An Adaptive Fault Tolerant Framework and Latency Aware Service Placement Scheme for Fog Computing Environment

Thesis submitted in fulfillment of the requirements for the Degree of

DOCTOR OF PHILOSOPHY

By

PANKAJ SHARMA

Department of Computer Science Engineering and Information Technology
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
Waknaghat, Solan-173234, Himachal Pradesh, INDIA
April, 2024

DECLARATION OF SCHOLAR

I hereby declare that the work reported in the Ph.D. thesis entitled **"An Adaptive Fault Tolerant Framework and Latency Aware Service Placement Scheme for Fog Computing Environment"** submitted at **Jaypee University of Information Technology, Waknaghat, INDIA** is an authentic record of my work carried out under the supervision of **Prof (Dr.) P K Gupta** I have not submitted this work elsewhere for any other degree or diploma. I am fully responsible for the contents of my Ph.D Thesis.

22/4/2024

Pankaj Sharma

Enrollment No. 176202

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat -173234, INDIA.

Date: /04 /2024

SUPERVISOR'S CERTIFICATE

This is to certify that the work in the thesis entitled **"An Adaptive Fault Tolerant Framework and Latency Aware Service Placement Scheme for Fog Computing Environment "** submitted by **Pankaj Sharma** enrollment no. 176202, is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science and Engineering in the Department of Computer Science and Engineering, **Jaypee University of Information Technology, Waknaghat, INDIA.** Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Date: 22/04 /2024

22/04/2024

Dr. P K Gupta

Professor

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat -173234, INDIA.

ACKNOWLEDGEMENTS

First, I would like to express my deep respect and gratitude towards my guide, Prof. Dr. P K Gupta in the Department of Computer Science and Engineering and Information Technology, for their support throughout this research work. I want to thank them for introducing me to the fog computing field and allowing me to work under them. Without their invaluable advice and assistance, it would not have been possible for me to complete this thesis. I am greatly indebted to them for their constant encouragement and valuable advice in every aspect of my academic life. I consider it my good fortune to have an opportunity to work with such wonderful people.

I want to express my gratitude to our Honorable Vice Chancellor, Prof. (Dr.) Rajendra Kumar Sharma and Dean Academics and Research Prof. (Dr.) Ashok Kumar Gupta to promote the research and facilitate resources in the institution. I would also like to thank H.O.D (CSE) Prof. Dr. Vivek Kumar Sehgal and my doctoral committee members, Prof. Dr. Rajiv Kumar, Dr. Kushal Kanwar, and Dr. Pankaj Dhiman, for their valuable feedback and critical reviews during presentations and time to time help.

I am also grateful for the support received from the JUIT, Waknaghat. In particular, I thank Ravi Raina and all the staff at the Department of Computer Science and Engineering and Information Technology, JUIT, Waknaghat, who have been extremely helpful on numerous occasions. I thank my fellow PhD friends for their consistent help and valuable discussion. Last but not least, I would like to thank my family and parents for giving birth to me in the first place and supporting me spiritually throughout my life.

Pankaj Sharma

This Thesis is dedicated to My Parents and My Beloved Family Members.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AI	Artificial Intelligence
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ANN	Artificial Neural Network
AR	Augmented Reality
CC	Cloud Computing
CoT	Cloud of Things
CRBM	Checkpoints, Replication and Bee Mutation
DARA	Deadline Aware Resource Allocation
ELBS	Efficient Load Balancing Strategy
FC	Fog Computing
FN	Fog Node
F-RAN	Radio Access Network for Fog
FTDM	Fault Tolerant Data Management
GA	Genetic Algorithm
HMM	Hidden Markov Model
IoT	Internet of Things
IIoT	Industrial Internet of Things
IoTSCA	Internet of Things Smart City Application
LB	Load Balancing
MEC	Mobile Edge Computing
MGA	Mobility Aware Genetic Algorithm
PSO	Particle Swarm Optimization
QoS	Quality of Service
QoE	Quality of Experience
VFC	Vehicular Fog Computing
VM	Virtual Machine

LIST OF SYMBOLS

Symbol	Definitions
$Sl_{i,j}$	New neighbour solution in Artificial Bee Colony
Cop	Cross-over probability
M_P	Mutation probability
Ft_i	Fitness measure of i^{th} swarm solution
Ft_j	Fitness measure of j^{th} swarm solution
Ft_a	Average fitness value
Ft_{max}	The maximum fitness value
$\phi_{i,j}$	A random real numbers
$Nl_{i,j}$	Lower bound of Jth variable
$Nl_{k,j}$	Uppar bound of Jth variable
Ri	Random Vector
D	Mobile device distance from the fog node
M_B	Mobility
T_{12}	Time to reach out from P1 to P2
P_A	Physical availability
F_XA	Fog node coverage area
P_AF	Physical availability factor
T_RT	Tier resident time
Md	Mobile Device
Fd	Fog Device
DT_{ij}	Distance between two fog node
α_{ij}	Inclination angle
θ_{ij}	Azimuth angle
$CarT_{ij}$	Cartesian coordinates
Q_{ij}	Rotation matrix

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION

1.1 Fog Computing

One of the inventions now receiving much attention is the IoT, which provides enormous advantages to researchers [1]. This evolution of the Internet of Things is approaching a point when many of the items already in our environment will access Internet connectivity capabilities to interact with each other without the participation of a human being. Initially, the Internet of Things was envisioned as a way to cut down on the amount of time humans spend manually entering data, as well as to make use of a variety of sensors to gather data from their surroundings and to enable the automated storing and processing of all of these data. Since the Internet of Things is defined by constrained calculations, its competitors in the storage and processing space face various difficulties, including performance, security, privacy, and dependability. The Cloud of Things (CoT), the combination of IoT devices with cloud storage, processing, and analytics, is the best solution to solve most of these problems. The CoT accelerates and reduces the expense of setting up and integrating sophisticated data processing & deployment while streamlining the process of obtaining and processing Internet of Things data.

Many benefits are achieved for different Internet of Things applications due to the integration of the Internet of Things combined with Cloud Computing. Creating new Internet of Things apps is difficult because a diverse collection of devices runs on different platforms [2]. This is because apps built for the internet of things produce vast amounts of information via sensors and other hardware and generate a massive amount of data, resulting in recommendations about various courses of action. It will take considerable bandwidth to upload all these files to the cloud. Cloud computing and other forms of fog computing may help solve these problems [3].

