SECURE DELAY AWARE SCHEDULING AND LOAD BALANCING ALONG WITH REDUCED ENERGY FOR DEADLINE SENSITIVE APPLICATIONS IN FOG COMPUTING ENVIRONMENT

Thesis

Submitted in fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

By

SHIVI SHARMA

In

COMPUTER SCIENCE & ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
AND INFORMATION TECHNOLOGY
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT SOLAN, HIMACHAL PRADESH, INDIA

AUGUST 2020

…Dedicated To

My family …

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the Ph.D. thesis entitled **“Secure Delay Aware Scheduling And Load Balancing Along With Reduced Energy For Deadline Sensitive Applications In Fog Computing Environment”** submitted at the **Jaypee University of Information Technology, Waknaghat, Himachal Pradesh, India**, is an authentic record of my work carried out under the supervision of **Dr. Hemraj Saini.** I have not submitted this work elsewhere for any degree or diploma. I am fully responsible for the contents of my Ph.D. thesis.

Shivi

(Signature of the Scholar)
Shivi Sharma
Enrollment No.: 166204
Department of Computer Science& Engineering
Jaypee University of Information Technology, Waknaghat, Solan
(HP), India
Date: 31/08/2020

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the Ph.D. thesis entitled "Secure Delay Aware Scheduling And Load Balancing Along With Reduced Energy For Deadline Sensitive Applications In Fog Computing Environment" submitted by Shivi Sharma at Jaypee University of Information Technology, Waknaghat, Himachal Pradesh, India, is a bonafide record of her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

[illegible]

(Signature of Supervisor)
Dr. Hemraj Saini
Associate Professor
Department of Computer Science &Engineering
Jaypee University of Information Technology,
Waknaghat, Solan (HP), India
Date: 31/08/2020

ACKNOWLEDGEMENT

Indeed the words at my command are not adequate, either in form of spirit or express the depth of my humility and kindness before Almighty one without whose endless benevolence and blessings this tedious task could not have been accomplished. I would first like to thank Dr. Hemraj Saini, Associate Professor, Department of Computer Science Engineering and Information Technology my supervisor for giving me this opportunity to conduct my Ph.D. research work under his supervision at the Department of Computer Science and Information Technology. He has constantly encouraged and supported me throughout my stay at JUIT. His vision towards science and technology has motivated me to do high-quality work. It is because of his active support, I could pursue research of my interest and able to conclude my work in time. I have many useful pieces of advice about research, career, and life from him, which will be more valuable in my future professional career. Because of his availability, I never felt a lack of guidance. It has been a wonderful experience to work under his supervision. In them, I found that the best are not only efficient, effective, and results-driven, but at "core they are persons with the best qualities as human beings". Humanity, scientific potentials, kindness, coolness, simplicity, the novelty of ideas are the arms of their great personality. I am grateful and indebted to Dr. Hemraj Saini for his invaluable painstaking efforts taken towards my study and for showing me the real path of sincerity & dedication.

Besides my advisor, I express my sincere gratitude to Vice-Chancellor Prof. (Dr.) Vinod Kumar, Director and Academic Head Prof. (Dr.) Samir Dev Gupta for kind support and successful completion of this research work.

I would like to acknowledge Professors Satya Prakash Ghrera for helping me with various academic issues. I would like to mention special thanks to Professor Vivek Sehgal for supporting me during my research. I owe my thanks to Dr. Amit, Dr. Ravinder Bhatt, Dr. Tiratha Raj, Dr.Rajinder Sandhu, Dr.Geetanjali, and other faculty members for providing me feedback, moral support, valuable suggestions, and necessary facilities during my research work. I also like to thank Mr. Amit Kumar Shrivastava and the Lab Technicians of all computer labs in JUIT, for their help and cooperation in helping me with various office-related issues. I am also obliged to the JUIT library staff that allowed me to use their facilities during my work. I am also grateful to all my friends, colleagues, seniors, and juniors for their love care, and support.

Finally, I would like to acknowledge my family for supporting me throughout my life. It will never be possible for me to pay for the price of sacrifices, which my parents have made for my success. I am especially grateful to my mother Mrs. Shanta Sharma and my father Dr.D.D Sharma for teaching me the most valuable lesson of life that my job in life is to remain happy, to learn, and to understand myself. This journey would not have been possible without their inspiration, love, and prayers. I would like to thank my younger brother Ashish Sharma for his love and care. He always remained by my side during my happy and hard times to motivate me. Last but not the least; I would like to thank my husband Bharat Sharma who has been a constant source of encouragement during this journey. Thank you for always been there whenever I need you to just listen and help me out in every possible way. I am truly thankful for having you in my life.
Finally, I thank anyone else whose contribution I could have forgotten.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AI	Artificial Intelligence
ABC	Artificial Bee Colony
ANN	Artificial neural network
ASTM	Abstract Syntax Tree Metamodel
BLA	Bees Life Algorithm
BP	Back Propagation
CC	Cloud Computing
CDC	Cloud Data Center
CH	Cluster Head
CMaS	Cost makespan aware scheduling
CS	Cloud Server
DASLB	Delay Aware Scheduling And Load Balancing
DVFS	Dynamic Voltage and Frequency Scaling
EC	Energy Consumption
EDF	Earliest Deadline First
FASWO	Fast Artificial Fish Swarm Optimization
FATASD	Fog Assisted Task Allocation and Secure De-duplication
FC	Fog Computing
FCFS	First Come First Serve
FLA	Fuzzy logic algorithm
FN	Fog Node

HM	Hybrid Multiplier
HP	High Priority
IoT	Internet of Things
IIoT	Industrial Internet of Things
KNN	K Nearest Neighbor
LP	Low Priority
LB	Load Balancing
PSO	Particle Swarm Optimization
PS	Proxy Server
RDD	Resilient Distributed Datasets
SA	Scheduling Algorithm
SLVMR	Schedule Length of VM ratio
VM	Virtual Machines

ABSTRACT

Fog Computing (FC) is the extension of cloud computing (CC) to meet the need of modern age technologies like the Internet of Things (IoT), Artificial Intelligence (AI), 5G, and other such aspects. Fog Computing extends its services to cloud computing by storing the data on Fog Nodes (FN) locally instead of increasing the burden on the cloud. The chief feature of Fog Computing is to present the user with the ideal elucidation, which is efficient and fast. The major critical issues that Fog Computing experience are that of load balancing (LB) and energy consumption (EC). The present thesis has designed a framework for scheduling deadline sensitive applications to provide an efficient solution for Load balancing using optimization techniques and to minimizing energy consumption in Fog Computing. Further, it has focused on designing the Four-Tier Architecture in fog computing environment for Delay Aware Scheduling and Load Balancing (DASLB) and developed a model to provide Fog-Assisted Task-Allocation and Secure De-duplication (FATASD) using Two-Fitness based One-to-One Matching Algorithm (2FBO2) and Multi-Objective Whale Optimization Algorithm (MoWo) in Cluster-based IIoT (Industrial Internet of Things). Fog-assisted has been a topic of interest in the research community. With the upsurge in the utilization of IoT devices, the transmission of duplicate data increased. To avoid this, the present research has used Cluster-based IIoT for the implementation of de-duplication and task allocation on fog layers. The first objective of the present study is to design a framework for scheduling deadline sensitive applications. This has been achieved in the study in two stages. In the first stage, an effective solution has been designed for load balancing using optimization techniques. The present study has thus focused on the drop in the runtime of schedule length and minimization of energy consumption rate. To achieve these results, the present research has proposed an algorithm based on the Artificial Bee Colony (ABC) optimization to balance the load. In the second phase, a system has been designed to decrease the usage of energy and Service-Level Agreement (SLA) violations in Fog Computing. During the execution of IoT applications in fog environments, load balance is still an issue. Thus, to tackle the issue, the present research is dedicated to finding an efficient solution for Load balancing using an optimization technique. The existing researches are focused on decreasing the consumption of energy without taking into consideration the violation of the SLA. Owing to the inconsistency in scheduling, the consumption of energy became more because of a lack of host classifier. The present research has focused on scheduling of job & resources which are energy aware based on the idea of artificial intelligence. Many jobs are considered for calculating the helpfulness of the proposed system, 70.2MJ and, 66.5 MJ respectively, are average values measured in the number of jobs without and with the optimization algorithm 0.776 and 0.742 respectively, are the average values witnessed for violation of SLA with and without optimization algorithms. Therefore, in Fog Computing, a reduction of 3.9 % in SLA violation was observed when an optimization algorithm is applied. Hence, it has been established that the performance of the proposed work has led to a reduction in energy consumption.

The second objective is addressing the issues of Load balancing and Delay Aware Scheduling using a newly designed Four-Tier Architecture in a fog environment. Load balancing and scheduling are the most important aspects of Fog Computing, which prominently influence the performance of an arrangement. The methods preceding are suffering from the failures of fog-node, scarcity of resources, Distributed or Centralized Sort-of Environment, etc. To achieve better performance of the problems of prior works, the present research has suggested a Four-Tier Structure. By the placement of the Fog Computing Model between IoT devices and the Cloud Model, idleness is reduced by optimum scheduling and Load balancing. To manage the enormous expanse of data sensing from various IoT devices, the proposed fog computing system in this research comprises four particular levels that isTier1, Tier2, Tier3, and Tier4. Tier 1 is the base level consisting of IoT modules, Router-based on Dual Fuzzy-Logic Algorithm, workloads (applications) are divided into two levels, High Priority (HP) and Low Priority (LP), respectively. Fuzzifier considers four metrics: Arrival Time, Minimum Execution Time, Maximum Completion Time, and Task Size. Assignment with High Priority is transferred to the 3rd Stage (Fog-Tier). A novel fog called Artificial Fractals was invoked in the third tier. Fog Node is clustered utilizing the clustering algorithm (K-mean++). The request is transferred to the Cloud-Tier if an IoT system does not receive the essential asset. Proposed work has been approved by the implementation of Video Surveillance & Object Tracking (VSOT) application in *iFogSim* Platform and output is assessed in-terms of Scheduling Time (SC), Response Time (RT), Delay, Energy Consumption (EC), and load-balancing (LB) rate. We test the simulation results for different output parameters. Then verified that the proposed Four Tier Fog Architecture is superior to previous works.

The third and final objective of the study is to provide Fog Assisted Task Allocation and Secure Deduplication (FATASD). As an outcome of increasing IoT devices, the transmission of duplicate data over the Internet has increased. This increase in transmission due to duplicate data has then increased the pressure on the data center resources. The scenario of transmission of duplicate data causes the Cloud Server (CS) to delay the services to be provided to users on time. To deal with this scenario, distribution and secured deduplication have been designed, that is, the elimination of redundant information in computer data for fog enabled IIoT. By using novel security and optimization algorithms for IIoT applications, we had designed architecture that will allow better task allocation and stable fog de-duplication. In this objective, the researcher offered job distribution and protected data de-duplication in cluster-based Industrial IoT. This chapter is focused on the third objective of the present research that is to provide FATASD by using 2FBO2 and MoWoA (Multi-objective Whale Optimization Algorithm) in Cluster-based Industrial IoT (IIoT- Industrial Internet of Things). For this, a system called Fog-assisted Internet-of-Things (FaCIIoT) has been designed which entails five things such as Fog Nodes (gateways), IoT devices (Sensors), trusted authority, Cloud Server, and proxy server (PS). The index structure is founded on the Merkle hash tree. To attain data privacy, the researcher suggested a safety algorithm for data encryption. The simulation of the suggested model is executed using iFogSim. It is a Java-based, open-source network simulator. *IFogSim's* works to simulate the surrounding that consists of a vast amount of IoT devices and Fog Node Simulation is shown to execute the suggested as well as a prior job comparison due to user satisfaction, average latency, to name a few aspects. The suggested system has demonstrated that it is performing better than prior jobs.

In future studies, the optimal method alongside the idea of Artificial Intelligence to cope with the overload issue in Fog Nodes with the least Energy Consumption rate has been suggested. We've intended to integrate task offloading to reduce any fog-node failures and work on other real-time applications such as healthcare services. Further research can be completed on another technique for data replication techniques in Fog network data management, which may additionally minimize overall reliance and delay.

Keywords: Fog computing, Internet of Things, Energy Consumption, Scheduling, Load Balancing

CHAPTER 1
INTRODUCTION

1.1 Introduction

Cisco [1] presented the term "Fog Computing" to designate the extension of cloud computing (CC) to meet the requirements of new-age technologies like 5G, Internet of Things, Artificial Intelligence, and other such aspects. The services provided to IoT users by Fog Computing (FC) include data processing and storage services. The Fog Computing extends its services to Cloud computing by storing the data on Fog Node locally instead of increasing the burden on the cloud by sending the data to be stored in the cloud. In this manner, Fog Computing improves the performance of the cloud and increases its efficiency. Fog Computing decreases the rate at which data to be processed, transferred to the cloud thereby facilitating the cloud in the storage and analysis of data as well. This service of Fog Computing results in decreasing network traffic and latency [2]. Figure 1.1 below presents a pictorial representation of Fog Computing.

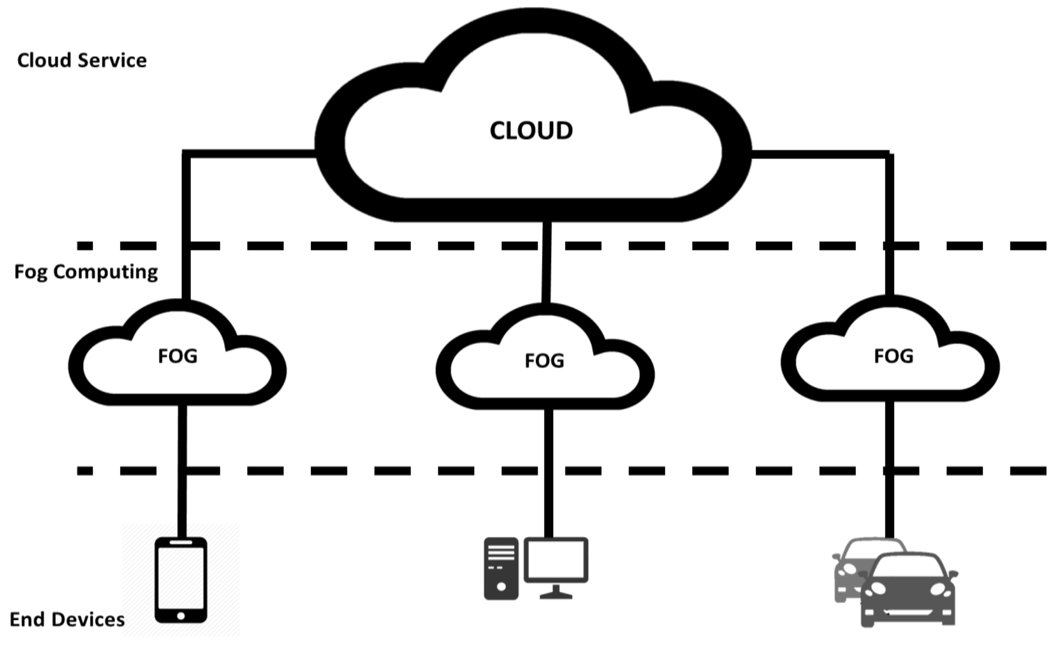


Figure 1.1: Basic design of Fog Computing

Fog computing positions fog-node all over the network. Devices from the controller switch act as Fog-node that are then being deployed in target areas like within a vehicle or office floor. The data generated by IoT devices are then analyzed using one of these nodes. Here in the Fog-node analyzes the data without the data being sent back to the cloud, which reduces the pressure on the cloud, thereby reducing latency and overcrowding of jobs for the cloud. Cloud Computing and Fog Computing differ on the ground that Cloud Computing provides centralized access to resources. On the other hand, Fog Computing provides decentralized access, which is local. Table 1.1 below presents the major differences as follows:

Table 1.1 Cloud Computing v/s Fog Computing

Characteristic	**CC**	**FC**
Location Awareness	No	Yes
Geo-distribution	Centralized	Supported
Client-Server Distance	Multiple hops	One hop
Real-time Interactions	Support Centralized	Support
Mobility Support	Low	Support
Latency	High	Low
Service Site	Within the internet	Local Network Edge
Delay	High	Low

Though Fog Computing provides privileges to Cloud computing some aspects need to be addressed to enhance the efficiency and effectiveness of Fog Computing as the number of IoT devices is increasing. It comprises the innovative concept of supplying IoT devices with storage space and computing capability. It consists of mentioned segments in sequence from bottom to top: IoT Devices (Lower Tier), Fog- nodes (Middle Tier), and Cloud (Top Tier). IoT Layer aims to link any physical object such as wearable devices, cameras, vehicle sensors, and home appliances, which creates massive data volumes. The cloud gives foundation level assistance, which upholds information investigation and capacity frameworks. The majority of IoT applications need less latency. In terms of latency, the cloud impacts their performance to overcome this issue fog computing has been implemented, where services that are far from the cloud can offer and thereby diminish the overcrowding and latency of the network. Figure 1.2 shows the three-tier framework.

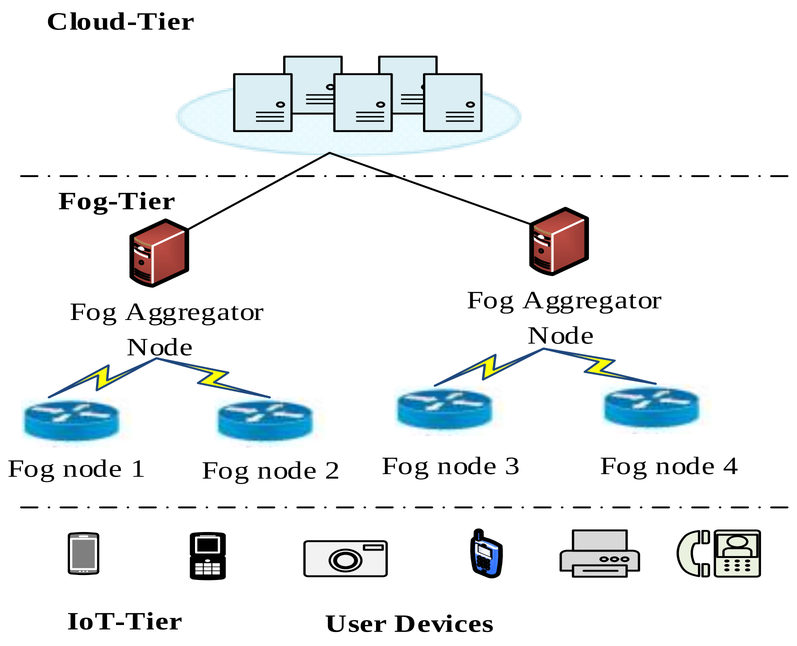


Figure 1.2: Three-Tier Framework of Fog Computing

IoT applications require enormous handling capacity, information storage to permit constant dynamic and quick broadband information streaming frames. For IoT based applications, IoT and distributed computing integration are needed [5]. Cloud storage is a capable approach focused on the desired Quality-of service (QoS) and the Pay-As much You-Use Pricing Model for on-demand access. It provides storage and high computational capabilities [4]. While cloud storage can offer many services, such as storing large volumes of data, IoT applications face challenges such as lack of sufficient network capacity, latency, mobile support, and location awareness. To overcome these issues, fog computing was introduced to leverage the capabilities of cloud solution [8], as it provides large-scale connectivity between IoT devices and cloud computing environments, which ensures better facilities without the constraints. Fog computing is a virtualized platform with a resource pool that offers creative business models for processing, storing, networking of end-users, and traditional cloud data centers. This methodology is appropriate for low idleness, video web-based, gaming, augmented reality, and so on brilliant applications. Still, there are some challenges in Fog computing that have been displayed in Figure 1.3 [3].

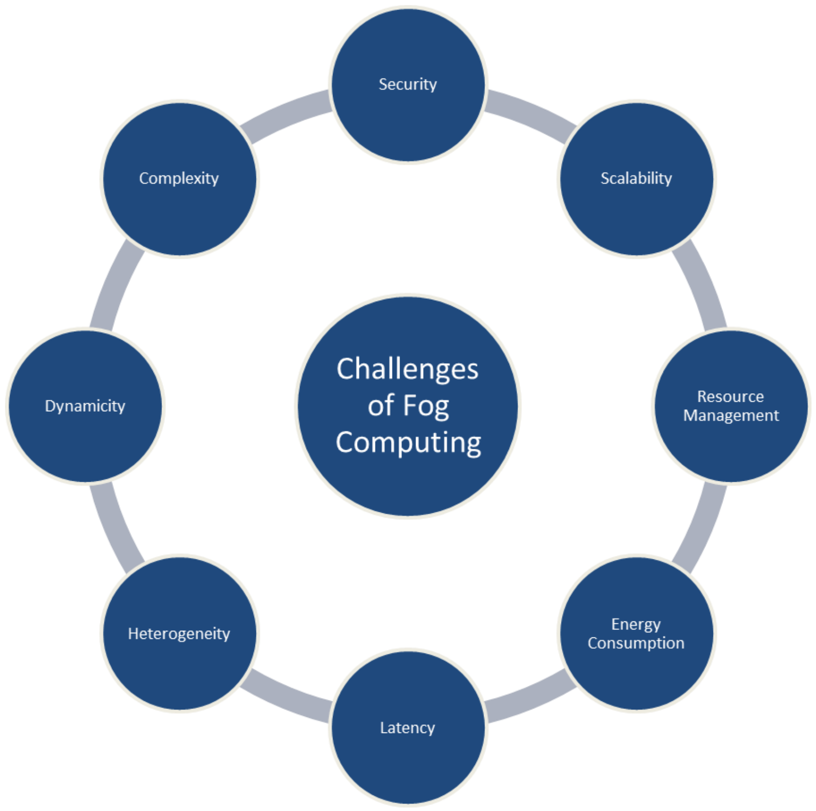


Figure 1.3: Challenges of Fog Computing

The present study has recognized and defined different fog-related research challenges, which are needed to be taken care of so that the future of fog computing shall become a capable and noticeable resolution for evolving services with extremely varied necessities. Scheduling is one of the major aspects responsible to arise the need to accomplish resource productivity and dodge bottlenecks. Thus, the present research is focused on dealing with some of these issues, which have been presented in the subsequent sections.

1.2 Problem Statement

The major critical issues that Fog Computing experiences are *Load Balancing* (*LB)* and *Energy Consumption* (EC). Load balancing is the process of proficiently distributing inward bound network traffic across the backend servers or the server pool. Computation and storage are the major requirements of the users. In the fog environment, the allocation of resources should consider the current load at Fog Nodes. In addition, schedulers and resource managers should be invoked to take into consideration the most appropriate Fog Nodes [4]. This causes a load imbalance problem. Herein, some of them remain idle while others become overloaded. The load-balancing scheme in Fog Computing has three stages. The first stage involves a load balancer that collects the material related to the allocation of workload. The second phase revolves around making decisions on the paramount probable data dissemination. Finally, the data communicated from one overloaded node to another is done in the third stage.

The increased use of Cloud Computing has also introduced the world to a significantly new issue of increased energy consumption. To increase the efficiency and processing of Cloud Data Centers (CDC), the numbers of PCs are being increased. This leads to an increase in energy consumption. Thus, to deal with this ever-increasing energy consumption, Fog Computing offers a solution to cloud infrastructure. The reduction of energy consumption through Fog Computing has been a challenge for the community of current researchers. Resource allocation techniques for requests of the users is the aspect on which the consumption of energy is dependent across fog servers. Thus, it is important to develop an energy-aware scheduling process to save on energy in the Fog Computing environment as well. Based on the above discussion; the present research is focused on presenting the solutions to issues of Load Balancing and Energy Consumption.

1.3 Aim and Objective of the Study

The research aims to design a "Secure Delay Aware Scheduling and Load balancing for Deadline Sensitive Applications in Fog Computing Environment". The objectives have been defined below:

- To design a Framework for Scheduling Deadline Sensitive Applications.
    - ➢ Utilization of optimization technique to address the challenge of Load balancing.
    - ➢ Reducing Service level violation and controlling the consumption of energy in Fog Computing.
- Addressing the issues of load balancing and Delay Aware Scheduling using a newly designed Four-Tier Architecture in a fog environment.
- To provide Fog Assisted Task Allocation and Secure Deduplication in cluster-based Industrial IoT.

1.4 Contributions

The thesis contribution in the present research has been focused on three aspects, which have been discussed below-

1.4.1 Load Balancing as a resolution for Fog Computing

One of the most significant issues of Fog Computing that need to be addressed is Load balancing Even in a heterogeneous environment of Fog Computing where there are so many nodes and resources available to share the increasing pressure, Load Balancing remains a major challenge. The present study has, thus, focused on the reduction of the schedule length runtime and the minimization of the energy consumption rate. To achieve these results, the present research has proposed an algorithm. This algorithm is based on Artificial Bee Colony Optimization (ABC) optimization to balance the load for load management in the Fog Computing environment.

1.4.2 Reducing Energy Consumption and Violation of SLA

In the past decade, there has been an exponential increase in the use of Cloud Computing. This growth is being expected to increase even more in the coming years owing to an increasing number of IoT devices. With an increase in usage, real-time jobs to be performed by CDCs are also increasing. It is difficult for traditional data centers for performing in such a scenario due to inadequate resource bandwidth. A solution to this aspect of Cloud Computing has been presented to the world in the form of Fog Computing. Fog Computing is a complement of Cloud computing that has extended the model of Cloud Computing to the network edge. This has led to the development of a new variety of services and applications based on infrastructure. Fog Computing is a transitional technology that lies between the IoT sensors or devices and CDCs. It provides storage service, networking, and computing to enhance the cloud-based service offerings for its sensors and devices. Owing an increase in so many aspects of Cloud Computing has led to an increase in Energy Consumption. The reduction of energy consumption via Fog Computing has surely been a challenge for the researchers. Resource allocation techniques for requests of the users is the aspect on which the consumption of energy is dependent across fog servers. It enables processing at the edge with the probability to communicate with the cloud.

1.4.3 Delay Aware Scheduling and Load Balancing: The Solution in a Four-Tier Architecture

Load Balancing and scheduling are the two significant aspects of Fog computing that prominently influence the performance of an arrangement. Many researchers have been trying to come up with a model that will be able to facilitate more efficient and effective task scheduling and Load Balancing. Presently, numerous research exertions have been discovering the concept of scheduling in Fog computing. These innovative challenges can be presented by a set of the Fog node. Optimal workload allocation or dynamic Load Balancing is an additional significant issue in the fog environment. However, the Fog node is not currently balanced. The previous approaches presented by other researchers are lagging owing to aspects like Fog Node failures, resource shortage, distributed, or centralized environment to name a few. To mitigate all such issues in the previous researches, the present research has been focused on decreasing Energy consumption and latency in the fog environment via both Load balancing and efficient task scheduling. To deal with the issues of the scheduling of tasks and Load balancing found in the previous works, the present research has proposed a four-tier architecture.

The Fog-computing environment consists of four separate physical Tiers, that tier 1, Tier 2, Tier 3, and Tier 4.

- Tier 1, which is also known as the IoT-based Devices Tier, is the bottom tier where the physical sensors IoT devices are present.
- Applications are segregated into HP and LP in Tier 2. For classification, we proposed the Dual FLA, which considers the four parameters.
- The classified HP applications (tasks) by the router are transmitted to Tier 3, which is Fog-Tier.
- In Tier 4, instead of using a distributed environment or central control, the researcher proposed a new structure called artificial fractals. This proposed scheme is planned in the scheduler. The request is sent to the neighboring Fog Node via an IoT device. In addition, to achieve better allocation of requests, capacity, and current usage of every Fog Node is updated periodically.
- When needed resources are absent, then the required resource request is sent to the cloud.

The projected system has validated by evaluating the performance metrics, mentioned below, on the VSOT application:

- Response Time for a different number of tasks
- Scheduling time/length for Number of Tasks (Seconds)
- Load Balancing rate
- Delay
- Energy Consumption (KJ)

1.4.4 Task Allocation and Secure Deduplication- Assistance from Fog Computing

As an outcome of increasing IoT devices, the transmission of duplicate data over the Internet has increased. This increase in transmission due to duplicate data has then increased the pressure on the data center resources. The scenario of transmission of duplicate data causes the CS to delay the services to be provided to users on time.

1.5 Thesis Organization

The following sections are addressed in the present research to give a clear picture of the ways and methods to secure delay aware scheduling and load balancing (DASLB) for deadline sensitive applications in the Fog Computing environment.

Chapter 1 Introduction: The background of the study has been discussed in this chapter while giving an overview of Fog computing. It further discusses the Problem Statement and contributions of the study.

Chapter 2: Taxonomy on Large-scale data: This chapter provides an in-depth discussion on the taxonomy of large-scale data created by devices based on it. The chapter discusses and analyzes the work and research done by other researchers on techniques used in handling large-scale data, data migration over the cloud, or fog based applications, and problem-related to the deadline sensitive application.

Chapter 3: Fog Computing Environment- the State of Art for LB: This chapter presents techniques for Load Balancing in a fog environment, problem-related for task scheduling and Load Balancing, modeling of Load Balancing, research, and challenges for deadline sensitive application.

Chapter 4: Framework for Scheduling Deadline Sensitive Applications on Fog Computing: This chapter presents the literature review and system model for an efficient solution for Load Balancing using optimization techniques, tackling SLA violation, and reducing consumption of energy in the fog environment using Artificial Neural Network. Finally, at the end of the chapter results and discussion has been presented.

Chapter 5: Fog Environment Issues of Load Balancing and Delay Aware Scheduling - A Four-Tier Architecture Solution: This chapter presents the literature review followed by the proposed system on four-tier architecture. This architecture has been designed to handle the Delay Aware Scheduling and Load Balancing in a fog environment. Herein, system model, system design and architecture, experimental setup, a simulation environment, and simulation tool on Delay

Aware Scheduling and Load Balancing in Fog Environment. Finally, results and discussions have been presented.

Chapter 6: Task Allocation Using Fog Assistance and Secure Deduplication: The introduction, state of art, system model, including system design and architecture, simulation environment including simulation tool in Task Allocation using Fog Assistance and Secure Deduplication has been presented here. Further, it consists of a case study on air pollution monitoring. Finally, results and discussions have been presented.

Chapter 7: Conclusion and Future Directions: This chapter presents the conclusion and future directions on secure Load balancing and Delay-Aware Scheduling in the Fog-computing environment.

CHAPTER 2

A TAXONOMY ON LARGE SCALE DATA

2.1 Introduction

In times of increasing dependence on technology, now resolution in the field of communication has taken place. This has been attributed to the fact that around two billion people globally accessible information, multimedia, communicating through messages, audio and video calls, and many other different aspects. Owing to this scenario, in the recent two decades, the numbers of internet-based, technology-dependent smart gadgets have increased and are increasing with every passing day [5]. This then gives birth to an innovation in the field of communication called the youth, which facilitates the decision-making of smart gadgets [6].

With an increase in the utilization of IoT applications that enable real-time decision-making, there has been a surging increase in the requirement of enormous handling power, information storage, and high-speed broadband systems to stream data [5]. These requirements are now fulfilled by Cloud Computing, which is now being complemented by Fog Computing. Cloud Computing and Fog Computing provide the ability to handle large data streams thereby supporting a huge amount of it based application [7] [8]. Cloud Computing provides a huge potential to IoT devices; some challenges are to be faced by Cloud Computing when dealing with IoT devices. These challenges include issues associated with latency, mobility support, lack of limited network bandwidth, and awareness of the location. To handle these and some other challenges associated with Cloud Computing, another technology are now available to support Cloud Computing in the form of "Fog Computing". Fog Computing uses an open application programming interface [9]. Instead of batch processing, Fog Computing facilitates conducting real time-sensitive interaction along with some mobility techniques. The architecture of Fog Computing has been presented below, wherein its three layers have been described [10]:

i. **Device layer**: It consists of different smart devices that facilitate detecting feature data physical objects. Further, the device layer shares the processing and storage information.
ii. **Fog layer**: It consists of network Fog Nodes like base stations, switches, routers, and

gateways. The transmission, storage of temporary data, and computation are facilitated with these nodes. The latency-sensitive applications are supported by this layer

iii. **Cloud layer**: It consists of multiple storage devices and superior servers that provide different application services. It supports extensive computation, analysis, and stores a lot of information. The architecture of Fog Computing has been presented in Figure 2.1.

