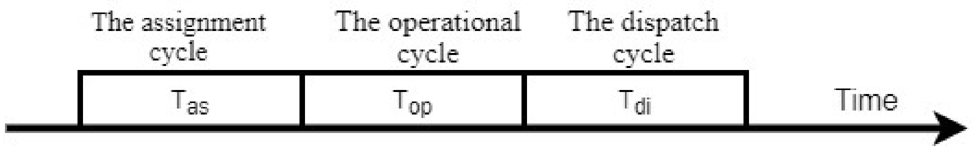

FIGURE 4.2: The offloading Time slots

4.3.2 Problem statement

An IoT device I_{ijk} randomly generates application task($A\tau_i$). $A\tau_i$ consist of offloadable part. If offloadable part does exist then I_{ijk} issues an offloading request Or_k to the smart gateway SG_k. SG_i forwards the request to the selected Fog node F_j which is considered optimal at the time of request. The selection process is driven by heuristic algorithm. The algorithm ensures that the selected Fog node to which an offload process op_{ijk} is initiated minimizes latency, response time etc. during time slot s_t. The offload process happens at the fog. In case an offload process can not happen completely in one time slot, the process is postponed to next time slot in a w the process can be invoked on a cloud thereby invoking a cloud process an offload process invoked on the fog is considered active otherwise the process is idle.

Since the tasks are forwarded to the fog through the gateway, we assume at a certain time(t) the gateway has at least one offloading requests. Further the cloud has unlimited resources and our solution will forward a task to cloud only when there is no fog to solve the task. Therefore, the main concern is to find a fog for each task available at the gateways.

Table 4.2: The time constructs required during offloading process

Offloading decision Model	Formula	Def. of symbols	Description
Local processing model	$\tau_{lp} = (1 - \lambda) \times \frac{\Omega_{iot}}{f_{iot}}$	(i) λ is the fraction of work load that is offloaded, (ii) Ω_{iot} is work load generated at the IoT, (iii) f_{iot} is the processing power of the IoT device	Time required to process a fraction of workload generated at IoT device this includes User interfaces, peripheral management programs etc. [16, 33, 148]
Processing offloaded work load at the Fog	$\tau_{lp} = \lambda \times \frac{\Omega_{iot}}{f_{fog}}$	(i) f_{fog} is processing power of the fog processor	Time required to process a fraction of a workload generated at IoT and offloaded to the Fog for processing. [16, 33, 148]
Transmission Time to selected fog	$\tau_{tf} = \lambda \times \frac{\Omega_{iot}}{\beta_{tf}}$	β_{tf} = is the transmission rate on the link(s) between the selected fog node and the IoT device	Amount of time required to transmit the offloaded workload to the fog device [16, 33, 148]
Transmission Time to the cloud	$\tau_{tc} = \lambda \times \frac{\Omega_{iot}}{\beta_{tc}}$	β_{tc} = is the transmission rate on the link(s) between the selected cloud and the IoT device	Amount of time required to transmit the offloaded workload to the cloud [16, 33, 148]

Offloading decision Model	Formula	Def. of symbols	Description
Local processing model	$\tau_{lp} = (1-\lambda) \times \frac{\Omega_{iot}}{f_{iot}}$	(i) λ is the fraction of work load that is offloaded, (ii) Ω_{iot} is work load generated at the IoT, (iii) f_{iot} is the processing power of the IoT device	Time required to process a fraction of workload generated at IoT device this includes User interfaces, peripheral management programs etc. [16, 33, 148]
Processing offloaded work load at the Fog	$\tau_{lp} = \lambda \times \frac{\Omega_{iot}}{f_{fog}}$	(i) f_{fog} is processing power of the fog processor	Time required to process a fraction of a workload generated at IoT and offloaded to the Fog for processing. [16, 33, 148]
Transmission Time to selected fog	$\tau_{tf} = \lambda \times \frac{\Omega_{iot}}{\beta_{tf}}$	β_{tf} = is the transmission rate on the link(s) between the selected fog node and the IoT device	Amount of time required to transmit the offloaded workload to the fog device [16, 33, 148]
Transmission Time to the cloud	$\tau_{tc} = \lambda \times \frac{\Omega_{iot}}{\beta_{tc}}$	β_{tc} = is the transmission rate on the link(s) between the selected cloud and the IoT device	Amount of time required to transmit the offloaded workload to the cloud [16, 33, 148]

4.3.3 Formation of optimization framework

An optimization framework for computational offloading in clustered fog-cloud of things is presented in this sub section. The optimisation framework determines the optimal computational offloading strategy.

Optimisation related entities

1. **Gateway:** From our model presented in Figure 4.1, all IoT devices are associated with a smart Gateway forming a cluster, therefore all the tasks generated from the lower levels of the model are surrogated by the gateway. In our optimisation framework, IoT devices are encapsulate by the gateway.
2. **The Fog:** This middle tier computing devices at the edge that is responsible for computational offloading. In case none of the fogs in the clusters can perform computation offloading, then the tasks are forwarded to the upper tier of the network model.
3. **The cloud:** Form the upper tie computing environment that is responsible for computational offloading if and only if there exists no fog from the computing environment to perform computation offloading.

Overall objective function

$$\text{Maximize Task allocation: TA} = \sum_{j=1}^{G} \sum_{i=0}^{T_j} \sum_{k=1}^{F} x_{i,j,k} \tag{4.7}$$

subject to:

$$\sum_{k=1}^{F} x_{i,j,k} \leq 1$$

$i = 1, 2, ...T_j; \quad j = 1, 2, ...G$

$$d_{i,j,k} * x_{i,j,k} \leq D_{i,j,k}, i = 1, 2, ...T_j; \quad j = 1, 2, ...G$$
$$m_k \leq M_k, k = 1...F$$
$$c_k \leq C_k, k = 1...F$$
$$x_{i,j,k} = \{0, 1\}$$

Table 4.3: The symbols used in formation of optimization framework

Symbols	Description
g_j	The j^{th} Gateway $j = 1, 2...G$
$T_{i,j}$	The i^{th} Task at j^{th} Gateway $i = 1, 2...T_j$
f_k	The k^{th} fog to which offloading is performed $k = 1, 2...F$
$D_{i,j}$	Maximum delay tolerance between the task $T_{i,j}$
$d_{i,j}$	Delay experienced during the processing of task $T_{i,j}$
t_d	Round trip time
t_c	Computational time at the fog
$x_{i,j,k} = \{0, 1\}$	Boolean indicator if $x_{i,j,k} = 1$ then that fog is chosen.
	this indicator is used to generate the assignment table
h_{jk}	Number of hope counts between g_i and the selected fog
l_d	the average link delay
m_{ij}	the memory requirement for tij
M_k	the maximum available memory at the f_k
$c_{i,j}$	the required processing power required for a task ti, j
C_k	the maximum available processing power available at the Fog f_k

$$\begin{aligned} d_{i,j} &= t_d + t_c \\ &= t_d \text{ is twice the delay between the selected fog and the gateway} \\ t_d &= 2 \times h_{j,k} \times l_d \end{aligned} \tag{4.8}$$

$$\begin{aligned} d_{i,j,k} &= t_c + 2 \times h_{j,k} \times l_d \\ &= \text{delay assigned if } t_{i,j} \text{ is assigned to a fog } f_k \\ &= x_{i,j,k} \times d_{i,j,k} \le D_{i,j} \end{aligned} \tag{4.9}$$

Resource requirement

(i) Central Processing Unit (CPU) requirement for task $t_{i,j}$ is measured in CPU cycles as follows

$$c_k = \sum_{j=1}^{G} \sum_{i=0}^{T_j} x_{i,j,k} \times c_{ik} \leq C_k \tag{4.10}$$

(ii) Memory requirement for task $t_{i,j}$ is measured in bits as follows

$$m_k = \sum_{j=1}^{G} \sum_{i=0}^{T_j} x_{i,j,k} \times m_{ik} \leq M_k \tag{4.11}$$

4.4 The modified Dynamic PSO for computational offloading

In this section, we present the mDyPSO A PSO motivated computational offloading algorithm applied in multiple cluster topology of fog-cloud of things ecosystem. This algorithm is based on dynamic particle swarm optimization mechanism, which forms a basis of many selection and scheduling problems [218, 219]. Studies in [220, 221] proposed PSO mechanism to solve multi-objective problems. Since this study involves Fog-cloud of things clustered in multiple domains whose topology and search space may vary. It becomes central to treat the problem as a multi-objective particle swarm optimization problem.