Computing in a cloudy environment is what Cisco first referred to as fog computing. It is an emerging technology that offers various advantages for various areas, particularly the Internet of Things (IoT). IoT users may use services that include fog computing's processing and data storage, analogous to cloud computing. This fog computing notion is founded on the hint that fog devices should be equipped with

local data processing capabilities and local storage rather than sending information to a remote server. The cloud & fog are two different terms for the same thing: storage, computation, and networking resources. The IoT uses fog computing to enhance efficiency and performance while reducing the amount of information that must be sent to a remote server for processing, analysis, & storage. Instead of being transmitted into the cloud, this information is gathered via sensors routed edge network devices, which will be processed and temporarily stored. This will result in a reduction in both the amount of network traffic and the amount of delay.

1.1.1 Fog Computing: Definition

Fog computing is a relatively new archetype of computation that is spread. Cisco created it in 2012 to help with cloud computing and boost QoS (quality of service) to generate various supported IoT applications [4]. The literal interpretation of the word "fog" reveals the characteristics of this meteorological phenomenon: when clouds are high in the atmosphere, fog descends to the earth and surrounds humans. There are several definitions for fog that all center on the same concepts. One of these definitions is that a fresh computing paradigm is called fog computing. It might be considered to further cloud computing's reach at a network's periphery to lessen the cloud's load. While researchers characterized fog computing as a novel computing paradigm discovered in transcend challenges associated with cloud computing, researchers also defined it as an outgrowth of cloud computing, fog computing, and services located at the user's proximity edge. The word "*fog*" refers to the distribution of data processing the periphery of a net.

In summary, fog computing refers to a "*small cloud that is located on the end user's network edge*". The cloud performs tasks such as preprocessing data, assigning roles and policies, filtering specific tasks, or caching certain information earlier, sending it to an enormous cloud where it can be stored and thoroughly analyzed. Because of Fog's commitment to being neutral, the company has provided new capabilities and solved several cloud computing issues.

1.1.2 Fundamentals of Fog Computing

The Internet of Things will include billions of additional gadgets, most of which will be connected to the network. This will have limited access to specific resources. The need for an intermediary computer layer becomes apparent to overcome the

difficulties presented by these devices and satisfy the prerequisites imposed by the application domain. The idea of computing in the fog is the most recent offspring of technological innovations that have evolved from the physical isolation operational elements. It's the computational layer that is physically closer to the sensors and actuators of the perception layer. This layer offers computing, networking, and storage services. This Fog layer affords the various distinctiveness addressed in the subsequent sections to handle these services and meet the demands placed on IoT systems[4].

1.1.3 Characteristics of the Fog Layer

This paragraph compares and contrasts the characteristics of this Fog layer, which serves as a computational middleware between the perceptual and cloud levels. Unlike the Cloud layer, the Fog layer is physically located nearer perception [5]. As a result of this closeness, the Fog layer has various benefits that help define it. One of its many instant advantages is that the Cloud is aware of its users' whereabouts. This level of awareness is made possible due to the devices comprising the Fog layer being dispersed over a vast geographical area. As shown in Fig. 1.1, every gateway part of the fog layer manages a subset of the Sensory layer nodes [6]. These gadgets with limited resources will be positioned close to one another, making it simple for the controlling gateway to find each device. Here, location awareness can meet various functional and non-functional needs associated with IoT applications. Some examples of these requirements include mobility and security. Unlike centralized Clouds, fog layers are more widely distributed throughout the sky. This is one of the other closely linked properties of the Fog layer. The concept of centralization in this situation is relative; from the perspective of the client side, the Cloud is a consolidated layer. From the cloud's organizational structure perspective, the servers are globally dispersed; nevertheless, this distribution is not at the scale expected to emerge through the layer of Fog [7]. To provide just one example, consider cloud computing companies like Amazon, which have many data centers in various geographic areas. Because of the relative proximity to the gateways and the widespread nature of the deployment, the fog layer's regional spread presents a unique challenge compared to other layers.

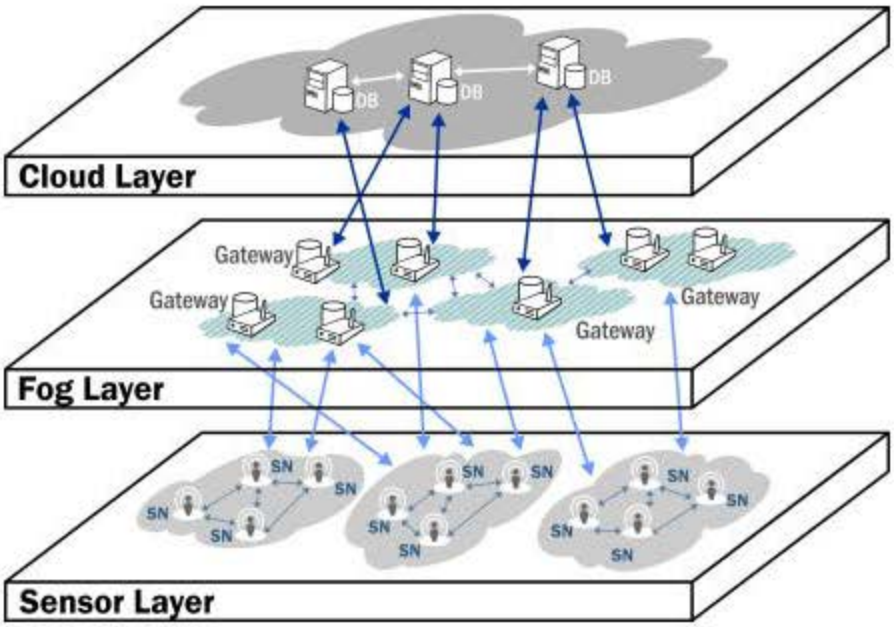

Figure. 1.1: IoT based on Fog: A high-level overview

The benefits of location awareness & widespread geographic dispersion help meet the needs of mobility-based gadgets or "things" operating at the perception layer. Since the Fog layer is so close to the nodes, it is possible to use the perception layer's sensors and actuators in real-time interaction. Among the most notable features of the fog layer is its widespread geographic dispersion, which, in turn, contributes to its ability to provide low communication latency. Specific application fields for the IoT, like healthcare or automotive; rely heavily on such functionality [8].