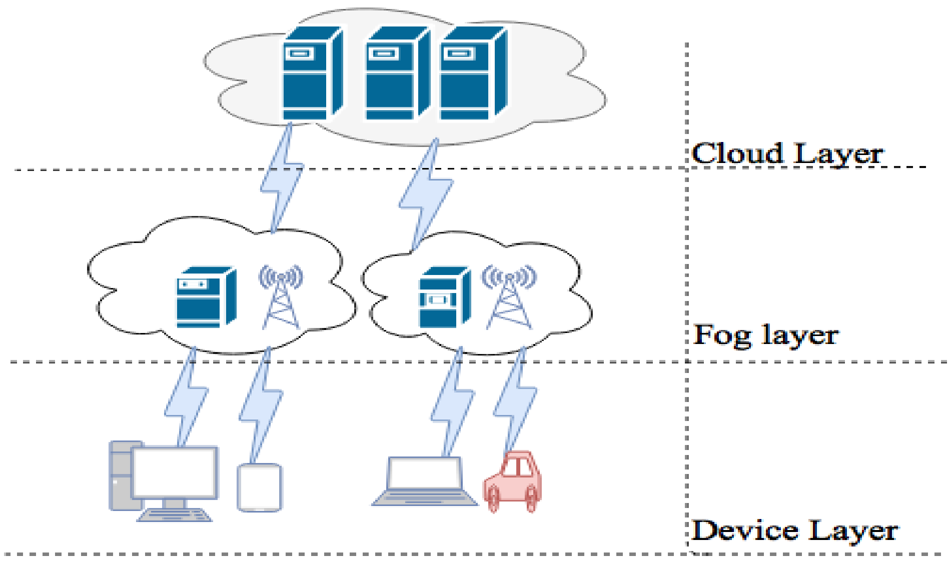


Figure 2.1: Fog Computing Architecture

A cloud-based application acts as a user of Fog computing, like data centers, switches, routers, and smart gateways along with Fog Nodes, which is a cloud-based edge component. Fog Nodes (FN) are resource-constrained devices that support computing, transmission protocol, capacity assets, and portability. Fog Nodes fulfill the requirements of communication overheads, computer applications, huge dissemination of scale, and constraint of latency.

Fog Computing facilitates Cloud Computing in providing communication among IoT devices and Cloud Computing thereby improving the quality of services and reducing the traffic and management load. While studying the aspects of Fog Computing, it is thus imperative to study various techniques used in handling large-scale data, data migration over fog based on applications. Before moving ahead with designing a system for fog environment for secure DASLB for deadline sensitive applications, it is significant to understand the dimensions (big) aspects of it growing data. Owing to this, the present chapter discusses the various techniques used in handling large-scale data, data migration over clouds, or fog based applications, and problems associated with a deadline sensitive application.

2.2 Techniques used in handling large scale data

A good amount of unstructured data is around floating globally, including the data, which are untreatable manually or by simple applications. Aspect like the World Wide Web, business services, applications and networks, and similar aspects produce data exponentially. This has been attributed to the progress of influential storage and assembly tools. Planned awareness and information cannot be gathered easily since this massive growth of data is neither understood straightforwardly extracted automatically. These aspects caused the growth of data mining and data science [11], which is a renowned subject that is omnipresent in the present age of information.

Currently, the amount of data being taken care of by systems has exceeded the ability of processing systems of previous times [12]. The ascending of new technologies like Fog Computing along with the decrease in hardware price is leading to the ever-increasing rate of information online. This scenario presents a "Big" issue with the data analytics community. Hence, "Big Data" is stated as a collection of large velocity, variety, and volume of data that needs speedy processing [13].

Distributed computing has been traditionally accepted by data-scientists before the introduction of the Big Data initiation with the help of various regular and prolonged algorithms. The researcher replaced their distributed versions to analyze the learning process. However, for most of the current massive problems, nowadays a distributed approach becomes obligatory because architecture lot is unable to handle such big issues. Numerous stages for processing done on a Large-scale have tried to solve the issue associated with Big Data in the past decade [14]. These stages attempt to reduce the difference in the distributed technologies to the typical user. Difficult schemes are obligatory to generate and preserve these stages, which simplifies the custom of dispersed computing. Further, platforms of a large amount of data also necessitate supplementary algorithms that provide funding for related jobs, like big data pre-processing. Typical algorithms for those jobs necessitate the re-designed if the author wants to study large-scale datasets.

MapReduce is the major outline that allowed the treating of large-scale datasets [15]. This groundbreaking implement was envisioned to develop and create large data sets in a distributed manner. By applying Map and Reduce, the employer is capable to take into consideration an accessible and spread device short of upsetting the technical shades, like retrieval of failure, data separating, or communication of job. Apache Hadoop [17], [18] appeared as the most standard open-source application of MapReduce, preserving the above-mentioned characters. Owing to its excessive admiration, both Hadoop and MapReduce are not intended to gauge. Apache Spark [19] was presented as a substitute to Hadoop, skillful enough to execute speedy distributed computing via in-memory primitives. This was achievable due to its capability of packing data into recall and re-using it frequently, this implement incapacitates the issue of recitative and network processing presented by MapReduce. Moreover, Spark is a general-purpose structure that permits to execute of many distributed programming prototypes on it, for example, Hadoop) [20]. The spark is made over a new abstraction module known as *"Resilient Distributed Datasets*" (RDDs). This multipurpose module permits governing the perseverance and handling the segregating of data, amid other characteristics.

Certain opponents to Apache Spark have also come into existence, particularly from the side of streaming. Apache Storm is an open-source distributed instantaneous processing aspect, which processes a gigantic number of tuples per second along with being fault-tolerant. Apache Flink [21] is a current top-level Apache development being brought into existence for the processing of batch data and distributed stream. Together substitutes attempt to mend the gap "Online" created because of Spark, which takes into consideration a "*mini-batch streaming processing"* in place by a method that involves clean streaming.

The superiority and performance of the information mined by data mining in different aspects are subject to not only the framework and working on the approach but also relied on the appropriateness and data quality. Sadly, destructive characters as missing values, noise, unreliability, and redundant data, and massive magnitudes are samples and present encouraging the data used to acquire and mine the information. It is a renowned fact that reduced data quality will bring the process to inform proving to be of low quality. Therefore pre-processing of data [22] is a chief and most important phase whose key objective is to get concluding data sets. Such data sets take precise and valuable for additional algorithms for mining of data.

2.3 Data Migration overcloud or fog based on applications

Some studies [23], [24] presented the "*Cloud Adoption Toolkit*" as a methodology for understanding the viability of cloud migration. The following aspect has been taken into consideration:

1. Costs associated with cloud adoption
2. Risk management
3. Considering trade-offs amid risks of migration and benefits related to the cloud.

The author of the study [24] has observed and worked on the edge issue of viability and possibility. The suggested cost modeling approach is advanced and based on the present costs, associated with infrastructure, short of utilization. The author used the ability to control activity from the approaches to power, avoid over-provisioning on the new cloud. The cost approximation tool presented [25] was global as Plan for Cloud. The authors suggested exhibiting an application and its essential substructure and organization to approximate the running costs associated with the cloud.

A study conducted by [26] recommended a method for "*enterprise applications*" migration to the cloud. Their key focus on representing the current system and installing a "cloud permitted" version. The main jobs in this progression are also computerized. The distinguished strong suit of [26] involves-

(1) Advanced automation

(2) The capacity to maintain organization or records in the form of data sets.

The Process designed by the REMICS project [27] backs heritage aspects migration to the" data cloud model" which is schemes with an emphasis on cloud services architecture. It emphasizes the re-design of the code, with related issues like testing, interoperability, and implementation.

The practice is stated by strategies to become features from applications of COBOL and produce Java. Likewise, the ARTIST practice and outline [28] funds the migration of the heritage aspects to the cloud combined with REMICS, this increases feasibility valuation and business process alteration. The authors contend that when wandering to the cloud, these are natural and important doings. Neither study looks freely at migration nor database modeling A model-based tool called Clouding [29] mechanically monitors a system's appropriateness for a cloud platform. It gets model architecture through the source code, using a log file approach. Composed of a cloud-focused environment approach, it makes it easier to understand whether the device is cloud-focused, compliant, prepared, linked, or enhanced.

The CloudSim simulation platform is a long version of Cloud MIG, but the modules for exhibiting and simulating are no different. This restricted coupling prevents their reprocessing in evaluating cloud migration.

A study conducted by [30], [31], it was suggested a new process for moving the database to the cloud from the current organization. It is based on difficult movements of databases where a noteworthy amount of data presented and the parts of the software continue to work on in-house software servers. The assets of this approach involve its stage of the element, the assessment with huge systems based on real world (example-NovaERM and SimTechSWfMS), and the associated "*Cloud Data Migration*" Tool [31]. This will facilitate different organizations to migrate their databases. Other main approaches in the field of migration and database transformation include "Minimal Schema Extraction"[62] and the business-based information discovery system.

The current method for mining KDM models via SQL takes into consideration a midway ASTM-based model [10, 63]. The "Abstract Syntax Tree Metamodel" is an "ADM meta-model"[65] that makes it easier to standardize the syntax tree that is constructed from the source. To achieve advanced interoperability the meta-model is intended here. A change involving model-to-model methodology is then executed on the ASTM model. That results in the KDM model being developed. A midway model necessitates two alterations to be established and preserved.

In a study conducted by [32], it was suggested to mechanize methods to mine a data design via source code of the software and summarize inheriting databases through the research services [33]. These will probably be of attention to businesses using legacy databases. To classify and mine the statements of SQL it encompasses a static examination of a system's source code. These SQL statements take into consideration to produce a design that individually involves the columns and or tables. The drawback of this method though is that executing it is problematic. There are various programming languages such as SQL alternatives and "database access libraries;" and to be precise, to detain all the inquiries embedded in the code

On the other hand, DBL Modeler takes SQL dumps generated from database IDEs to ease this difficulty and to sustenance a widespread variety of systems. These yields SQL in a typical output arrangement regardless of in what way a systematic inquiries its database.

Apart from the researches discussed in the above section, many investigations, researchers examine the arena of migration of cloud. Most particularly, [34] studied forty-three papers to recognize their characteristics, compare, and research excellence. The researchers detected what the way devices maintain cloud migration is not in abundance even with the amount of effort in the zone and numerous researches stressing this problem.

- After a detailed review of the available publishing on fog computing, roughly more than 800 papers were classified based on their title and abstract and it was found that 39 percent of papers were introduced. 21 percent were based on scheduling challenge, 18 percent were based on security problems, 11 percent were based on the case study, 11 Percent were concept articles. Figure 2.2 and Figure 2.3 shows the total no of research paper and distribution of fog computing papers concerning the survey respectively.

- Based on the survey, scheduling, Energy Consumption, and security in fog computing were found some of the main concerns.

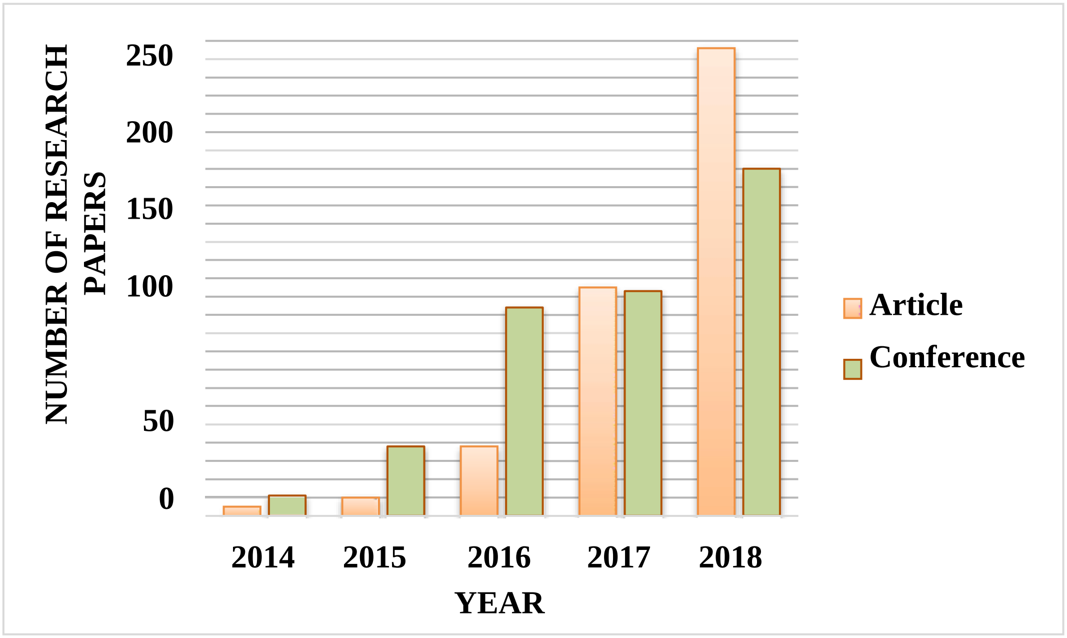


Figure 2.2: Number of research papers

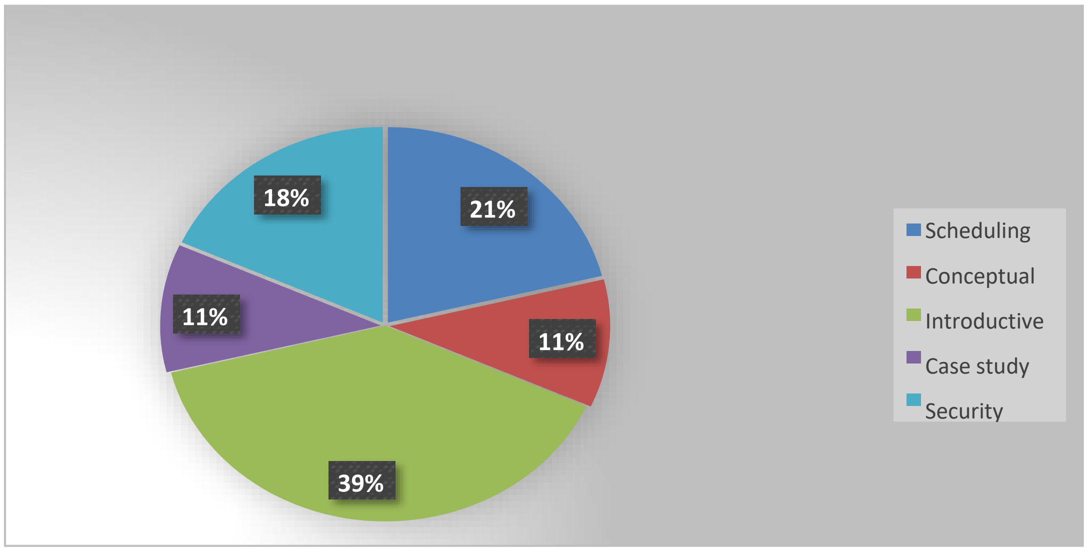


Figure 2.3: Distribution of Fog computing papers concerning the survey.

2.4 Related work regarding Fog computing and scheduling

Usually, the fog-node was deployed in a fog environment to study data collected from IoT devices. This segment addresses specific research issues in smart IoT applications like scheduling and load-balancing (LB). In the fog, Bittencourt et al. [45] implemented a mobility-aware application schedule, with three scheduling strategies invoked to affect mobility applications in the fog environment. They are concurrent strategy, first come served (FCFS), and strategies to delay goals. Yi et al., [82] proposed an energy-efficient schedule designed to minimize energy consumption over heterogeneous IoT flow computing architecture. Rahbari et al. [111] developed a search-optimized knapsack-based symbiotic organism schedule that is suitable for little traffic and little latency networks. The simulated result indicated an 18 % enhance in energy use, 15 %cost of execution, 1.17 % overall network use, and 5 percent lifetime of the sensors in schedule increases. Madni et al., [86] suggested various heuristic methods for proficient cloud-based scheduling of tasks. Consequently, the effective algorithm selection for task scheduling is complicated since the algorithmic procedures are implemented in various assumptions. FCFS (First Come First Serve), Minimum Completion Time (MCT), Minimum Execution Time (MET), is the most extensively used task scheduling algorithms. Yin et al., [112] implemented task scheduling and resource allocation, containers in fog networks intended primarily for smart developed applications, the planned task scheduling, and resource redeployment schemes successfully minimize task delays and improve the productivity of tasks in fog nodes.[113],[114] different optimization technique and survey of recent technology is proposed in recent research,

2.4.1 Related work on Load Balancing

Balancing of Load is a key factor that defines resource allocation efficiency and management strategies. Deng et al., [110] tackled the issue of allocating workload that can be designed to reduce power usage and service latency. Pham et al., [88] a cloud-fog environment, given a schedule of tasks. In this work, a fog provider can take advantage of the relationship between their fog nodes and leased cloud nodes to run user-friendly large-scale offloading applications efficiently. Because of this, a heuristic-based algorithm with the main purpose of finding a compromise between the costs of cloud resources is proposed.

CHAPTER 3

FOG COMPUTING ENVIRONMENT- STATE OF ART FOR LOAD BALANCING

3.1 Techniques for Load balancing in fog environment

Balancing the load is a chief factor that recognizes resource distribution, performance, and management strategies. The author of the study [35] stated that issues relating to loading balancing come up with different applications and the program is parceled into smaller jobs, which are then completed concurrently. The specific operation is to be conducted by them in the application of distributed and parallel computer systems. The researcher recommended and load arrangement formed with the help of genetic algorithms as a solution to the dynamic Load Balancing problem.

In a study conducted by [36], "*Particle Swarm Optimization (PSO)*" was taken into consideration in VMs for the accomplishment of the load balancing. The author presented a discussion on the situation where the reduction in the cost is done and not at the expense of the service quality and maintaining the SLAs. A new suitability means were planned for the reduction in the cost and examination of the server related load. Moving in the same direction, [37] for optimization of Load balancing, employed swarm intelligence for Load balancing optimal. The author presents the issue of optimization to be NP that is a difficult tissue and correspondingly sees into the under loaded condition of VM if any. The author has proposed the scheme of load balancing for VMs by optimizing the QoS service parameters and throughput. Further, [38] conducted a research where the author presented a framework of Fog Computing using Cloud Atomization Technology. This cloud atomization technology is used to modify physical nodes in VM nodes in increasing stages. The author then considered a graph partitioning theory to create algorithms for Load balancing for Fog Computing based on dynamic graph partitioning. The process of Load balancing has proficiently organized system resources with the lessening of the ingesting of the node.

[39] that in an environment of Fog Computing introduced a well-systemized algorithm meant for Load Balancing have stated it. In this meeting users' requirements, maintenance of consistency of data with reduced complications is done, surged throughput, utilization of the network, and

scheduling of tasks are done within the time limit.

3.2 Problems related to load balancing and scheduling of task

In FC, proper management of resources is important for better resource utilization. However, there are many different types of challenges in FC as discussed below [40]-

- Dynamic and Fluctuating workloads:

Workload fluctuations occurs in fog computing. In two ways; pre-determined and post-determined. A predetermined fluctuation situation is analyzed earlier and hence resource allocation is done efficiently within an appropriate period.

- Ensuring efficient resource utilization:

In a post-planned scenario as per requirement resource allocation should be done directly this is known as “auto-allocation”. For fog schedulers to be assured about the successful utilization of resources to be efficient, the incoming workload should be resource allocated. For task allocation to nearby processor, scheduling techniques need to be improved.

- Heterogeneous physical nodes in fog data centers:

Scheduling of tasks is done within the accessible nodes. These nodes are spread at the different architecture and different locations, varied memory, different network performance, and power of computation.

- Increased scheduling granularity than existing scheduling:

The size of issues of the problem of scheduling has grown from comfortable, informal scheduling of the process with little movement of data to exhaustive VM scheduling and VM reallocation.

CHAPTER 4

FRAMEWORK FOR SCHEDULING DEADLINE SENSITIVE APPLICATION ON FOG COMPUTING

The first objective of the present study is to design a framework for scheduling deadline sensitive applications. This has been achieved in the study in two stages. In the first stage, an effective solution has been designed for Load Balancing using optimization techniques. In the second phase, a system has been designed to reduce the consumption of energy and SLA violation in Fog Computing.

4.1 An Efficient Solution for Balancing load Using Optimization Technique

4.1.1 Introduction

The resource requirement in its applications is done using fog or Cloud Computing nodes. Efficiency in achieving and using these resources is obtained by managing Load Balancing. Avoid aspects that hinder the efficient utilization of resources like low load, bottlenecks, and overload via efficient Load Balancing. During the execution of its applications in fog environments, load balance is still a challenge. Thus, to tackle the issue, the present chapter of the research is dedicated to finding an efficient solution for Load Balancing using an optimization technique.

4.1.2 Related Work

In a study conducted by [41], it was suggested that scheduling of tasks is extremely important for the appropriate operation of a framework of the parallel processor. The author further in the study inspected the elective paradigm based on a genetic algorithm. This paradigm was elected to fix the issues related to scheduling avoiding the requirement to present any limited assumptions, which are explicit to the problem, like the scenario while employing heuristics. Further, [35] stated that issues relating to load balancing come up in different applications, and the program is parceled into smaller jobs which are then be completed concurrently. The specific operation is to be conducted by them in the application of distributed and parallel computer systems. The researcher suggested a Load Balancing system of a vigorous nature produced via genetic algorithms to resolve the vigorous Load Balancing issues.

In a study conducted by [36], *PSO* was taken into consideration in VMs for the accomplishment of the Load Balancing. The author presented a discussion on the situation where the reduction in the cost is done and not at the expense of the service quality and maintaining the SLA. A new suitability means were planned for the reduction in the cost and examination of the server related load. Moving in the same direction, [37] for optimization of Load balancing, employed swarm intelligence for the optimization of load balancing. The researcher presents the issue of optimization to be NP that is a hard problem and correspondingly sees into the under the loaded condition of VM if any. The author has proposed the scheme of Load balancing for VMs by optimizing the QoS service parameters and throughput. The research study conducted by [42], presented the suggestion of effective allocation of resources.

In Fog Computing, efficient resource allocation (ERA) is applied to the expert of the cloud for checking the projected method performance. The authors found that after the implementation of this method, this mechanism might assign the resources in an improved way and superior to the traditional algorithms via the transfer of data, complete reaction, and deployment of bandwidth. Further, [38] conducted a research where the author presented a framework of Fog Computing using Cloud Atomization Technology. This technology of Cloud Atomization Technology is used for altering physical nodes in changing stages in VM nodes. [43] Examined the perfect allocation of workload in the direction of intake of power and balanced delay. The authors found that Fog computing improves Cloud computing performance. Further, a scheduling issue was presented by [44] in the diverse staged configuration of fog and Cloud computing. The research presented that scheduling systems intended to acclimate to diverse classes of application as per the requirement of movable customers; the captivating benefit of both the Fog Computing and Cloud Computing [45] did some analysis on the scheduling of service requests over Virtual Machines (VM). In this scenario, a trade-off was held in reserve between the consumption of energy and the make span. The author confirmed that in Fog Computing, the performance of the met heuristic method, which authorized to delay the heterogeneity of resources. [46] Researched the new expertise in FC about service, safety, and structural related difficulties. Fog computation classification was also delivered grounded on the standard chief difficulties and features and investigation problems have been planned. Table 4.1 shows the techniques used by the different researchers in the cloud /fog layer.

4.1 Summary of Different Aspects and Methods Utilized by Different Researchers

Sr. No.	Researcher	Fog Layer (FL) / Cloud layer (CL)	Parameters	Techniques	Limitation
1	(Pandey, Wu, Guru, &Buyya, 2010)	CL	Load Balancing, Transmission Cost	PSO	No latency issue is covered
2	(Li, 2011)	CL	Load Balancing and minimizing make span	Ant Colony Optimization	Data reduction affect performance
3	(Zhao, 2009)	CL	Task scheduling	Genetic Algorithm (GA)	Limitation to few policies
4	(Ge, 2010)	CL	Scheduling	Genetic Algorithm	Reliable energy resources are required
5	(Guo, 2012)	CL	Optimal Time & Cost	Multi-objective-Particle Swarm Optimization (MOPSO)	VM limited data centers
6	(Zhu, 2011)	CL	Load Balancing	Multi-agent Genetic Algorithm	Huge data is required to train the system
7	(Netjinda, Sirinaovakul, &Achalakul, 2012)	CL	Cost	Cost optimization) Particle Swarm Optimization (PSO)	Reliable energy resources are required
8	(Junwei, 2013)	CL	Average Cost & task completion time	Complement ary of multiple objectives	Load need to be balanced with latency reduction
9	(Wan, 2018)	FL	Load balancing and scheduling	Improved (PSO) algorithm	Fail for multitasking
10	(Bitam, Zeadally, &Mellouk, 2018)	FL	Job scheduling	Bees Swarm optimization	Min. time needs to be reduced

4.1.3 The issue to be addressed

Via an investigation conducted, all the above-mentioned limitations have been addressed in this research work, as mentioned in subsequent sections. In the current objective, we deal with different parameters: energy consumption, SLA violation, utilization of VM, Reduction of cost. In the present study, it was found that the scheme of Load Balancing grounded on Fog Computing comprises three phases, namely an assembly of data, the accurate choice, and relocation of data. In phase one, the load balancer collects the evidence for the distribution of assignments and found if there is an imbalance of load. In phase two, a choice to estimate the best conceivable data distribution is made. In the final phase, communicated data from one overloaded node to another under the loaded node. If the characteristics of nodes are not augmented then the communication may be for the different node, which is overloaded. So, in Fog Computing, to decrease the likelihood of unwelcome distribution, the idea of an optimization method with the balancing of the load is utilized.

The proposed work model has a couple of points, which are considered on the VMs architecture. They are listed as follows:-

a) The processing unit has three VMs with different cost values
b) The parent task will be executed first
c) The children will be ranked and they will be executed as per their ranking
d) The proposed model has to look for a balancing framework

4.1.4 Proposed System Model

The model proposed for presenting an efficient solution for Load Balancing using optimization technique consists of certain aspects based on VMs architecture, have been enlisted below-

- There are different cost values associated with three VMs of the processing unit
- The task to be completed first is the parent task
- The execution of the children is done based on their ranking
- A balancing framework is the aim of achievement of the proposed system

The start time and end time of each process will be present. An allocation of VM with one job cannot jump to another job without concluding the job. Parallel execution of jobs is allowed. The ABC architecture is adopted for prioritization of the jobs. The sorting of the jobs is done via connections in the graphs through ABC architecture. Here, firstly connections of high jobs are executed if it lives up to the requirements of the ABC architecture. Balancing of the load is done using ABC as shown in algorithm 1. Every task will have an end and a start beginning time wherein FT is the finish time and ST is the start time. The algorithm above is based on the job priority order of the allotted area. Cuckoo fittest [47] is presented to regularize the rank of jobs. If an egg is found to be defective by a cuckoo, discard all its eggs. In this case, the brink is arbitrariness in the average cost of an energy power job.

If an arbitrary disparity is an entire cost, it is prepared and the threshold is matched with a dissimilarity Algorithm in table 4.2 shows load balancing using ABC Algorithm.

When a program is distributed across several processors, the job execution model consumes more energy (Agarwal, Yadav, & Yadav, 2015). In an energy model, the following factors related to energy consumption fitted with a mathematical model (He, Ren, Shi, & Fang, 2016):-

i. Energy is used to process the data and results.
ii. Energy used to transmit the data and energy required to maintain the communication so that a mobile application's split into-

a) Instructions to be executed
b) Input data used and Output data generated
c) Computing process
d) Transmission of data

The job that fulfills the requirements is prepared for its state; otherwise, it is positioned last. In the specified situation, the procedure taken into consideration differs by single or double positions on the job as the search of cuckoo has some arbitrariness associated with it. Table 4.2 shows Load balancing using the ABC optimization technique.

Table 4.2: Load balancing using ABC

```
a) Engaged_Bee → Existing _Bee
b) Onlooker_ Bee →Judgment _Maker

1.Fxn situate (Load_ coasting, Jobs)
2. Initialize Load (L) = 0;
3. Initialize Priority Order(P.O) = [ ];
4. for
5.     J == 1 ;  Job
6. for
7.     I==1 : High _Connection_ count
8. Load (L)= Load+ Load –costing (J.I) ;
9. End for
10. End for
11. Total L = L /( JOB * (High_ Connection-count));
12. P.O =  (numel(Jobs));
13. FALSE_Food Source = FALSE;
14. FALSE_Food Source = FALSE;
15.  for
16. 1<=0 :

Total No._Bee /  High__Connection

        $Employee = \sum_{I=0}^{power\_Node1} \frac{Energy\_Node\ (k)}{I}$

        Employed_Bee * random ( ) < Load (L)*random())

17. If
18. If
19. FALSE_Food Source = TRUE ;
20. temp= (P.O )[last];
21. Adjust (P.O one position);
22. Else
23. End if
24. End for
25. STOP
```

Let Pcbe the energy consumed for processing instruction in a task and Pt be the energy used for transmission of data. Then choose to perform the task on fog, and then the energy consumption is given by the following formula (Baccarelli, Cordeschi, Mei, Panella, Shojafar, &Stefa, 2016):

$$E_{cloud} = P_t + D/B \qquad (4.1)$$

Where D is data in bytes to be processed and B is network bandwidth. Assume that the output generated is D` then it is safely being assumed that D` is smaller than D. Now if the mobile device/smart device performs the task than the energy consumed is:

$$E_{mobile} = P_c * I + P_t * D/B \qquad (4.2)$$

Where, 'I', is the numerals of instructions for the execution of a task, then energy saved (E_s) is:

$$E_s = P_t * D/B - P_c * I - P_t * D'/\mathrm{B} \qquad (4.3)$$

$$E_s = [P_t * D/B] - P_c * I - [P_t * (D * K)] \qquad (4.4)$$

Where K=D`/D and K are called the compression ratio.

$$E_s B = P_t D(1 - K) – P_c * I \qquad (4.5)$$

Energy is saved if this equation is positive and only then it would be preferable to offload the task (Puthal, Obaidat, Nanda, Prasad, Mohanty, &Zomaya, 2018). First, classification is done on incoming requests as to whether it is CPU bound or input, output memory. The incoming request follows one or the other algorithms and every algorithm further be classified into time, logarithmic algorithm, a polynomial-time algorithm with an exponential time algorithm, etc. In a CMOS integrated circuit such as a modern CPU, the power consumption is reduced by reducing the frequency as given below (Byers, &Wetterwald, 2015)

$$P = CfV^2 + P_{static} \qquad (4.6)$$

Where 'C' is considered as the transistor gate capacitance, 'f 'is the frequency and 'V' is the supply voltage and static denotes that it is not performing any operation. The voltage required for normal and static operations is dependent on the frequency on which the circuit is being clocked. This voltage may be reduced by reducing frequency. This will result in an important reduction in power consumption by reducing voltage. It is pertinent to mention here that energy is only being protected if the power consumption is reduced enough to run the workload, to cover the extra time it takes to process at the lower frequency. Thus, it is a tradeoff between energy saving and time-saving (Hong, Lillethun, Ramachandran, Ottenwälder, & Koldehofe, 2013). The

Energy E is given by Power*Time hence, E is proportional to V^2.

$$E \alpha V^2 (V \alpha f) \qquad (4.7)$$

Morally the job in the second position has to be scheduled for the process job in the third position. This will involve some arbitrary processing price, which will then be applied from VM1 to VM2 and VM3. If a particular arbitrary cost is taken, say from 1 to 3, which are 15. This will then be included in the total cost. Henceforth, the total cost would differ as presented in Table 4.3 and Figure 4.3 showing the different costs associated with the job.