4.4.1 Traditional Dynamic Particle Swarm Optimisation algorithm

The traditional particle swarm optimisation is best suited for problems that are represented in n-dimension space [33]. Using a particle with defined velocity, acceleration and communication channels between them, they are made made to drift towards the best suited solutions known as the best fit among the other potential solutions [200]. The dynamic variation of Particle swarm optimisation considers that the swarm size may not be the same or the space consist of varying topology. Therefore in such a case acceleration is weighted by random term and average of the fitness may be considered [222]. in algorithm 5 a dynamic particle swarm optimisation is presented.

```
Method DyPSO():
    $noParticles \leftarrow p = (1, 2, ...)$
    $avaragefit \leftarrow 0$
    $currentfit \leftarrow 0$
    $fitness$
    $personalBest \leftarrow 0$
    $bestFit \leftarrow max(personalBest_p)$
    $globalBest \leftarrow bestFit$
    for all the fogs in the fog list do
        if $currentFit \leq avarageFit$ then
            $currentFit \leftarrow avaragaFit$
            compute $fitness$
        end
        if $personalBest \leq fitness$ then
            $personalBest \leftarrow fitness$
        end
    end
    select a fog with $bestFit$ as $globalBest$
    for All the fog in foglist do
        Compute $particleVelocity$ using equation 4.5
        compute $particlePosition$ using equation 4.6
    end
End DyPSO
```

Algorithm 5: TradDyPSO

4.4.2 Dynamic Task allocation algorithm

Algorithm 7 presents the dynamic task allocation in the fog-cloud of thing ecosystem. Application tasks arrive at the gateway for offloading to the optimal-fog that provides sufficient resources to execute the task with minimum latency.For each of the cluster mDyPSO is initiated to determine an optimal fog node for a cluster. The best fit fog amongst all the clusters is assigned the function of a global best fit to whom the tasks are allocated. If all the fogs are busy and can not execute the tasks allocated in the threshold allocated the task is forwarded for processing to the cloud.

```
Method Dynamic-task-allocation():
    $resource - bank \leftarrow 0$
    $resource - required \leftarrow 0$
    $min - reserve\ sum - of - resources\ fog \leftarrow fog_1, fog_2, ...fog_j$
    $clustersize \leftarrow m$
    $cloud \leftarrow cloud_1, cloud_2...$

    for all incoming application tasks $A\tau_1, A\tau_2, A\tau_3...$ do
        for each of the clusters $K$ do
            $local - optimal - fog \leftarrow DyPSO(fog)$
            if local-optimal-fog is better than global-optimal-fog then
                $global - optimal - fog \leftarrow local - optimal - fog$
            end
            allocate-task(global-optimal-fog)
        end
        if all fogs are busy then
            allocate-task(cloud)
        end
    end
End Dynamic-allocation
```

Algorithm 6: Dynamic Task allocation algorithm

```
Method mDyPSO():
    $local - optimal - fog \leftarrow 0$
    $global - optimal - fog \leftarrow 0$
    $fog \leftarrow fog_1, fog_2, ...fog_j$
    $gateway \leftarrow gw_1, gw_2, ...gw_m$
    $clustersize \leftarrow m$
    $cloud \leftarrow cloud_1, cloud_2...$
    $required_memory \leftarrow m_{i,j}$
    $available_memory_fog \leftarrow M_k$
    $required_cpu_cycle \leftarrow c_{ij}$
    $available_cpu_cycles_fog \leftarrow M_k$

    for each of the clusters $i$ do
        for all incoming application tasks $A\tau_{i,1}, A\tau_{i,2}, A\tau_{i,3}...$ do
            for each fog do
                if $m_{i,j} - M_k \leq 0$ then
                    if $c_{i,j} - C_k \leq 0$ then
                        $f_k \leftarrow DyPSO(fog)$
                        assign $A\tau_{ij}$ to $f_k$
                    end
                end
            end
        end
    end
mDyPSO
```

Algorithm 7: Modified dynamic Particle Swarm Optimisation

4.5 Comparative study between DO2QIEO, SiPSO and mDyPSO

4.5.1 Experimental environment and settings

System configuration

Table 4.4 shows the system configuration and development environment. IFogSim was used to develop and simulate our proposal. In addition, we run our simulation on a machine installed with windows 10, IntelCore i_3, 8^{th} generation. Each processor speed is 2.1GHz and 16GB.

TABLE 4.4: The system configuration

Name	Description
Simulation tool	IFogsim
Operating System	Windows 10
Development Platform	Eclipse IDE for Java Devlopers (2021-03)
Processor	IntelCore i_3 8^{th} Generation 2.1GHz 2.3GHz
Installed Memory	16GB

The choice of using IFogSim in this study is motivated by simplicity underlying the it's architecture in terms of application placement, load balancing, resource application and network utilization [171]. Secondly, IFogSim is free and popular event driven simulation tool used modeling the IoT-fog environment. the underlying architecture enables creation of physical, logical and management components that constitute the fog-cloud of things ecosystem. IFogSim gives us the capacity to evaluate resources management policies based on network usage, energy consumption and other operational costs [223].

Simulation parameters

TABLE 4.5: The simulation parameters

Item	Description/number
Number of IoT nodes	[60-70] nodes
Number of Fog nodes	[2,3,4,5]
Number of cloud	1 hybrid cloud
No of application tasks	[10,20,30,40,50,60,70,80]
Bandwidth	10000
simulation Area	1200 x 200

In our simulation, we consider number of IoT devices ranging from 60 to 70 devices per cluster. Each cluster is bound by a smart gateway. Further a hybrid cloud and fog devices ranging from 2 to 5 was considered. Lastly, a simulation area of 10000 and simulation area of 1200×200. Other parameters that included the task arrival rates, simulation time, traffic types, traffic arrival rates are set to default setting.

4.5.2 Evaluation parameter

(i) Application latency: - in this study we define application latency as the total round trip taken by a data packet to travel to and from the IoT device. The lower the application latency the better the offloading strategy. Higher application latency can strangle network reducing the performance.

(ii) Network usage: - we define network usage as the amount of data that travel back and forth across a network due IoT applications, the fog devices and network users.

(iii) Energy consumed during application execution: This parameter refers to total amount of energy consumed when an application is launched to execute on an IoT device. We note that IoT devices are often constrained by battery life. The less the energy consumed the better the mechanism for running the applications that may require intensive application.

4.5.3 Application latency for mDyPSO as compared with DO2QIEO and SiPSO

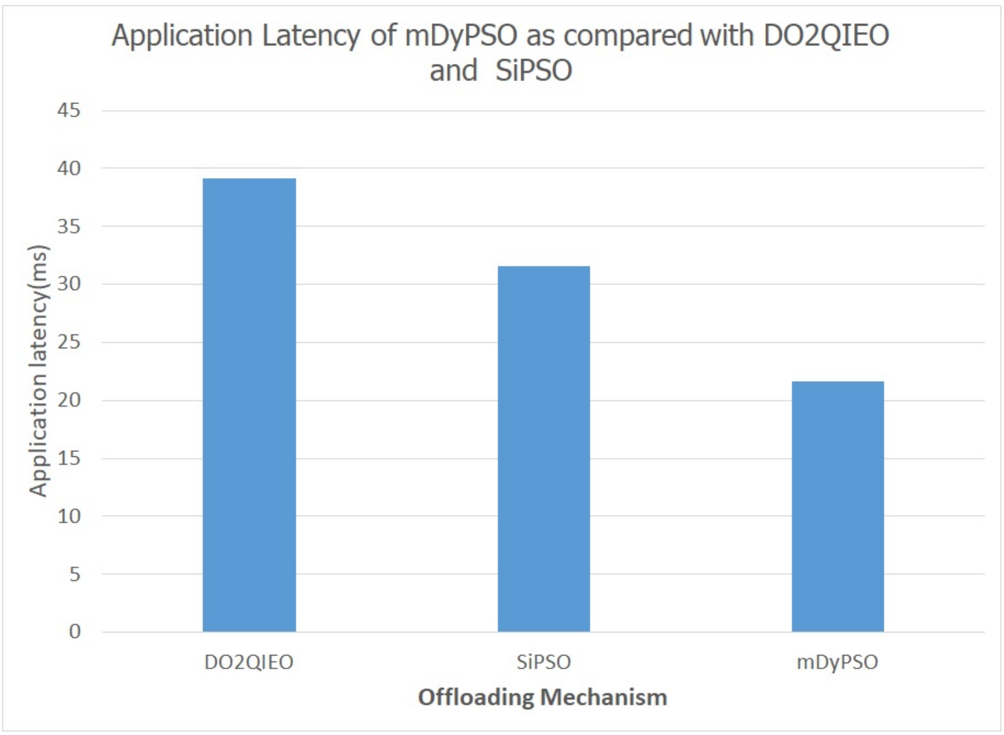

FIGURE 4.3: The application latency for mDyPSO and SiPSO

Figure 4.3 displays the application latency achieved by IoT devices connected to a network using three offloading mechanisms: DO2QIEO from [202], SiPSO from [33], and an improved nature-inspired offloading method based on mDyPSO. Simulation data shows that DO2QIEO achieved an application latency of 39.16 ms, SiPSO scored 31.61 ms, while mDyPSO achieved a latency of 21.61 ms. Offloading mechanisms and multitasking techniques can conceal application latency, and well-designed offloading algorithms can significantly enhance performance. Our modified nature-inspired offloading approach outperforms the work presented in [202] in terms of execution time, primarily due to the incorporation of blockchain in our model. The blockchain consensus mechanism plays a vital role in ensuring secure data transmission at the edge, even though it affects the execution time of the system. It is worth noting that a blockchain-based offloading approach provides better security at the edge, especially when dealing with sensitive data, such as in the case of applications like the Internet of Health Things (IoTH). In contrast, the neural fuzzy model implementation in our study does not offer the same level of security. Therefore, we conclude that the careful design of mDyPSO can enhance offloading performance by approximately one-third.

4.5.4 Computational Delay for mDyPSO as compared with DO2QIEO and SiPSO

In the experiment, the computational delay of mDyPSO is compared with DO2QIEO and SiPSO, as shown in Figure 4.4. The computational tasks are fixed at 80, and the number of fog nodes and cloud nodes are fixed at five and one, respectively. The results demonstrate a gradual increase in computational delays at the IoT devices as the number of nodes increases. For a small number of nodes, the offloading mechanism supported by mDyPSO performs better and is more gradual. In contrast, the gradient of DO2QIEO and SiPSO is slightly higher. This difference can be explained by the fact that mDyPSO is effective in optimizing complex structures in high dimensions compared to DO2QIEO. Additionally, the mDyPSO mechanism does not require large amounts of data to achieve convergence, making it feasible for handling both chained and non-chained data structures.