1.1.4 Organization and Design of the Fog Layer

The Fog layer is capable of being structured productively to fulfill the needs. To begin, consider a network gateway or a wireless hotspot serving customers in the immediate area. The purpose of this kind of gateway is to forward data packets from the network to the Internet-connected back-end infrastructure system. Multiple access points may be organized in the broader environment to provide users with continuous connections across the desired region. This can be done by providing the users with a network that spans the whole environment. So many different types of devices connect to the Fog layer, and it is possible to imagine this layer as a network of gateways that extends across a broader region. In addition to transmitting network packets, these networked intelligent gateways can process or store the data, depending on the context.

The Fog layer is shown in Fig.1.1, and it is the layer in which dispersed Smart gateways communicate with one other, the Cloud, and the sensor layer. Fog layer gateways must have a network interface to support many wireless network protocols.

1.1.5 The main critical aspects of Fog computing

- The ability to pre-process data (filtering, eliminating repetitions, resolving conflicts, gathering, & classifying) to convert it from raw to smooth data, which decreases the amount of the data and enables it to flow on the link and offload the burden for the cloud and offers locality awareness, was made feasible by the large - scale deployment of Fog nodes at the edge of the network near the user.
- Fog may make quick decisions before data is uploaded to the cloud or in an urgent situation.
- Even when Internet connectivity is lost, the Fog nodes may help increase service availability and mobility.
- Data security may be enhanced by the fog applying a policy or encrypting it before sending it to the cloud.
- Fog may also support new use cases, including augmented and virtual reality apps that run in real time.

1.1.6 Comparison between Cloud and Fog

This cloud delivers a tremendous quantity of resources and services to a customer at a cheap cost via three tiers of service (degrees of control), namely:

- Software as a Service (SaaS) enables users to access cloud services like storage, computing, and other functions. A third party offers online software applications; users may only use these programs to handle their documents online.
- Platform as a Service (PaaS) enables users to create and run cloud-based applications that can be accessed anytime and anywhere. This is because some businesses will deploy applications on the cloud so that their staff or clients can use them without worrying about resources, data security, backups, management, etc.

- A user of Infrastructure as a Service (IaaS) has virtual access to and control over specific resources, such as the OS, the LAN, and the hard drive via the Internet. Specific programs need high-performance computing (HPC) before assessing how well they handle large datasets; some applications need virtualized root access to a resource to manage and use it however they see fit.

The fact that all these service levels were developed using vast amounts of raw data that had already reached the cloud would negatively impact the link's capacity, bandwidth, speed, etc [9].

The Fog computing as edge model offered several solutions to deal with these problems, improve cloud quality, and broaden the scope of services that may be provided to end users.

1.1.7 Characteristics of Fog Computing

Computing in the fog is a development of cloud computing that is more specifically applicable to IoT devices. Fig. 1.2 illustrates how fog computing bridges endpoints and the cloud by giving end devices networking, computation, and storage space. Fog nodes are the name given to these gadgets. Anywhere there is a network connection, they can be used. Fog nodes can be considered any device with computing, storage, and a network connection, including industrial controllers, switches, routers, embedded servers, and security cameras [10].

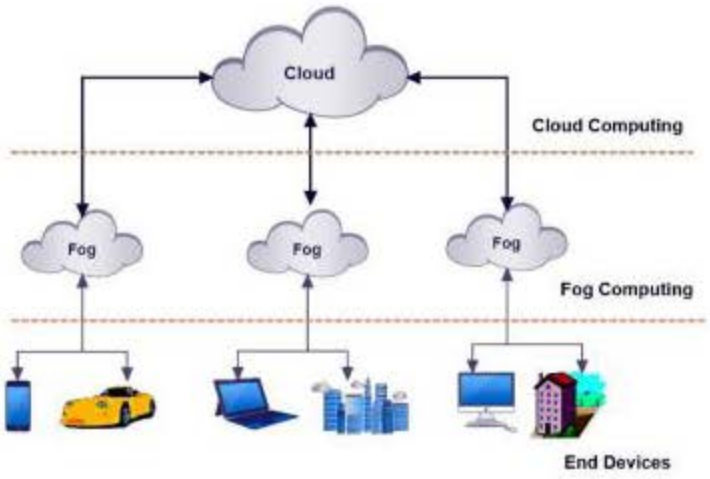

Figure 1.2 A cloud extension that is closer to end-user devices

The fundamentals of the cloud are said to be fog computing. The following succinct description of cloud computing characteristics:

- Low latency and location awareness: These are supported by fog computing, allowing nodes to be placed in various places.
- Geographical distribution: These services & applications are offered. Unlike the centralized cloud, the fog allows for the implementation of solutions everywhere.
- Scalability: Broad sensor networks monitor the environment. Resources for computation and storage are dispersed throughout the fog and can support such large end devices.
- Mobility support: As a significant component of fog applications, it facilitates mobility strategies like location ID separating protocol (LISP), necessitating a distribution to a directory system.
- Unlike cloud computing apps, which employ batch processing, fog computing applications provide real-time interactions between multiple fog nodes.
- Heterogeneity: There is a wide range of possible configurations for fog nodes, also known as end devices, and they must be deployed in compliance with the specifications of the platforms they were designed for. The fog might function on any number of different systems.
- Interoperability: A fog system's components may communicate and cooperate across several domains and service providers.
- Support for cloud engagement and online analytics: The fog, which is in the network's center for ingesting and processing data close to the end users' devices, there must be a connection between the cloud and end devices.

1.1.8 Cases and examples of Fog usage

i) E-health system

This one is among the most significant applications within the smart city context. Using IoT infrastructure, it aims to provide healthcare at higher standards and during emergencies. The patient must wear specific clothing or devices equipped with sensors to monitor vital signs, including pulse rate, hypertension, sugar level, effluence in the air, temperature, and more. This data must be uploaded to the cloud (server provider) for collection and analysis to foresee future threats. The patient will

be informed if there is any unusual behavior or if a health emergency is anticipated. Even in an emergency, a family member or an ambulance may be called.