The structure of job 2 is presented below. Firstly, contemplate the second job:

1. For the execution of Job 2, their are3 options;
2. The aspects to be followed before the allocation of the job 2 any VM, subsequent must be obliged
    - The free slot has to be therefore VM to implement this request for the job;
    - When compared with other available VMs, the cost should be least at the VM;
    - Less EC is caused by reducing time consumption

The following architecture, explains the calculation (Figure 4.1 and 4.2). The execution cost is:

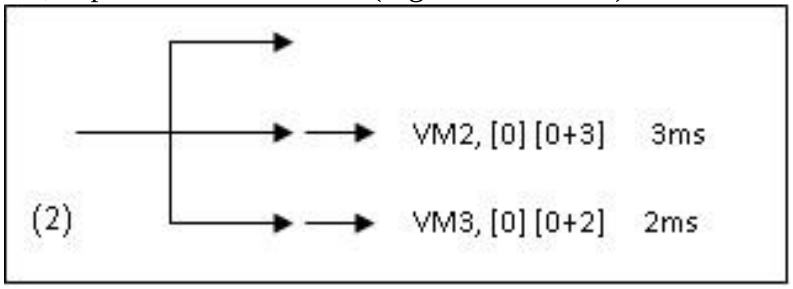


Figure 4.1: Assignment of VMs

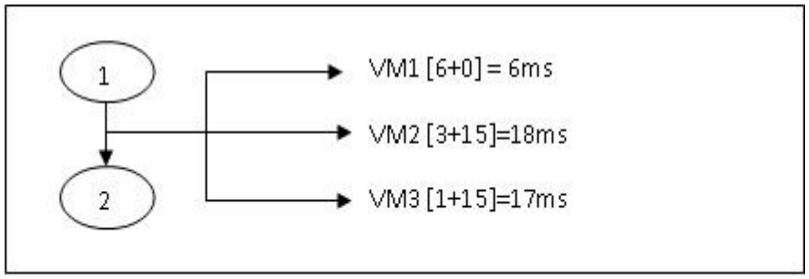


Figure 4.2: Assignment of jobs to VMs

Table 4.3: Costing of VMs for different jobs

Job No.	**Costs of Job at VM**		
	VM1	**VM2**	**VM3**
1	2	4	6
2	4	3	2
3	1	4	2
4	2	1	4
5	3	1	6
6	4	1	3
7	13	11	12
8	5	6	7
9	11	2	6
10	6	1	4

The calculation is explained using the following architecture shown in Figure 4.1 and Table 1 shows the costing of different VMs.

The total cost of execution is:

$$Total\ Cost = \text{Ec} + \text{Tc} \qquad (4.8)$$

Where:

$EC = Execution\ Cost$

$TC = Transfer\ Cost$

TC is applied in case the parent is implemented on some other VM. Job 1 has job 2 connected before it. Thus, in a scenario where job1isexecutedonVM1 then other than VM1, VM2, and VM3 have to bear the transfer cost.

The model of the job has two situations that are single and multiple parent jobs. , Node 3 and node 5 are the parents of node 8. The complete scenario has been depicted in figure 4.3.

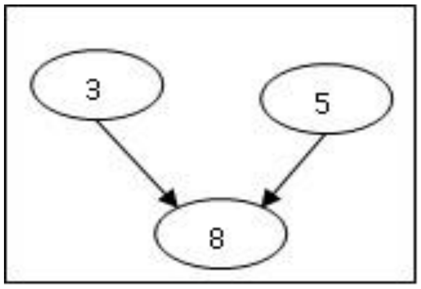


Figure 4.3: Multiple parent case

Multiple parents create a scenario wherein job 8 (figure 4.5) cannot be executed unless and until both parent 3 and parent 5 are executed. In case, one of the two-parent nodes has competed for the execution while the other parent node is in the process of execution, job 8 will wait. In the present case, presumptuous that 3 are completed on VM2 while 5 is executed on VM1. The structure of execution is presented in Figure 4.4. It shows that the fifth job is executed on VM2 while VM1 supports the execution of job 3. The graphs have been presented just for demonstration aspects. The actual model might differ. Now, the presented model will run both jobs 3 and 5 as job 8, which has two parents. The execution structure is shown in Figure 4.4.

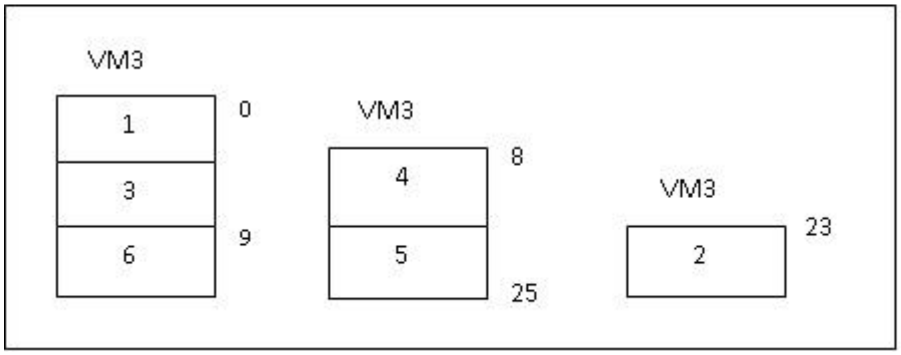


Figure 4.4: Execution structure

4.1.5 Performance Parameter

The proposed algorithm has been presented in this section of the chapter. For the initial phase, the researcher has considered 1000 jobs for the simulation phase, parameters of performance, including SERVER, SLR, ECR, and EC. The proposed algorithm MATLAB. The description of the algorithm has been presented below:

1. **SLR**: At all VMs, the SLR is the completion time of the accomplishment of the job model. It is given by the SLR equation as below:

$$SLR = \sum_{k=0}^{n} Jobexecutiontime \qquad (4.9)$$

Where,

$n = total\ number\ of\ jobs\ in\ the\ job\ model.$

2. **SLVMR**: In the case when jobs are executed at one VM only, it is the ratio of the SLR to the time. It is given by the equation as below:

$$SLVMR = \frac{SLR}{Jobtime\_oneVM} \qquad (4.10)$$

1. **EC**: It is the total consumed energy to execute all the tasks.
2. **ECR:** If all tasks are executed at one VM, the following is the formula:

$$ECR = EC/Energy\ cousmed\ by\ 1\ vm \qquad (4.11)$$

Table 4.4: Model environment

Tool	**MATLAB**
No. of Job Model	100
Max. No. of Jobs	1000
Simulation Area	1000* 1000 m^2

The computational cost associated with selected queries depicted in Figure 4.5, has been considered throughout the research. The setup for the simulation of the proposed model is as follows in Table 4.4. Simulation implemented using MATLAB R2013a on a 64-bit machine having a core 7 processor and 8GB of Memory. We assume the simulation area 1000*1000 m^2 and a certain number of nodes are randomly distributed in the area. Every node is modeled as an independent functional entity and moves randomly according to a random waypoint model. These associated costs are attained with the uppermost arrangement of VM for the architecture of ABC. The approximation of the population and generation of a number of the algorithm are the most important features that should be turned all around to attain solutions that are optimal at the time of optimization. The computation of SLR via the overall number of tasks has been presented in Figure 4.6. 1000 jobs have been included in the computation. As presented, the red line has presented the SLR results, DAG while the results of the proposed SLR by the blue line. The proposed model establishes a vast alteration of almost 500 ms for the accomplishment of 1000 jobs. Decent SLR will yield a decent SLRVM ratio. A reduced amount of SLRVM recognizes the development of the structure as the denominator is the time consumed at VM1, which is most of the time more than the numerator. The total number of jobs owing to energy computation is presented in Figure 4.7. For computation, 1000 jobs have been considered. As presented, the orange line has presented the consumption of energy results, DAG while the results of proposed consumption by the blue line.

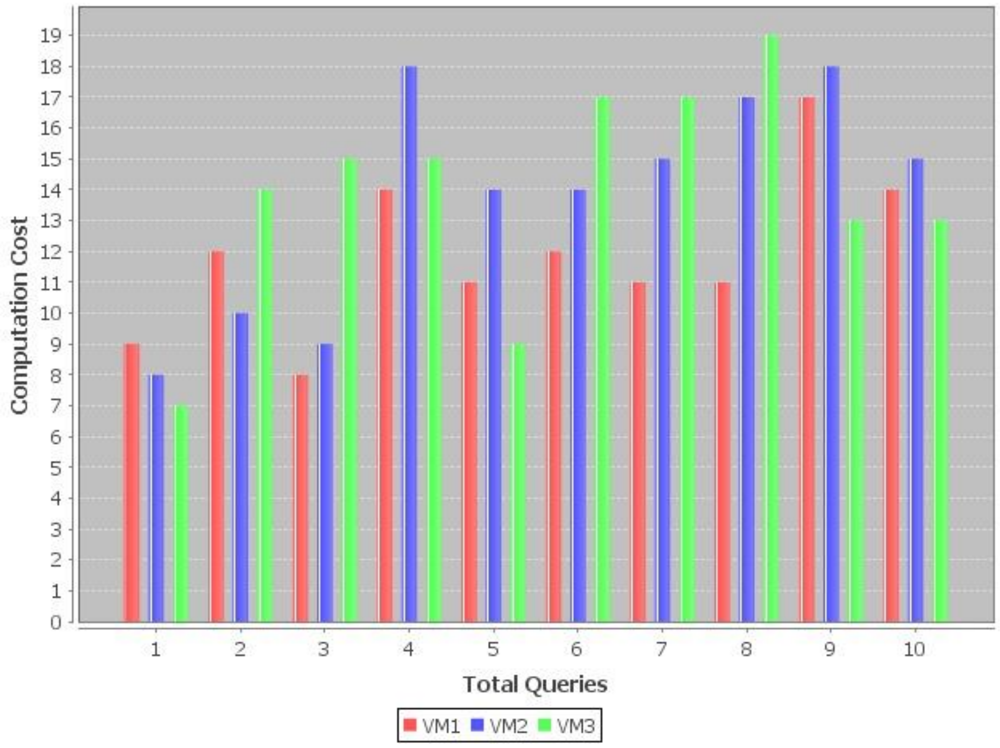


Figure 4.5: Computational cost of sample queries

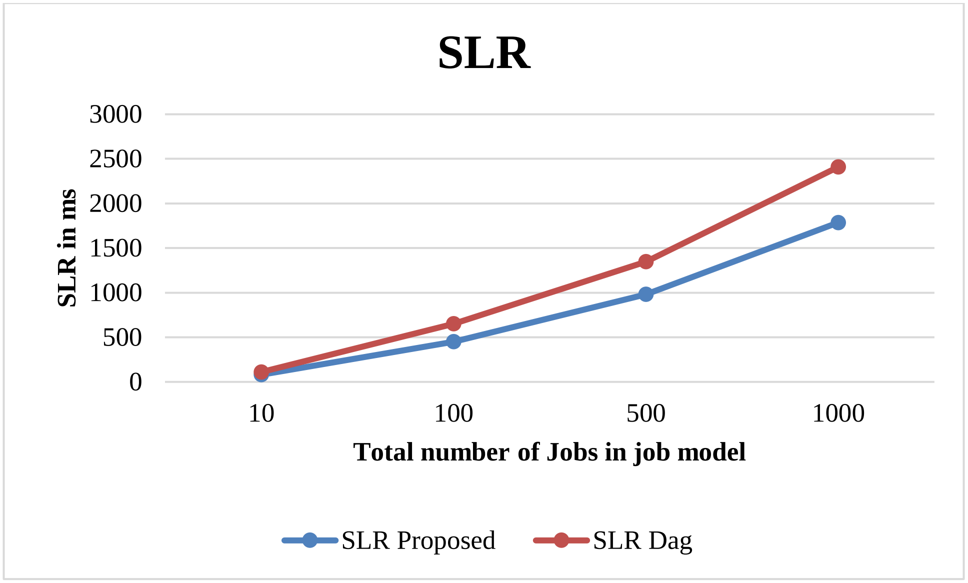


Figure 4.6: SLR (ms) for DAG and proposed

According to equation 4.11, the maximum energy consumed for the proposed model is 229 mJ for 1000 tasks. In the DAG model, the high-energy consumption of 400 mJ was found in the same count of jobs. Figure 4.8 shows the graphical representation of SLRVM proposed with SLR, DAG, and ECRVM proposed with ECRVM, DAG. A comparative analysis has been shown in the figure for the same. The comparison has been drawn for several jobs. As shown, X- jobs are considered whereas the- axis defines the SLRVM and ECRVM computation.

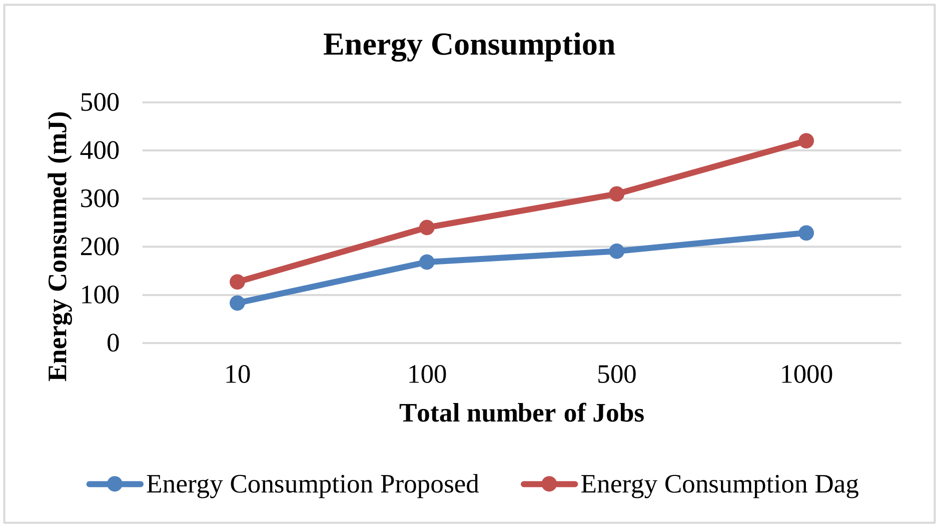


Figure 4.7: Energy Consumption (mJ)

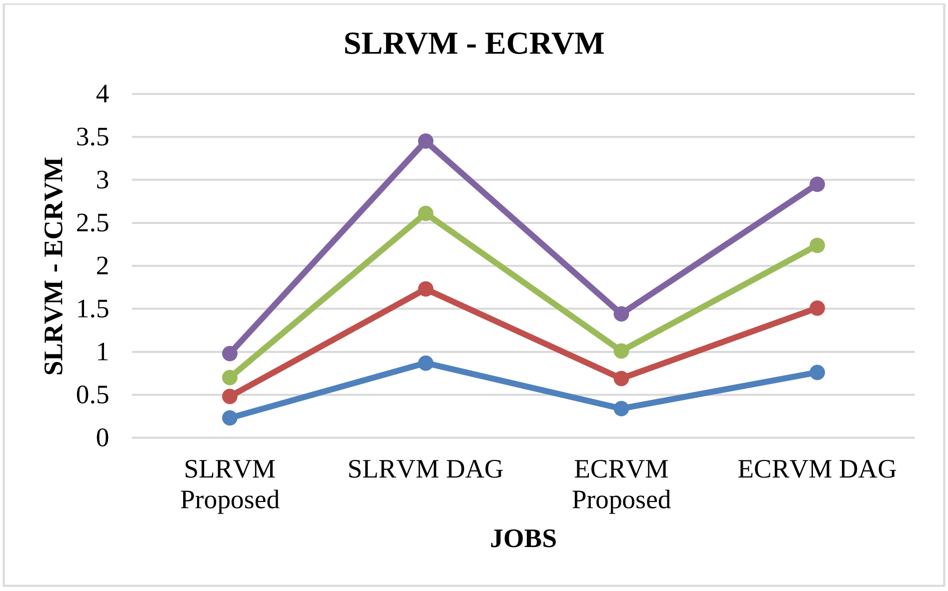


Figure 4.8 SLRVM and ECRVM

4.2 Reducing Consumption of Energy and SLA Violation via ANN

The proceeding section worked on the first phase of objective one that is to provide an efficient solution for Load balancing using the optimization technique. To fulfill the first objective that is to design a framework for scheduling deadline-sensitive applications, the second phase is also to be proved which revolves around minimizing Energy Consumption and SLA violations in Fog Computing using ANNs. The succeeding section will work on this aspect of the research.

4.2.1 Introduction

Fog computing is stated to be a platform of virtualized nature that provides computation, storage, & networking services amongst end devices along with conservative data centers for CC. However, these have not been entirely situated crossways the edge of the network [48]. The Fog computing environment has state-of-art mechanisms of switches, proxy servers, routers, networking, Base Stations (BS), and set up boxes that are situated at the neighboring closeness of devices and sensors as presented in Figure 4.9. The mechanisms are presented in Figure 4.9, having distinct computing, storage, and capability of networking that facilitate in sustenance the implementation of service applications. The effective implementation of FC via consumption of power, network traffic, service latency, and content sharing, and other such aspects [49]. The idea of consumption of energy is relative to traditional with the use of properties related to data centers. Similarly, associated cost inclines to be straight compared with energy ingesting. For resolving the rigid nature of difficulty connected to the distribution of service, it requires explanations that are suboptimal since they typically are inclined to be extremely efficient. Owing to this, researchers have led to demonstrate numerous heuristic events for speaking this experiment. Heuristic methods have been responsible for the issues about their greedy characteristics; they incline to be typically stuck inside a local optimal and are incapable to move in the direction of an appropriate explanation. Methods of the Meta-heuristic aspect are extremely Well-organized since they incline to be not dependent on the problems [50]. Different methods grounded on meta-heuristic methods are inclined to use a trade-off exploration and randomization.

Randomization has fetched around providing a temperately attired method for moving away from a native to worldwide hunt. It is inclusive of increasing and solidifying the designs which are meta-heuristic The system of broadening allow distinction for determining the world examination at a worldwide stage by being able to generate various solutions while in specific region intensification search for the optimum solution. Combining these features leads to assuring the accomplishment of optimality at an international level [51]. Numerous systems are considered to utilization for reducing energy ingesting inside a fog server, certain of them to comprise facility based on the allocation of the virtual machine, consolidation, multiple-core architecture-based, thermal-aware-based bio-inspired computation-based methods, and other such aspects. Service-based issues of allocation necessitate policies related to scheduling for diverse systems. Holding the ability to take advantage associated with mobile computing with the storage of resources position at network edges has fetched considering. With the progression of technology and the Internet, organizations all around the globe have estimated that increased providers of cloud come up and present increasingly different purposes of numerous resources and services. Nonetheless, the arrangement of data centers in clouds leads to a growing number of computational devices being used every year, increasing the consumption of energy and depraved pressure on the environment. The studies conducted previously have stressed the reduction of interruptions of transmissions capable to take benefits of distributed abilities of storage at network edges.

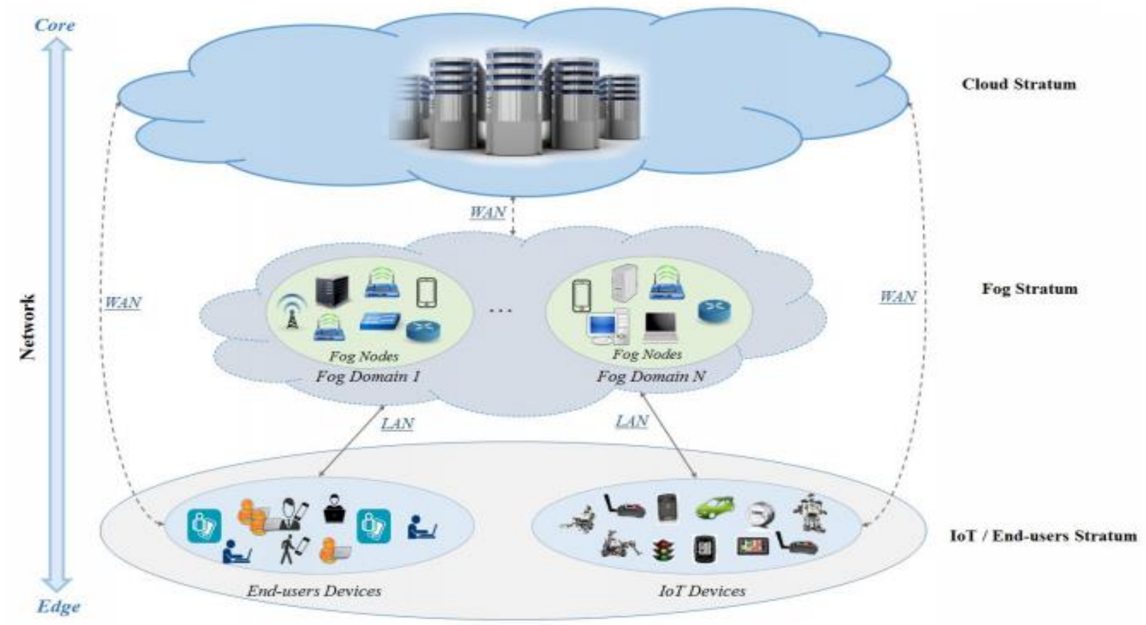


Figure 4.9: Fog Computing Environment

The architecture F-RAN has been presented to offer super low latency services. Alternative work assessed the association of latency-driven jobs in computing the F-RAN networks where the node of F-RAN offloads their job for totaling for the nearest load-free F-RAN nodes. Another research emphasized refining the use of storage or computing across network edges by using Fog computing for effectively dealing through a highly complicated multi-tier network architecture through the means of heterogeneous node abilities and bendable network resources concerning; Communication bandwidth, transmission power, and capacity of storage [52]. Figure 4.10 presents the fog system with an architecture based on three tiers that are IoT, cloud, or end-user and fog layers.

4.2.2 Related Work

For the management of Energy Consumption in data centers, two main techniques are there namely "*VM consolidation technique*" and "*Dynamic Voltage and Frequency Scaling (DVFS) technique*". The DVFS method vigorously regulates the operating frequency of the chip and voltage based on the different requirements of the application program implemented on the chip for computing power. This takes place to attain the determination of saving energy. The three chief groups of DVFS approaches are inter-task [53] interval-based [54] – [56] and treats methods [57]. Interval-based approaches alter the frequency of the processor by forecasting the prospect of CPU usage, inter-task approaches allocate diverse jobs to diverse processor speeds, and intra-task approaches have regulated the frequency of processor based on the program structures. Though DVFS approaches develop the utilization of energy, the scope for optimization still prevails.

The issue of consolidation of VM is an NP-hard problem [58]. Thus, the issue of VM consolidation is many times articulated as a problem of optimization to find a way out which is near optimal. The present VM consolidation methods are bifurcated into two groups that are decision-making based on a statistical analysis of historical data and threshold-based heuristics. The researchers of the study [59] presented a static threshold-based heuristic approach, wherein the researcher fixed two thresholds for the prediction of the host state: an under loaded and overloaded threshold. If the utilization of CPU is equivalent to or less than 10%, then the migration of all the VMs should be facilitated to save power. In this scenario, a PM is under loaded.

In case, the utilization of CPU is equivalent to or more than 90%, then the migration of some

VMs should be practiced so that the next time overloading be ignored. Herein, a PM is overloaded. Many times, a static threshold approach is not appropriate because, CDCs, the amount of work complex and is dynamic. Many pieces of research [60] – [62] recommended two dynamic thresholds-based heuristic policies: "*Inter Quartile Range*" and "*Median Absolute Deviation*".

A neural network framework called PRACTISE [63] is used for forecasting the utilization of resources. The researchers have found that when compared to the basic neural network model and ARIMA [64], PRACTISE has superior precision in prediction. PRACTISE has also been used to understand scheduling VM plan to provide for the highest requirements in other research [65]. Prediction approaches based on the neural network are more complex and are frequently taken into consideration for intermediate to extended-term prediction. Similarly, linear regression [62] facilitates generating an assessed upcoming utilization of resources of a PM from analyzing statistics of its historical resource consumption. It is frequently used for short-term estimate. A "*The research heightened linear regression*" approach is used to estimate the forthcoming capacity of job handling in [66]. Unlike other linear regression methods used for prediction, the proposed robust model compensates the prediction and move towards over-prediction by the accumulation of the fault straight or meandering to the estimation. In [67]the study presented an optimization of the "*ant colony meta-heuristic VM placement algorithm*", the study designed the placement problem of VM as a "*multidimensional bin packing* "issue and marked VM placement plan based on the assignment and the connected resource necessities.

4.2.3 The issue to be addressed

The deficiencies of Fog Computing present the positives of Cloud Computing which provides admission to on-demand for resource computing, which Is physically terminated and chiefly ascendable. Many researchers in the field of Fog Computing have already provided their research based on scheduling techniques but many problems still prevail [68]. The Energy Consumption rate is the first challenge. Owing to the instabilities in scheduling, the consumption of energy is on the higher side. Figure 4.10 presents the SLA for Fog Computing. For checking the energy-conscious job positioning, the approximation of the impression of energy for scheduling a specific job on an accurate server is needed.

The proposed method of FC helps the scheduling which is energy-aware and assets with ANN. Its goal is to decrease the consumption of energy in the fog environment without degrading the system performance and SLA violation. The task allocations are done to energy-aware nodes and as per the task completeness constraints imposed by the users [69]. In the present research, the researcher created an algorithm that was issued for the job list optimization by the consumption of energy rate based on fitness function and combined with (Artificial Neural Network) ANN that facilitates in categorizing the most appropriate host according to the needs of the jobs.

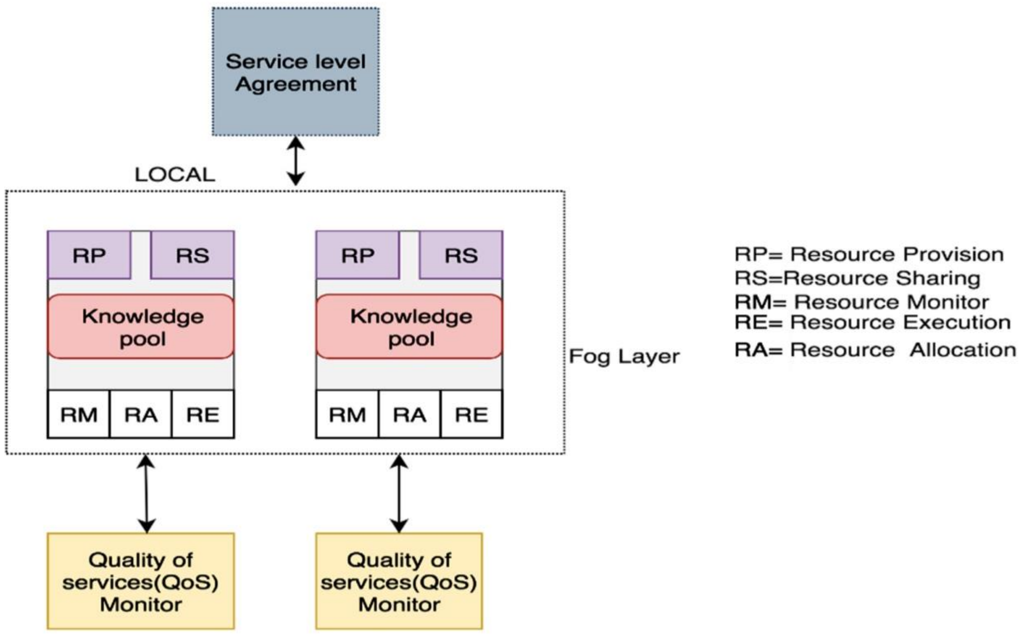


Figure 4.10: SLA in the fog Environment

4.2.4 Proposed Techniques

The method proposed in the present research Fog Computing environment permits, scheduling which is energy-aware of the resources of the system and servers utilizing the idea of ANN along with the altered varied objective scheduling method for the task. In the proposed system, it is taken into consideration to reduce the consumption rate of energy without degrading the performance of the system and a violation of the SLA. This will facilitate in discovering practicable scheduling which improves a group of the objective function. Here, the MMJS algorithm is taken into consideration for optimization of the list of jobs based on the consumption rate of energy grounded on the function based on fitness and give back the optimized job_list and host_list. Optimized job_list and host_list are beneficial for the distribution of jobs with reduced consumption of energy and it behaves as if the input of ANN.ANN facilitates the categorization of the finest host based on the necessities of the task and distributes them for decreasing the greater options of consumption of energy and violation of SLA.

The neural system designed by the researcher has been established to have an exceptionally accurate expectancy ability with reduced overhead to suit the distinctive real-time application, which is applicable in the fog environment. The present algorithm, which is an amalgamation of MMJS with ANN, facilitates discovering job distribution scenarios the fare optimal and facilitates the progression of server regulation usage in Fog Computing scenarios by concluding the unexploited servers. The neural system-generate before has been established to have a remarkably detailed expectation ability with reduces the overhead to suit in exclusive on-going situations. The application of the present neural network is for the sustenance of calculation as to make appropriate decisions, and to make the procedure increasingly easy. Figure 4.11 presented below shows the use of ANN in the distribution of jobs. The algorithms Table 4.5 & Tables 4.6 are present as follows:

Table 4.5: Multi-objective Job Scheduling (MMJS)

```
START
Input: Host_List (HL), Job_List (JL), and Deadline
JL.Sort_Decreasing_ Optimization ()
For
Each job in JL, List do
Min_Power←max
Allocated_ host<--null
If
The host has sufficient resource for Job
Power←estimate_power (H, J)
If
Power<man_power then
Propose___Fitness (HL, Deadline, Job);
If it is fulfilled Fitness_Fxn
                              Optimized__Host←host
End if
End if
Return; Optimized HL & JL
End (Fxn)
End for
```

Output: Optimized HL &JL

Table 4.6: MMJS using Artificial Neural Network (ANN)

```
START
Input: Optimized Host_ List (HL) and Job_ list(JL)
Use ANN to configure Parameters: Neurons (N): Epochs (E)
Performance parameters: Validation_Points; MSE; Gradient; Mutation

Training Techniques: Levenberg Marquardt (Trainlm)
  Data Division: Random
For
Each job and Host
Group (G) =    Work categories according
to host ability
End
Initialized the ANN using Group & Training data
Set training parameters as needed, and train the system
            Net = Train (Group, Jobs,)
Return;
Net as ANN Trained Job Allocation Structure
End (Fxn)
Output: Jobs__Allocation
```

As VM is transported forth on attention to deal with the problems of the client in Fog Computing, the neural network anticipates the upcoming load of requests on servers reliant on documented notice.

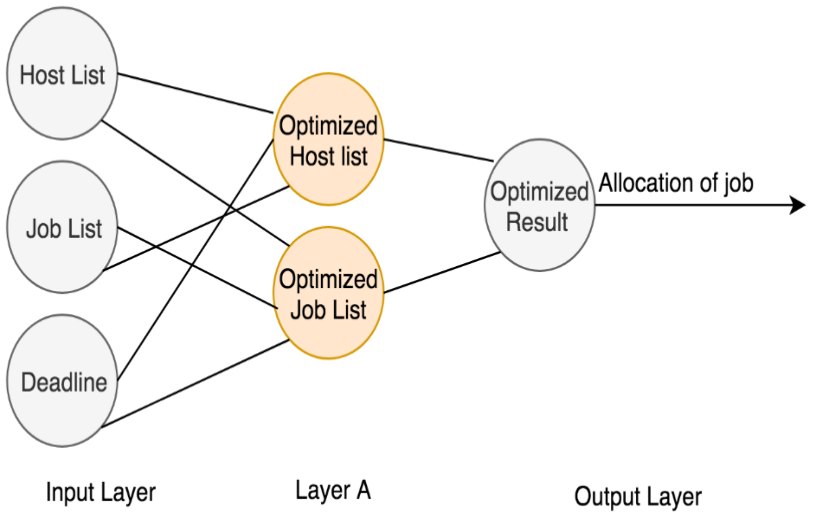


Figure 4.11: ANN used for optimized allocation of jobs

4.2.5 Results and Discussions

The present section is focused on the outcomes attained later in the simulation of the proposed method. For the execution, the researcher has measured ANN and MMJS algorithms. Throughout the process of placement of jobs, the jobs are distributed to the appropriate host following the resources based on ANN and MMJS. MMJS has a function that is multi-objective whose design is taken to be the basis of the consumption rate of energy throughout the distribution of jobs in the direction of hosts. Before this, the tasks are distributed to hosts based on the structure trained via ANN. As per the observation, before the application of ANN and MMJS, consumption of energy and violation of the SLA has reduced. Consumption of energy and violation of the SLA has been processed using the formulas presented below: For simulation 100 jobs have been considering and the simulation area is 1000*1000, simulation is done in MATLAB.

$$SLA\ Violation = \sum_{i=1}^{n} SLA(host, jobs) \qquad (4.2.1)$$

Where,
n= Iteration number

Consumption of energy is Well-defined mathematically via:

$$Energy\ consumption = \sum_{i=1}^{m} job_e + \sum_{i=1}^{l} host_e \qquad (4.2.2)$$

As presented in the equation above,

$job_e = energy\ of\ jobs$

$host_e$=energy of hosts

Table 4.7 and Figure 4.12 presents the attained values of consuming energy in the Fog Computing environment based on the job number calculated with & without the algorithm of classification and optimization. The X-axis and Y-axis respectively present the jobs and value witnessed for energy consumed by the resources.