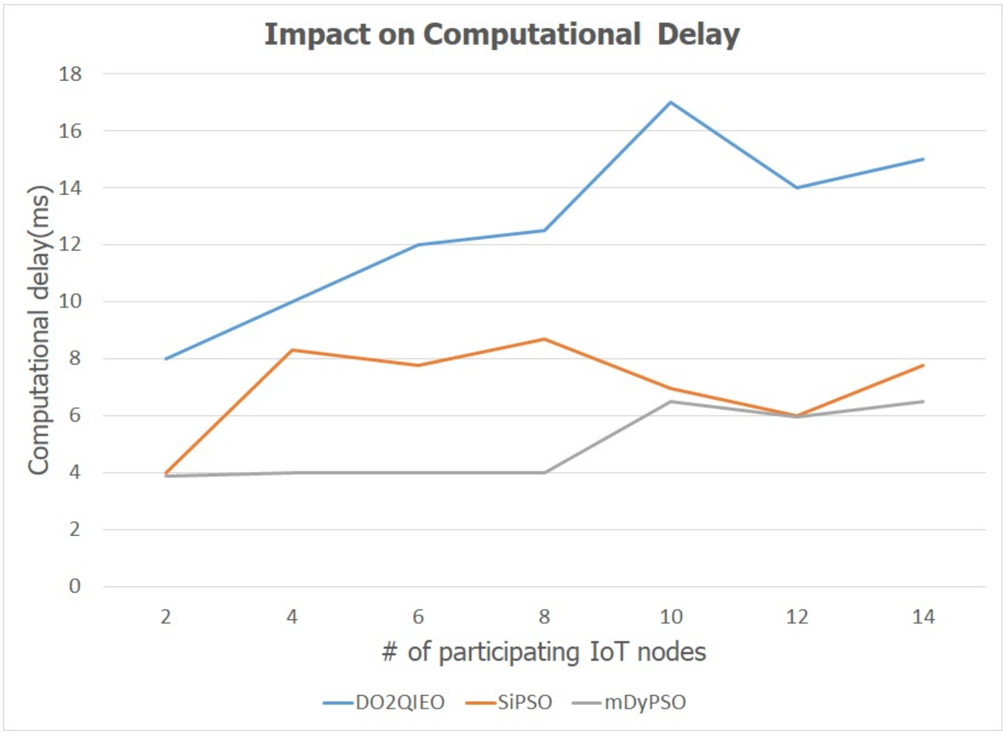

FIGURE 4.4: The computational delay for mDyPSO as compared with DO2QIEO and SiPSO

4.5.5 Energy consumption for mDyPSO as compared to SiPSO

The graph displayed in Figure 4.5 compares the energy consumption of two offloading strategies, namely SiPSO and mDyPSO. From the graph, it can be observed that the energy consumption during offloading using mDyPSO is lower than the energy consumption when using SiPSO. The reduced energy consumption can be attributed to the fact that mDyPSO is designed to optimize PSO parameters as the swarm grows, and the parameters are tuned to minimize the number of iterations required. Even as the number of nodes in the network grows, mDyPSO remains an effective offloading algorithm that minimizes energy consumption.

It is important to note that energy consumption is directly related to energy conservation. Therefore, we can conclude that mDyPSO provides a better offloading strategy in terms of energy conservation at the IoT device.

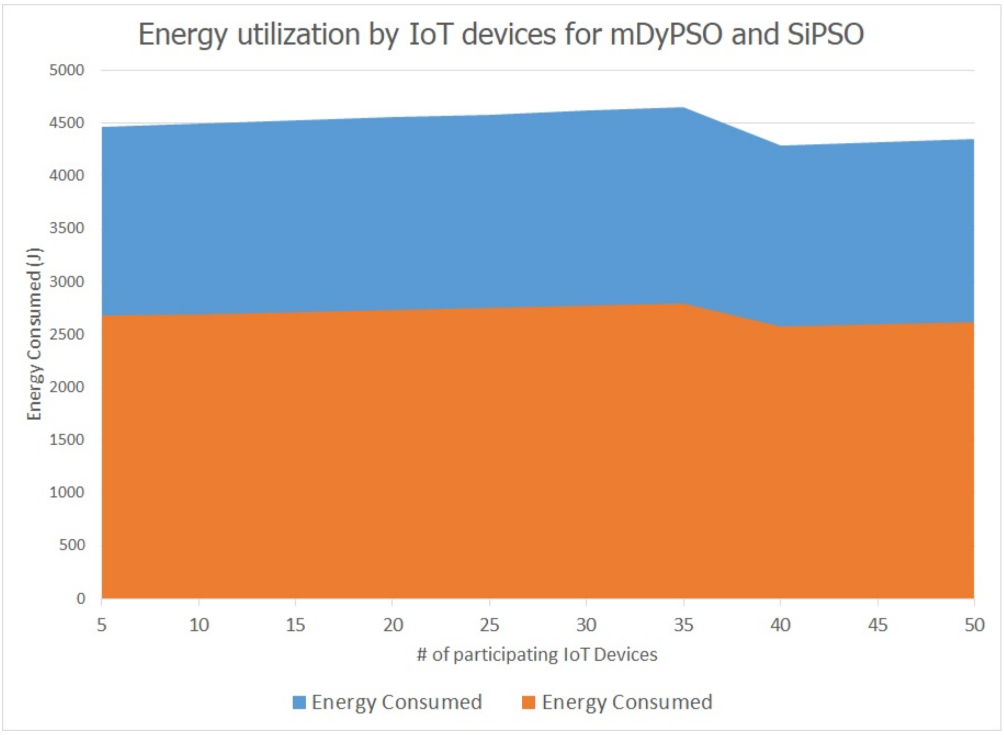

FIGURE 4.5: Energy consumption for both mDyPSO and SiPSO

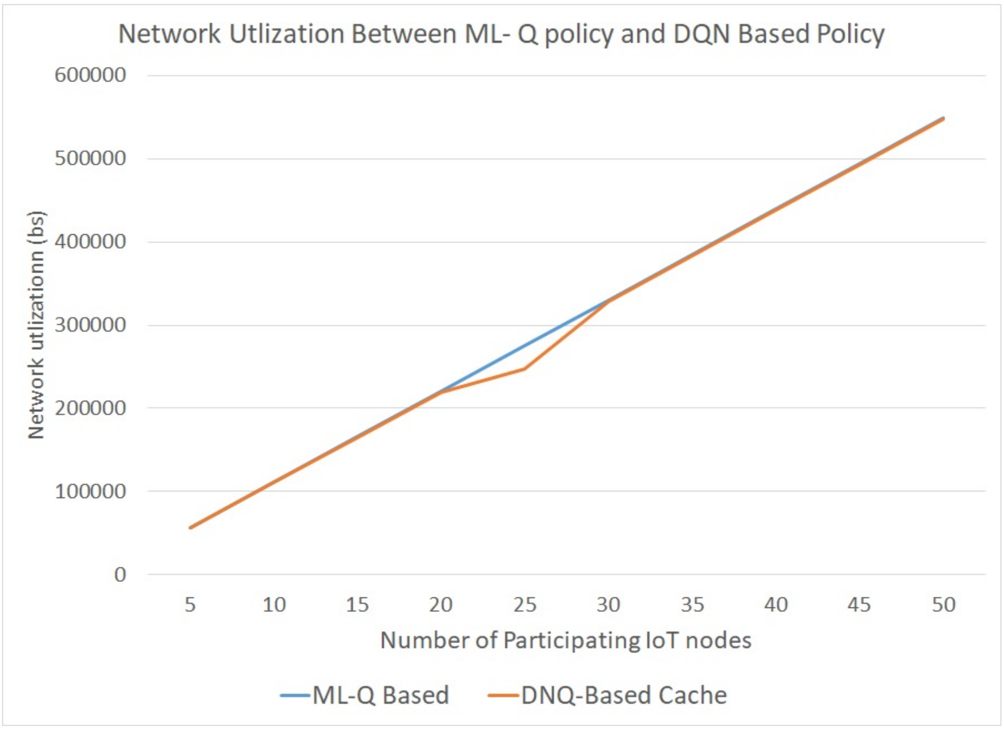

FIGURE 4.6: The network utilization based on ML-Q based and DQN-based cache policy

4.5.6 Network utilization between the ML-Q based and DQN based Policy

The comparison between the network utilization based on ML-Q based further computation offloading and DNQ-based cache mechanism was presented in Figure 4.6. The experiment consisted of 80 computational application tasks, five fog nodes, and one cloud node. By varying the number of participating IoT devices, the results showed a gradual linear increase in network utilization. The ML-Q based policy showed a slightly better utilization of the network, although the difference was not significant. One of the reasons for this is the manageable dimension of the Q table in offloading mechanisms. In many offloading scenarios, there exists an optimal fog node for performing offloading. Furthermore, literature notes that solutions with large dimension DQN-based mechanisms are desired.

4.6 Summary

Fog computing aims at improving responsiveness of real time application. From our study we conclude that Application latency can be reduced through adoption well-designed mechanism in the fog-cloud of things infrastructure. Secondly, we achieve lower response by adapting computational offloading at the edge through proper resource utilization and load balancing. Our Nature inspired computational offloading in clustered fog of things ecosystem exhibits better network utilization as the network grows, a typical characteristics of IoT networks. Further our study show that network performance is reduced by one third, energy consumption is reserved and application latency is moderately lower than earlier proposed mechanism in SiPSO. In future, we hope to explore models for computational offloading with hybrid cloud in IoT ecosystem, 5G-enabled services for task offloading in fog-cloud of things ecosystems and future perspectives for fog-cloud of things computing cooperation.