Because it affects people's lives, this system is susceptible, primarily to delays. Sadly, the cloud cannot accommodate the sudden influx of many functions because of the scheduling of these services from meeting this demand. For this, fog computing is used, which allows it to serve an infinite number of patients concurrently without compromising latency.

The city (smart city) will have a highly populated Fog node network, and patients may wirelessly connect to any node closest to them. All detected data will be sent to Fog to collect and categorize it. Then, instead of transferring enormous amounts of data, It will compile aggregate statistics for all data sets and upload just those statistics to the cloud, where they may be stored indefinitely and studied in the future. This will reduce the link's bandwidth and capacity and improve performance. In addition, if an emergency arises, the fog will be able to recognize it immediately and make the appropriate choice without having to get in touch with the leading cloud.

ii) Traffic organization system

The prerequisites for these structures are rapid responses & tools to gather data immediately to analyze automobiles and the best path based on the traffic volume on each street. A network of Fog nodes that could be embedded in things like connected automobiles or traffic lights is required to operate such a system. Cars will constantly send signals to the nearest traffic signals (Fog node). The node will count items & transmit that transfer valuable information and applications to the cloud, combine the data, make a decision, & send alerts to the fog nodes once again. Finally, cars get them from fog nodes. Another possibility with comparable systems is that a group of intelligent cars might work together to achieve a common objective.

iii) Energy control system

Each appliance in a connected device periodically notifies the server supplier (cloud) of its power consumption rate. However, this causes network congestion and enables the cloud to track the user's habits to provide data on his devices and energy use.

Alternatively, fog nodes might be used to avoid disclosing unnecessary or otherwise sensitive user data by caching data, calculating device-level average consumption, and then sending that data to the cloud. This improves the performance and the utilization

of network and cloud resources.

iv) Face recognition system

Instead of acquiring their system, some organizations utilize this system that depends on the cloud, which has all the data on their workers, to do computations, produce results, and track attendance. Consequently, each employee will upload a picture to the cloud, which will analyze it, run an algorithm, and then provide the results.

There will be a great deal of lag and extra effort for the cloud and network to handle if many workers submit their images through a connection.

This issue may be resolved using the Fog to modify the picture before transmitting it. Instead of sending the whole picture to the cloud, the fog node will perform a feature extraction work on the image. This will free up network resources and shift jobs to the cloud—virtual reality, e-learning, video conferencing systems, and other technologies all leverage related concepts.

There are several more applications that fog may enhance and make more complex, including mobile learning, decision-making, e-governance, shopping, crowd management, etc.

1.2 Architecture of Fog Computing

A "fog computing" technique moves data processing and storage to the network's edge. The fog shares little processing power, storage, and networking capabilities between client computers and the mainstays of cloud computing's server farms. Low and predictable latency is the primary goal of fog computing for time-sensitive IoT applications.

This section provides an overview of architecture specified in software, radio access networks for fog, and fog computing's hierarchical design to comprehend better how fog computing enhances capabilities at the network's periphery regarding processing, communication, storage, & service (F-RAN). One of the most fundamental and extensively used structures in three-tier fog computing is Fig. 1.3, which depicts the building's layout. Following is a discussion of the tiers:

Tier 1–Things/End Devices: In this tier, gadgets connecting to the Internet of Things, such as sensor networks, mobile gadgets used in the European Union (including smartphones, tablets, cards, vehicles, and wristwatches), and more. The

phrase "terminal nodes" is often used to describe these last stops (TNs). These TNs will presumably be GPS-equipped (GPS).

Tier 2–Fog: This is the "computer fog" layer because it mimics the virtual fog's look. Fog nodes consist of the hardware in a network, such as a switch, router, router bridge, and Access points at this level. These fog nodes can cooperate to share resources for computing and storage.

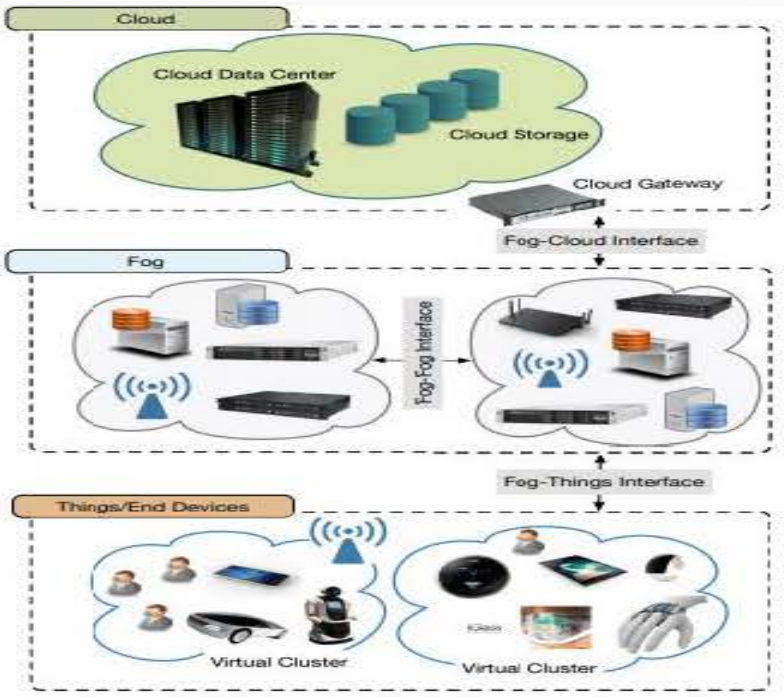

Figure. 1.3: A three-tiered design for fog computing.

Tier 3–Cloud: The highest tier is home to traditional cloud servers and data centers. This tier has an adequate amount of computational and storage resources.

1.2.1 Layered architecture for fog computing

Fig. 1.4 shows fog computing's six layers: Transit, pre-processing, monitoring, virtualization, temporary storage, and security. The physical TNs and the virtual sensor nodes are the main elements of the physical and virtualization layer. The monitoring layer oversees the desired task and keeps an eye on energy usage concerns that are caused by the hardware. Actions connected to data management, filtering, and cutting are carried out in pre-processing—the data is solely in temporary storage for a predetermined amount of time. In the security layer, the problems that pertain to security are addressed and resolved. Finally, the transport layer sends data to the cloud [11].