From Figure 4.12, it is understandable that the job number is increasing the consumption of energy values. 70.2mJ and 66.5 mJ respectively, are average values measured in the number of jobs without and with the optimization algorithm.

Table 4.7: Energy Consumption Evaluation

No. of jobs	EC(mJ)	EC with MMJS and ANN
10	60	55
20	63	60
30	65	59
40	71	70
50	69	65
60	73	71
70	75	70
80	77	73
90	80	77
100	69	65

Table 4.7 and Figure 4.12 presents the attained values of consuming energy in the Fog computing environment based on the job number calculated with & without the algorithm of classification and optimization. The X-axis and Y-axis respectively present the jobs and value witnessed for energy consumed by the resources. From Figure 4.12, it is understandable that the job number is increasing the consumption of energy values. 70.2mJ and 66.5 mJ respectively, are average values measured in the number of jobs without and with the optimization algorithm. Therefore, it is witnessed that the consumption of energy value is reduced by 5.3mJ.

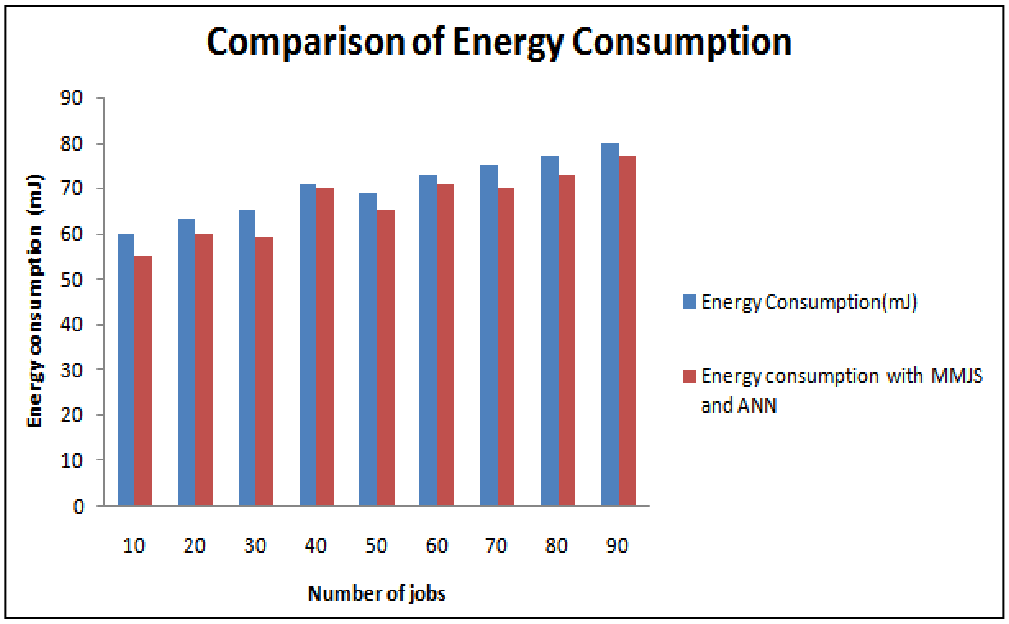


Figure 4.12: Comparison of Energy consumption (mJ) w.r.t the number of jobs

Table 4.8: SLA Violation evaluation

No. of Jobs	SLA Violation	SLA violation with MMJS and ANN
10	0.65	0.63
20	0.82	0.78
30	0.75	0.72
40	0.88	0.82
50	0.83	0.81
60	0.89	0.85
70	0.69	0.68
80	0.62	0.60
90	0.78	0.72
100	0.85	0.81

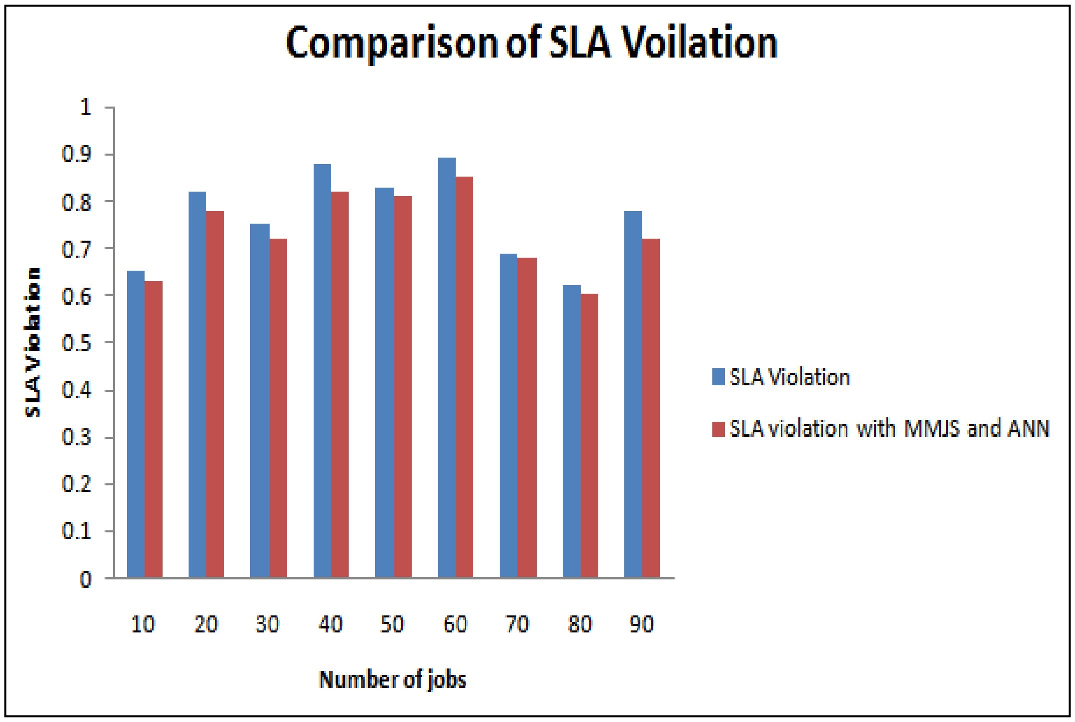


Figure 4.13: Comparison of SLA violation w.r.t the number of jobs

The above Table 4.8 and Figure 4.13 present the violation of SLA in Fog computing received for a dissimilar number of jobs. In the figure above, the blue bar has presented SLA violation without optimization technique and the red bar presents the violation of SLA values detected when ANN and MMJS methods are focused on the system. 0.776 and 0.742 respectively, are the average values witnessed for violation of SLA with and without optimization algorithms. Therefore, in Fog computing a reduction of 3.9 % in SLA violation is observed when an optimization algorithm is applied.

4.3 Summary

In Fog computing, problems of consumption of energy and violation of the SLA for job scheduling have been practiced. However, several ways present for resource management for data centers that are energy aware. The existing researches are focused on decreasing the consumption of energy without taking into consideration the violation of the SLA. Owing to the inconsistency in scheduling, the consumption of energy became more because of a lack of host classifier.

The present research has focused on scheduling of job & resources which are energy aware based on the idea of artificial intelligence. Many jobs are considered for calculating the helpfulness of the proposed system. 70.2mJ and 66.5 mJ respectively, are average values measured in the number of jobs without and with the optimization algorithm. 0.776 and 0.742 respectively, are the average values witnessed for violation of SLA with and without optimization algorithms. Therefore, in FC, a reduction of 3.9 % in SLA violation is observed when an optimization algorithm is applied. It has been established that the performance of the proposed work has led to a reduction in energy consumption.

CHAPTER 5

FOG ENVIRONMENT ISSUES OF LOAD BALANCING AND DELAY AWARE SCHEDULING - A FOUR-TIER ARCHITECTURE SOLUTION

5.1 Introduction

Nowadays, in both fog and cloud environments, quite a lot of studies are trying to dig into the idea of scheduling [70]–[74]. User mobility and application classification are the key issues in scheduling, which is presented by a group of nodes in Fog Computing. [75]. Load Balancing is the main problem in Fog computing because users requesting network resources access is intended for storage & process largely. The resource allocation in fog should consider the present pressure on Fog Nodes; it also cites schedulers and resource managers towards selecting the appropriate Fog Nodes meant for computing services [76]. Moreover, Fog Nodes in Fog Computing Networks vary based on the rate of processing, input data, which causes the issue of load imbalance problem that is few Fog Nodes, are burdened, and few remain idle. Such problems hardly reduce the functioning of the entire fog environment [77]. Consequently, the important aspect associated with fog environment includes issues related to the optimal allocation of workload or dynamic balancing of load along with task scheduling. Hence, because of this, numerous graph partitioning and meta-heuristic algorithms have been put forward [78], [79]. But then again, because of growing task influx rates and mostly deployed figure of the number of Fog Nodes, the Fog Nodes is presently imbalanced. To overwhelm the aforementioned issues, in this study we present the system architecture of four-tier fog computing for both task scheduling and load balancing with reduced energy consumption and delay.

5.2 Related Work

Fog- Node is mainly used in the study of data, collected from the IoT devices within a fog environment. Here the researcher will explain particular research challenges like Load Balancing as well as task scheduling, within IoT applications. [80] In situated an energy-effective scheduling scheme, which is to reduce the energy intake for IoT flows on the diverse form of FC architectures. The SA, in-between edge nodes, and other similar aspects do not take into account the task migration. All such things make the mechanism more complex. Further, periodic tasks are not taken into account because of the tasks, which are a real-time take on the heterogeneous environment. [81] Presented "*knapsack-based scheduling*" improved by search associated with the interdependent organisms that are simulated via simulator iFogSim. It depicts enhancements in the consumption of energy by 18%, the cost of execution by 15%, total usage of the network by 1.17%, and the lifetime of the sensor by 5% within the scheduling. Network with low latency and traffic is appropriate for scheduling based on Knapsack. [82] Proposed containers that are primarily made for the application of smart manufacturing; these are also introduced for task scheduling as well as for the allocation of resources within the fog network. Moreover, the allocation of resources and scheduling of task plans efficiently lowers down the delays in the task. In FN, it also improves the concurrency in many tasks. It needs an appropriate workable environment as Well as virtual devices whenever a user makes a request. The terminal devices along with FN are unable to calculate processing and are given high-grade computation, it happens if unloading of computation tasks occurs. If in the FN rate of traffic network, utilization of resources is high then time for execution of the task will also be high.

In the milieu of fog, [83] introduced application scheduling which was aware of mobility. As the world is moving towards technological advancements, the need for computing of agile users remains increasing. The architecture of FC follows a hierarchical process and decisions about storage locations. These storage locations are dependent on the geo-locations of users and the constraints of the application. There are three types' strategies in the schedule that generated to influence application; these are Delay Priorities Strategies, First Come First Serve (FCFS), and Concurrent Strategy. However, there are many underlying problems of FC like ownership, cloudlet placement, and business model. The authors of the study [84] suggested that the Cloud environment introduced varied heuristic methods for the ideal scheduling of tasks within the environment of the cloud. The suitable choice of an algorithm for scheduling of job is difficult as the methods associated with algorithms are executed in different assumptions. Minimum Execution Time, FCFS, MinMin algorithm, Minimum Completion Time, and Max-Min Suffrage have regularly used task SAs. In the working of Management tactics and allocation of resources, LB plays a major role. In fog aided cloud milieus, power utilization and balanced delay are the limitations within optimum workload allotments.

A study by [85] suggested that the fixed workload allotment issues, it formulated aiming to reduce the utilization of delay in service and power. Moving forward authors stated that if optimization is done within centralized behavior, the total system might fail. Moreover, the essential exchange of data and the extra cost of communication have not been examined appropriately. The research conducted on the study [86] stated that within cloud-fog milieu have exhibited task scheduling. To execute the major offloading applications, the fog provider overuses the association within its hired cloud nodes and its FN. This leads to a heuristic algorithm that is put forward that aims to get the stability within the cost of cloud resource and make span, but it doesn't take into account limitations of execution of the workflow and also the functioning of applications working.

In research conducted by [87], the researchers have used the scheduling of jobs, which are based on our priority, where different components have been suggested. Since modules increase in number, therefore, it lowers down the response time and the overall cost of computation. The SA does not improve if threshold values ($T1,2$) change dynamically. The researchers of the study [88] introduced a unique architecture, which locally handled the user's request. If the requirement of additional resources comes up it is managed by multiple regions, not just by a region. The fog Message module was essential to send off information related to the load of the fellow FN or regions as stated by researchers. [39] That in an environment of FC introduced a well-systemized algorithm meant for Load balancing have stated it. In this to meet users' requirements maintenance of consistency of data with reduced complications is done, surged throughput, utilization of network, and scheduling of tasks are done within the time limit.

Researchers in [89] introduced latency aware application component administration meant for fog milieus. In the fog environment, the focus of this component is to confirm the QoS of applications based on meeting deadlines of delivery of service and adjust resource utilization. A resource scheduling module was introduced by [90] grounded on the mobility of users in FC. Execution requests for IoT applications are redirected to centers of cloud data that are dispersed in the fog layer throughout. The time of response has increased with the help of the rate of mobility of users. In an environment of cloud, a research study [91] introduced the *"Genetic Algorithm".* Both RC-GA helped in increasing the consumption of power of resources as well as upsurges the execution of tasks. Moving on it also, lowers down the errors that are random with the help of "*wheel* *selection of roulette*" an operative of GA, but processing time increases for devices that are at a long distance from the cloud. In Fog Computing for scheduling of task [92] proposed technique of classification of data mining. The job is selected in a way that reduces the time of completion on Fog nodes execution, as authors introduced the "*I-Apriori algorithm*", as well as a set of rules of the association, are produced. It lowers down both time of waiting and time of completion, but a high overhead rate in the environment of fog is there. From ToI equipment to lowers down the latency inflow of users was mainly contributed by in the environment of fog by work Load balancing introduced by [93]. Therefore, to dodge the issues of congestion load in the middle of FN are calculated. For an extended period, important tasks are kept waiting in this; hence, the number in the arrival of the task regarding latency grows largely. To reduce all the above-stated issues, the present study has focused on reducing the consumption of energy and latency.

5.3 The issue to be addressed

The section here focuses on techniques of Load Balancing, task scheduling, and performance metrics. First, the time of scheduling for the execution of the task of the user is defined. Pham et al., [88] introduced quid pro quo problems in the cost of cloud and makespan when in such environment scheduling of large scale application is done. *"Cost-makespan aware scheduling heuristic"* is a SA that aims to obtain the steadiness between the utilization of cloud resources in the execution of the application and working cost, which is fixed as, projected in this thesis, but it is unsuitable for systems, which need to be executed on a very large scale. The utilization of time in both Fog computing and cloud computing is increasing the consumption of time

5.3.1 Problem statement

In this section, we present the proposed problems based on task scheduling and load balancing techniques with performance metrics. Firstly scheduling time for the user's task execution is described. *Pham et al., [97]* presented the tradeoff issue between the makespan and cloud cost when scheduling large-scale applications in such an environment. Hence, this paper proposed a scheduling algorithm called cost-makespan aware scheduling heuristic. The major objective is to obtain the balance between the performance of mandatory cost and application execution for the use of cloud resources, but it is not suitable for large-scale application (smart grid) since the huge volume and ever-increasing service requests. Then an average energy consumption problem is analyzed [94]. A DEBTS (Delay Energy Balanced Task Scheduling) is proposed to minimize the overall energy consumption, which reduces service delay and jitter. Initially, fog node locations, available fog node connections, traffic demands, and channel conditions are changed over time. The centralized controller has taken high-energy consumption and if it is failed, the whole network leads to failure. Thirdly, the response time for several tasks execution is high. In [4], and [98] response time is high when the number of tasks is increasing. In [94] bees life algorithm (BLA) is proposed, which determines an optimal tradeoff between allocated memory and CPU execution time for providing fog computing services for mobile users. This is because it does not consider the dynamic arrival of new requests while other requests are being executed for the fog-computing environment. In [38] graph partitioning is proposed to achieve less response time by load balancing among fog nodes. Fourthly, delay at particular fog nodes for task execution or concerning the number of arrived tasks is high [96], which leads to simple fog architecture. An improved non-dominated sorting genetic algorithm II (NSGA-II) is proposed for fog clusters and nodes.

5.4Proposed System Model

DEBTS is introduced to reduce the total consumption of energy, which lowers down service delays. Primarily, connections of FN, location of FN, the traffic demand, and conditions vary over some time. The consumption of high energy is taken up by a centralized controller if failed it results in the failure of the whole network, in [94], [95] with increment in several tasks, response time increases. To determine the balance between assigned memory and time of execution of the CPU of FC services of movable users "*Bees life algorithm (BLA)"* was introduced in [94]. Due to this fact, frequent and changing requests arrival are not considered but rest are executed. In [90], to get quicker response time between FN through LB was suggested to be done through Graph Partitioning. Moreover, the execution of task lag in a node concerning tasks that arrive is high [96]. For FN and clusters, a better *"Non-dominated Sorting Genetic Algorithm II (NSGA-II)"* was proposed.

5.4.1System model

In this section proposed fog computing architecture for task, scheduling and load balancing techniques are described. The proposed model is depicted in figure 5.1; we consider a set of IoT devices N $(i = 1 \dots n)$, and set of F $(f_i = 1 \dots n)$ fog nodes for the high-performing servers around those IoT devices. From an IoT device, a cloud will be placed, this IoT device proposes many services of a cloud application for the request of the user and that offers computing resources and storage used for the processing of Fog Nodes. To carry out tasks, the group of Fog nodes is clustered. The research motivation is shown in Table 5.1

Table 5.1: Major Research Motivations

Four-Tier Fog Architecture for Delay aware Task Scheduling and Load Balancing		
Tiers	*Tier working nature*	*Algorithm used*
Tier-1 (IoT devices)	Data acquisition	-
Tier-2 (Routers)	Application/task classification (high-priority and low priority)	Dual Fuzzy Logic Algorithm
Tier-3 (Fog nodes)	Fog nodes clustering	K-means++ clustering
	Task scheduling	Earliest Deadline First Scheduling
	Load balancing (current load of fog nodes)	Artificial Neural Network
Tier-4 (Cloud)	Tasks which are executed in longer delay will consider it	Earliest Deadline First Scheduling Algorithm

A cloud will be located far from the IoT devices, which offer several cloud application services for the user request and give computing resources and computation storage for the fog nodes. The IoT devices can access the fog node via a router (wireless network) like Bluetooth or WiFi. Each user request will be classified into high-priority tasks and low-priority tasks using a dual fuzzy logic algorithm. The set of fog nodes (artificial fractals) are clustered to execute tasks. Each fog node consists of three components: fog scheduler, load monitor, and communication component.

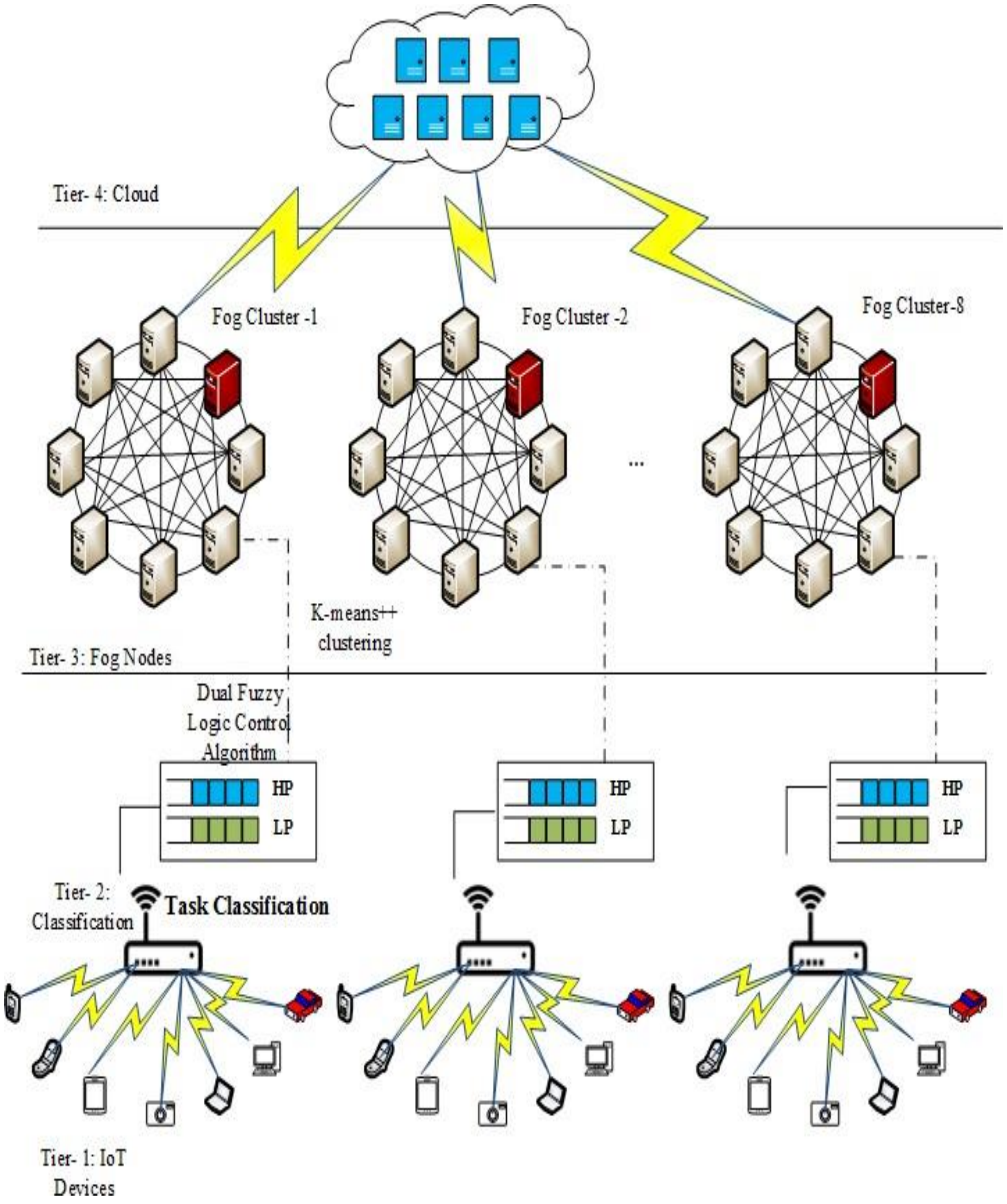


Figure 5.1: Proposed System Model

In Figure 5.1,

$$HP = high\ priority\ tasks\ (class\ 1)$$

$$LP = low\ priority\ tasks\ (class\ 2)$$

Q1 and Q2 are the queues into which users' requests are divided with the help of a *“dual fuzzy logic controller algorithm.”*

The message module, fog scheduler, and load monitor are three components of Fog Nodes. The CDC (tier 4) makes the acceptance or rejection of the user task decision. Every Fog Nodes at the fog. The scheduler plans tasks to achieve the target. The Load Monitor is used for monitoring load in all Fog Nodes. The work states a before user delay, already accepted tasks should be performed. Sometimes Fog Nodes is unable to execute the user task in a given time; this might indicate a lack of resources of Fog Nodes to allot requests. The components of communication transfer its activities once it detects Fog Nodes usage to head Fog Nodes also known as seed nodes. The tasks given by the user cannot be executed if overloading occurs in Fog Nodes usage, in such a situation a *“heartbeat message”* is generated by components of communication to seed nodes so that they reallocate the tasks to the closest Fog Nodes. Algorithms and research used in this thesis are explained in Table 5.1.

Tier-1: It is named *“IoT Devices Tier” and* it is present in the location of physical sensors. Here workload and acquirement of data are rationed out to cloud or else to fog tier This Tier has *“sensor nodes”* and *“actuators”* that are smart IoT devices. These devices grasp data that is geographically spread and sends it to the router.

Tier-2: “*Task Classification using Dual FLA*”: scheduling of jobs is a significant subject in a fog-based cloud environment. This is because, in fog and cloud environments, many numbers of jobs are important to execute. For efficiency, an algorithm for task classification is used and the task is first measured in terms of HP or LP. Tasks that are high in priority are difficult to meet the target even that is given by the user, these tasks are real-time. Figure 5.1 shows the proposed system model. Table 5.2 describes the set of fuzzy rules for task classification.

Table 5.2: Rules for Tasks Classification

	**Fuzz_ Inputs
	Low= L
	High=H**
			**Fuzzy_Output
			Low= L
			High=H**

FLC_1	T_S	T_{AT}	O_1
	L	L	H
	H	L	L
	H	H	L
	L	H	H
FLC_2	**Fuzzy_Inputs**		**Fuzzy_Output**
	$minTET$	$maxTET$	O_2
	L	L	H
	H	L	L
	H	H	L
	L	H	H

In a first fuzzy logic control ($FLC1$) in Table 5.2, factors used

$TS - Task\ size$

$TAT\ -\ Task\ arrival\ time$

In the second Fuzzy Logic Control ($FLC2$) in Table 5.2, factors used:

$minTET = minimum\ task\ execution\ time$

$maxTET = maximum\ task\ execution\ time$

Within all the tasks following tasks are important, these are the execution of the task, minimum size task, time of arrival .O1 and O2 are the outputs as per FLC1 and FLC2. Figure 5.2 shows every set of tasks allocated to every queue.

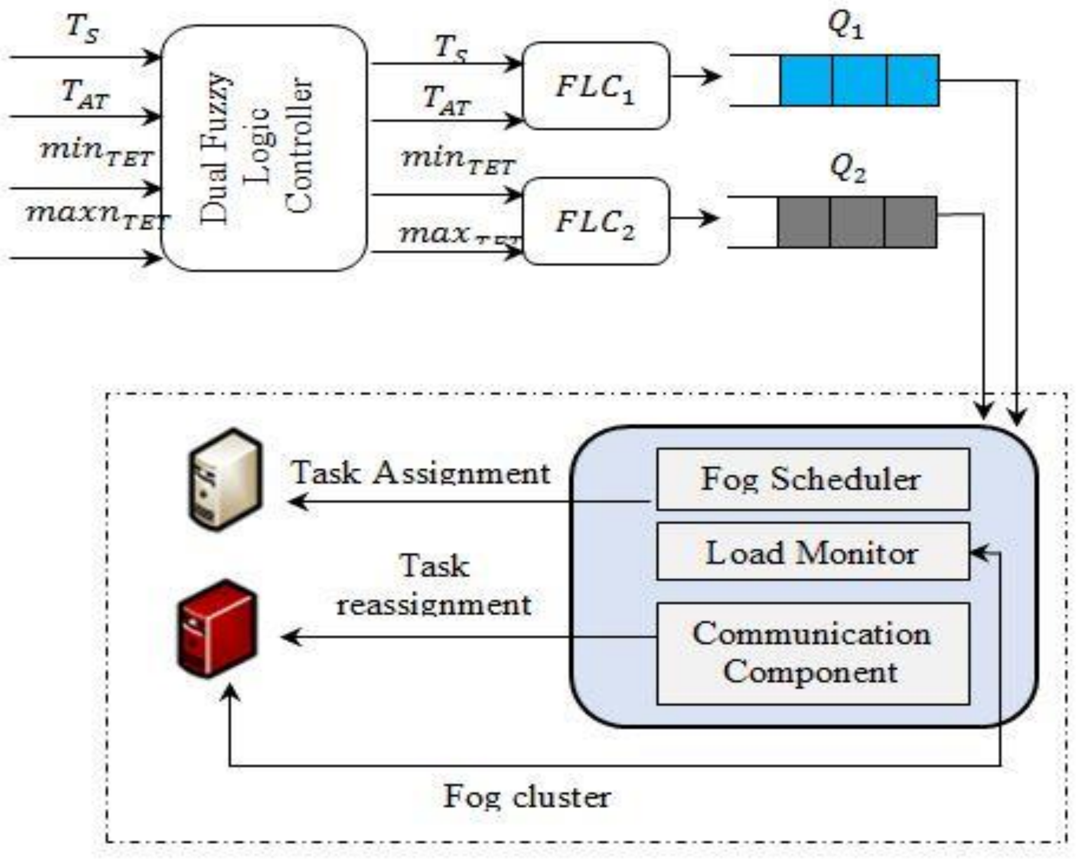


Figure 5.2: Task Classification and Scheduling in Fog Cluster

In an orthodox "*FLA*", the calculative difficulty is more because of many "*fuzzy rules",* the number of requests which are coming does not get updated vigorously as well. Thus, the researcher has suggested the "*dual fuzzy logic controller"* algorithm. The consumer job that has an HP is performed before the job, which has low priority.

Tier-3: Task Scheduling and Load balancing:

Here we will throw some light on the process of scheduling HP jobs amid FN. Here, the researcher suggests using a new framework in the fog tier in place of using a distributed environment or central control like Artificial Fractals. This shall be discussed in detail in the sub-sections. Each cell chooses a seed node to achieve communication among the cells (head for all other FN) to communicate to the external of the nodes. Every FN has three constituents, which are maintained namely Load Monitor, Fog Scheduler, and message module. The use of possessing these modules on every node is to make it accountable for communicating, scheduling, and monitoring. In this thesis, the EDF task scheduling arrangement is suggested in the scheduler. The researcher uses a message module to transfer the associated load data to adjacent FN. The present usage prediction for every node is completed by an ANN. Mobility of FN and user mobility is also observed through assessing the User to FN distance and vice-versa at a time‘t’.

5.4.2 System Design and Architecture

FN clustering which uses K-means++ clustering algorithm-

This arrangement comprises 64 Fog Node as eight series of FN have been considered. These are clustered applications of the algorithm of the K-Means ++ clustering algorithm. There are eight nodes in each fractal and hence eight fractals are formed for 64 FN. The IoT device to the neighboring FN. transmits a requirement. The seed node as per its priority level, deadline in the FN, and available resources, governs the importance of the request. For this, current usage and every FN capacity are periodically updated based on the time interval. To improve the message, each fractal chooses a one-seed node (which is signified in red color). For the grouping of FNs into the layout, adjacent FNs are assembled into a fractal. One configures the fractal scale effortlessly & this framework facilitates informing the fog fractals because the FN constantly converse with contiguous FN with lesser energy intake and high bandwidth. The FN distance is measured using the “*Euclidean-Distance”* method.

Distance between the 2fog fractals (ai=1…n) & (bi=1…n) is calculated as:

$$\|f(a) - f(b)\| = \sqrt{\sum_{i=1}^{n} f(a_i) - f(b_i)^2}$$

$$= \sqrt{(f(a_1) - f(b_1)^2 + \cdots + (f(a_n) - f(b_n)^2} \qquad (5.1)$$

The K-means++ suggested here is to select the cluster center through a new approach. Supposing, X be the shortest distance to the nearest center from a point.