Chapter 5

Conclusion and Future work

5.1 Conclusion

In this thesis, a general pipeline based on Particle swarm optimisation algorithms, neural-fuzzy and machine learning algorithms was considered for secure offloading in the Fog-cloud of things ecosystem. Our focus was to create a general purpose mechanism for computational offloading introduced in chapter 1. In Chapter 2, we provide extensive survey on Fog computing so as to give a foundation to solutions proposed in Chapter 3 and Chapter 4 respectively. Chapter 2 focused the thesis in terms of concepts, architecture, standards, tools, and applications of cloud-based controlled ecosystem across range of network terminals.

In Chapter 3, we considered SecOFF-FCIoT: a detailed PSO-fuzzy-machine learning based secure offloading in Fog-Cloud of things for smart city applications. In addition, the experimental evaluation of the study was provided. In this chapters, we proposed layered system architecture for offloading scheme is secure and effective for balancing the trade off between latency and energy consumption. The neuro fuzzy model is proposed to eliminate the invalid resources and also optimal fog node is chosen by Particle Swarm Optimisation. The results of our implementation show that the delay is marginal and has negligible energy consumption.

In chapter 4, we continued to explore heuristic approach that permits offloading to optimal offsite fog by developing modified dynamic PSO(mDyPSO) mechanism. We compared our results with the SecOFF-FCIoT which used traditional simple PSO(SiPSO)in non clustered networks. Our simulation results show that mDyPSO out performs SiPSO in terms of application latency, network usage and energy utilization. We note that mDyPSO offloading mechanism improves network performance up to one third. we conclude that mDyPSO mechanism performs well in clustered fluctuating topology by one third. Therefore, considering multiple computational parameters to modify PSO yield better offloading Results. A conclusion to the thesis is provided in chapter 5.

Summary of the main contribution

This thesis presents a new taxonomy for Fog Cloud of Things (FCIoT) ecosystem, aimed at guiding future research in fog and IoT domains. Additionally, we have proposed SecOFF-FCIoT, a secure computation offloading scheme in the Fog-Cloud-IoT environment, which uses machine learning strategies to achieve efficient offloading in the Fog-IoT setting. Further, an improved SecOFF-FCIoT, a modified nature-inspired computational offloading mechanism has been proposed for smart city applications. This mechanism implements dynamic Particle Swarm Optimization (PSO) in a clustered IoT environment. Furthermore, we implement a classifier based on Load Input Ratio (LIR) to categorize data based on complexity and sensitivity in the Fog layer. Lastly, a surrogate entity implementation that ensures stability in mobile-enabled and fluctuating network conditions has been included.

In general, a comprehensive presentation of nature-inspired computational offloading, incorporating machine learning mechanisms has been given. The thesis highlights the latent power and limitations of Particle Swarm Optimization (PSO), neural-fuzzy techniques, and machine learning approaches in enhancing performance in the FCIoT ecosystem through computational offloading.

5.2 Future Directions

Currently, a number of state-of-art research dedicated to the concept of offloading technologies for IoT application is ongoing. These works are of great significance to the collaboration of the edge computing with theories and methods of decision making. Generally, the perception of hybrid task offloading empowers IoT application in different ways. These include performance of complex task in constrained ecosystem of less computing capacity and limited battery power. The future studies will involve i) introducing new technologies such as blockchain in decentralizing security, privacy and trust issues at all levels of continuum from IoT devices to the cloud. ii) Another emerging challenging and interesting problems in fog computing is parallel programming for Fog and Edge computing environments. The classes of these problems may include; federated learning, parallel programming models for data optimisation and management across the fog systems, 5G /6G enabled programming models and their application in fog computing.

5.3 List of publications

1. A.A. Alli and M.M. Alam, ”Seccoff-fciot: Machine Learning Based Secure Offloading in Fog-Cloud of Things for Smart City Applications. ” *Internet of Things*, Vol 7, p. 100070, Sep. 2019

2. A.A. Alli and M.M. Alam, ”The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications.” textitInternet of Things, vol 9, P.100177, 2020

Bibliography

[1] ETSI. ETSI GS MEC 002. Mobile Edge Computing (MEC); Technical Requirements. *ETSI White Paper*, 1:1–40, 2016. URL http://www.etsi.org/standards-search.

[2] Keith Dyer. The Mobile Network ≫ MEC architecture, 2016. URL http://the-mobile-network.com/2016/04/etsi-establishes-the-foundations-of-mobile-edge-computing/mec-architecture/.

[3] Joshua E Siegel, Sumeet Kumar, and Sanjay E Sarma. The future internet of things: secure, efficient, and model-based. *IEEE Internet of Things Journal*, 5(4):2386–2398, 2018.

[4] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017.

[5] Massimo Ficco, Christian Esposito, Yang Xiang, and Francesco Palmieri. Pseudo-dynamic testing of realistic edge-fog cloud ecosystems. *IEEE Communications Magazine*, 55(11):98–104, 2017.

[6] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In *International conference on wireless algorithms, systems, and applications*, pages 685–695. Springer, 2015.

[7] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper - Cisco, 2016. URL https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[8] Habib Ur Rehman, Muhammad Asif, and Mudassar Ahmad. Future applications and research challenges of iot. In *2017 International Conference on Information and Communication Technologies (ICICT)*, pages 68–74. IEEE, 2017.

[9] Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. Security and privacy challenges in industrial internet of things. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2015.

[10] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. *National institute of standards and technology*, 53(6):50, 2009.

[11] Karthik Kumar and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51–56, 2010.

[12] Yuan Ai, Mugen Peng, and Kecheng Zhang. Edge computing technologies for internet of things: a primer. *Digital Communications and Networks*, 4 (2):77–86, 2018.

[13] Hany Atlam, Robert Walters, and Gary Wills. Fog computing and the internet of things: A review. *Big Data and Cognitive Computing*, 2(2):10, 2018.

[14] Ashkan Yousefpour, Genya Ishigaki, Riti Gour, and Jason P Jue. On reducing iot service delay via fog offloading. *IEEE Internet of Things Journal*, 2018.

[15] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

[16] Minghui Min, Dongjin Xu, Liang Xiao, Yuliang Tang, and Di Wu. Learning-based computation offloading for iot devices with energy harvesting. *arXiv preprint arXiv:1712.08768*, 2017.

[17] Mohammad Aazam and Eui-Nam Huh. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*, pages 687–694. IEEE, 2015.

[18] Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra. Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing*, 6(1):46–59, 2018.

[19] sdxcentral. Mobile edge computing vs. multi-access edge computing. *sdxcentral.com*, 2018. URL https://www.sdxcentral.com/mec/definitions/mobile-edge-computing-vs-multi-access-edge-computing.

[20] Afnan Fahim, Abderrahmen Mtibaa, and Khaled A Harras. Making the case for computational offloading in mobile device clouds. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 203–205. ACM, 2013.

[21] Clinton Dsouza, Gail-Joon Ahn, and Marthony Taguinod. Policy-driven security management for fog computing: Preliminary framework and a case study. In *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on*, pages 16–23. IEEE, 2014.

[22] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[23] Real Carbonneau, Kevin Laframboise, and Rustam Vahidov. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3):1140–1154, 2008.

[24] Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17, 2015.

[25] Geoffrey I Webb, Michael J Pazzani, and Daniel Billsus. Machine learning for user modeling. *User modeling and user-adapted interaction*, 11(1-2):19–29, 2001.

[26] Thuy TT Nguyen and Grenville J Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys and Tutorials*, 10(1-4):56–76, 2008.

[27] John M Mulvey. Machine learning and financial planning. *IEEE Potentials*, 36(6):8–13, 2017.

[28] Rahul C Deo. Machine learning in medicine. *Circulation*, 132(20):1920–1930, 2015.

[29] Gianni D'Angelo and Salvatore Rampone. Diagnosis of aerospace structure defects by a hpc implemented soft computing algorithm. In *2014 IEEE Metrology for Aerospace (MetroAeroSpace)*, pages 408–412. IEEE, 2014.

[30] Gianni D'Angelo and Salvatore Rampone. Feature extraction and soft computing methods for aerospace structure defect classification. *Measurement*, 85:192–209, 2016.

[31] Tara A Estlin and Raymond J Mooney. Learning to improve both efficiency and quality of planning. In *IJCAI*, pages 1227–1233, 1997.

[32] Knud Lasse Lueth. State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating, 2018. URL https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[33] Adam A Alli and Muhammad Mahbub Alam. Secoff-fciot: Machine learning based secure offloading in fog-cloud of things for smart city applications. *Internet of Things*, 7:100070, 2019.

[34] Yang Yang. Multi-tier computing networks for intelligent iot. *Nature Electronics*, 2(1):4, 2019.

[35] Syed Noorulhassan Shirazi, Antonios Gouglidis, Arsham Farshad, and David Hutchison. The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective. *IEEE Journal on Selected Areas in Communications*, 35(11):2586–2595, 2017.

[36] Mithun Mukherjee, Lei Shu, and Di Wang. Survey of fog computing: Fundamental, network applications, and research challenges. *IEEE Communications Surveys & Tutorials*, 20(3):1826–1857, 2018.

[37] Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito, and Omer Rana. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology (TOIT)*, 19(2):18, 2019.

[38] Ranesh Kumar Naha, Saurabh Garg, Dimitrios Georgakopoulos, Prem Prakash Jayaraman, Longxiang Gao, Yong Xiang, and Rajiv Ranjan. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE access*, 6:47980–48009, 2018.