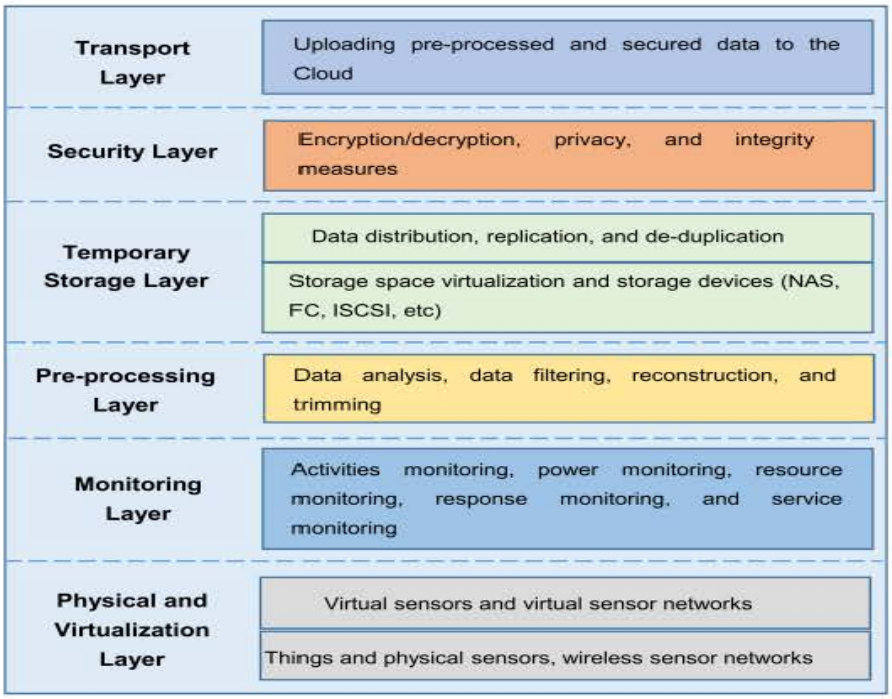

Figure 1.4 Layered architecture of fog computing.

Physical, virtual, and virtual sensor networks are only a few different types of nodes involved in the physical and virtualization layers. These nodes are managed and maintained by determined by the sorts of services they need and the demands placed on them. Various sensors are dispersed around the landscape to detect the environment and transmit the data they acquire via gateways to higher layers, where it will be further processed and filtered. The monitoring layer keeps track of the network's components, resource utilization, and the presence of sensors and fog nodes [12]. This layer keeps track of everything the nodes are working on, including what tasks they are working on, when they are working on them, and what the next step will need them to complete. The functionality and status of all apps and services installed on the infrastructure are kept under surveillance. The energy usage of fog devices is also measured and monitored. Because fog computing uses various devices, each with a unique degree of power requirements, energy management may be both timely and efficient.

The pre-processing carries out the functions of data management. This layer analyses gathered data and data filtering and trimming to extract significant information. Next, the data that have been pre-processed are stashed away momentarily within the layer of temporary storage. After the information has been uploaded to the cloud, keeping

them in local storage is no longer necessary, and they may have the quick storage medium removed from their possession.

The process of encrypting and decrypting data is a function that is carried out inside the security layer. In addition, it is possible to apply integrity safeguards to the data to prevent tampering with them. The next step, which takes place at the transport layer, involves uploading the data already pre-processed to the cloud. This provides the ability to extract and provide more helpful services in the cloud. Only a subset of the captured data is sent to the cloud to make the most optimal use of the available power. The data is processed by the gateway device linking the Internet of Things to the cloud before being uploaded to the cloud. A smart gateway is the name given to this particular sort of gateway. The cloud is the destination for the data sent from smart gateways after being acquired by sensor networks and other Internet of Things devices. The information sent to the cloud and saved there is ultimately used to produce services for end users. To make the most of the fog's limited resources, a communication protocol for fog computing must be as lightweight as possible while maintaining flexibility. Therefore, the fog's application situation should be considered while selecting the communication protocol.

The architecture of FC is also used as inspiration for the design of the fog node. In light of the characteristics of FC, the architecture will be developed in two stages: computing and networking.

1.3 Fog Services Types

Fog services may also be used to achieve fault-tolerant fog computing. When a fog node hosting an IoT smart city application (IoTSCA)-supporting fog service dies, it may be replaced by a comparable service offered by a different healthy fog node situated near enough for the application to continue running without interruption. Depending on the kind of service, there are a few possible routes When optimizing and implementing fault tolerance for a fog service. Certain services need to be considered more than others to achieve fault tolerance. To further comprehend various potential fault-tolerance problems and solutions for these kinds, it is crucial first to explore the multiple sorts of fog services [13].

1. *Stateless vs. Stateful Services*: Fog services vary significantly when managing state data. The most recent state data may affect the operations and results of a

stateful fog service since it is updated with each service invocation.

2. *Non-Real-Time Services Vs. Real-Time Services:* Real-time fog services must act and respond immediately. If not done in time, these activities and reactions are meaningless. If activities or answers are delayed, the service-using application may suffer. Fog computing supports this service better than cloud computing. Fog nodes near IoT devices may improve control by being closer to information and activities. Services updated in real time can do various IoTSCA tasks rapidly and accurately. However, fog computing may allow non-real-time services without time limits; they are more straightforward to fault-tolerate services that are not real-time.

3. *Single-Level vs. Multiple-Level Services:* Single-level services are executed within a fog node and finish their activities without requiring other services, the cloud, or IoT devices. Single-level services at the same fog node may call each other. However, a multiple-level service uses other node services to perform its duties. Cloud, fog, and IoT devices are additional nodes. Another fog node with identical capabilities may handle a single-level service., which is the main difference in failure tolerance.