Steps in K-means++ Clustering are mentioned below:

1) Choose1cluster-center ($C1$) arbitrary from R
2) Choose a new cluster-center Ci, selecting $x \in R$ with a probability of $X2 \sum X2\ x \in R$
3) Do until $k$centers C = {$C1$, $C2 \ldots Ck$}
4) From the group of cluster-center Ci, $i \in$ {$1 \ldots k$}, use a “*k-means”* clustering algorithm

EDF based Task Scheduling

Jobs are planned with EDF. The fog fractals communicate with one another with the help of a communication constituent, after combining the fog fractals. With a huge quantity of FN, the first deadline job, which has a smaller job size, gets the maximum priority. In this research, the job level of priority has been transformed and coped while in process, it means the job priority is altered at the time of its execution. The former tactics offered [24] the "*fixed priority scheduling"* algorithm. It was further found ineffective at implementing jobs. If two jobs have similar deadlines, then it randomly selects any job. The algorithm I describe the 'The task scheduling' at Fog node.

ANN-based Load Balancing

Here the ANN algorithm achieves load balancing. Herein, the workload of all Fog Node is disseminated equally by application of back Propagation (BP) Learning Algorithm. The suggested back Propagation Algorithm is easy &operational; however, it will work on nodes if operative training sets are accessible. Algorithm governs present requirements and consequently assigns resources to the user job.

If the Fog node surpasses the preset threshold, it will activate Load Balancing. In this study, the researcher "presented an active "*Load balancing threshold policy"* centered on the Artificial Neural Network. Artificial Neural Network is sent to the fog Node for calculating the existing requirement of fog Node

The primary purpose of recommending this system is its easiness and simplicity to be used for any prediction request. Therefore, in this study workload shared and jobs disseminated amid several Fog Node are assured even when a single Fog node is either idle or overloaded. The main aim of Load Balancing in fog calculating is to decrease the usage of energy. The recommended Artificial Neural Network framework is divided into three layers. Table 5.3 shows the algorithm used in task scheduling.

Table 5.3: Task scheduling using EDF

```
START
Input: Record of FOG _NODE (F→f (i=1…n) ),
Existing User_Task (S)
For all f in F do
Start HP tasks from Fog_Scheduler at first
If
 f_i is available to process S∈ High_Priority
then
Do task execution
Repeat STEP III  until finding the fitting FOG_NODE
Do task execution for  High_Priority tasks
f_i is available to process S∈ Low_Priority then
Do task execution
Else
        Total No. __  Bee = (J*Power_ Jobs);
for
Repeat STEP III until finding the fitting
FOG_NODE
Do task execution for all Low_Priority tasks
End if


Output: Tasks are scheduled to fog nodes and organization task execution
```

Current load Computation:

$$L(f_n) = \frac{\sum_{s=1}^{S} A(T_s)}{T} \tag{5.2}$$

$$Ave_L^i = \frac{\left(\sum_{j=1}^{N}[RET_j(s)+ET(S)]\right)}{N} \tag{5.3}$$

Where S represents the total number of tasks executed at each fog node, $A(T_s)$ represents the task size and T represents the simulation time. The average load of each fractal can be computed by equation 5.3. Where N represents the number of Fog Nodes present in fractal i, $RETj$is the remaining time for presently executing the tasks, which are being administered by node j, and (S) shows all the job implementation time. The hidden layer calculates a weight value as per the Fog node's present; the output layer forecasts the balanced workload between 'N' Fog Nodes.

Dynamic Load Balancing policy: Threshold-based on $AveLi$ is calculated as follows

$$DLB_{Threshold} = L * \ Ave_L^i \qquad (5.4)$$

Where,

L= constant multiplier

Because of the job's (based on EDF) dynamic implementation and plotting of a job to the FN, brink for and LB needs vigorous nature, because static value performs badly in the real-time environment, which has controlled supply.

Tier-4: In *Cloud Data Centers (CDC*s), if fog fractals fail in the implementation of a job, to provide request data packets to users, the CDC behaves like user requirements supervisor it helps in applications demanded by users. If the cloud server performs badly for user requests, then it transmits the communication to the whole CDC.

5.5 Experimental Setup

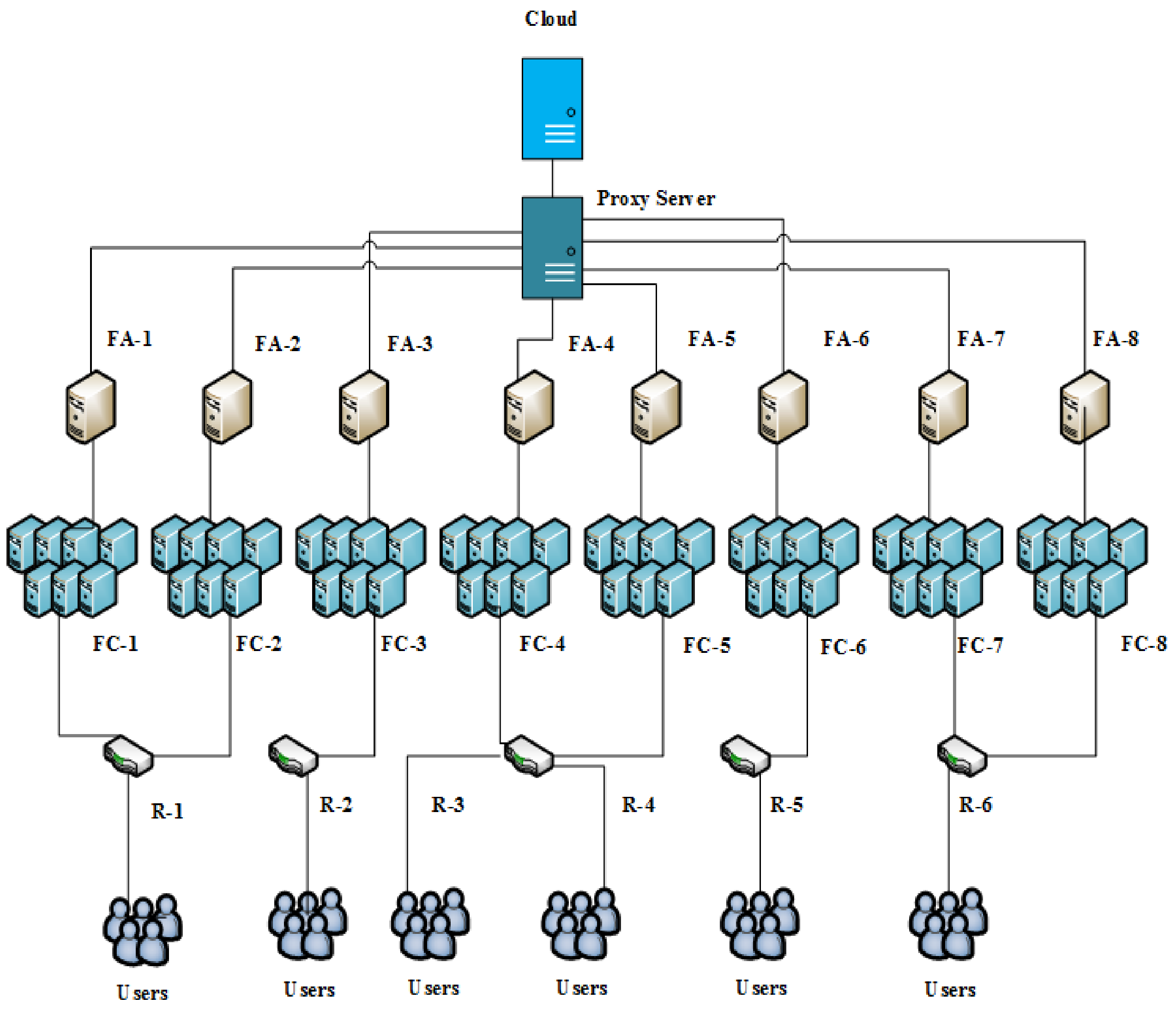
The researcher has developed a Fog Computing architecture thereafter compare it with other Fog Computing environments. The Simulation Environment Design is presented in Table 5.4. One simulation study for application namely, Video Surveillance/Object Tracking. It relies on a set of intelligently distributed cameras that able to track movement. This Application is used for experimental verification and validation

5.5.1 Simulation Model

The researcher used *iFogSim* for the execution of the proposed architecture as shown in Figure 5.3. The *iFog Sim* is selected because it can easily augment with well-established Cloud-Sim Simulators. In our experiment, the system configuration is set with a schedule cycle of 100 milliseconds. There are 64 fog nodes considered for allocating computing resources to users. The 64 users send requests to the cloud concurrently and the data of each data request can be a range of 10000 kb – 20000 kb. The latency is considered for each task, which ranges from 10 – 100 milliseconds (as per the data volume). Figure 5.3 illustrates the iFogSim network topology. In this work, we set up only one fog data center as the user base (UB1) and one datacenter (DC1) in the cloud layer. The fog can select the nearest data center if there are no more processes in the fog layer.

Table 5.4: Simulation Environment

Simulation tool			iFogSim (CloudSim 3.0.3)
System configuration	CPU		Pentium (R) Dual-Core CPU E5700 @ 3.00 GHz
	Topology type		Fully connected
	Memory		2.00GB
	OS		Windows 7 ultimate[x86] -64 bit processor
	Language		Java
	Development kit		JDK 1.8
	IDE		Netbeans 8.0
iFogSim Entities	Number of fog nodes		64 (8*8)
	Number of users		64
	Application module	MIPS	1000
		Memory	10Megabyte
		Bandwidth	1000KBs
		RAM capacity	64GB
	Fog node	Storage capacity	1Gigabyte
		Bandwidth	10000KBS
		Resource cost	3.0
		Memory cost	0.05
		Storage cost	0.001
	Proxy Server	Delay	100ms (between proxy server – cloud)
	Mobile devices	Delay	1ms (between mobiles and the parent fog device)



R-1, 2, 3, 4, 5, 6 - Routers

FC-1, 2, 3, 4, 5, 6, 7, 8 - Fog Clusters

FA-1, 2, 3, 4, 5, 6, 7, 8 - Fog Aggregators

Figure 5.3: .iFogSim Network Topology

In the starting of the algorithm, the researcher described the creation of a Fog Device as presented below in Table 5.5-

Table 5.5: Fog Node Design

```
1: START
2: Construct processor_list;
3: Construct hosts (Input, Virtual Machines, Cost per storage & OS);
4: Construct storage_list;
5: Set parameters of Upper _Bandwidth, Lower_
    Bandwidth, Latency,
6: Request mapping to modules
7: End
```

100 ms is stipulated with system configuration in this experiment. The number of Fog Nodes taken into account for assigning computing resources is 64 to handlers or users, hence 64 requests are made to the cloud alongside.1000kb-2000kb is the range of data in every data request. 10-100 ms per data volume is the range of latency in every task.*" iFogSim network topology*" is shown in Figure 5.4. DC1and UB1 in the cloud uses only a single FN as illustrated in Figure 5.4. The closest data center is picked up in fog in the absence of processes.

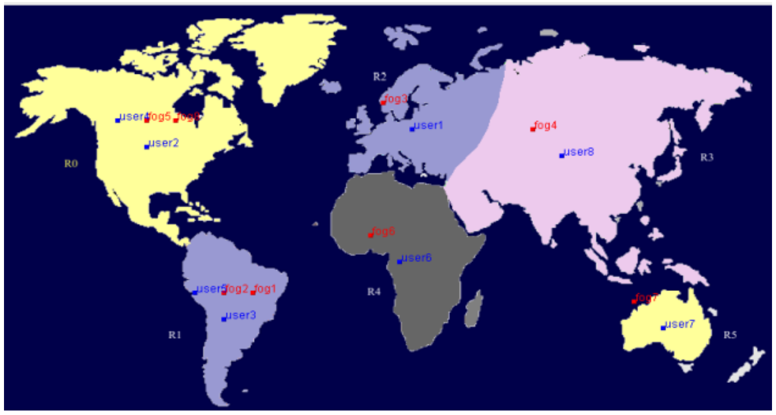


Figure 5.4: UB1 and DC1 in the region

Parameters Definition

Calculations and definitions for parameters are defined as-

a. Response Time: The total time it takes between the requests generated by the user until the appearance of the response. It is the sum of the request time (T_R) and behavior of the processing time (T_{BP}). It is computed by:

$$RT = T_R + T_{BP} \tag{5.5}$$

Where,

b. Scheduling Time:

Also known as, scheduling Length is among the important elements in resource allocation.

It defines start times for each task (S_{TE}) and simulation ending time (S_{ET}).

$$ST = S_{TE} + S_{ET} \tag{5.6}$$

c. Load Balancing Rate:

It is measured by the load distribution among fog nodes to reduce the congestion problem.

It can be estimated by each fog node's current workload and it can be mathematically written as.

$$L = \frac{N_{Ts}}{N_{FNs}} \tag{5.7}$$

where N_{Ts} is the number of tasks and N_{FNs} is the number of fog nodes.

d. Delay:

It is the time duration to execute the whole task assigned to the fog node. In our evaluation tests, we estimated in milliseconds. It is computed by,

$$d = C_T - SS_T \tag{5.8}$$

where C_T represents the current time and SS_T represents the starting time of the simulation.

e. Energy Consumption:

It means the total energy consumed by the whole system. Energy consumption is usually measured by any of the components of the system such as sensors, fog nodes, etc. Kilo Joule (KJ) measures it. It is computed by, where C_{EC} is the current energy consumption, CT is the current time, LUT is the last energy value updated time.

$$Energy = C_{EC} + (CT - LUT) \tag{5.9}$$

5.6 Result and Discussion

5.6.1 Comparison Analysis

Starting, comparison of earlier approaches with introduced fog architecture is done as depicted in table 5.6, and detailed description given ([97], [98], [94], [95], [96], [4], [99], [95], [71]). Accordingly, a comparison has been done. We compare the performance of our proposed fog architecture with previous approaches as shown in Table 5.6, for the most significant features described in the theoretical phase. The previous approach is done in three-tier architecture; we have taken different parameter to overcome the above-said issue. To provide a framework for load balancing and energy consumption in four-tier architecture, then the performance metrics are taken into consideration for comparison.

Table 5.6: Drawbacks of Previous Approaches

Researcher	**Main Feature**	**Drawbacks**
Bitam et al., [94]	Allocated ample time for smartphone users to provide memory and CPU organization.	Does not recognize diverse applications for novel arrivals
Su et al., [96]	NSGAII based algorithm for FG	latency is high
Pham et al., [97]	Systems of large size in a time-constrained setting	• Not suitable for huge size application • Take high scheduling time for execution
Ningning et al., [38]	Graph partitioning for load balancing between nodes in the fog	• Reaction time for task execution is high
Yang et al., [98]	Reduces overall energy consumption and does away with jitter and operation delays	• The centralized guideline have improved energy consumption
Xu et al., [4]	• Distribution and control of the capital in the fog system	• congestion on the network
Chen et al. [99]	Timer Load Balancing implemented to plan properly. • Introduce Scheduling for .NET remote services.	• lead to high execution time
Topcuogluet al. [95]	Heterogeneous Earliest Finish Time (HEFT) is recommended for fast scheduling and control of a critical processor route	• Priority-conscious scheduling
Wang et al. [71]	Multi-population genetic algorithm (MPGA) is proposed, considered for load balancing and task scheduling	• High processing time

a. Comparison of Response Time

A measure of completion time of the request of the user starts from the initial response provided. Comparison between response time and earlier approach with graph partitioning [95], and BLA [94].

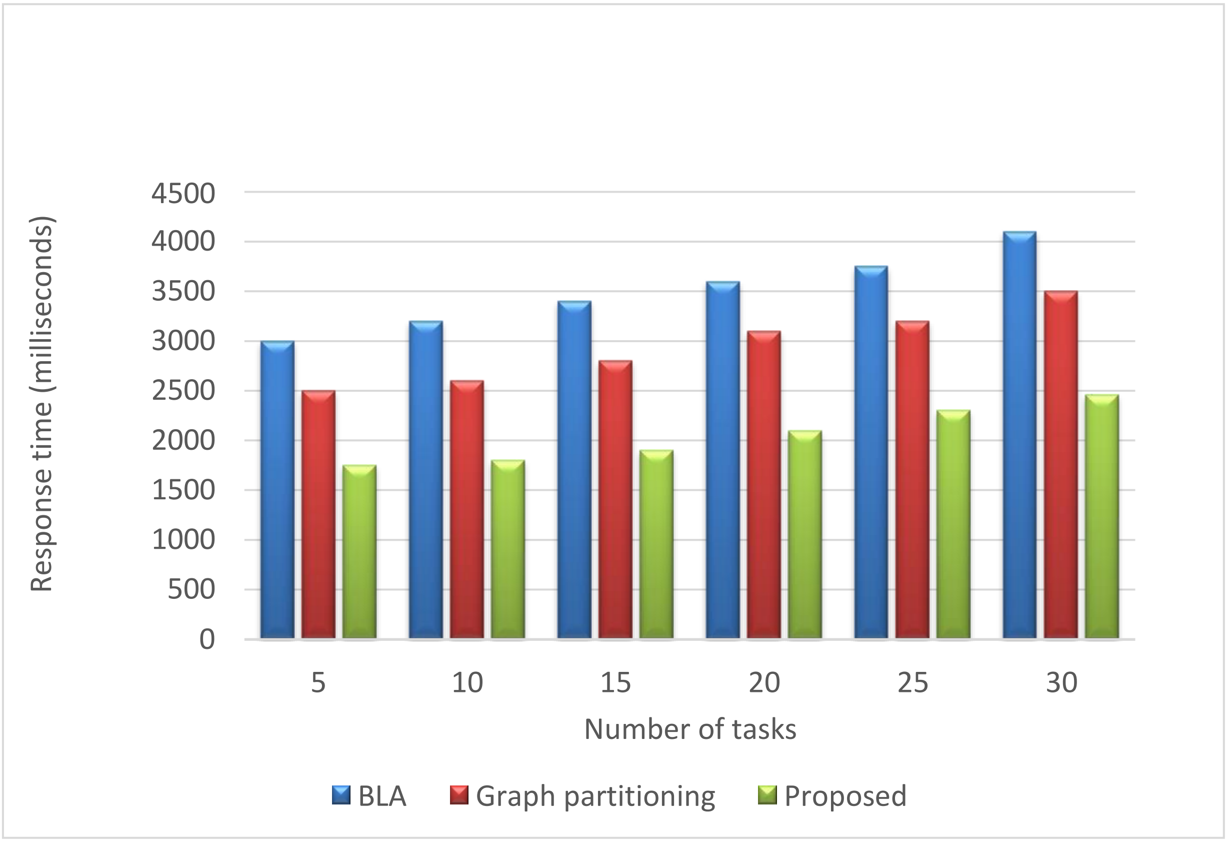


Figure 5.5: Comparison of Response Time

Response time working (Figure 5.5) is affected directly with increased requests and load misbalancing of the load. Portioning of the graph and increased response time occurs because the current FN load is not considered in the BLA [94] algorithm. Say there are many jobs with five tasks each. To complete these 5 tasks 1750 milliseconds is required a reduction of 30% when compared with graph partitioning and a reduction of 45% when compared with BLA. Comparison of scheduling time

Task scheduling issues hamper FC and these issues must be controlled by a decent to decrease the time of execution of time. CMaS [32], against time scheduling are depicted in Figure 5.6.

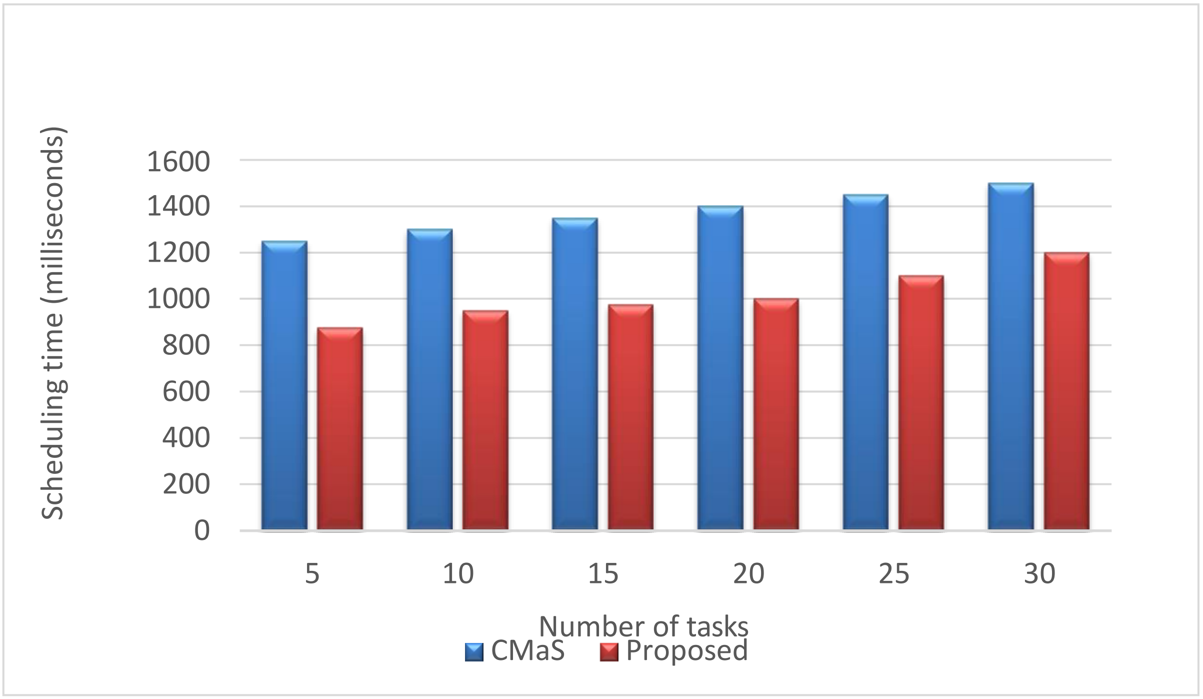


Figure 5.6: Comparison of Scheduling Time

The simulation describes the effect of scheduling tasks with growing tasks in numbers. CMaS is grounded upon the deadline of the task and is "*cost makespan aware SA*". 18.72% times is decreased in this algorithm. For that reason, the fog environment gives improved managing network size when input data is in high numbers.

On a Fog Node, executed tasks are limited in number in the environment of fog. The rate of Load balancing is reduced by linking the varied tasks of the user in a sole Fog Node. Figure 5.7 depicts DRAM with rated Load balancing for the suggested fog environment. The work focuses on the distribution of load throughout the network. However, the total number of jobs on the Fog Node calculates load dissemination amid Fog Node. Job execution is organized effectively because of the administration of the current load. For example, 5 jobs are kept at every Fog Node.

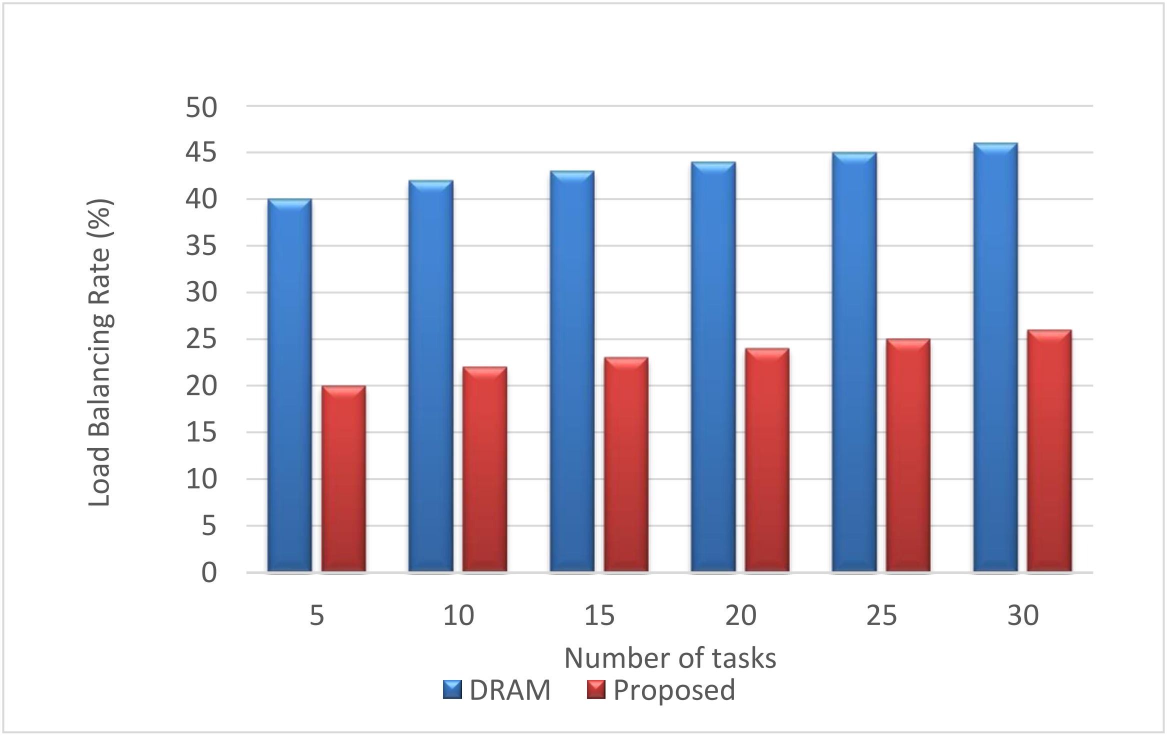


Figure 5.7: Comparison of Load balancing Rate

a. Comparison of Load Balancing Rate

The earlier method is known as the "*Dynamic Resource Allocation Method"* amplified congestion and led to overloading. However, our suggested fog framework calculates present use and takes the help of a load monitor at every Fog Node. If a Fog Node raises its present load by its threshold level ($\leq L$), a heartbeat communication is dispatched to the seed node so that it allocates existing user requests to some other adjacent Fog Node.

b. Comparison in respect to Delay

Fog Computing must speak about the matter of decreasing the total delay and thus increasing reliability along with scalability. In the suggested circumstance of fog architecture, consideration delay is observed for the simulation and authorized for the earlier tactic called, NSGA-II [96]. Figure 5.8 shows the assessment of the delay along with the number of tasks.

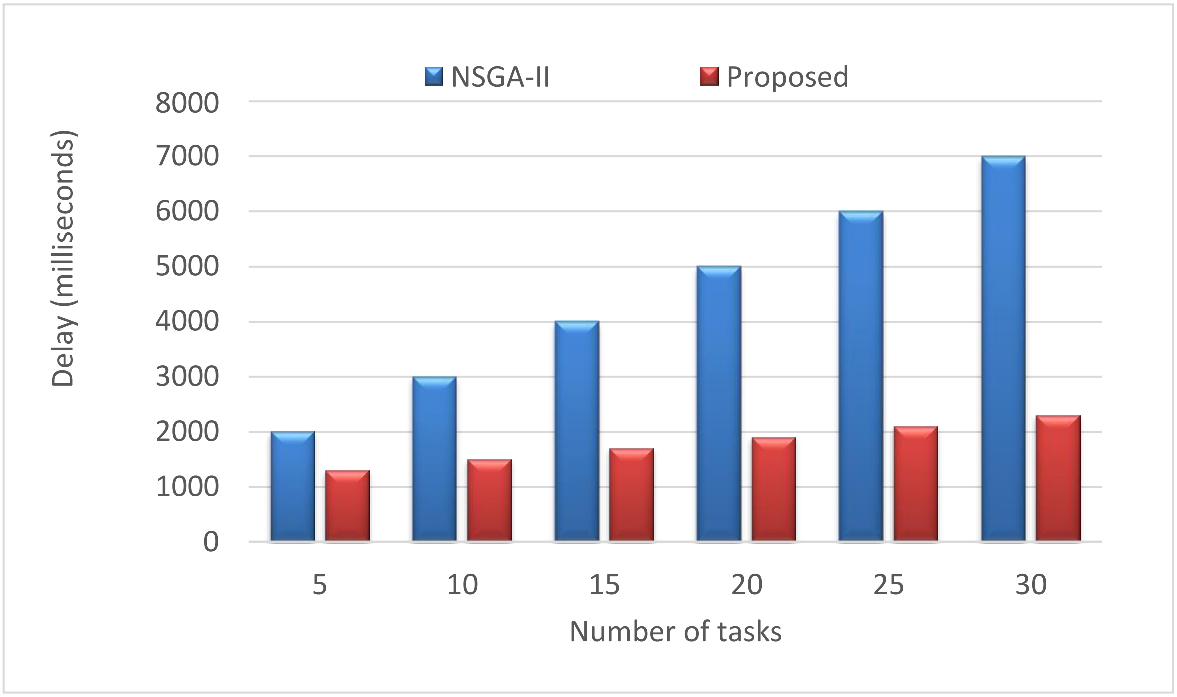


Figure 5.8: Comparison of Delay Vs. Number of Tasks

The number of jobs from users ranges from 5 to 30 milliseconds and for every five jobs, there is a 200-millisecond delay in NSGA-II at the FN, but the recommended fog architecture reduces the delay by 50% for the equal amount of jobs of the same size. Research also suggested complex fog structure i.e. Artificial Fractal with systematized enormous bulk of data. Increasing the number of jobs and job size also increases delays. Figure 5.9 shows the comparison delay in comparison with the number of Fog node. 64 Fog Node is taken in this simulation arrangement. Fog nodes resources, in the NSGA-II algorithm, are sensibly used, but Fog Node failure is a drawback in NSGA-II. The graph (Figure 5.9) clearly shows that increasing the number of Fog Node and increases the delay and it is more in NSGA-II, but the research, delay minutely differs on growing the number of Fog Node.

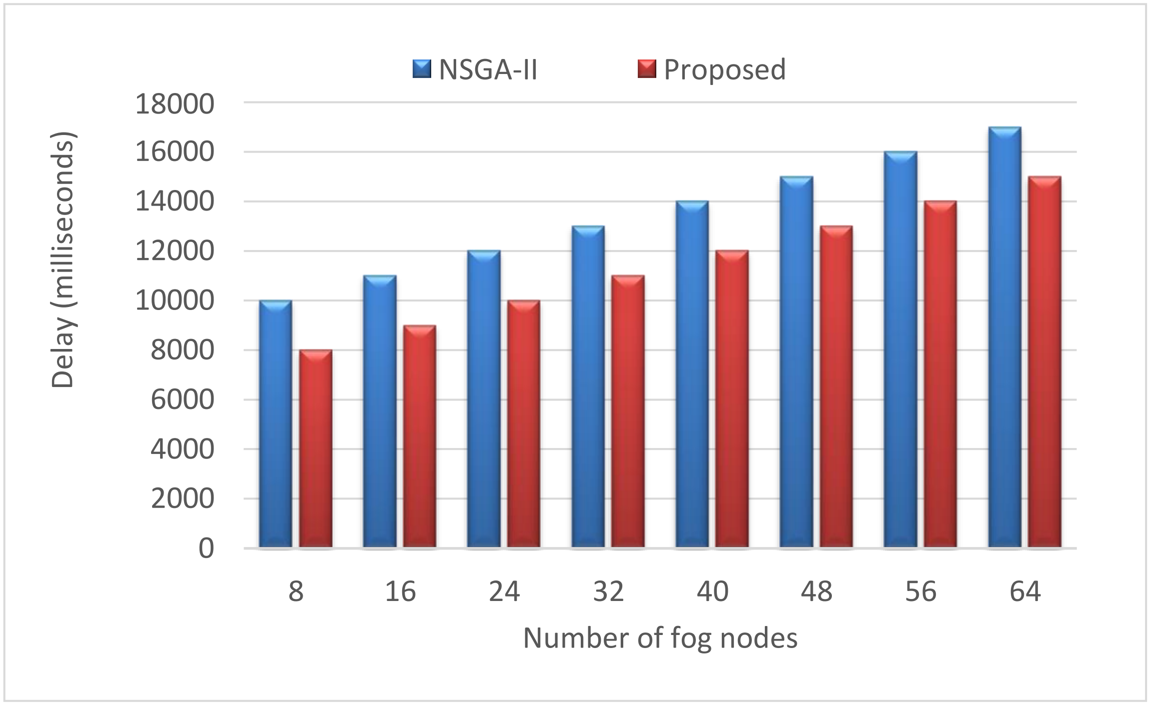


Figure 5.9: Comparison of Delay Vs. Number of FN

c. Comparison of Energy Consumption

For a comparison of Energy Consumption, the researchers measured the total energy taken for job scheduling. However, an effective SA is essential to reduce Energy Consumption. Figure 5.10 shows the Energy Consumption comparison for DEBTS vs. Proposed. The researcher assessed energy intake performance throughout job implementation and job scheduling. Figure 5.10 displays the results gotten by EC with several jobs. The DEBTS raises the Energy Consumption for less quantity of jobs as we use 11000000KJ ($1.1\times104mW$), however, the recommended fog environment uses 5500000KJ. It is because our suggested framework uses a new approach for job scheduling and job implementation. Hence, in the research, FogNode's artificial fractals and present usage are suggested to increase their performance. Hence, it is decided that the recommended research, four-tier fog architecture performed better than the earlier methods in term of decreasing response time (45% & 30%) on comparing to Graph partitioning and BLA, Scheduling time (18.72%) on comparing with CMaS[97], Load Balancing rate (45%) as compared to DRAM Delay (50%) compared to NSGA-II [96], and EC as compare to DEBTS.