[39] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[40] Orlando Arias, Jacob Wurm, Khoa Hoang, and Yier Jin. Privacy and security in internet of things and wearable devices. *IEEE Transactions on Multi-Scale Computing Systems*, 1(2):99–109, 2015.

[41] 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*, pages 13–16. ACM, 2012.

[42] Daniele Catteddu. Cloud computing: benefits, risks and recommendations for information security. In *Web application security*, pages 17–17. Springer, 2010.

[43] IEEE Standard Association et al. Ieee 1934-2018-ieee standard for adoption of openfog reference architecture for fog computing, 2018.

[44] 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*, 98:27–42, 2017.

[45] Mung Chiang and Tao Zhang. Fog and iot: An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6):854–864, 2016.

[46] Dongxin Lu and Tao Liu. The application of iot in medical system. In *2011 IEEE International Symposium on IT in Medicine and Education*, volume 1, pages 272–275. IEEE, 2011.

[47] K Natarajan, B Prasath, and P Kokila. Smart health care system using internet of things. *Journal of Network Communications and Emerging Technologies (JNCET) www. jncet. org*, 6(3), 2016.

[48] Prasanth Lade, Rumi Ghosh, and Soundar Srinivasan. Manufacturing analytics and industrial internet of things. *IEEE Intelligent Systems*, 32(3): 74–79, 2017.

[49] Khadija Akherfi, Micheal Gerndt, and Hamid Harroud. Mobile cloud computing for computation offloading: Issues and challenges. *Applied computing and informatics*, 14(1):1–16, 2018.

[50] Manas Kumar Yogi, K Chandrasekhar, and G Vijay Kumar. Mist computing: Principles, trends and future direction. *arXiv preprint arXiv:1709.06927*, 2017.

[51] Ambrose McNevin. Edge computing in broadcasting — Industry Trends — IBC, 2017. URL https://www.ibc.org/tech-advances/edge-computing-in-broadcasting/1881.article.

[52] OpenfogConsortium. OpenFog Reference Architecture for Fog Computing . *Reference Architecture*, 1(February):1–162, 2017. ISSN 2047-4954. doi: 10.1049/iet-net.2015.0034.

[53] Dario Sabella, Alessandro Vaillant, Pekka Kuure, Uwe Rauschenbach, and Fabio Giust. Mobile-edge computing architecture: The role of mec in the internet of things. *IEEE Consumer Electronics Magazine*, 5:84–91, 10 2016. doi: 10.1109/MCE.2016.2590118.

[54] By Dario Sabella, Alessandro Vaillant, Pekka Kuure, and Uwe Rauschenbach. Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things. *IEEE Consumer ElEctronics magazine*, 1(September):84–91, 2016. doi: 10.1109/MCE.2016.2590118.

[55] Andrea Giordano, Giandomenico Spezzano, and Andrea Vinci. Smart agents and fog computing for smart city applications. In *International Conference on Smart Cities*, pages 137–146. Springer, 2016.

[56] Behailu Negash, Tuan Nguyen Gia, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, Tomi Westerlund, Amir M Rahmani, Pasi Liljeberg, and Hannu Tenhunen. Leveraging fog computing for healthcare iot. In *Fog Computing in the Internet of Things*, pages 145–169. Springer, 2018.

[57] Chao Zhu, Giancarlo Pastor, Yu Xiao, and Antti Ylajaaski. Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges. *IEEE Communications Magazine*, 56(10):58–63, 2018.

[58] Ashish Rauniyar, Paal Engelstad, Boning Feng, et al. Crowdsourcing-based disaster management using fog computing in internet of things paradigm.

In *2016 IEEE 2nd international conference on collaboration and internet computing (CIC)*, pages 490–494. IEEE, 2016.

[59] Jianhua He, Jian Wei, Kai Chen, Zuoyin Tang, Yi Zhou, and Yan Zhang. Multitier fog computing with large-scale iot data analytics for smart cities. *IEEE Internet of Things Journal*, 5(2):677–686, 2017.

[60] Nanxi Chen, Yang Yang, Tao Zhang, Ming-Tuo Zhou, Xiliang Luo, and John K Zao. Fog as a service technology. *IEEE Communications Magazine*, 56(11):95–101, 2018.

[61] Martin Strohbach, Holger Ziekow, Vangelis Gazis, and Navot Akiva. Towards a big data analytics framework for iot and smart city applications. In *Modeling and processing for next-generation big-data technologies*, pages 257–282. Springer, 2015.

[62] Juma Kasadha, Adam A Alli, Aisha Kasujja Basuuta, and Abdulhamid Mpoza. Social media taxation and its impact on africa's economic growth. *Journal of Public Affairs*, page e2004, 2019.

[63] Yin Zhang, Xiao Ma, Jing Zhang, M Shamim Hossain, Ghulam Muhammad, and Syed Umar Amin. Edge intelligence in the cognitive internet of things: Improving sensitivity and interactivity. *IEEE Network*, 33(3):58–64, 2019.

[64] Mehdi Mohammadi and Ala Al-Fuqaha. Enabling cognitive smart cities using big data and machine learning: Approaches and challenges. *IEEE Communications Magazine*, 56(2):94–101, 2018.

[65] Amir M Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658, 2018.

[66] Rabindra Kumar Barik, Harishchandra Dubey, Kunal Mankodiya, Sapana Ashok Sasane, and Chinmaya Misra. Geofog4health: a fog-based sdi framework for geospatial health big data analysis. *Journal of Ambient Intelligence and Humanized Computing*, 10(2):551–567, 2019.

[67] Hasan Ali Khattak, Hafsa Arshad, Saif ul Islam, Ghufran Ahmed, Sohail Jabbar, Abdullahi Mohamud Sharif, and Shehzad Khalid. Utilization and

load balancing in fog servers for health applications. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):91, 2019.

[68] Ida Syafiza M Isa, Mohamed OI Musa, Taisir EH El-Gorashi, Ahmed Q Lawey, and Jaafar MH Elmirghani. Energy efficiency of fog computing health monitoring applications. In *2018 20th International Conference on Transparent Optical Networks (ICTON)*, pages 1–5. IEEE, 2018.

[69] Pedro H Vilela, Joel JPC Rodrigues, Petar Solic, Kashif Saleem, and Vasco Furtado. Performance evaluation of a fog-assisted iot solution for e-health applications. *Future Generation Computer Systems*, 97:379–386, 2019.

[70] Gunasekaran Raja and Anil Thomas. Safer: Crowdsourcing based disaster monitoring system using software defined fog computing. *Mobile Networks and Applications*, pages 1–11, 2019.

[71] Tiago M Fernández-Caramés, Iván Froiz-Míguez, Oscar Blanco-Novoa, and Paula Fraga-Lamas. Enabling the internet of mobile crowdsourcing health things: A mobile fog computing, blockchain and iot based continuous glucose monitoring system for diabetes mellitus research and care. *Sensors*, 19(15): 3319, 2019.

[72] Fatimah Alghamdi, Saoucene Mahfoudh, and Ahmed Barnawi. A novel fog computing based architecture to improve the performance in content delivery networks. *Wireless Communications and Mobile Computing*, 2019, 2019.

[73] Aparna Kumari, Sudeep Tanwar, Sudhanshu Tyagi, Neeraj Kumar, Reza M Parizi, and Kim-Kwang Raymond Choo. Fog data analytics: a taxonomy and process model. *Journal of Network and Computer Applications*, 128: 90–104, 2019.

[74] Cheol-Ho Hong and Blesson Varghese. Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. *ACM Computing Surveys (CSUR)*, 52(5):1–37, 2019.

[75] Dimitri Belli, Stefano Chessa, Burak Kantarci, and Luca Foschini. Toward fog-based mobile crowdsensing systems: State of the art and opportunities. *IEEE Communications Magazine*, 57(12):78–83, 2019.

[76] Yaohua Sun, Mugen Peng, and Shiwen Mao. Deep reinforcement learning-based mode selection and resource management for green fog radio access networks. *IEEE Internet of Things Journal*, 6(2):1960–1971, 2018.

[77] Tian Wang, Yuzhu Liang, Weijia Jia, Muhammad Arif, Anfeng Liu, and Mande Xie. Coupling resource management based on fog computing in smart city systems. *Journal of Network and Computer Applications*, 135:11–19, 2019.

[78] Mohit Taneja and Alan Davy. Resource aware placement of data analytics platform in fog computing. *Procedia Computer Science*, 97:153–156, 2016.

[79] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. Edge affinity-based management of applications in fog computing environments. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, pages 61–70, 2019.

[80] Abebe Abeshu and Naveen Chilamkurti. Deep learning: The frontier for distributed attack detection in fog-to-things computing. *IEEE Communications Magazine*, 56(2):169–175, 2018.

[81] Tian Wang, Guangxue Zhang, Md Zakirul Alam Bhuiyan, Anfeng Liu, Weijia Jia, and Mande Xie. A novel trust mechanism based on fog computing in sensor–cloud system. *Future Generation Computer Systems*, 2018.

[82] Adam A Alli, Mugigayi Fahadi, and Cherotwo Atebeni. Blockchain and fog computing: Fog-blockchain concept, opportunities and challenges. *Blockchain in Data Analytics*, page 75, 2020.

[83] Esubalew Alemneh, Sidi-Mohammed Senouci, Philippe Brunet, and Tesfa Tegegne. A two-way trust management system for fog computing. *Future Generation Computer Systems*, 106:206–220, 2020.

[84] Wassim Masri, Ismaeel Al Ridhawi, Nour Mostafa, and Pardis Pourghomi. Minimizing delay in iot systems through collaborative fog-to-fog (f2f) communication. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 1005–1010. IEEE, 2017.