1.4 Fault Tolerance in Fog Computing

In contrast to the cloud, devices are spread out throughout a network in the Fog computing paradigm, making failure more likely. Since the Fog is still developing, whether or not Fog computers can tolerate faults has not been investigated. However, the cloud computing paradigm is where fault tolerance has received the most research. Most frequently, availability is used to gauge fault tolerance. Proactive and fault tolerance methods are used, depending on whether they are utilized at the workflow or task level in the cloud to handle errors—techniques for reducing the impact of system failures via reactive fault tolerance after they have already happened. Workflow rescue, job migration, task resubmission, replication, rollback and recovery, user-defined error handling, and checkpointing and restarting of jobs are techniques based on this policy. To prevent recovery from faults and errors, anticipating potential failures and swapping out questionable parts with fully working alternatives is what proactive fault tolerance is all about. Cloud computing's limited proactive fault tolerance strategies include self-healing, preemptive migration, and software

rejuvenation.

A system's fault tolerance enables it to continue operating after a component fails. This malfunction might be caused by hardware, software, or a network issue. The consequence of the fault tolerance approach is a fully functional system that operates with reduced capabilities rather than completely shutting down. Much of the current research on wireless networks, distributed computing, cloud computing, and peer-to-peer computing is based on fault tolerance, which primarily employs more resources to cover inevitable accidents or failures. This is particularly true in terms of scheduling [13].

The primary backup approach, which duplicates the fragments of a job and allows each principal subtask to have a backup for increased resilience, is often addressed concerning this subject. If there is a restriction on the number of fog nodes, as would be the case when installing gateways for the Internet of Things, then the system's performance might suffer due to this strategy. This is because the backup resources would be squandered.

Virtual machine (VM) procedures for backup are being thoroughly researched as an alternative to using physical nodes that prevent the occupation of additional physical nodes. One of the premises upon which these techniques are based is that the main virtual machine (VM) of a chunk and the backup VM of that chunk cannot be deployed in the same physical nodes together. This method works well in cloud computing environments that use potent nodes. However, fog nodes' performance may suffer significantly when loading two or more virtual machines due to their limited capacity. Although several overlapping techniques are being investigated to conserve virtual machine resources, these methods still need a compromise between reducing latency and protecting tasks.

1.5 Research Gaps

The researchers have provided a straightforward but all-encompassing technique for bringing multi-part IoT applications to fog networks while maintaining QoS. The reviewed literature reveals that the reliability of the fog node, which is responsible for communication between IoT devices and cloud infrastructure, most published works have not been remunerated [28][29]. The fog paradigm must provide real-time responses and also be able to handle server faults in real time [30]. Fog computing

still faces challenges, mainly when dealing with mobile nodes. Fog computing also tackles an issue of mobility when the tasks originate from ubiquitous mobile applications in which the data sources are moving objects [5]. The mobile user's location and direction should be predicted for allocating the scheduling request to minimize the overall latency [32] effectively. The best cost path optimization is required to reduce packet transmission time [26][27]. The following research gaps(RG) have been designed based on the literature survey, which can also be explored in terms of the IoT-Fog framework:

RG1. Existing Fault Tolerance & Service Placement techniques were based on static priority and did not deal with real-time server faults. The reliability of the fog node has not been taken into consideration.

RG2. The issue of mobile user's request allocation needs to be addressed.

RG3. QoS parameters should be optimized for packet transmission time minimization.

1.6 Research Objectives

To fulfill the above-mentioned research gaps, this research aims to design an adaptive fault-tolerant framework and latency-aware service placement scheme for a fog computing environment. The research objectives as framed to achieve this task are as follows:

1) To develop an Adaptive fault tolerant service by using reactive techniques.

2) To design a mobility-aware service placement for allocation of service.

3) To optimize network path in fog environment.

1.7 Thesis Organization

This research uses the following segments to represent the means and schemes for adaptive fault tolerance framework and mobility-aware and latency-sensitive scheduling algorithms in the Fog computing environment.

Chapter 1 Introduction: This chapter provides an overview of fog computing. It also covers the study's contributions and new problems in fog computing.

Chapter 2: Fog Computing Environment- the State of the Art: The classification of IoT and fog computing is covered in detail in this chapter. The research and work

of other researchers on handling errors in fog computing are discussed and analyzed in this chapter.

Chapter 3: CR-BM-based Framework for Fault Tolerance :

The framework projected in this chapter carries out adequate Quality of Service (QoS) conscious practices built on the amalgamation of Checkpoints and Replication (CR) for fault diagnosis, and the Bee-Mutation (BM) algorithm has been employed for efficient allocation of service to the best fog devices. The fog service monitor has been set up to track the effectiveness of the nodes in the fog layer. The CR and BM modules analyze the service monitor in cooperation for the proposed hybrid architecture to be fault-tolerant. Additionally, the suggested CR-BM pedestal hybrid structure performance has been assessed using a range of performance metrics.

Chapter 4: Efficient Mobility Aware Scheduling Algorithm

This chapter proposed an adaptive service discovery and mobility-aware scheduling algorithm presented. Simulation results showed that the proposed algorithms can provide uninterrupted service to IoT devices.

Chapter 5: Particle Bee Optimization for Fault Tolerance

In the fog-supporting scenario, this chapter provided a helpful strategy that increases overall performance by determining the optimal cost path and shortens packet transmission time by choosing the nearest neighbor among fog nodes. The proposed approach also protects against live server failure by repeating server activity.

Chapter 6: Conclusion and Future Directions: This chapter summarizes the findings and plans for fault tolerance and QoS-based frameworks in the fog computing environment.

CHAPTER-2

REVIEW OF LITERATURE

2.1 Introduction

Fog computing is a kind of distributed computing that uses "edge" nodes to access cloud-like services. In fog computing, "Fog" is the same as everyday life. It is possible to encounter fog between clouds and the ground in the real world, and this concept is employed in fog computing to symbolize the region between clouds and the physical world. In other words, fog nodes are placed in the path that connects the cloud to the user's endpoint devices. The cloud and fog concepts appear relatively comparable regarding management, computing, communication, and data storage. However, there are a few crucial differences. Fog is tailored to use cases that need near-instantaneous responses but can still function with a little more delay. However, the cloud has application performance limitations and is susceptible to location and latency because of its centralized nature and users' propensity to spread out over enormous distances. Fog computing, like cloud computing, enables the use of several forms of service provisioning, such as "Software as a Service," "Platform as a Service," and "Infrastructure as a Service" (IaaS). The following features, exclusive to fog computing, set it apart from other approaches to computing:

- Fog nodes know their logical position within the more extensive system and the latency costs associated with connecting with other fog nodes; fog computing has the lowest possible associated latency. Because fog nodes are positioned close to end-user devices, the processing of incoming data and response times of these devices are significantly faster than those of cloud services.