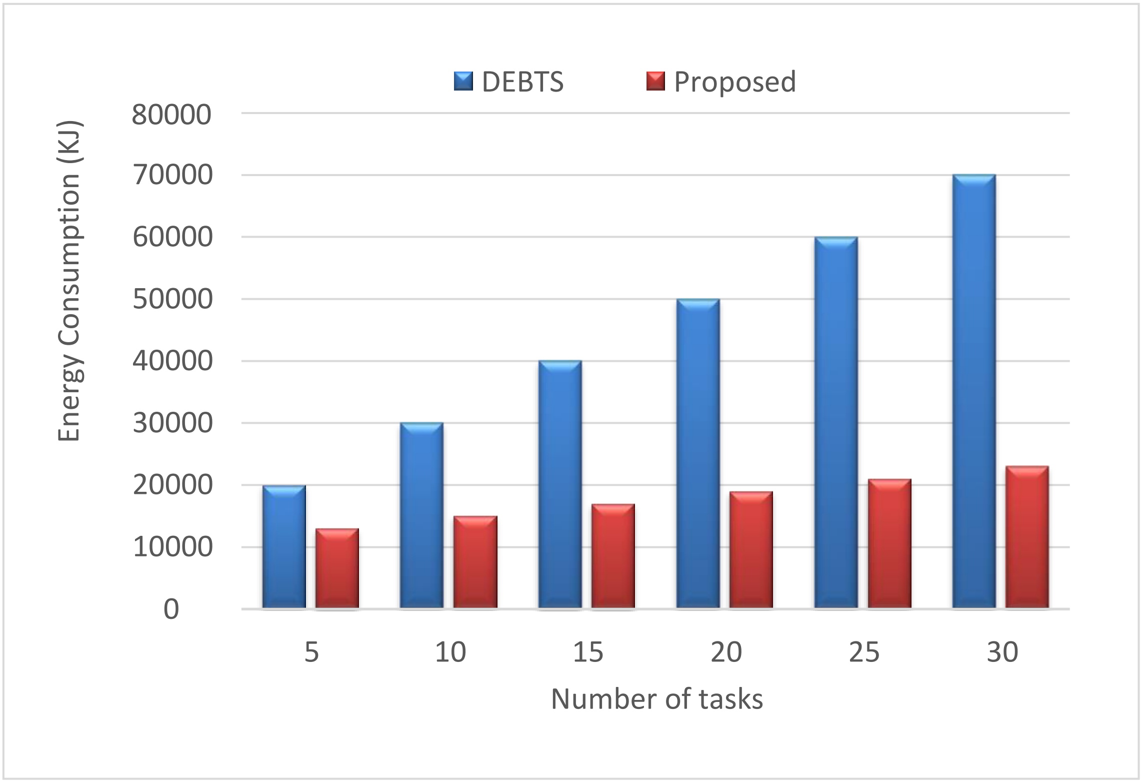


Figure 5.10: Comparison of Energy consumption

5.7Summary

Task scheduling and load balancing are the most significant factors of fog computing which greatly impact the performance of a system. The previous approaches are suffered due to resource shortage, fog node failures, type of environment (centralized or distributed), and so on.

To overwhelm the issues of previous works, we proposed novel four-tier architecture for task scheduling and load balancing. Tier-1 organizes the number of IoT objects and the large volume of data generated and transmitted concurrently. In tier-2, we classified the user applications into high priority and low priority using a dual fuzzy logic algorithm. Then high-prioritized tasks are forwarded to the fog tier. The fog nodes are clustered using the k-means++clustering algorithm. The node with the least loaded, which is near to the user, has considered the priority for task execution. The current load is predicted using an artificial neural network, and the earliest deadline first scheduling algorithm is considered for task scheduling at fog nodes. The user request, which is not executed by the fog node, is executed at the cloud tier. Finally, we evaluate the simulation results for various performance metrics and proved our proposed fog architecture is outperformed the previous approaches.

CHAPTER 6

TASK ALLOCATION USING FOG ASSISTANCE AND SECURE DE-DUPLICATION

6.1 Introduction

In current years, the number of devices connected to IoT is increasing. However, IoT devices sense data and forward them to the respective data centers or end-users. Due to the growth rate of the number of IoT devices, the number of duplicates data during sensing has increased, data deduplication scheme is required in which the redundant or similar data is found out and eliminated. This will improve the storage requirement of the data. IoT devices submit the replicate sensing data that must be eliminated to store it to the cloud server. On the other hand, task allocation is performed via fog computing for several IoT devices. It reduces latency, communication overhead, and communication costs for IoT devices. We jointly consider these two issues in this research. In the present chapter, job distribution and secure de-duplication have been designed that is the elimination of redundant information in computer data for fog-enabled IIoT. In this chapter, the distribution and protected data de-duplication in cluster-based Industrial IoT. This chapter is focused on Fog Assisted Task Allocation and Secure De-duplication by using MOWO in Cluster-based Industrial IoT (IIoT- Industrial Internet of Things) and 2FBO2, which is the third objective of the present research, is to provide.

6.2 Related Work

In [102] authors have suggested a multi-criteria centered decision-making tactic to accomplish job distribution amongst numerous nodes, it is executed on the edge nodes. Here, in peer topology, jobs executed at nodes in a native method are offered. The suggested decision-making structure uses two decisions for ideal job distribution. Because of high latency, energy consumption is also high. In [103] with the help of fog computing, spatial crowdsourcing supported job owners based job distribution and data aggregation has been suggested. The server gathers sensed information from people using mobiles. Collection of data is an explicit job and since it is extracting out enormous spatial crowd sensing data, it has received major attention. To help the server in allocating and collecting data in a restricted privacy fashion, FNs are positioned in various places. Due to ineffective cryptography arrangements, there is a lack of privacy available to mobile users.

In [104], fog-computing technology based on IIoT (Industrial Internet of Things) was offered, which was executed for the smart factory systems. Here, a “hierarchical fog servers” centered placement was raised. In any system first, the “High priority” requests are scheduled as they are based on urgent requirements. In addition, the workload assignment algorithm was taken into consideration to divest a large number of FNs to higher fog tiers. The adaptive configuration of Fog Nodes was applied in [105] over the IIoT atmosphere. It provides a lot of IoT facilities. It enhances the performance of manufacturing organizations. The researcher suggested the application of adaptive FNs configuration. However, these studies are not used for any real-time claim.

In [106] Fog aided IIoT environment suggested smart resource partitioning. First, the authors have used Zipf’s law to calculate the association amid computing control layer’s popularity rank from the IIoT’sdata processing layer. The research was done to demonstrate effective accomplishment concerning delay time, response time as well as response rate. This may lead to huge intricacy because of the large amount of FNs, which were organically disseminated in the fog layer. In [107], the researcher had suggested that latency in healthcare systems based on the IoT architecture can be reduced by the hybrid method. In this method, packets were allocated to various virtual machine processors, which are accessible to the gateway. To provide the right service to any request, indexing is required.

In [108], researchers have suggested an indexing system that is energy-efficient over a fog environment. The system was created for the IIoT environment to give the utmost suitable services to end-users. This index was built on the ground of relationships among users. Here, multi-level indexing was executed along with several indexing operations. This method does not suit the dynamic IoT atmosphere as it has more latency. Just like indexing, recent researches do not focus on security as well. It is crucial for fog enabled cloud environment. In [109], the researcher suggested, which concentrated on creating communication over fog computing, confirming the identity of numerous parties. In this research, the data from healthcare was encoded and uploaded on CS. Time taken to encrypt and decrypt is more. Fog based security structure is employed with intellectual traffic control system framework. This speaks about many security outbreaks such as

Sybil, Denial of Service, and in this computation, time is more.

In Fog based IoT atmosphere, energy proficient cluster-based routing is a crucial aspect that is offered in [110]. The author presented a new P-SEP, which led to a reduction of 8% and 9% usage of energy in FEAR and FECR, consecutively. In addition, 74% of network lifetime is augmented to 83% and 74% in FEAR and FECR consecutively. Due to the selection of the node that has large residual energy as CHs, the clustering process has become ineffective. The optimal allocation of FNs is not executed correctly due to the random selection of neighboring fins. "Adaptive block compressed sensing" was suggested in [111], it is established by implementing an additional "sensor cloud data acquisition" technique in a fog environment. Adaptive block compressed sensing is a method of compressing images, that was suggested in the WSN layer. The disadvantage of this system is that it is very large and complex and hence it is not lightweight. Also due to virtual-clusters creation in the WSN layer causes large energy consumption in fog-nodes. In [112], IIoT is used along with fog enabled cloud framework and various sources produce a huge quantity of data that was produced from various sources and for indexing such diverse data sources, an AVL tree was created in a CS. This led to large query-processing charges and time-consuming to rebalance the tree. To guarantee data privacy, K Nearest Neighbor or secure KNN was suggested however, it is highly costly, thus hence time for data searching is more. In addition, KNN is unsuitable for dense areas and treating huge quantities of data. Figure 6.1 shows the Fog assisted cloud of IIoT.

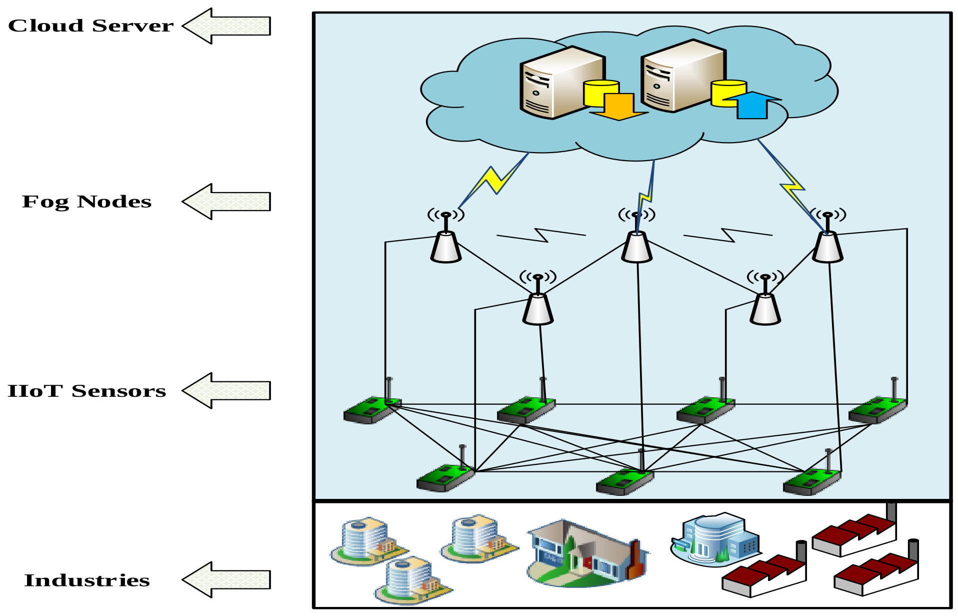


Figure 6.1 Fog assisted cloud of IIoT

6.3 System model

In this section, the researcher shows the system framework for the suggested job distribution and secure de-duplication through FaCIIoT.The suggested structure for air pollution monitoring comprises four layers (Figure 6.1). The description as follows: The main motivation of this objective is based on research challenges underlying fog-enabled IIoT. IoT devices suffer from large latency from the cloud server. Despite mobile crowd sensing, the cloud server does not provide services to users on time. Therefore, our endeavor in this study is to design a task allocation and secure deduplication for fog enabled IIoT, using the concept of novel secure and optimization algorithms. Our proposed system architecture of air pollution monitoring in FaCIIoT estimate several air pollutants such as NO_2,CO_2,SO_2,O_2,$PM_{2,5}$,PM_{10},etc. In this paper, we presented task allocation and secure data deduplication in cluster-based Industrial IoT. The main contributions of this paper are summarized as follows:

- Firstly registers all IoT devices to the cloud server via trusted authority using Elliptic Curve Cryptography (ECC) with Hybrid Multiplier
- Then we cluster similar IoT devices using the Multi-Objective Whale Optimization (MoWo) algorithm based on node residual energy, node degree (no. of neighbor connections), and distance between nodes.
- For secure data deduplication, SHA-3 is presented in which hash values are generated for the verification of data deduplication. After hash generation, CH sends Duplicate Service Check (DSC) and Duplicate Service Response (DSR). Then we select the optimal fog node from the CH using the 2FBO^2 (two fitness-based One-to-One Matching) algorithm.
- Before transmitted to the cloud server, data packets are encrypted using ECC HM private key. Here packets are encrypted using the ECC HM algorithm and decrypted at the proxy server.
- The experimental results show that the proposed scheme is outperformed the previous works based on the QoS metrics: average latency, energy consumption, user satisfaction, network lifetime, and security strength.

6.3.1 System Design and Architecture

The suggested structure framework for the processes is depicted in figure 6.2.

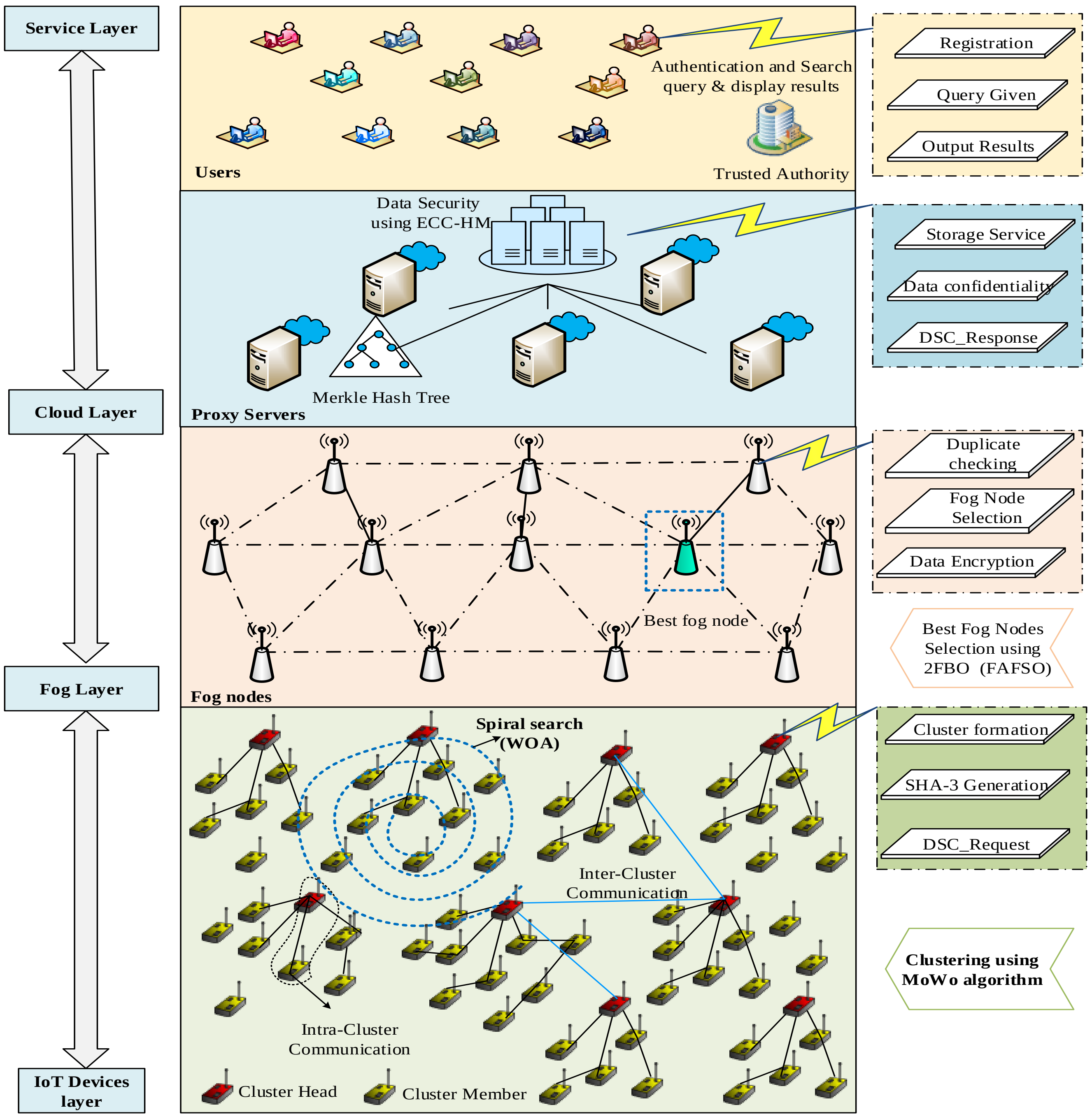


Figure 6.2: System Architecture

A small explanation of layers is presented in the following section-

- **IoT devises layer**

Here, all air pollutants present in the atmosphere are detected by the IoT Devices and by using optimal FNs, this data is transferred to the CS for processing. Herein, sensors for notable air polluting gas such as CO, VOC, SO2, PM, CO2, and NOx are considered.

A. Cluster Head Selection and Formation

a) At first, similar sensors are grouped according to three parameters: node residual energy, node degree, and distance between nodes. Each cluster contains one CH and two or more cluster members. In data transmission, sensors can communicate within one cluster, and CHs can communicate via other CHs. The proposed MoWo clustering algorithm cluster sensors are based on the fitness (optimum) value of three parameters. The whale optimization algorithm is the best optimization algorithm, which outperforms PSO and ACO optimization algorithms. This algorithm utilized multi-objective to solve the optimization problem for clustering. In MoWo, the threshold value is optimized and it is considered for cluster formation. A general procedure for WOA follows:

Prey Encircling: This optimization algorithm is based on Humpback whales' behavior, which is called Search Agents. Whales encircling the prey i.e. Optimal Agent. Here the problem is to determine the best and current solutions that are available for the desired prey. When optimal search agents are identified, all search agents update their positions towards the direction of the optimal solution, which follows:

$$\overrightarrow{DR} = \left|\overrightarrow{c_1} \times \overrightarrow{y''}(T) - \overrightarrow{y'}(T)\right| \qquad (6.1)$$

$$\overrightarrow{y'}(T+1) = \overrightarrow{y''}(T) - \overrightarrow{c_2} \times \overrightarrow{DR} \qquad (6.2)$$

Where $\overrightarrow{DR}$ is the direction for the optimal solution, $\overrightarrow{c_1}$ and $\overrightarrow{c_2}$ are coefficient vectors, T is the iteration number, $\overrightarrow{y'}(T)$ and $\overrightarrow{y''}(T)$ are the current and best position vector respectively. At the end of each simulation, $\overrightarrow{y''}(T)$ is updated to get an optimum solution. Also, $\overrightarrow{c_1}$ and $\overrightarrow{c_2}$ are computed in the following way:

$$\overrightarrow{c_1} = 2\vec{R}_1 \qquad (6.3)$$

$$\overrightarrow{c_2} = 2\vec{A} \times \vec{R}_2 - \vec{A} \qquad (6.4)$$

Where $\vec{R}_1$ is a random vector, which ranges from 0 and 1 and $\vec{A}$ is a linear vector that is minimized in each iteration, starting from 2 to 0.

b) **Exploitation:** It works based on the Bubble-Net Attacking method, which is used two-step processes:

- **Shrinking Encircling Approach:** In this scheme, the value of $\overrightarrow{c_2}$ is randomly selected between -1 and 1. Hence, the search agent new position is determined at any plane (Between the agent's position and the position of the recent optimal agent)
- **Spiral Position Update Approach:** In this scheme, the distance between the search agent and the optimal agent is computed. Then the spiral shape is imitated for Humpback whales shaped movement, which is computed as follows:

$$\overrightarrow{y'}(T+1) = \overrightarrow{Dıstance.}\, e^{\omega r}.\, cos(2\pi r) + \overrightarrow{y''} \qquad (6.5)$$

$$\overrightarrow{y'}(T+1) = \begin{cases} \overrightarrow{y''}(T) - \overrightarrow{c_2} \times \overrightarrow{DR} & if\ p > 0.5 \\ \overrightarrow{Dıstance.}\, e^{\omega r}.\, cos(2\pi r) + \overrightarrow{y''}\ if\ p \geq 0.5 \end{cases} \qquad (6.6)$$

Where $\overrightarrow{Dıstance}$ is the distance between $\overrightarrow{y''}$(T) and $\overrightarrow{y'}(T)$, ω is the constant variable, r is the random number between -1 and 1 and p is the random variable between 0 and 1. Besides, this algorithm updates the search agent position by 50% level of probability to choose the shrinking encircling approach and spiral approach which is defined in Eqn. $\overrightarrow{y'}(T+1)$.

c) **Exploration:** In this step, whales search for prey, and thus randomly chosen search agents will act as the optimal search agent, which is followed by other search agents. The main purpose of this step is to update other search agents based on their current positions. In addition, this uses a random value between the range of -1 and 1. $\overrightarrow{c_2}$ enables the search agents to search and travel far away from the Reference Agent. This step can be written as:

$$\overrightarrow{DR} = \left|\overrightarrow{c_1} \times \overrightarrow{y'}(Rand) - \overrightarrow{y'}(T)\right| \qquad (6.7)$$

$$\overrightarrow{y'}(T+1) = \overrightarrow{y'}(Rand) - \overrightarrow{c_2} \times \overrightarrow{DR} \qquad (6.8)$$

Where $\overrightarrow{y'}(Rand)$ is a random search agent, which is chosen between the set of populations.

Fitness Computation using MoWo

The importance of clustering is to minimize the end-to-end delay between sensors during data transmission and also improve the stability of clusters.

$$fitness = \{\max\_n_{RE}, \max\_n_D, \min\_n_{d-n}\} \qquad (6.9)$$

The sensor node with highn_{RE}, n_D and less n_{d-n} is selected as CH and cluster members are connected to CH according to their values of n_{RE}, n_D and n_{d-n}. The cluster formation is based on the fitness function, which is named$fitness$.

This function is used to given join requests to nodes other than CH. It depends upon certain criteria that are given below:

- **CH_Node_Degree:** An IoT device D_i would join a CH_j, which contains higher n_D than any other CH present in the coverage. Hence, it is expressed as

$$CH_{fitness}(D_i, CH_j) \propto \frac{1}{n_D(CH_j)} \qquad (6.10)$$

- **CH_Residual Energy:** An IoT device D_i would join aCH_j, which contains higher n_{RE} than any other CH present in the coverage. Hence, it is expressed as

$$CH_{fitness}(D_i, CH_j) \propto n_{RE}(CH_j) \qquad (6.11)$$

- **Distance from IoT device to CH:** IoT devices are resource-constrained, which consume less amount of energy. However, a node with a minimum distance to CH will consume small energy. Hence, it is expressed as:

$$CH_{fitness}(D_i, CH_j) \propto \frac{1}{d(D_i, CH_j)} \qquad (6.12)$$

The aforementioned equations are combined to get the final $CH_{fitness}$ and therefore it can be written as:

$$CH_{fitness}(D_i, CH_j) \propto \frac{n_{RE}(CH_j)}{d(D_i, CH_j) \times n_D(CH_j)} \qquad (6.13)$$

$$CH_{fitness}(D_i, CH_j) = \tau \times \frac{n_{RE}(CH_j)}{d(D_i, CH_j) \times n_D(CH_j)} \qquad (6.14)$$

Where τ is a constant value (τ =1). Eqn(14) is used in cluster formation, which is based on the device $CH_{fitness}$ and CH sends join requests to nodes having the highest weight value. The algorithm for CH selection and formation is depicted in the following. The related algorithm is presented below in Table 6.1:

Table 6.1: Algorithm for Clustering using MoWo

```
STEP I   BEGIN

STEP 2   START (NTIALIZATION)
                 2.1 Initialize: Search Agent
                     Assign: Random Vector
                 2.2 Initialize (c_1 ) , (c_2 ) , 〖MAX〗 _iterations&A ,
                 2.3End  (INTIALIZATION)
STEP 3   START (EVALUATION)
         3.1 Calculate fitness value using (6.9)
          3.2 An optimal agent (CH) is setting the best search agent
         3.3 End (EVALUATION)
STEP 4   START (UPDATE)
                   4.1 Update Parameter :
                    4.2 Update Parameter :(c_1 ) , (c_2 ) , A ,R_1, r&p
                     4.3 Execute : Exploration & exploitation to
                        the Search agents

                     4.4Adjust Threshold
STEP 5   Continue step 3 and 4
         Until reach max. Iterations
STEP 6   Reiterate 2 & 5 until we get the best_Solution
STEP 7   END
```

comprises of multiple cluster members and CH. In data broadcast, sensors interconnect within one-cluster and similarly, CH connects to other Cluster Heads. The Whale Optimization Algorithm (WOA) is the best algorithm for optimization. WOA. WOA with multi-objective (MOWO) is utilized to handle the clustering optimization issue. The augmented threshold value is used by MOWO for the Clustering process is dynamic, but we have not encouraged the frequent clustering process. Therefore, it considers the new arrival of requests and assigns the computing resources.

B. ECC based Hybrid Multiplier

Initially, the cryptography algorithm is used for data packets encryption, which relatively less time-consuming process and highly efficient for organizes real-time data processing. In this work, we proposed an ECC-HM algorithm is proposed for packet encryption. ECC-based HM performs than conventional ECC. In conventional ECC, multiplication is very slow and requires many computational resources since, in conventional ECC, multiplication is frequently used. One of the advantages of our proposed ECC based HM speeds up and more favorable for large numbers. This ECC based HM algorithm follows the procedure of two Vedic Sutras such as Nikhilam, UrdhvaTirakbhyam as well as Karastuba. These are used to perform unique multiplication for large size of numbers. In ECC elliptic curves are divided into two fields: Prime Field and Binary Field and these representations are followed:

- **Prime field:** Elliptic curve of this field is written by

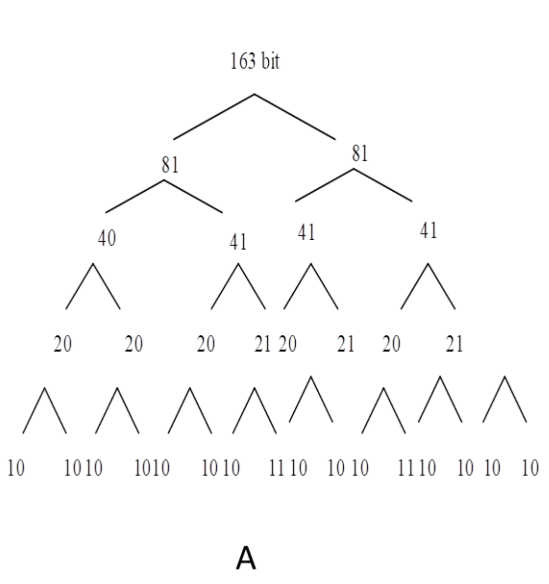
  $$Y^2\% \, P = (X^3 + AX + B)\%P \tag{6.15}$$

  Where A and B are the parameters, X and Y are the curve points.

- **Binary field:** Elliptic curve of this field is written by

  $$Y^2 + XY = (X^3 + AX^2 + B) \tag{6.16}$$

Both prime and binary fields are optimal for the field of secure system designs, which outperforms in terms of speed, but the binary field-based elliptic curve has given a better performance without consideration of Carrying and Modular Reduction. The mapping process of the data packet to elliptic curve points ensures data confidentiality. The ECC based HM encryption invokes several scalar/double multiplications and modulo consumes more power and area. Therefore, the delay increases gradually and the computations are minimized using multiplication, which consumes less resource and utilized to offer maximum frequency and speed. Let us take an example of the Karastuba Hybrid Multiplier for Binary Field (163bits) and Prime Field (192bits). . For example, consider multiplier in Figure 6.3.



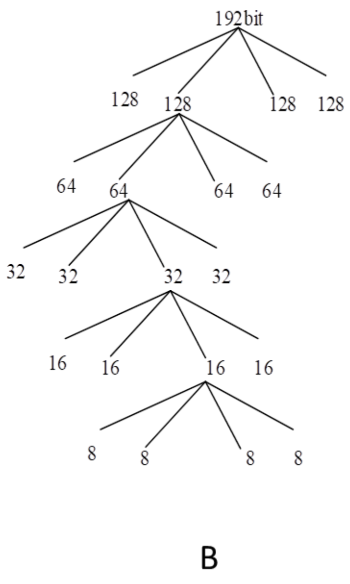


Figure 6.3: Karastuba Hybrid Multiplier (A) 160bits and (B) 192bits

- Secure De-duplication

Presently, SHA-3 has become a significant algorithm for data duplication verification. After the nodes are authorized to TA for CH selection and the formation of the cluster, SHA-3 is utilized to jumble combined data of CH for the data duplication verification. The integrity of data is maintained while transferring the data packets from the fog-nodes to the CS. The input of the data packet of SHA-3 is of random lengths &the output is consists of hashed values or digested messages. Consider Table 6.2 for hash generation.

Table 6.2: Properties of SHA-3

SHA3(512bits)	
Parameters	*Variants*
Block size (bits)	576
Capacity	1024
Word size (bits)	64
Rounds	24
Operations	AND, OR, Rot, and Not
Security strength	256
Output size (bits)	512

After hash generation, DSC_request is sent by GH to an adjacent and quality FN. FN then confirms if this data packet has been collected or not. If packets are stocked in momentary storage of FNs, the DSC_response will be sent immediately. After that, the files are stored in CH through FNs.

- Fog Layer

In the Fog layer, CH selects the node, which is nearest to the FN by using 2FBO^2, it uses the process of the Fast Artificial Fish Swarm Optimization (FASWO) algorithm. This algorithm is has taken its inspiration from the collective movement of fish and their different behaviors. The primary usage of FASWO is because of its tolerance parameter setting, global searchability, and good robustness. This algorithm is a prolonged form of AFSA. In orthodox AFSA, the speed with which determines the finest solution is a difficult job, and therefore here, the speed improves by suggesting FASWO. The overall behavior of AFSA is as given: af_prey, af_swarm, af_follow, and af_move. To increase the speed of AFSA to get improved results quickly: The research takes into consideration two features given below:

Brownian motion:

It is centered around the *"normal distribution*" and it's defined through a real-valued stochastic process.