[85] Dinh Nguyen, Zhishu Shen, Jiong Jin, and Atsushi Tagami. Icn-fog: An information-centric fog-to-fog architecture for data communications. In

GLOBECOM 2017-2017 IEEE Global Communications Conference, pages 1–6. IEEE, 2017.

[86] Bhawna Suri, Shweta Taneja, Hemankur Bhardwaj, Prateek Gupta, and Udit Ahuja. Peering through the fog: an inter-fog communication approach for computing environment. In *International conference on innovative computing and communications*, pages 73–81. Springer, 2019.

[87] Paola G Vinueza Naranjo, Zahra Pooranian, Mohammad Shojafar, Mauro Conti, and Rajkumar Buyya. Focan: A fog-supported smart city network architecture for management of applications in the internet of everything environments. *Journal of Parallel and Distributed Computing*, 132:274–283, 2019.

[88] Simar Preet Singh, Anand Nayyar, Rajesh Kumar, and Anju Sharma. Fog computing: from architecture to edge computing and big data processing. *The Journal of Supercomputing*, 75(4):2070–2105, 2019.

[89] Elena Hernandez-Nieves, Guillermo Hernández, Ana-Belén Gil-González, Sara Rodríguez-González, and Juan M Corchado. Fog computing architecture for personalized recommendation of banking products. *Expert Systems with Applications*, 140:112900, 2020.

[90] Neda Maleki, Mohammad Loni, Masoud Daneshtalab, Mauro Conti, and Hossein Fotouhi. Sofa: A spark-oriented fog architecture. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 2792–2799. IEEE, 2019.

[91] Pradip Kumar Sharma, Mu-Yen Chen, and Jong Hyuk Park. A software defined fog node based distributed blockchain cloud architecture for iot. *Ieee Access*, 6:115–124, 2017.

[92] Rabindra K Barik, Harishchandra Dubey, Chinmaya Misra, Debanjan Borthakur, Nicholas Constant, Sapana Ashok Sasane, Rakesh K Lenka, Bhabani Shankar Prasad Mishra, Himansu Das, and Kunal Mankodiya. Fog assisted cloud computing in era of big data and internet-of-things: systems, architectures, and applications. In *Cloud computing for optimization: foundations, applications, and challenges*, pages 367–394. Springer, 2018.

[93] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, 18(1):129–140, 2013.

[94] Ting-Yi Lin, Ting-An Lin, Cheng-Hsin Hsu, and Chung-Ta King. Context-aware decision engine for mobile cloud offloading. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2013 IEEE*, pages 111–116. IEEE, 2013.

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

[96] Daniel E O'Leary. Big data', the 'internet of things'and the 'internet of signs. *Intelligent Systems in Accounting, Finance and Management*, 20(1): 53–65, 2013.

[97] Tim Stack. Internet of things (iot) data continues to explode exponentially. who is using that data and how? — cisco blog — cisco systems, 2018. URL https://blogs.cisco.com/datacenter/internet-of-things-iot-data-continues-to-explode-exponentially-who-is-usin

[98] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, pages 37–42. ACM, 2015.

[99] matt Watson. What Is Function-as-a-Service? Serverless Architectures Are Here!, 2017. URL https://stackify.com/function-as-a-service-serverless-architecture/.

[100] Noah Slater. Backend as a Service, 2014. URL https://www.engineyard.com/blog/backend-as-a-service.

[101] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.

[102] Tuyen X Tran, Parul Pandey, Abolfazl Hajisami, and Dario Pompili. Collaborative multi-bitrate video caching and processing in mobile-edge computing

networks. In *Wireless On-demand Network Systems and Services (WONS), 2017 13th Annual Conference on*, pages 165–172. IEEE, 2017.

[103] Shan Zhang, Peter He, Katsuya Suto, Peng Yang, Lian Zhao, and Xuemin Shen. Cooperative edge caching in user-centric clustered mobile networks. *IEEE Transactions on Mobile Computing*, 17(8):1791–1805, 2018.

[104] Engin Zeydan, Ejder Bastug, Mehdi Bennis, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, and Mérouane Debbah. Big data caching for networking: Moving from cloud to edge. *IEEE Communications Magazine*, 54(9):36–42, 2016.

[105] David Applegate, Aaron Archer, Vijay Gopalakrishnan, Seungjoon Lee, and Kadangode K Ramakrishnan. Optimal content placement for a large-scale vod system. In *Proceedings of the 6th International COnference*, page 4. ACM, 2010.

[106] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Communications Surveys and Tutorials*, 19(3):1657–1681, 2017. ISSN 1553877X. doi: 10.1109/COMST.2017.2705720. URL http://ieeexplore.ieee.org/document/7931566/.

[107] Peter Graffagnino, Dave Hyatt, Richard Blanchard, Kevin Calhoun, Gilles Drieu, Maciej Stachowiak, Don Melton, and Darin Adler. Methods and apparatuses for providing a hardware accelerated web engine, July 31 2012. US Patent 8,234,392.

[108] Noriyuki Takahashi, Hiroyuki Tanaka, and Ryutaro Kawamura. Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing. In *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on*, pages 233–234. IEEE, 2015.

[109] Azarias Reda, Edward Cutrell, and Brian Noble. Towards improved web acceleration: leveraging the personal web. In *Proceedings of the 5th ACM workshop on Networked systems for developing regions*, pages 57–62. ACM, 2011.

[110] Radovan Novotnỳ, Radek Kuchta, and Jaroslav Kadlec. Smart city concept, applications and services. *Journal of Telecommunications System & Management*, 3(2):1, 2014.

[111] Oran Chieochan, Anukit Saokaew, and Ekkarat Boonchieng. Iot for smart farm: A case study of the lingzhi mushroom farm at maejo university. In *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–6. IEEE, 2017.

[112] Maxim Chernyshev, Zubair Baig, Oladayo Bello, and Sherali Zeadally. Internet of things (iot): Research, simulators, and testbeds. *IEEE Internet of Things Journal*, 5(3):1637–1647, 2017.

[113] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

[114] A Kertesz, T Pflanzner, and T Gyimothy. A mobile iot device simulator for iot-fog-cloud systems. *Journal of Grid Computing*, pages 1–23, 2018.

[115] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[116] Antonio Coutinho, Fabiola Greve, Cassio Prazeres, and Joao Cardoso. Fogbed: A rapid-prototyping emulation environment for fog computing. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2018.

[117] Yukun Zeng, Mengyuan Chao, and Radu Stoleru. Emuedge: A hybrid emulator for reproducible and realistic edge computing experiments. In *2019 IEEE International Conference on Fog Computing (ICFC)*, pages 153–164. IEEE, 2019.

[118] ChingYao Huang, John Zao, Chih-Yuan Huang, Kha Tho Nguyen, Yang Yang, Ming Tou Zhou, Xiliang Luo, Jen-Shun Yang, Vince Hsu, Yi-Chieh Peter Chang, et al. Trusted worthy fog computing testbed developed in great china region. In *2017 IEEE Fog World Congress (FWC)*, pages 1–6. IEEE, 2017.

[119] Antonio Brogi and Stefano Forti. Qos-aware deployment of iot applications through the fog. *IEEE Internet of Things Journal*, 4(5):1185–1192, 2017.

[120] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Computing Surveys (CSUR)*, 51(6):116, 2019.

[121] Redowan Mahmud, Fernando Luiz Koch, and Rajkumar Buyya. Cloud-fog interoperability in iot-enabled healthcare solutions. In *Proceedings of the 19th international conference on distributed computing and networking*, page 32. ACM, 2018.

[122] Cindy-Pamela Lopez, Marco Santórum, and Jose Aguilar. Autonomous cycles of collaborative processes for integration based on industry 4.0. In *International Conference on Information Technology & Systems*, pages 177–186. Springer, 2019.

[123] Arslan Munir, Prasanna Kansakar, and Samee U Khan. Ifciot: Integrated fog cloud iot: A novel architectural paradigm for the future internet of things. *IEEE Consumer Electronics Magazine*, 6(3):74–82, 2017.

[124] Vittorio Cozzolino, Aaron Yi Ding, Jorg Ott, and Dirk Kutscher. Enabling fine-grained edge offloading for iot. In *Proceedings of the SIGCOMM Posters and Demos*, pages 124–126. ACM, 2017.

[125] B Di Martino, M Rak, M Ficco, A Esposito, SA Maisto, and S Nacchia. Internet of things reference architectures, security and interoperability: A survey. *Internet of Things*, 1:99–112, 2018.

[126] Mohammad Aazam, Imran Khan, Aymen Abdullah Alsaffar, and Eui-Nam Huh. Cloud of things: Integrating internet of things and cloud computing and the issues involved. In *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014*, pages 414–419. IEEE, 2014.

[127] Jiehan Zhou, Teemu Leppanen, Erkki Harjula, Mika Ylianttila, Timo Ojala, Chen Yu, Hai Jin, and Laurence Tianruo Yang. Cloudthings: A common architecture for integrating the internet of things with cloud computing. In *Proceedings of the 2013 IEEE 17th international conference on computer supported cooperative work in design (CSCWD)*, pages 651–657. IEEE, 2013.

[128] Shweta Kaushik and Charu Gandhi. Fog vs. cloud computing architecture. In *Advancing Consumer-Centric Fog Computing Architectures*, pages 87–110. IGI Global, 2019.

[129] Dai H Tran, Nguyen H Tran, Chuan Pham, SM Ahsan Kazmi, Eui-Nam Huh, and Choong Seon Hong. Oaas: offload as a service in fog networks. *Computing*, 99(11):1081–1104, 2017.