- Compared to the centralized cloud, services and applications operating in geographically distributed divisions are excellent candidates for fog computing backgrounds.

- Fog computing allows maintaining and processing a variety of data kinds via the use of various forms of network communicative capability.

2.1.1 Comparative Study of QoS and Service Placement based schemes

This section carried out a review based on a series of parameters that the various

researchers considered for all edge paradigms. Table 2.1 analyzes these parameters based on which research gaps have been found and objectives derived.

Table 2.1: Issues addressed in various paradigms

References	Comparative Study					Paradigm's
	Fault Tolerance	Mobility	Service Placement	QoS Considerations	Latency Sensitive	
Luiz et al. [13]	✖	✓	✖	✖	✖	Fog
Wang et al. [14]	✖	✖	✓	✖	✖	Fog
Satria et al. [15]	✓	✖	✓	✖	✖	Mobile Edge
Verba et al. [16]	✖	✖	✓	✖	✓	Fog
Liu et al . [17]	✖	✖	✖	✖	✓	Fog
Baranwal et al . [18]	✖	✖	✓	✓	✖	Mobile Edge
Kimovski et al . [19]	✖	✓	✓	✓	✖	All edge Paradigm's
Ghosh et al . [20]	✓	✓	✖	✓	✓	Cloud, Fog, IoT, and Edge
Waqas et al . [21]	✖	✓	✖	✖	✖	Fog
Alarifi et al. [22]	✓	✖	✖	✖	✖	Fog and Cloud
Souza et al. [23]	✓	✖	✖	✖	✖	Fog and cloud
Skarlat et al .[24]	✖	✖	✓	✓	✖	Fog
Wang et al . [25]	✖	✓	✖	✖	✖	Fog
Verma et al . [26]	✖	✓	✓	✖	✖	Fog
Mahmud et al . [27]	✖	✖	✖	✓	✓	Fog

2.1.2 Fog Computing Frameworks and Fault Tolerance Schemes

Different research studies consisted of a full investigation of various frameworks and fault tolerance schemes in a fog computing environment are presented in this section. In-depth taxonomies, performance metrics, and parameter analyses were also given.

In [28], Al-Khafajiy et al. proposed an IoT-based fog computing framework to enhance the QoS for IOT apps. This proposed system allows for collaboration amongst Fog nodes at a particular location to make data processing possible in a shared manner. This meets the quality of service requirements and provides for the most significant number of service requests to be fulfilled.

In [29], García-Valls et al. discussed designing and validating a framework that improves the service time of chosen activities at the fog servers. More specifically, this improvement increases the service time of those activities that distant patients request. It does this by taking advantage of the capabilities of modern processors to parallelize activities that can be executed on reserved cores.

In [30], Verma et al. presented a novel architecture in the Fog Computing environment based on a scheduling algorithm to handle the increasing number of final users with less complexity and more efficiency. The issue of network delay is overcome by an innovative kind of computing paradigm that is known as fog computing. The main distinguishing feature of fog compared to cloud computing is that it operates on the periphery of a network. One additional benefit is that it makes it possible to implement location-dependent applications.

In [31], Mazumdar et al. have developed a collaborative strategy to provide on-time service; other fog nodes must take up some of the work of a single node struggling under strain. A three-layered IoT-fog-cloud system framework is presented to overcome this limitation and satisfy the needs of real-time programs. An evaluation of the proposed framework work is performed by a simulation study using performance indicators such as latency and service response rate.

In [32], Apat et al. discussed the role and importance of the generic three-tier architecture's fog orchestration node. The primary goal of this design is to facilitate the equitable allocation of workloads among a system's resources (physical equipment, software, network bandwidth, etc.) that are dynamically associated with the Internet of Things (IoT). In addition, the author discussed a few difficulties related to resource management that need to be accounted for to provide improved QoS to customers.

In [33], Wen et al. mentioned a Byzantine fault-tolerant networking strategy and two resource allocation algorithms for IoT fog computing. They aimed to create a

protected fog network that they dubbed SIoT Fog. This network would withstand Byzantine faults and make the transmission and processing of IoT extensive data more efficient. They focused on latency as a metric for examining the simulation's findings. The findings of the simulation indicate that their tactics may be able to contribute to the development of an effective and dependable fog network.

In [34] Araujo et al., the researcher presented a resilient agent-based fog computing architecture. This architecture utilizes machine learning and a statistical model to forecast the time until an instance is revoked. It also contributes to the refinement of fault tolerance parameters and the reduction of total execution time. Experiments showed that their model is accurate to a great degree, attaining a success rate of 94%, which shows that it is useful when applied to realistic settings.

In [35], Samann et al. recommended a Fault Tolerant Data Management (FTDM) method in healthcare IoT fog computing. FTDM effectively organizes and manages the data produced by IoT devices in the healthcare industry through clearly defined components and stages. The research included energy usage, execution cost, network utilization, latency, and execution time as performance evaluation factors. The simulation results show that the suggested FTDM technique lowers heat output by 3.97 times, cost of execution by 5.09 times, network usage by 25.88 times, latency by 44.15%, and time to execute 48.89 times. When compared to the existing Greedy Knapsack Scheduling (GKS) strategy. The recommended approach is thus quite successful in these circumstances.

In [36], Bakhshi et al. explain how container-based architectures are often used for cloud computing and how they may play a crucial part in creating fog computing infrastructures. The authors have examined the difficulties of implementing containerization at the fog layer and have chosen to concentrate on one of those difficulties: the supply of fault-tolerant persistent storage. In addition, the article provided a container-based fog architecture that uses so-called storage containers. These containers combine the fault-tolerance techniques already built into storage units using a decentralized consensus mechanism to guarantee data integrity.