Levy Flight:

It is centered on the "*random walk procedure"*

Fitness Computation using FASWO

The primary metrics used for job distribution are Task Length & Processing Delay. In many previous works, authors had selected only above-mentioned metrics for selecting the fog-node, however, none had considered the characteristics of FNs. In this study, selecting the idle FN is grounded on two fitness values calculated as:

$fitness\ F_1 = \{\min\_T_L, \min\ \_T_{PD}\}$ (6.17)

$fitness\ F_2 = \{\max\ \_N_{RE}, \min\ \_N_{d\,(FN_i,CH_j)}\}$ (6.18)

Hence the sum of two fitness functions (2FBO^2) is calculated as:

$$F = \sum F_1 + F_2 \tag{6.19}$$

Table 6.3: Best Fog node selection using FASWO

```
I    BEGIN

2    START (INITIALIZATION)
       2.1 Initialize $AF\_total$,$AF\_step$,$AF\_delta$,$AF\_visual$,$AF\_best$
       2.2 Initialize:$Iteration\_Time$, Try_Number
       2.3 End(INITIALIZATION)

3    START (EVALUATION)
   Calculate fitness_AF using (6.19)
   Fix the best AF is a finest_solution (Best Fog_Node)
   End (EVALUATION)

4    START (UPDATE)
      4.1 Update Parameter: AFs → $AF\_best$

5    Continue Step 2
     Until reach max. Iterations
6      if not ;
       end the process;
       and return_output
7    END
```

The algorithm shown above is using FAFSWO for 2FBO^2, which shows for the job given by the device CH_j, how the idle fog-node is selected for its execution Best fog node selection is done using the algorithm shown in table 6.3.

- Cloud Layer

CSs are positioned after the Proxy server. For cataloging the data communicated by various IoT devices, the researcher provided a "*Merkle hash-tree*" (MHT). The Proxy-server provides the primary node installed amid the CS and FN. The data searching and data storage and recovery are placed based on the user query in the proxy server. The new MHT index technique is the basis of the operation of the proxy server. It is useful to decrease the calculative and communication overheads among proxy servers and IoT devices. As mentioned above, data communicated by various IoT devices will also be cataloged from the Root-node towards all leaves of internal-nodes. The MHT internal-nodes are symbolized as *i* and *j* which are left and right index trees respectively. The mathematical model for MHT follows:

$$M = RH\,(In\,(i) * L\,(j)) \qquad (6.20)$$

Where i=1,2,…n-1,j=1,2,…j-1, L is the leaves of the internal node, In is the internal node and RH is the root node.

Figure 6.4 shows the graphic depiction for "*Merkle hash tree*",

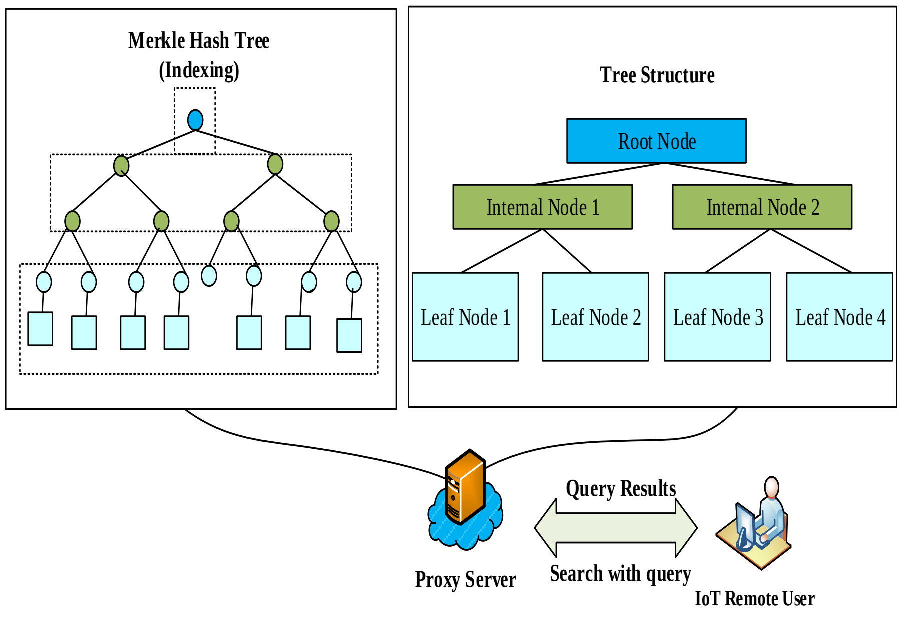


Figure 6.4: Merle Hash Tree for Indexing

- Services Layer

  IIoT data are recovered here post confirmation of the user's certification. If a user has registered successfully in TA, then the searching outcome is given for users. To increase the speed of searching, in this study the researcher creates a searchable index with the help of MHT.

6.4 Experimental Results and Discussion

Here, covers the simulation of the above-mentioned System Model shown in table 6.4. Subsequently, the performance of the above-mentioned proposed schemes is discussed by using numerous metrics.

6.4.1 Simulation environment

The simulation of the suggested system is executed using iFogSim. It is a Java-based, open-source network simulator. iFogSim's works to simulate the surrounding that consists of a vast amount of IoT devices and FN. The simulation topology in figure 6.5

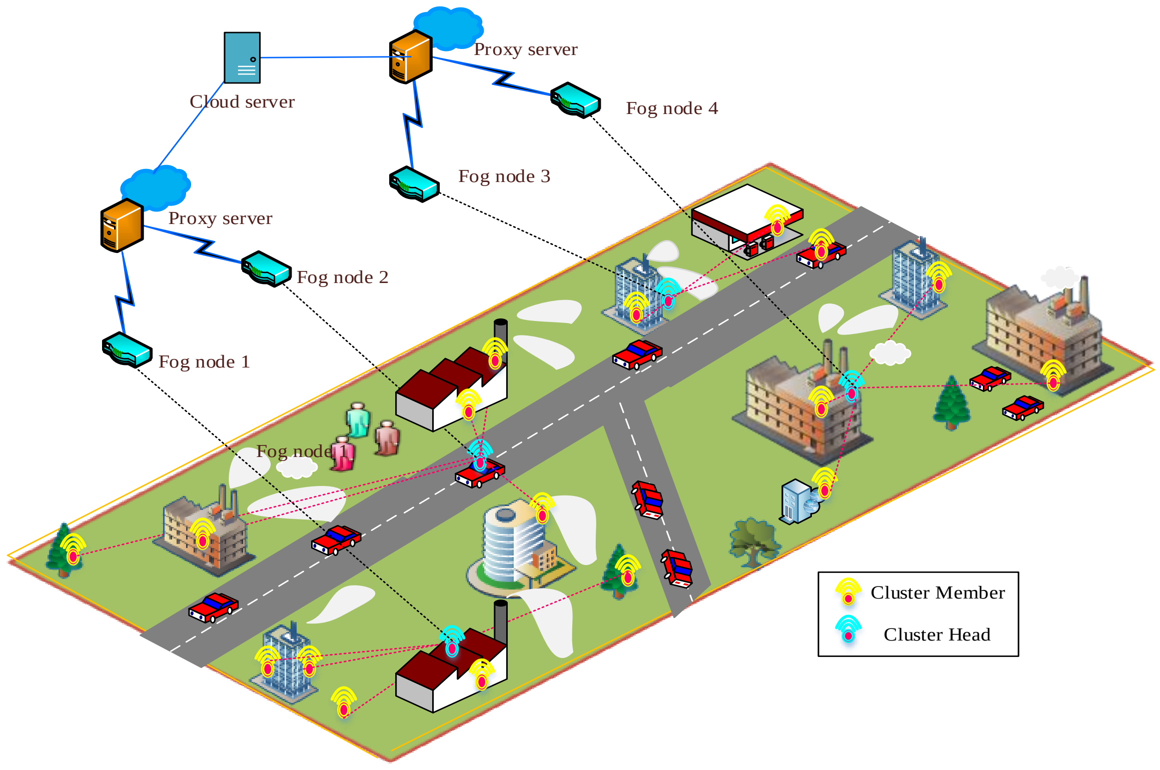


Figure 6.5 Simulation topology

Table 6.4: Simulation Settings

Simulation tool				**iFogSim (CloudSim 3.0.3)**
System configuration	CPU			Pentium (R) Dual-Core CPU E5700 @ 3.00 GHz
	Topology type			Fully connected
	Memory			4.00GB
	OS			Windows 7 ultimate[x86] -64 bit processor
	Language			Java
	Development kit			JDK 1.8
	IDE			Netbeans 8.0
iFogSim Configuration	# of fog nodes			5
	# of IoT devices			20
	Application module	MIPS		1000
		Memory		10Megabyte
		Bandwidth		1000KBs
		RAM capacity		10
	Fog node	Storage capacity		1Gigabyte
		MIPS		2800
		RAM (GB)		4
		Bandwidth	Uplink	10000
			Downlink	10000
		Resource cost		3.0
		Memory cost		0.05
		Storage cost		0.001
	Proxy Server	Delay		100ms (between proxy server – cloud)
		MIPS		2800
		RAM (GB)		4
		Bandwidth	Uplink	10000
			Downlink	10000
	Cloud Server	Delay		1000ms and longer (between devices)
		MIPS		44800
		RAM (GB)		40
		Bandwidth	Uplink	100
			Downlink	10000
	IoT Devices	Delay		1ms (between IoT sensors to fog devices)
		MIPS		1500
		RAM (GB)		2
		Bandwidth	Uplink	10000
			Downlink	10000
MoWo	Chosen Optimization Parameters	Population size I		50
		$MAX_{iteration}$		500
		Spiral constant ω		1.5
		Distance parameter α		0.3
		Energy parameter β		0.3
		Density parameter γ		0.3
FASWO	Chosen Optimization Parameters	AF_total		20
		AF_visual		2.5
		AF_step, AF_delta		0.7, and 8
		Try number		50

6.4.2 Case Study

The proposed scheme is tested over the Air Pollution Monitoring application. As presently a lot of businesses and automobiles are present closer to the city and they emit a large number of air toxins, there occurs an issue of data redundancy and hence the need to observe and govern the data duplicity which is transferred over the fog network. Repetition is taking place currently in many areas, which causes poor QoS and leads to stark issues in CSs. Air Pollution is injurious to the people's health as well as to the environment, hence it must be tackled for the people and the environment benefit.

However, presently Air Pollution monitoring is stimulated and major jobs are executed in IIoT which are fog enabled. Sensors are installed in this IIoT platform to monitor pollution in the air. Consider Figure 6.4. In this research work, air pollutants are categorized in the 3 classes as mentioned below:

- Primary air pollutants.
- Secondary air pollutants:
- **Others**

Table 6.5 presents a list of IoT sensors used here for measuring air pollution in industrial area. Functionalities of each sensor are illustrated.

Table 6.5: Sensors and Functionalities

Sensor type	Functionality
MQ-135 gas sensor	For measuring air quality
MQ-2 gas sensor	To detect CO, Alcohol, Smoke/Propane, H2, LPG, and CH4
MQ-7 gas sensor	Detecting CO, and suited sensing concentrations CO in the air
MQ-3 gas sensor	Detects Benzine, Hexane, CO, Alcohol
Buzzer sensor	For giving an alarm to inform about Unhealthy Air or Exceeding chemical values on each sensor
LM-35 sensor	For measuring temperature inputs
MiCS 4514	For measuring NO_2 and CO
MiCS 2614	For measuring O_3
DHT11	For measuring Humidity and Temperature

6.5 Result and Discussion

6.5.1 Evaluation measures

Average latency, user satisfaction, network lifetime, network lifetime, energy consumption, and security strength are mainly concerned with the combination of IoT, fog, and cloud computing paradigms. The definition of these important measures can be following.

(i). Average Latency: It is the time required to respond to the user's given request at a time. The average latency is defined as the sum of time taken to process all requests given by the IoT device. It is written by:

A_l= (min+max)/2 (6.21)

Where A_l is the average latency and its unit is milliseconds (ms). It is computed at a minimum and maximum amount of time. The minimum time is zero and the maximum time is the time required for processing a single request.

(ii). User Satisfaction: It is a metric that finds how well a service response, from the fog/cloud will satisfy the user's requirement. It is not the same for users with requests for specific services. Hence, it differs based on the user's service request, arrival time and distance to the fog/cloud servers. It is written by:

$U_s = S_{RT} + S_{IT} + S_Q$ (6.22)

Where U_s is user satisfaction, S_{RT} is the service response time, S_{IT} is the service-initiated time, and S_Q is the service quality.

(iii). Network Lifetime: It reflects the performance of the entire designed architecture. These metrics are affected by heavy energy consumption at both IoT and fog nodes. It is the time duration of how long the network will active for sensing, data collection, and transmission. It is written by:

collection and transmission. It is written by:

$N_{LT} = \sum_{i=1}^{n} B_i / C_i$ (6.23)

Where $N_{LT}$is the network lifetime, $\sum_{i=1}^{n} B_i$ is the life of device I, and C_i is the coverage of device i.

(iv). Energy Consumption: It is defined as the amount of energy consumed at IoT devices to collect information and making communications to other devices. Energy consumption for IoT device to transmit the 1-bit message at distance D is computed as:

$\varepsilon_{TX}(L, D) = E_{TX-Elec}(L) + E_{TX-AMP}(L, D)$ (6.24)

Where $\varepsilon_{TX}(L, D)$ is the energy consumption for message transmission at size L with distance D

(v). Security Strength: It is essential to analyze the designed system for task allocation and deduplication. It ensures user satisfaction and is supported for massive data storage at cloud servers. It is computed by the following.

$$S_{st} = K_s + M_s + E_t + D_t \quad (6.25)$$

Where $S_{st} =$ is the security strength, K_s is the key size, M_s is the message size, E_t is the encryption time, and D_t is the decryption time.

6.5.2 Comparative Analysis

As earlier mentioned, different performance evaluation measures are evaluated and compared with four previous works, namely, task allocation and secure deduplication (TA & SD), fog-based energy-efficient routing protocol (FEER), adaptive block compression sensing (ABCS), and secure data storage and searching in IIoT (SDSS-IIoT).

Average Latency

Some of the IIoT applications such as fire accidents, healthcare, and air pollution require very severe constraints in latency i.e. 10's of ms. When enabling data collection and processing features include clustering and classification of data at the device layer or fog layer, we can reduce the latency. Figure 6.6 shows the performance of the average latency concerning the number of IoT devices.

The fog layer lies in the cloud layer is to minimize the latency. However, previous mechanisms (TA&SD, FEER, ABCS) were required a large amount of time for processing data sensed by different IoT devices. The average latency for twenty IoT devices for the proposed scheme is 2.7ms which is minimum than previous works such as TA&SD, FEER, and ABCS for 8.9ms, 10.54ms, and 11.54ms, respectively. When compared to FEER and ABCS, TA&SD requires minimum average latency due to avoiding redundant copies in the fog layer. Mathematical computations of these three works are more and processing time for data transmission, and collection is large. We proposed 2FBO2 for optimum fog node selection. It take less waiting time for data transmission. Also, we proposed SHA-3 (512bits) for hash generation, which eliminates duplicate data in the fog layer.

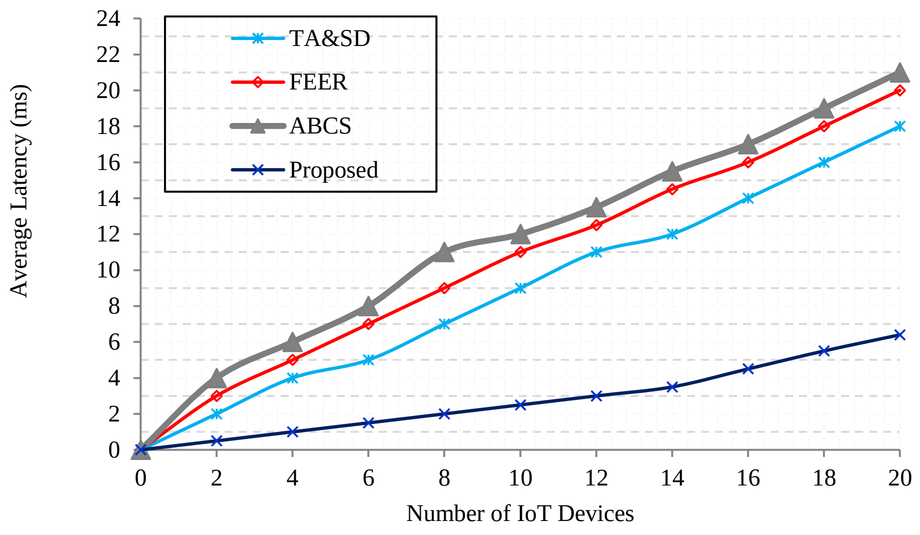


Figure 6.6: Average latency vs. No. of devices

User Satisfaction

As mentioned above, User satisfaction governs the quality of service provided to the user and depends on the thoughtfulness of QoS parameters. The user satisfaction performance of the proposed work in comparison to earlier work for several IoT devices is shown in Figure 6.7.

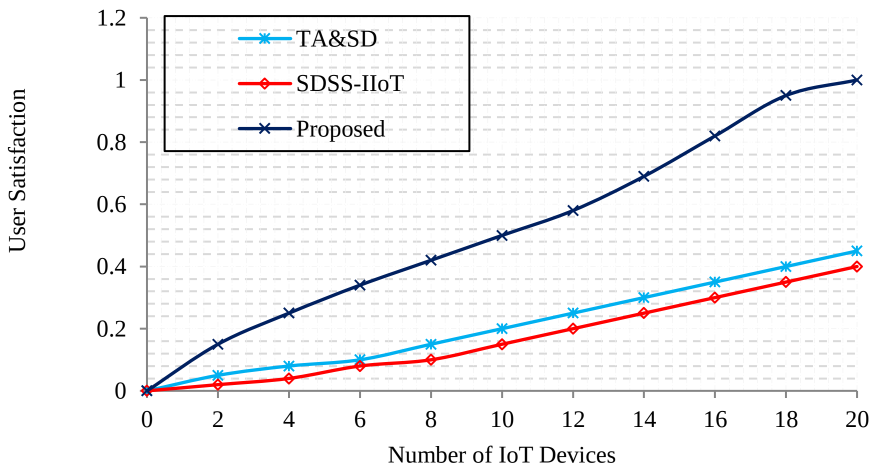


Figure 6.7: User satisfaction vs. No. of devices

From the results, it is found that the average user satisfaction of the proposed system is 0.5 in-comparison to 0.21 and 0.17 for TA&SD and SDSS-IIoT respectively. Hence, the proposed system pleases users with faster response times and better quality of service. Indexing is an essential component of data communication services used by IoT users, which also ensures that data is stored in a particular order and its accurate retrieval. The user satisfaction in TA&SD is low due to inappropriate management of CSs resulting in an extra delay in user services execution and in-fact many times, service provided to users is inappropriate. In this research work, better user satisfaction is achieved by the system with the application of the Merkle hash- tree along with the selection of idle FNs for the execution of user service.

Network lifetime

In this study, the research assessed the network lifetime for evaluation. Figure 6.8 shows the comparison of various systems on network lifetime performance concerning the number of IoT devices.

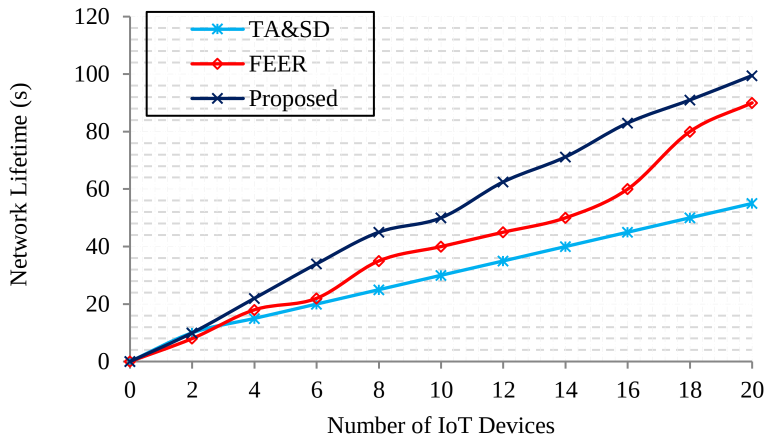


Figure 6.8 Network lifetime vs. the number of devices

From the results as mention in the Figure 6.8 graph, it is observed that the average network lifetime for the proposed system is the 50s, which is higher than previous works that are 29.54s and 40.72s for TA&SD and FEER respectively. In FEER, though network scalability is enhanced, latency in the network is diminished and network traffic is lessened, however, it has less network lifetime due to the routing of tasks among fog-nodes. In contrast, TA&SD holds a few network lifetime, which is mainly because of selecting poor FNs. Here researchers did not focus on selecting idle FNs at IoT devices. Generally, IoT devices and FNs are resource-constrained. The suggested system increases the network lifetime in-addition to decreases the energy consumption rate.

Energy Consumption:

The high-energy consumption issue is addressed by suggesting energy-effective jobs e.g. clustering. Figure 6.9 demonstrates a comparison of energy consumption by the proposed system and various earlier systems concerning the number of devices. From the results in the graphs, According to the graph, it is observed that the average energy consumption in the suggested system is 0.26J, which is much lesser than the usual consumption of energy in TA&SD and FEER, which are 1.5J and 1.21J respectively. This is achieved by the selection of the best FN for the execution of the task and energy-efficient cluster formation. In contrast, in earlier systems, enormous energy is consumed in calculations required for the transfer of packets from sensors to the fog-nodes.

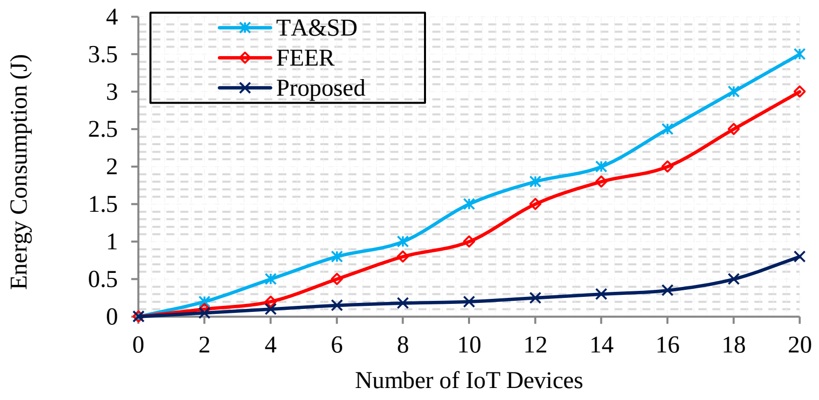


Figure 6.9: Energy Consumption vs. the number of devices

Further,

$TA\&S = 1.5J$
$FEE = 1.21J$

The suggested system framework uses less energy via the best Fog node selection and cluster formation.

Security Strength

Investigation of security for any real-time application is important, particularly in air pollution monitoring in IIoT. Security strength is computed based on several metrics such as key size, message size, encryption, and decryption time. We compare the proposed scheme with some previous works based on key size (in bits) starting from 64bits to 2048 bits. When key size increases, security strength may increases or decreases. Our proposed scheme has attained better security strength, which is depicted in Table 6.5, whereas previous works such as TA&SD and

SDSS-IIoT was attained less security strength due to ineffective security algorithms of Security Strength depends upon various metrics such as message size, key size & decryption, and time required for encryption. The performance of various systems including the proposed system for Security Strength concerning Key Size is shown in Figure 6.10 and table 6.5, concerning Message Size is shown in Figure 6.11 concerning Encryption Time is shown in Figure 6.12 and similarly, concerning Decryption Time is shown in Figure 6.13. Figure 6.11 shows the performance of the security strength concerning message size (bits). The average security strength for the proposed scheme is 0.61, which is higher than previous works such as TA&SD, SDSS-IIoT since they obtained 0.341, and 0.358, respectively. We proposed ECC-HM, which is a lightweight cryptographic algorithm, which gives high-security strength when message size increases. It consumes a minimal amount of time for encryption and decryption. Table 6.7 and Table 6.8 shows the performance of security strength with respect to the time of encryption and decryption. Pseudorandom function and secure KNN algorithm are not lightweight cryptography and thus it takes high processing time.

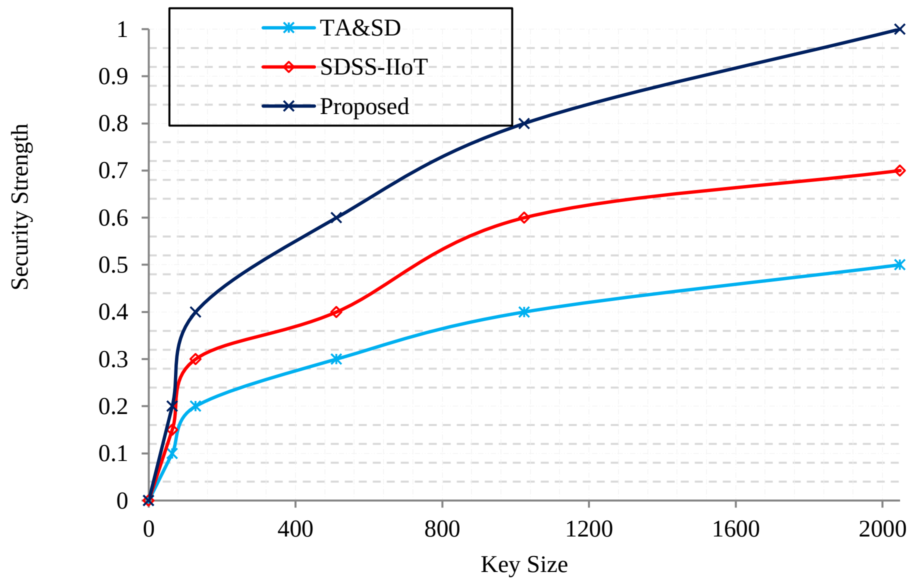


Figure 6.10: Security strength vs. key size

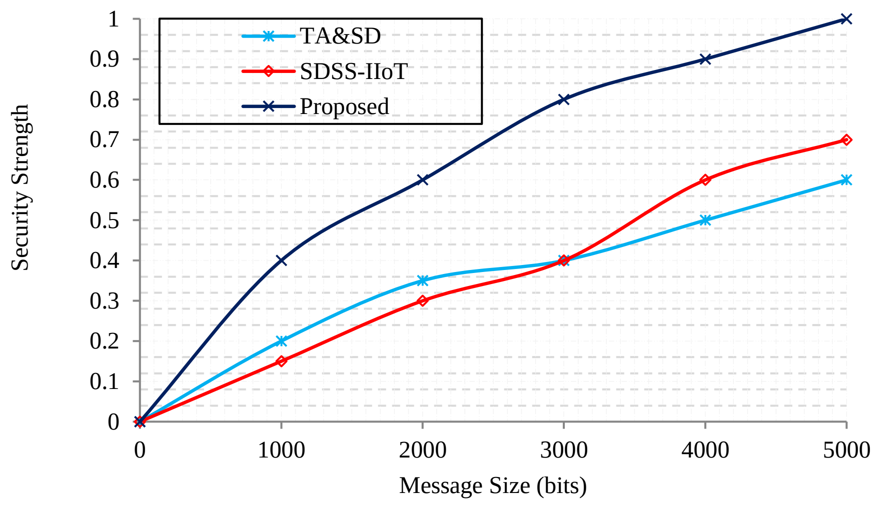


Figure 6.11 Security strength vs. message size

Table 6.6: Security Strength For Proposed Vs. Previous Works

Key size	**Security Strength**		
	TA & SD	**SDSS IIoT**	**Proposed**
64bits	0.1	0.15	0.2
128bits	0.2	0.3	0.4
512bits	0.3	0.4	0.6
1024bits	0.4	0.6	0.8
2048bits	0.5	0.7	1.0
Average	**0.25**	**0.358**	**0.5**

This research presented ECC-HM cryptographic algorithm, which is not only light in computation due to which encryption/decryption time is significantly reduced, but it also ensures that security strength remains high with an increase in the size of the message. In contrast, researchers in [39] had used a weak BLS Pseudorandom Function for security.

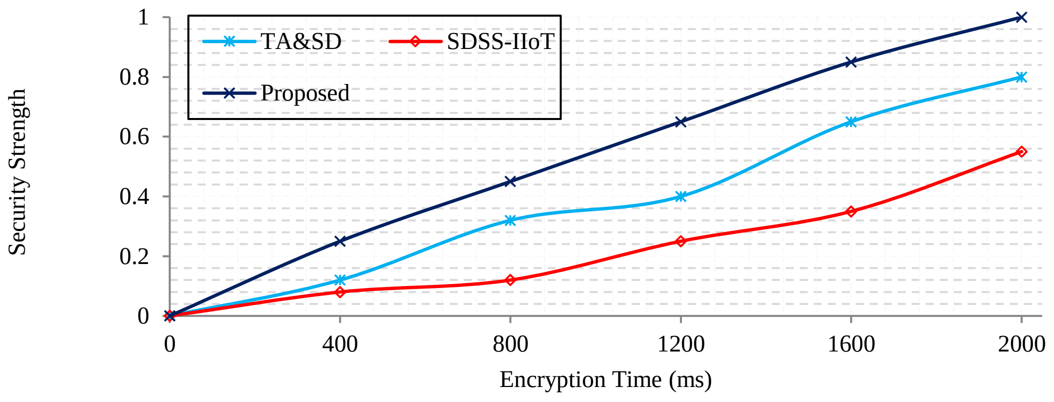


Figure 6.12: Security strength vs. encryption time

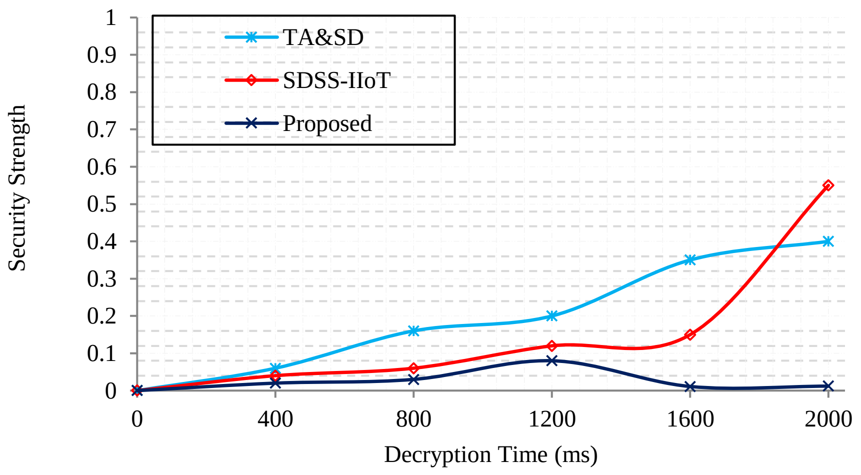


Figure 6.13: Security strength vs. decryption time

Table 6.7: Security Strength for the Proposed Vs. Previous Works with respect to encryption time

Encryption Time	**Security Strength**		
	TA&SD	**SDSS IIoT**	**Proposed**
400ms	0.12	0.08	0.25
800ms	0.32	0.12	0.45
1200ms	0.4	0.25	0.65
1600ms	0.65	0.35	0.85
2000ms	0.8	0.55	1
Average	**0.381**	**0.225**	**0.533**

Table 6.8: The Proposed Vs. Previous Works- Security Strength with respect to decryption time

Decryption Time	**Security Strength**		
	TA&SD	**SDSS IIoT**	**Proposed**
400ms	0.06	0.04	0.02
800ms	0.16	0.06	0.03
1200ms	0.2	0.12	0.08
1600ms	0.35	0.15	0.011
2000ms	0.4	0.55	0.012
Average	**0.195**	**0.153**	**0.0255**

Table 6.4 for different key sizes and Figure 6.11, it is observed that the suggested system has achieved an average security strength of 0.61, while earlier works - TA&SD and SDSS-IIoT had 0.341 and 0.358 respectively. The greater security strength in the proposed system in-comparison to earlier systems is attained by the elimination of unproductive security algorithms. The proposed system outperformed the earlier systems in all other metrics of Security Strength, which are Encryption Time and Decryption Time as summarized in Table 6.7 and Table 6.8 respectively as mentioned above. It can be inferred that the suggested system is performing better.