[130] Said El Kafhali and Khaled Salah. Efficient and dynamic scaling of fog nodes for iot devices. *The Journal of Supercomputing*, 73(12):5261–5284, 2017.

[131] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698, 2018.

[132] Yunsik Son, Junho Jeong, and YangSun Lee. An adaptive offloading method for an iot-cloud converged virtual machine system using a hybrid deep neural network. *Sustainability*, 10(11):3955, 2018.

[133] Liang Xiao, Xiaoyue Wan, Xiaozhen Lu, Yanyong Zhang, and Di Wu. Iot security techniques based on machine learning: How do iot devices use ai to enhance security? *IEEE Signal Processing Magazine*, 35(5):41–49, 2018.

[134] Duc Van Le and Chen-Khong Tham. A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 760–765. IEEE, 2018.

[135] Shuai Yu, Xin Wang, and Rami Langar. Computation offloading for mobile edge computing: a deep learning approach. In *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2017 IEEE 28th Annual International Symposium on*, pages 1–6. IEEE, 2017.

[136] Zhikai Kuang, Songtao Guo, Jiadi Liu, and Yuanyuan Yang. A quick-response framework for multi-user computation offloading in mobile cloud computing. *Future Generation Computer Systems*, 81:166–176, 2018.

[137] F Zhang, J Ge, Z Li, C Li, Z Huang, L Kong, and B Luo. Task offloading for scientific workflow application in mobile cloud. In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017)*, pages 136–148, 2017.

[138] Dmitry Kozyrev, Aleksandr Ometov, Dmitri Moltchanov, Vladimir Rykov, Dmitry Efrosinin, Tatiana Milovanova, Sergey Andreev, and Yevgeni Koucheryavy. Mobility-centric analysis of communication offloading for heterogeneous internet of things devices. *Wireless Communications and Mobile Computing*, 2018, 2018.

[139] Huber Flores, Xiang Su, Vassilis Kostakos, Aaron Yi Ding, Petteri Nurmi, Sasu Tarkoma, Pan Hui, and Yong Li. Large-scale offloading in the internet of things. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 479–484. IEEE, 2017.

[140] Xiao Ma, Chuang Lin, Han Zhang, and Jianwei Liu. Energy-aware computation offloading of iot sensors in cloudlet-based mobile edge computing. *Sensors*, 18(6):1945, 2018.

[141] Sungwook Kim. Nested game-based computation offloading scheme for mobile cloud iot systems. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):229, 2015.

[142] Jie Zhang, Zhili Zhou, Shu Li, Leilei Gan, Xuyun Zhang, Lianyong Qi, Xiaolong Xu, and Wanchun Dou. Hybrid computation offloading for smart home automation in mobile cloud computing. *Personal and Ubiquitous Computing*, 22(1):121–134, 2018.

[143] AN Gnana Jeevan and MA Maluk Mohamed. Dyto: Dynamic task offloading strategy for mobile cloud computing using surrogate object model. *International Journal of Parallel Programming*, pages 1–17, 2018.

[144] Gianni D'Angelo and Salvatore Rampone. Cognitive distributed application area networks. In *Security and Resilience in Intelligent Data-Centric Systems and Communication Networks*, pages 193–214. Elsevier, 2018.

[145] Jean Caminha, Angelo Perkusich, and Mirko Perkusich. A smart trust management method to detect on-off attacks in the internet of things. *Security and Communication Networks*, 2018, 2018.

[146] Yinghui Zhang, Jiangfan Zhao, Dong Zheng, Kaixin Deng, Fangyuan Ren, Xiaokun Zheng, and Jiangang Shu. Privacy-preserving data aggregation against false data injection attacks in fog computing. *Sensors*, 18(8):2659, 2018.

[147] Manuel Suárez-Albela, Tiago M Fernández-Caramés, Paula Fraga-Lamas, and Luis Castedo. A practical evaluation of a high-security energy-efficient gateway for iot fog computing applications. *Sensors*, 17(9):1978, 2017.

[148] Luis Belem Pacheco, Eduardo Pelinson Alchieri, and Priscila Mendez Barreto. Device-based security to improve user privacy in the internet of things. *Sensors*, 18(8):2664, 2018.

[149] Roohie Naaz Mir et al. Resource management in pervasive internet of things: A survey. *Journal of King Saud University-Computer and Information Sciences*, 2018.

[150] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, pages 44–51. Ieee, 2009.

[151] Yuhui Shi and Russell C Eberhart. Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1945–1950. IEEE, 1999.

[152] Yuhui Shi et al. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 81–86. IEEE, 2001.

[153] Abdulwahab A Alnaqi, Hossein Moayedi, Amin Shahsavar, and Truong Khang Nguyen. Prediction of energetic performance of a building integrated photovoltaic/thermal system thorough artificial neural network and hybrid particle swarm optimization models. *Energy Conversion and Management*, 183:137–148, 2019.

[154] Xudong Ye, Bing Chen, Liang Jing, Baiyu Zhang, and Yong Liu. Multi-agent hybrid particle swarm optimization (mahpso) for wastewater treatment network planning. *Journal of environmental management*, 234:525–536, 2019.

[155] Hao Pu, Taoran Song, Paul Schonfeld, Wei Li, Hong Zhang, Jianping Hu, Xianbao Peng, and Jie Wang. Mountain railway alignment optimization using stepwise & hybrid particle swarm optimization incorporating genetic operators. *Applied Soft Computing*, 2019.

[156] K Thangaramya, K Kulothungan, R Logambigai, M Selvi, S Ganapathy, and A Kannan. Energy aware cluster and neuro-fuzzy based routing algorithm for wireless sensor networks in iot. *Computer Networks*, 2019.

[157] George S Atsalakis and Kimon P Valavanis. Surveying stock market forecasting techniques–part ii: Soft computing methods. *Expert Systems with Applications*, 36(3):5932–5941, 2009.

[158] Navneet Walia, Harsukhpreet Singh, and Anurag Sharma. Anfis: Adaptive neuro-fuzzy inference system-a survey. *International Journal of Computer Applications*, 123(13), 2015.

[159] Jose Vieira, F Morgado Dias, and Alexandre Mota. Neuro-fuzzy systems: a survey. In *5th WSEAS NNA international conference on neural networks and applications, Udine, Italia*, pages 87–92, 2004.

[160] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[161] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Thirtieth AAAI Conference on Artificial Intelligence*, 13(7):2094–2100, 2016.

[162] Sascha Lange, Martin Riedmiller, and Arne Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

[163] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[164] Robert H Crites and Andrew G Barto. Improving elevator performance using reinforcement learning. In *Advances in neural information processing systems*, pages 1017–1023, 1996.

[165] Petar Kormushev, Sylvain Calinon, and Darwin Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3): 122–148, 2013.

[166] Kehua Su, Jie Li, and Hongbo Fu. Smart city and the applications. In *2011 international conference on electronics, communications and control (ICECC)*, pages 1028–1031. IEEE, 2011.

[167] Jiong Jin, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. An information framework for creating a smart city through internet of things. *IEEE Internet of Things journal*, 1(2):112–121, 2014.

[168] Tongxiang Wang, Xianglin Wei, Chaogang Tang, and Jianhua Fan. Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints. *Peer-to-Peer Networking and Applications*, 11(4):793–807, 2018.

[169] Yucen Nan, Wei Li, Wei Bao, Flavia C Delicato, Paulo F Pires, Yong Dou, and Albert Y Zomaya. Adaptive energy-aware computation offloading for cloud of things systems. *IEEE Access*, 5:23947–23957, 2017.

[170] Chaogang Tang, Shixiong Xia, Qing Li, Wei Chen, and Weidong Fang. Resource pooling in vehicular fog computing. *Journal of Cloud Computing*, 10 (1):1–14, 2021.

[171] Adam A Alli and Muhammad Mahbub Alam. The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications. *Internet of Things*, page 100177, 2020.

[172] Pedro García López, Marc Sánchez-Artigas, Gerard París, Daniel Barcelona Pons, Álvaro Ruiz Ollobarren, and David Arroyo Pinto. Comparison of faas orchestration systems. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 148–153. IEEE, 2018.

[173] Ayed Salman, Imtiaz Ahmad, and Sabah Al-Madani. Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems*, 26(8):363–371, 2002.

[174] Kenneth A De Jong and William M Spears. Using genetic algorithms to solve np-complete problems. In *ICGA*, pages 124–132, 1989.

[175] Thinh Quang Dinh, Jianhua Tang, Quang Duy La, and Tony QS Quek. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications*, 65(8):3571–3584, 2017.

[176] Zhenyu Zhou, Pengju Liu, Junhao Feng, Yan Zhang, Shahid Mumtaz, and Jonathan Rodriguez. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. *IEEE Transactions on Vehicular Technology*, 68(4):3113–3125, 2019.

[177] Yi-Hsuan Kao and Bhaskar Krishnamachari. Optimizing mobile computational offloading with delay constraints. In *2014 IEEE Global Communications Conference*, pages 2289–2294. IEEE, 2014.

[178] Adam A Alli, Kalinaki Kassim, Nambobi Mutwalibi, Habiba Hamid, and Lwembawo Ibrahim. Secure fog-cloud of things: Architectures, opportunities and challenges. In *Secure Edge Computing*, pages 3–20. CRC Press, 2021.

[179] Sanjay P Ahuja and Nathan Wheeler. Architecture of fog-enabled and cloud-enhanced internet of things applications. *International Journal of Cloud Applications and Computing (IJCAC)*, 10(1):1–10, 2020.