6.6 Summary

Air pollution is one of the major problems in the industrial sector in which deduplication is the critical factor to minimize storage capacity and latency of cloud server and fog nodes, respectively. The main purpose of this application consideration is that today air pollutants range exceeds its threshold range. Therefore, it causes severe health issues for people. To overwhelm this issue, we designed this paper over the Fog assisted Cloud environment for IIoT, which is carried out by two processes, namely Task Allocation and Secure Deduplication. For task allocation from IoT devices to fog nodes, we selected optimal CH, which generates a hash using aggregated data using SHA-3. After SHA-3, DSC_request is sent to optimal fog node, which process request and send DSC_response message to CH. To reduce latency for data transmission, we determined optimum fog nodes using the 2FBO^2 (FASWO) algorithm. A proxy server is deployed between the cloud servers and fog nodes, which is employed to build a searchable index for searching users by the given query (data retrieval). Index construction is based on the Merkle hash tree. To achieve data confidentiality, we proposed a security algorithm called ECC based HM algorithm for data encryption. Finally, simulation is conducted to implement the proposed as well as previous works comparison in terms of average latency, user satisfaction, energy consumption, network lifetime, and security strength. Our proposed scheme has proved that it is outperforming previous works.

CHAPTER 7

CONCLUSION

7.1 Conclusion

Scheduling and Energy consumption is a crucial aspect of Fog Computing, which has a significant impact on system performance. The current study's main aim is to reduce energy consumption and to optimize the load. For this, we have implemented an ABC optimization algorithm for load supervision and stabilization, to attain a lesser consumption of energy rate and decreased runtime length of the task. The proposed algorithm not only stabilizes the fog-computing load but also improvises the whole framework performance. A recurring framework considered sends a maximum of 1000 jobs and at least 10 jobs. The fitness function is formulated to attain the primary objective of reducing loads on VM. To execute the VMs tasks, an implementation system with initiation time and execution time is offered. The ABC Algorithm is used to rank the connected components based on the total energy consumed and finish time for the execution of whole tasks. New limits of Schedule Length of VM-Ratio (SLVMR) for the proposed framework are obtained which is not deliberated in previous studies. SLR at one VM, which is utilized the smallest time frame is compared with SLR of the proposed algorithm. In addition to the above, the performance of the proposed algorithm is evaluated by using ECR and ECRVM as well.

The performance results from the proposed framework are compared with the DAG structure. The comparative outcome analysis revealed that the proposed model has superior performance of both SLRVM & ECRVM concerning the regular DAG Model. It was found that the most SLRVM of the proposed model is 28, while it is 88 for DAG Model. In addition, the highest value of ECRVM of the proposed task model is 43, while it is 7 for DAG Model. Similarly, the proposed model considerably outperformed DAG Model in all other assessments because of the standardized ABC algorithm application in the proposed model. The proposed work has further futuristic scope by application of dynamic machine learning algorithms and open load management.

In this study, a matter of energy usage and SLA violation for task planning in fog computing have been spoken. However, many explanations are available for energy-conscious resource managing in fog computing for data centers however, the present studies emphasize decreasing energy usage without bearing in mind the SLA violation. Due to the discrepancy in planning, energy usage was more because of the deficiency of the host classifier. This study deals with task scheduling and resources energy awareness resources in fog computing conceptualized centered on artificial intelligence. ANN systems & MMFS have been used for reducing the SLA violation and energy usage in the suggested structure. MMFS is required for enhancing the tasks according to the energy ingestion rate based on several objective fitness functions. ANN is required for categorizing the finest host based on task requests and has assigned them consequently for decreasing energy consumption rates and SLA violations. Various tasks are occupied for computing the efficiency of the suggested task. The average value calculated for energy usage through the number of tasks with and without optimization algorithms 66.5 MJ and 70.2mJ correspondingly, and the average values perceived for SLA violation without and with optimization, algorithms are 0.742 and 0.776. It has been determined that the performance of the suggested job through SLA violation and energy consumption has been decreased.

Load balancing and task scheduling are important features, which critically influence the performance of the structure. The prior methods suffered because of node failures, environment type (for example distributed or centralized), resource shortage, and so on. To get a better of the problems of prior works, the present research has suggested a four-tier structure for load balancing and task scheduling.

The researcher has conducted this research to provide task allocation, which is fog-assisted, clustering of IIoT data based on MOWO, and further secure data de-duplication by using 2FBO^2. This has been shown through two procedures, namely job Distribution, and Protected Deduplication. For job distribution to FNs from IoT devices, the researcher designated optimal CH that produces a hash with the help of gathered data. To decrease latency for data communication, the research proposed to recognized idle FNs by using an advanced 2FBO^2 (FAFSWO) algorithm. The Merkel hash-tree is used to store data in an index structure. To ensure the privacy of data, the researcher suggested a safety algorithm for data encryption called ECC based HM algorithm.

Lastly, simulation is showed to execute the suggested as well as prior job comparison due to user satisfaction, average latency to name a few. The suggested system has demonstrated that it is performing better than prior jobs.

7.2 Future Work

This study has unlocked an extensive possibility for future studies. The way forward for future work shall be based on the application of Artificial Intelligence in the optimization method, which may further reduce the energy consumption rate and provide a resolution to fog computing overload issues. Further, it is suggested to implement other data handling techniques in search of better data de-duplication algorithms in fog computing systems. This is to achieve any further additional decreases in process delay and overall dependency. Edge AI (Artificial Intelligence) and services, which is a new paradigm, can be used as future work, as it embedded Artificial intelligence locally or near the source of streaming data. While interacting with an IoT enabled equipment, the data from the sensors or similar IoT devices need to be analyzed and processed in milliseconds, without having to send them to an offshore cloud location. It will provide low latency for onsite data processing, real-time analytics, and machine learning in the smallest compute footprint. It will not only improves the latency but also improves security. The potential of Fog Computing in 5G-enabled Edge AI environments extends way beyond the automotive industry, bringing the benefits of AI at the edge to all manner of enterprise, vertical market, smart cities, smart vehicle, etc.

BIBLIOGRAPHY

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *MCC ’12: Proceedings of of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012.

[2] Z. Wen, R. Yang, P. Garraghan, P., T. Lin, J. Xu, and M. Rovatsos, “Fog orchestration for the Internet of things services,” *IEEE Internet Computing”.*, vol. 21, no. 2, pp. 16–24, 2017.

[3] S. N. Gollaprolu Harish, B. Harish, and M. Shaik, “A Review on Fog Computing and its Applications,” *Int. J. Innov. Technol. Explore. Eng.*, vol. 8, no. 6, pp. 358-369, 2019.

[4] X. Xu., Fu, S., Cai, Q., Tian, W., Liu, W., Dou, W., & Liu, A. X.,”.Dynamic resource allocation for load balancing in fog environment”. *Wireless Communications and Mobile Computing*, pp. 1–15, 2018.

[5] A. Al-fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things : A Survey on Enabling Internet of Things : A Survey on Enabling Technologies, Protocols, and Applications”, *IEEE communications surveys & tutorials* vol.17, no. 4, pp. 2347–2376, 2015.

[6] P. Suresh, J. V. Daniel, and R. H. Aswathy, “A state of the art review on the Internet of Things (IoT) History, Technology and fields of deployment,” *International conference on science engineering and management research (ICSEMR)*, pp. 1-8, 2014.

[7] I. Lee and K. Lee, “The Internet of Things (IoT): Applications, investments, and challenges for enterprises,” *Bus. Horiz.*, vol. 58, no. 4, pp. 431–440, 2015.

[8] S. Madakam, R. Ramaswamy, and S. Tripathi, “Internet of Things (IoT): *A literature review”. Journal of Computer and Communications*, vol 3, pp. 164–173.

[9] M. Chiang and T. Zhang, “Fog and IoT: An Overview of Research Opportunities,” *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, 2016.

[10] P. Hu, S. Dhelim, H. Ning, and T. Qiu, “Survey on fog computing: architecture, key technologies, applications, and open issues,” *J. Netw. Comput. Appl.*, vol. 98, pp. 27–42, 2017.

[11] X. Wu, X. Zhu, G. Wu, and W. Ding,” Data mining with big data”, *IEEE transactions on knowledge and data engineering”*, vol. 26, no. 1, pp. 97–107, 2014.

[12] W. Jiafu, “Fog computing for energy-aware load balancing and scheduling in smart factory."

IEEE Transactions on Industrial Informatics, vol. 14,pp. 4548-4556,2018.

[13] A.Fernández, del Río, S., López, V., Bawakid, A., del Jesus, M. J., Benítez, J. M., & Herrera, F. (2014). “Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks”. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. vol. 4, no. 5, pp. 380–409, 2014.

[14] D. J and G. S., “Mapreduce: Simplified data processing on large clusters,” in *OSDI ’04: 6th Symposium on Operating Systems Design and Implementation*, pp. 137–150, 2004.

[15] Apache Hadoop, “Apache Hadoop,” *Apache Hadoop*, 2020. [Online]. Available: http://hadoop.apache.org/.

[16] J. Lin, “Mapreduce is good enough? if all you have is a hammer, throw away everything that’s not a nail,” *Big Data*, vol. 1, no. 1, pp. 28–37, 2013.

[17] G. Wu, W. Bao, X. Zhu, Xi. Zhang, “A General Cross-Layer Cloud Scheduling, Framework for Multiple IoT Computer Tasks”, *Sensors*, vol. 18, pp. 1-20, 2018.

[18] M. Zaharia *et al.*, “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing.” in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation.* pp. 15–28, 2012.

[19] Apache Flink, “Apache Flink — Stateful Computations over Data Streams,” *Apache Flink*. [Online]. Available: https://flink.apache.org/.

[20] S.Mahendra, B. Gawali, S. K. Shinde, “Task Scheduling and Resource Allocation in Cloud Computing using a Heuristics Approach”, *Journal of Cloud Computing: Advances, Systems, and Applications*, vol. 7, issue.4, pp. 1-16, 2018.

[21] Z. Liu , J. Zhang, Y. Li, L. Bai, J.Yuefeng Ji,” Joint Jobs Scheduling and Lightpath Provisioning in Fog Computing Micro Datacenter Networks”, *Journal of Optical Communications and Networking*, Vol. 10, Issue.7, pp. B152-B163, 2018.

[22] A. Khajeh-Hosseini, D. Greenwood, J. Smith, and I. Sommerville, “The Cloud Adoption Toolkit: Supporting Cloud Adoption Decisions in the Enterprise.” *Softw. Pract. Exp.*, vol. 42, no. 4, pp. 447–465, 2015.

[23] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, “Cloud migration: A case study of migrating an enterprise IT system to IaaS,” in *3rd International Conference on Cloud Computing*, pp. 450–457, 2010.

[24] T. BInz, U. Breitenbücher, O. Kopp, and F. Leymann, “Migration of enterprise applications

to the cloud," *IT-Inf Techno*, vol. 56, no. 3, pp. 106–111., 2014.

[25] P. Mohagheghi and T. Sæther, "Software engineering challenges for migration to the service cloud paradigm: Ongoing work in the REMICS project," in *IEEE World Congress on Services (SERVICE),* pp. 507–514, 2011.

[26] G. Menychtas A, Konstanteli K, Alonso J, Orue-Echevarria L, Gorroñogoitia J, Kousiouris, "Software modernization and cloudification using the ARTIST migration methodology and framework," *Scalable Comput. Pract. Exp.*, vol. 15, no. 2, pp. 131–152, 2014.

[27] S. Frey and W. Hasselbring, "Model-based migration of legacy software systems into the cloud: The CloudMIG approach," *Softwaretechnik-Trends*, vol. 30, no. 2, pp. 155–158., 2010.

[28] S. Strauch, V. Andrikopoulos, D. Karastoyanova, and K. Vukojevic-Haupt, "Migrating e-Science Applications to the Cloud: Methodology and Evaluation. Cloud Computing with E-science Applications*", Boca Raton*.: Taylor & Francis, pp. 89-114, 2014.

[29] R. Pérez-Castillo, I. de Guzmán, I. Caballero, and M. Piattini, "Software modernization by recovering web services from legacy databases," *J. Softw. Evol. Process*, vol. 25, no. 5, pp. 507–533, 2013.

[30] M. Gholami, F. Daneshgar, G. Low, and G. Beydoun, "Cloud migration process—a survey, evaluation framework, and open challenges," *J. Syst. Softw.*, vol. 120, pp. 31–69, 2016.

[31] A. Y. Zomaya and Y. H. Teh, "Observations on using genetic algorithms for dynamic load-balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 9, pp. 899–911, 2011.

[32] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," *24th IEEE international conference on Advanced information networking and applications (AINA),* pp. 400–407, 2010.

[33] P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2292–2303, 2013.

[34] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Commun.*, vol. 13, no. 3, pp. 156–164, 2016.

[35] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real-Time Efficient Scheduling Algorithm for Load Balancing in Fog Computing Environment," *Int. J. Inf. Technol. Comput. Sci.,* vol. 14,

pp. 1–10, 2016.

[36] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, “An overview of fog computing and its security issues. Concurrency and Computation,” vol. 28, no. 10, pp. 2991–3005, 2016.

[37] A. Y. Zomaya, C. Ward, and B. Macey, “Genetic scheduling for parallel processor systems: Comparative studies and performance issues,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 8, pp. 795–812, 1999.

[38] S. Agarwal, S. Yadav, and A. K. Yadav, “An architecture for elastic resource allocation in fog computing,” *Int. J. Comput. Sci. Commun.*, vol. 6, no. 2, pp. 201–207, 2016.

[39] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, “Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption,” *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, 2016.

[40] L. F. Bittencourt, D.-M. J., R. Buyya, O. F. Rana, and M. Parashar, “Mobility-aware application scheduling in fog computing,” *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, 2017.

[41] S. K. Mishra, D. Putha, J. J. Rodrigues, B. Sahoo, and E. Dutkiewicz, “Sustainable Service Allocation using Metaheuristic Technique in Fog Server for Industrial Applications,” *IEEE Trans. Ind. Informatics*, vol. 14, no. 10, pp. 4497–4506, 2018.

[42] R. Mahmud, R. Kotagiri, and R. Buyya, “Fog computing: A taxonomy, survey, and future directions,” in the *Internet of everything*, Singapore: Springer, pp. 103–130,2018.

[43] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, “Secure and Sustainable Load Balancing of Edge Data Centers in Fog Computing,” *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 60–65, 2018.

[44] C.Chen, Y.Hao Chang, C.Chieh Yang, K. Lee, “Switching Supports for Stateful Object Remoting on Network Processors”, Journal of Supercomputing, vol. 40, Issue.3, pp. 281-298, 2007.

[45] X. Q. Redowan Mahmud, Kotagiri Ramamohanarao, Rajkumar Buyya, “Latency-aware Application Module Management for Fog Computing Environments”, ACM Transactions on Embedded Computing Systems, vol. 9, no. 4, pp. 1-21.2018.

[46] Pham and E. N. Huh, “Towards task scheduling in a cloud-fog computing system.” in *18th Asia-Pacific network operations and management symposium*, pp. 1–4, 2016.

[47] X. Masip-Bruin, E. Marín-Tordera, A. Alonso, and J. Garcia, "Fog-to-cloud computing (F2C): The key technology enabler for dependable e-health services deployment," in *2016 Mediterranean ad hoc networking workshop (Med-Hoc-Net)*, pp. 1–5, 2016.

[48] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1728– 1739, 2016.

[49] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *International Conference on Wireless Algorithms, Systems, and Applications*, pp. 685–695, 2015.

[50] K. Flautner, S. Reinhardt, and T. Mudge, "'Automatic performance setting for dynamic voltage scaling," *Wirel. Netw.*, vol. 8, no. 5, pp. 507–520, 2002.

[51] H. Hanson, S. W. Keckler, S. Ghiasi, K. Rajamani, F. Rawson, and J. Rub, "Thermal response to DVFS: Analysis with an Intel Pentium M," *Int. Symp. Low Power Electron. Des.*, vol. 7, pp. 219–224, 2007.

[52] C.-M. Wu, R.-S. Chang and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Futur. Gener. Comput. Syst.*, vol. 37, pp. 141–147, 2017.

[53] A. Wierman, L. L. H. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing system," in *IEEE International Conference on Computer Communications*, 2009, pp. 2007–2015, 2009.

[54] S. Lee and T. Sakurai, "Run-time voltage hopping for low-power real-time systems," in *Proceedings of the 37th Annual Design Automation Conference*, pp. 806–809, 2000.

[55] E. Feller, C. Morin, and A. Esnault, "'A case for fully decentralized dynamic VM consolidationinclouds," in *International Conference on Cloud Computing Science and Technology*, pp. 26–33,2012.

[56] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "*An overview of fog computing and its security issues*", Concurrency and Computation: Practice and Experience", vol. 28, no. 10, pp. 2991–3005, 2016.

[57] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurr. Comput. Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, 2012.

[58] A. Beloglazov, J. Abawajy, and R. Buyya, "'Energy-aware resource allocation heuristic for efficient management of data centers for cloud computing," *Futur. Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[59] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, "PRACTISE: Robust prediction of data center time series," in *11th International Conference on Network and Service Management*, pp. 126–134, 2015.

[60] D. Grimes, "Robust Server Consolidation: Coping with Peak Demand Underestimation," in *IEEE 24th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 271–276, 2016.

[61] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression-based CPU usage prediction algorithm for live migration of virtual machines in data centers," in *46th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 357–364, 2013.

[62] B. Guenter, N. Jain, and C. Williams, "'Managing cost performance and reliability tradeoffs for energy-aware server provisioning," in *Proceeding IEEE INFOCOM*, pp. 1332–1340, 2011.

[63] HF Sheikh, I Ahmad, Z Wang, S Ranka, "An overview and classification of thermal-aware scheduling techniques for multi-core processing systems", *Sustainable Computing: Informatics and Systems*, vol. 2, issue. 3, pp.151-169.

[64] E. Feller, L. Rilling, and C. Morin, "'Energy-aware ant colony based workload placement in clouds," in *12th IEEE/ACM International Conference on Grid Computing*, pp. 26–33, 2011.

[65] HF Sheikh, I Ahmad, D Fan, "An EvolutionaryTechnique for Performance-Energy-Temperature Optimized Scheduling of Parallel Tasks on Multi-Core Processors", *IEEE Transactions on Parallel and Distributed Systems (TPDS),* vol.27, issue.3, pp. 668-681, 2016.

[66] M.Aazam, Zeadally, S., & K. A Harras," Deploying Fog Computing in Industrial Internet of Things and Industry 4.0", *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674-4682, 2018.

[67] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications, and issues," in *In Proceedings of the 2015 workshop on mobile big data*, vol. 5, pp. 37–42, 2015.

[68] G. Wu, W. Bao, X. Zhu, and X. Zhang, "A General Cross-Layer Cloud Scheduling Framework for Multiple IoT Computer Tasks," *Sensors*, vol. 18, pp. 1–20, 2018.

[69] J.Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog Computing for Energy-aware Load Balancing and Scheduling in Smart Factory," *IEEE Trans. Ind. Informatics*, pp. 1–9, vol 23,2018.

[70] M. B.Gawali and S. K. Shinde, "Task Scheduling and Resource Allocation in Cloud Computing using a Heuristics Approach," *J. Cloud Comput. Adv. Syst. Appl.*, vol. 7, no. 4, pp. 1–16, 2018.

[71] Q. Liu, Y. Wei, S.Leng, and Y. Chen, "Task Scheduling in Fog Enabled Internet of Things for Smart Cities," in *17th IEEE International Conference on Communication Technology*, 2017, pp. 975-980.

[72] Z. Liu, J. Zhang, Y. Li, L. Bai, and Y. Ji, "Joint Jobs Scheduling and Lightpath Provisioning in Fog Computing Micro Datacenter Networks," *J. Opt. Commun. Netw.*, vol. 10, no. 7, 2018.

[73] S. E.Kafhali and K. Salah, "Efficient and dynamic scaling of fog nodes for IoT devices," *J. Super Comput.*, vol 60, pp. 1–24, 2017.

[74] X. Xu *et al.*, "Dynamic Resource Allocation for Load Balancing in Fog Environment," *Wirel. Commun. Mob. Comput. Hindawi*, pp. 1–16, 2018.

[75] H. Xiao, Z. Zhang, and Z. Zhou, "GWS- A Collaborative Load-Balancing Algorithm for Internet-of-Things," *Sensors*, vol. 18, pp. 1–17, 2018.

[76] M. U. Sharif, N.Javaid, M. J. Ali, W. A. Gilani, A.Sadam, and M. H.Ashraf, "Optimized Resource Allocation in Fog-Cloud Environment using Insert Select," in *21st International Conference on Network-based Information Systems*, pp. 611-623,2018.

[77] Y. Liu, K.A. Hassan, M. Karlsson, O. Weister, S. Ghong, "Active Plant Wall for Green Indoor Climate based on Cloud and Internet of Things", *IEEE Access*, vol. 6, pp. 33631-33644, 2018.

[78] F.Caramés, T.M., P.Suárez-Albela, M., & M.A. Díaz," A Fog Computing Based Cyber-Physical System for the Automation of Pipe-Related Tasks in the Industry 4.0 Shipyard", *Sensors*, vol. 18, no. 6, pp. 1-26.

[79] A.L. Clements, W.G. Griswold, J.E. Johnson, M.M. Herting, J. Thorson, A.C. Oxandale, M. Hannigan, "Low-Cost Air Quality Monitoring Tools: From Research to Practice (A

Workshop Summary), *Sensors*, vol. 27, no. 11, pp. 1-20, 2017.

[80] Z. Liu, & Li, S.," Sensor-cloud data acquisition based on fog computation and adaptive block compressed sensing", *International Journal of Distributed Sensor Networks*, vol. 14. Issue. 9, 2018.

[81] D. Rahbari and M.Nickray, "Scheduling of Fog Networks with Optimized Knapsack Symbiotic Organisms Search," in *Proceeding of the 21st Conference of Fruct. Association*, 2018.

[82] L. Yin and J. Luo, "Task scheduling and Resource Allocation in Fog Computing based on Containers for Smart Manufacturing," *IEEE Trans. Ind. Informatics,* pp. 1–9, 2018.

[83] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility- Aware Application Scheduling in Fog Computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, 2017.

[84] H. H.Madni, M. S. A.Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance Comparison of Heuristic Algorithms for Task Scheduling in IaaS Cloud Computing Environment," *PLoS One*, pp. 1–26, 2017.

[85] R. Deng, R. Lu, C. Lai, T. H. Luan, and H.Liang, "Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption," *IEEE Internet Things*, pp. 1–11, 2016.

[86] X. Q. Pham and E. Huh, "Towards Task Scheduling in a Cloud-Fog Computing System," in *The 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1-4. IEEE, 2016.

[87] T. Choudhari, M. Moh, and T. Moh, "Prioritized Task Scheduling in Fog Computing," in *ACMSE Conference Transactions Proceedings*, pp 1-8, 2018.

[88] T. D. Dang and D. Hoang, "FBRC: Optimization of Task Scheduling in Fog based Region and Cloud," *IEEE Trust.*, pp. 1109-1114, 2017.

[89] R. Mahmud, K.Ramamohanarao, and R.Buyya, "Latency- aware Application Module Management for Fog Computing Environments," *ACM Trans. Embed. Comput. Syst.*, vol. 9, no. 4, pp. 1–21, 2010.

[90] H. A. Manis, F. O. Oladipo, and E. Ariwa, "User mobility and resource scheduling and Management in fog computing to support IoT devices," in *The Seventh International Conference on Innovative Computing Technology*, pp. 11-21 2017.

[91] N. Zhang, X. Yang, M. Zhang, Y. Sun, and K. Long, “A Genetic Algorithm-based Task Scheduling for Resource Crowd-Funding Model,” *Int. J. Commun. Syst.*, vol. 31, pp. 1–10, 2018.

[92] L. Liu, D. Qi, N. Zhou, and Y. Wu, “A Task Scheduling Algorithm based on Classification Mining in Fog Computing Environment,” *Wirel. Commun. Mob. Comput.*, pp. 1–12, 2018.

[93] Q. Fan and N. Ansari, “Towards Workload Balancing in Fog Computing Empowered IoT,” *IEEE Trans. Netw. Sci. Eng.*, pp. 1–11, 2018.

[94] S. Bitam, S. Zeadally, and A. Mellouk, “Fog Computing Job Scheduling Optimization based on Bees Swarm,” *Enterprise Information Systems*, *12*(4), pp. 373-379, 2018.

[95] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, “Fog Computing Dynamic Load Balancing Mechanism Based on Graph Repartitioning,” *China Commun.*, pp. 156–164, 2016.

[96] Y. Su, F. Lin, and H. Xu, “Multi-Objective Optimization of Resource Scheduling in Fog Computing using an Improved NSGA-II,” *Wirel. Pers. Commun.*, pp. 1–17, 2017.

[97] X. Pham, N. D. Man, N. D.T.Tri, N. Q. Thai, and E. Huh, “Cost and Performance Effective Approach for Task Scheduling based on Collaboration between Cloud and Fog Computing,” *Int. J. Distrib. Sens. Networks,* vol. 13, pp. 1–16, 2017.

[98] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, “DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog Networks,” *IEEE Internet Things J.*, pp. 1–11, 2018.

[99] C.K. Chen, Y.-H. Chang, C.-C. Yang, and J.-K. Lee, “Switching Supports for Stateful Object Remoting on Network Processors,” *J. Supercomput.*, vol. 40, no. 3, pp. 281–298, 2007.

[100] H. Topcuoglu and S. Hariri, “Min-You Wu, Performance Effective and Low-Complexity Task Scheduling for Heterogeneous Computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, 2002.

[101] T. Wang, Z. Liu, Y. Chen, and Y. Xu, “Load balancing task scheduling based on genetic algorithm in cloud computing,” in *IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, pp. 146-152, 2014.

[102] T. Khalid, A. N. Khan, M. Ali, A. Adeel, ur R. Khan, and J. A., Shuja, “A fog-based security framework for intelligent traffic light control system,” *Multimed. Tools Appl.*, 2018.

[103] H.-Q. Wu, L. Wang, and G. Xue, "Privacy-aware task allocation and data aggregation in fog-assisted spatial crowdsourcing", IEEE Transactions on Network Science and Engineering, vol7 issue 1, pp.589-602.

[104] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial IoT Data scheduling based on Hierarchical Fog Computing", Industrial IoT data scheduling based on hierarchical fog computing: A key for enabling smart factory. *IEEE Transactions on Industrial Informatics, 14*(10), 4590-4602, 2018.

[105] L. Chen, P. Zhou, L. Gao, and J. Xu, "Adaptive Fog Configuration for the Industrial Internet of Things," *IEEE Transactions on Industrial Informatics", 14*(10), 4656-4664. vol. 1, no. 1, 2018.

[106] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart-resources partitioning for fog computing-enabled industrial Internet of Things." *IEEE Transactions on Industrial Informatics* Issue 14, vol no. 10, pp.4702-4711, 2018.

[107] S. Shukla, M. F. Hassan, L. T. Jung, and A. Awang, "Architecture for Latency Reduction in Healthcare Internet-of-Things Using Reinforcement Learning and Fuzzy Based Fog Computing," *Recent Trends Data Sci. Soft Comput.*, pp. 372–383, 2018.

[108] Miao, Dejun, et al. "An efficient indexing model for the fog layer of industrial internet of things." *IEEE Transactions on Industrial Informatics* vol 14 issue.10, pp. 4487-4496.,2018.

[109] J. Shen, H. Yang, A. Wang, T. Zhou, and C. Wang, "Lightweight authentication and matrix-based key agreement scheme for healthcare in fog computing," *Peer-to-Peer Netw.Appl.*, pp.1123-129,2018.

[110] Z. Liu and S. Li, "Sensor-cloud data acquisition based on fog computation and adaptive block compressed sensing," *Int. J. Distrib. Sens. Networks,* vol. 14, no. 9, 2018.

[111] J. Fu, Y. Liu, H.-C. Chao, B. Bhargava, and Z. Zhang, "Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing." *IEEE Trans. Ind. Informatics,* vol. 1, no. 1, 2018.

[112] B. Bathiya, S. Srivastava, and B. Mishra, "Air pollution monitoring using a wireless sensor network". *IEEE international WIE conference on electrical and computer engineering,* pp. 112-117,2016

[113] H, Wang, Liu, T., Kim, B. G., Lin, C. W., Shiraishi, S., Xie, J., & Han, Z. " Architectural Design Alternatives based on Cloud/Edge/Fog Computing for Connected Vehicles". *IEEE*

Communications Surveys & Tutorials, 2020.
[D.O.I 10.1109/COMST.2020.3020854]

[114] J Bellendorf., & Mann, Z. Á. (2020). "Classification of optimization problems in fog computing", *Future Generation Computer Systems*, *107*, 158-176.

.

LIST OF PUBLICATIONS

Journal

1. Sharma, S., &Saini, H. (2019). A novel four-tier architecture for delay aware scheduling and load balancing in fog environment. Sustainable Computing: Informatics and Systems, 24, 100355.[[Major indexing: **Science Citation Index Expanded,** Scopus) [https://doi.org/10.1016/j.suscom.2019.100355]

2. Sharma, S., &Saini, H. (2020). Fog assisted task allocation and secure deduplication using 2FBO2 and MoWo in cluster-based industrial IoT (IIoT). *Computer Communications*, *152*, 187-199.*[Major indexing: indexing:* **Science Citation Index** Expanded, Scopus] [https://doi.org/10.1016/j.comcom.2020.01.042]

3. Sharma, S., &Saini, H. (2019). Efficient Solution for Load Balancing in Fog Computing Utilizing Artificial Bee Colony. *International Journal of Ambient Computing and Intelligence (IJACI), 10*(4), 60-77.*[Major indexing: SCOPUS*, Emerging Sources Citation Index (ESCI)]

4. Sharma, S., Sharma, A., &Saini, H. (2019).Advanced Network Security Analysis (ANSA) in Big Data Technology. International Journal of Innovative Technology and Exploring Engineering. Volume 8, Issue 10. pp.2634-2640. *[Major indexing: SCOPUS,*]

5. Sharma, S. and Saini, H., "Minimizing Energy Consumption and SLA Violation in Fog Computing Using Artificial Neural Network", International Journal of Sensors, Wireless Communications, and Control (2020) 10: 1. [Major indexing: SCOPUS} [https://doi.org/10.2174/2210327910666200206155949]

Conferences:

1. Sharma, S., Rathee, G., & Saini, H. (2018, December). Big data analytics for crop prediction model using an optimization technique. In 2018 Fifth International Conference on Parallel, Distributed, and Grid Computing (PDGC) (pp. 760-764). IEEE.
2. Presented a Technical paper entitled, "Big data analytics for crop prediction model using optimization technique", Accepted in Futuristic Trends in Network and Communication Technologies (FTNCT2018).

3. Presented a paper entitled "BDA for crop prediction using GWO optimization" in the Fifth international conference on parallel distributed and grid computing (PGDC-2018) held in JUIT Waknaghat.
4. Presented a paper entitled " Agricultural data Clustering using FAO Heuristics" in the International conference on computational and automation engineering (ICCAI-2018) held at AMITY University Noida,7-9 Feb 2018
5. Presented a paper entitled "Advance Network security analysis using big data technology" in an International conference on advance in science and technology held at Swami Keshavand and Institute of Technology and Management Jaipur on 4,5 May 2018.

<u>Other Conferences</u>

- Presented a paper entitled "Big Data Analytic for Crop Prediction in Indian Scenario Using FAO Heuristic" Organized by HP Science Congress held at IIT Mandi (22-23 Oct 2018).
- Presented a paper entitled "Big data analysis for crop prediction: A case study of Solan District of Himachal Pradesh' at 2nd HP Science Congress held at Hotel Peterhoff, Shimla. (20 - 21 Nov 2017).
- Presented a paper entitled "Big Data Analytics for Agriculture data set in Indian Scenario" in International conference on advance in science and technology held at Swami Keshavand Institute of Technology and Management, Jaipur (4-5 May 2018).

Work under submission:

- Sharma, S. and Saini, H " Framework for deadline sensitive application using Conventional Neural Network", In Journal of Computing (JOC) [Minor Revision]

Shivi

Signature of Scholar

(Shivi Sharma)