[180] Carlo Puliafito, Diogo M Gonçalves, Márcio M Lopes, Leonardo L Martins, Edmundo Madeira, Enzo Mingozzi, Omer Rana, and Luiz F Bittencourt. Mobfogsim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory*, 101:102062, 2020.

[181] Haoxin Wang, Tingting Liu, BaekGyu Kim, Chung-Wei Lin, Shinichi Shiraishi, Jiang Xie, and Zhu Han. Architectural design alternatives based on cloud/edge/fog computing for connected vehicles. *IEEE Communications Surveys & Tutorials*, 22(4):2349–2377, 2020.

[182] Zainab Javed and Waqas Mahmood. A survey based study on fog computing awareness. *International Journal of Information Technology and Computer Science (IJITCS)*, 13(2):49–62, 2021.

[183] Gonçalo Carvalho, Bruno Cabral, Vasco Pereira, and Jorge Bernardino. Computation offloading in edge computing environments using artificial intelligence techniques. *Engineering Applications of Artificial Intelligence*, 95: 103840, 2020.

[184] Fei Gu, Jianwei Niu, Zhiping Qi, and Mohammed Atiquzzaman. Partitioning and offloading in smart mobile devices for mobile cloud computing: State of the art and future directions. *Journal of Network and Computer Applications*, 119:83–96, 2018.

[185] Zhiyuan Li, Cheng Wang, and Rong Xu. Computation offloading to save energy on handheld devices: a partition scheme. In *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 238–246, 2001.

[186] Mingjin Gao, Rujing Shen, Long Shi, Wen Qi, Jun Li, and Yonghui Li. Task partitioning and offloading in dnn-task enabled mobile edge computing networks. *IEEE Transactions on Mobile Computing*, 2021.

[187] Jianhui Liu and Qi Zhang. Adaptive task partitioning at local device or remote edge server for offloading in mec. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2020.

[188] Sachchidanand Singh. Optimize cloud computations using edge computing. In *2017 International Conference on Big Data, IoT and Data Science (BID)*, pages 49–53. IEEE, 2017.

[189] Giorgos Mitsis, Pavlos Athanasios Apostolopoulos, Eirini Eleni Tsiropoulou, and Symeon Papavassiliou. Intelligent dynamic data offloading in a competitive mobile edge computing market. *Future Internet*, 11(5):118, 2019.

[190] Xin-She Yang. Nature-inspired optimization algorithms: challenges and open problems. *Journal of Computational Science*, page 101104, 2020.

[191] Kinjal Chaudhari and Ankit Thakkar. Travelling salesman problem: An empirical comparison between aco, pso, abc, fa and ga. In *Emerging Research in Computing, Information, Communication and Applications*, pages 397–405. Springer, 2019.

[192] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[193] Praveen Kumar Tripathi, Sanghamitra Bandyopadhyay, and Sankar Kumar Pal. Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. *Information sciences*, 177(22):5033–5049, 2007.

[194] Wei Bin, Peng Qinke, Zhao Jing, and Chen Xiao. A binary particle swarm optimization algorithm inspired by multi-level organizational learning behavior. *European Journal of Operational Research*, 219(2):224–233, 2012.

[195] Amira S Ashour and Yanhui Guo. Optimization-based neutrosophic set in computer-aided diagnosis. In *Optimization Theory Based on Neutrosophic and Plithogenic Sets*, pages 405–421. Elsevier, 2020.

[196] Matheus Rosendo and Aurora Pozo. Applying a discrete particle swarm optimization algorithm to combinatorial problems. In *2010 Eleventh Brazilian Symposium on Neural Networks*, pages 235–240. IEEE, 2010.

[197] Indresh Kumar Gupta, Samiya Shakil, and Sadiya Shakil. A hybrid ga-pso algorithm to solve traveling salesman problem. In *Computational Intelligence: Theories, Applications and Future Directions-Volume I*, pages 453–462. Springer, 2019.

[198] Ammar W Mohemmed, Nirod Chandra Sahoo, and Tan Kim Geok. Solving shortest path problem using particle swarm optimization. *Applied Soft Computing*, 8(4):1643–1653, 2008.

[199] Han Huang and Zhifeng Hao. Particle swarm optimization algorithm for transportation problems. *Particle swarm optimization, ed. Aleksandar Lazinica, InTech*, pages 275–290, 2009.

[200] Jean L. Pierobom, Myriam Regattieri Delgado, and Celso A. A. Kaestner. Particle swarm optimization applied to task assignment problem. *ChemBioChem*, pages 1–8, 2016.

[201] Hina Rafique, Munam Ali Shah, Saif Ul Islam, Tahir Maqsood, Suleman Khan, and Carsten Maple. A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing. *IEEE Access*, 7:115760–115773, 2019.

[202] Arash Heidari, Mohammad Ali Jabraeil Jamali, Nima Jafari Navimipour, and Shahin Akbarpour. Deep q-learning technique for offloading offline/online computation in blockchain-enabled green iot-edge scenarios. *Applied Sciences*, 12(16):8232, 2022.

[203] Siyuan Sun, Junhua Zhou, Jiuxing Wen, Yifei Wei, and Xiaojun Wang. A dqn-based cache strategy for mobile edge networks. *Computers, Materials & Continua*, 71(2):3277–3291, 2022.

[204] Umber Saleem, Yu Liu, Sobia Jangsher, and Yong Li. Performance guaranteed partial offloading for mobile edge computing. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[205] Lan Li, Xiaoyong Zhang, Kaiyang Liu, Fu Jiang, and Jun Peng. An energy-aware task offloading mechanism in multiuser mobile-edge cloud computing. *Mobile Information Systems*, 2018, 2018.

[206] Pengtao Zhao, Hui Tian, Cheng Qin, and Gaofeng Nie. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. *IEEE Access*, 5:11255–11268, 2017.

[207] Sudip Misra and Niloy Saha. Detour: Dynamic task offloading in software-defined fog for iot applications. *IEEE Journal on Selected Areas in Communications*, 37(5):1159–1166, 2019.

[208] Bowen Zhou, Amir Vahid Dastjerdi, Rodrigo N Calheiros, and Rajkumar Buyya. An online algorithm for task offloading in heterogeneous mobile clouds. *ACM Transactions on Internet Technology (TOIT)*, 18(2):1–25, 2018.

[209] Amit Kishor and Chinmay Chakarbarty. Task offloading in fog computing for using smart ant colony optimization. *Wireless Personal Communications*, pages 1–22, 2021.

[210] Maryam Keshavarznejad, Mohammad Hossein Rezvani, and Sepideh Adabi. Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. *Cluster Computing*, pages 1–29, 2021.

[211] Jin Wang, Jia Hu, Geyong Min, Albert Y Zomaya, and Nektarios Georgalas. Fast adaptive task offloading in edge computing based on meta reinforcement learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(1): 242–253, 2020.

[212] Liang Huang, Xu Feng, Cheng Zhang, Liping Qian, and Yuan Wu. Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. *Digital Communications and Networks*, 5(1):10–17, 2019.

[213] D Shobha Rani and M Pounambal. Deep learning based dynamic task offloading in mobile cloudlet environments. *Evolutionary Intelligence*, pages 1–9, 2019.

[214] Jianhua Liu, Xin Wang, Shigen Shen, Guangxue Yue, Shui Yu, and Minglu Li. A bayesian q-learning game for dependable task offloading against ddos attacks in sensor edge cloud. *IEEE Internet of Things Journal*, 8(9):7546–7561, 2020.

[215] Jin Wang, Wenbing Wu, Zhuofan Liao, R Simon Sherratt, Gwang-Jun Kim, Osama Alfarraj, Ahmad Alzubi, and Amr Tolba. A probability preferred priori offloading mechanism in mobile edge computing. *IEEE Access*, 8: 39758–39767, 2020.

[216] Ni Zhang, Songtao Guo, Yifan Dong, and Defang Liu. Joint task offloading and data caching in mobile edge computing networks. *Computer Networks*, 182:107446, 2020.

[217] Xueying Zhang, Ruiting Zhou, Zhi Zhou, John CSCS Lui, and Zongpeng Li. An online learning-based task offloading framework for 5g small cell networks. In *49th International Conference on Parallel Processing-ICPP*, pages 1–11, 2020.

[218] Zhen Wang, Jihui Zhang, and Shengxiang Yang. An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals. *Swarm and Evolutionary Computation*, 51:100594, 2019.

[219] Mohd Azri Abdul Aziz, Mohd Nasir Taib, and Naimah Mohd Hussin. An improved event selection technique in a modified pso algorithm to solve class scheduling problems. In *2009 IEEE Symposium on Industrial Electronics & Applications*, volume 1, pages 203–208. IEEE, 2009.

[220] N Delgarm, B Sajadi, F Kowsary, and S Delgarm. Multi-objective optimization of the building energy performance: A simulation-based approach by means of particle swarm optimization (pso). *Applied energy*, 170:293–303, 2016.

[221] Li-biao Zhang, Chun-guang Zhou, Ming Ma, and Xiao-hua Liu. Solutions of multi-objective optimization problems based on particle swarm optimization. *Journal of computer research and development*, 7(41):7, 2004.

[222] Hemlata S Urade and Rahila Patel. Dynamic particle swarm optimization to solve multi-objective optimization problem. *Procedia Technology*, 6:283–290, 2012.

[223] Spiridoula V Margariti, Vassilios V Dimakopoulos, and Georgios Tsoumanis. Modeling and simulation tools for fog computing—a comprehensive survey from a cost perspective. *Future Internet*, 12(5):89, 2020.