In [37], Tong et al. presented a brand-new short-circuit diagnostic method with a high degree of diagnosis accuracy, a strong fault tolerance, and minimal delay time. Wide-area backup protection data is transformed into targeted information with tiny

package sizes and transported between the corresponding fog nodes to generate incidence matrices in each node. Utilizing the matrices for the location of the defect, each fog node may determine its diagnostic value. The findings demonstrate that the technique can maintain 100% correctness in all three mistake scenarios and provide a reduced latency when there is a significant network load.

In [38], Ozeer et al. suggested a fault tolerance technique that considers these three features of the fog-based IOT environment. To achieve fault tolerance, the application's state must be saved in a manner that is not coordinated. Notifications are sent out if a fault has been identified; this helps mitigate the effects of failures and dynamically modify the program. The data saved throughout the process of conserving the state are utilized for recovery, and this procedure considers the data's consistency concerning the physical world.

In [39] Naim et al. suggested that one of the paradigms for delivering high availability in computerized systems where application service is replicated to several nodes is called "fault tolerance." However, resident on-site applications, such as those found in small clinics, are still susceptible to outages and would need a certain amount of time and human intervention to recover. The middleware will detect failure through heartbeat checks, and simultaneous service replication will occur. If an outage is identified, the middleware will take the necessary steps to assume its role as the secondary service provider.

Table 2.2: Research Gaps related to various Fog Computing Frameworks and Fault Tolerance Schemes

References	**Features**	**Research Gaps**
AI-Khafajiy et al .[28]	Fog computing framework for enhancing QoS, but the reliability of fog nodes has not been considered.	RG1, RG3
García-Valls et al. [29]	Framework improves service time only.	RG1
Verma et al. [30]	Architecture overcomes network delay issues, doesn't explore, and gives closer attention to the processing of the fog node.	RG1
Mazumdar et al. [31]	A three-layered architecture is needed to overcome the issue of real-time programs, but aspects of fault tolerance have not been explored.	RG1
Apat et al. [32]	A layered architecture for allocating workload among the resources but not classifying the actual resource requirement.	RG1,RG2

Wen et al. [33]	A fault-tolerant strategy for developing an effective fog network didn't explore other parameters, i.e., response time, throughput, and associated overheads.	RG1,RG3
Araujo et al. [34]	A fault-tolerant-based architecture reduced execution cost only.	RG1
Samann et al. [35]	Fault-tolerant data management handles malfunctioning tasks and nodes but does not explore the QoS parameter.	RG1,RG3
Bakhshi et al. [36]	A fault-tolerant base using a decentralized consensus mechanism to preserve data integrity only.	RG1
Tong et al. [37]	A short circuit diagnostic-based fog framework has strong fault tolerance and minimal delay but is specific to short circuits and doesn't explore other features.	RG1
Ozeer et al. [38]	A fault-tolerant-based technique has been proposed for preserving data consistency concerning the physical world. This technique can be improved by providing fault monitoring and more sophisticated fault detection services.	RG1
Naim et al. [39]	A fault tolerant-based mechanism that provides the advantage of automated backup for applications. The proposed method is limited to delivering recovery only.	RG1

This section analyzed the fog computing environment's different architectures, frameworks, and fault-tolerant schemes. The authors also examined the essential qualities of fog computing frameworks and highlighted various problems associated with their architectural design, service measure quality, service placement, and communication modalities. Most IoT frameworks enclose sensing devices that monitor the environment in which it is established. The usage of IoT in an automated environment monitors complex systems. The scalability of the fog system provides an increased probability of failures. In some instances, hardware faults and software bugs can adversely affect the system's reliability and performance. To address these, a user-transparent and redundant replications fault-tolerant deployment and execution approach is needed. Hence, there should be a module for fault diagnosis to detect the fault accurately in the automated system. This fault-diagnosing model may save the system and humans from any catastrophic events.

2.1.3 Mobility Aware Scheduling Schemes

The articles understood the research concerns via a thorough taxonomy and identified significant obstacles in previous work. The articles also looked at current approaches to several problems, presented a meta-analysis based on QoS metrics, and described various tools for implementing Fog task scheduling algorithms. According to this systematic review, potential researchers found it simple to pinpoint particular research issues and future paths to improve scheduling effectiveness.

In [40], Kaur et al. have presented a mongrelized load corresponding approach (Tabu-GWO-ACO) for scientific processes and a Fog Figuring Infrastructure of Task Scheduling (FOCALB) for such presentations combines tabu search, Grey Wolf Optimization (GWO), and Ant Colony Optimization (ACO). The author summarized the findings and compared the simulated experiments' execution times, costs, and energy use to those of several current models, demonstrating that FOCALB minimizes the energy consumed by fog nodes, speeds up processing, and costs less to install costs.

In [41], Wadhwa et al. have created a novel scheme for efficient placement and organization of computational and memory. TRAM is a novel scheme for proficient allocation of resources and their management and also improves the system's overall performance. It is recommended to ensure the consumption of resources at the fog layer. The method tracks the degree of job concentration using the expectation maximization (EM) algorithm and determines the current resource condition. The experimental findings showed that TRAM effectively reduces task execution time, network use, energy use, and average loop delay.

In [42], Kochovski et al. addressed and developed a practical technique for using AI in construction projects through the DECENTER Fog Computation & Brokerage Platform. The researcher created a Mobility-aware Genetic Algorithm (MGA) for fog service placement. This MGA aims to facilitate the mobility of nodes while preserving the energy efficiency of the infrastructure and the Quality of Service (QoS) requirements of the applications. The projected framework offers promising energy reduction and delay violation results compared to alternative strategies.

In [43], Science et al. suggested a data offloading mechanism that considers deadlines and implemented it on fog nodes. After evaluating these parameters, the node

controller offloads the job to the appropriate fog node. According to the experimental findings, the suggested method achieves an overall delay time of 24% and 17% shorter than the two benchmark algorithms. The deadline-aware data offloading technique achieves a 79% likelihood of achieving the deadline for time-sensitive applications. Task scheduling and offloading strategies are the focus of the research; however, the impact of scalability has not been explored.

In [44], Choudhari et al. proposed the architecture, queuing & priority model for the priority assignment module. Presentation analysis demonstrates the suggested technique significantly lowers total cost and overall response time compared to other job scheduling algorithms. They think this research is essential for the development of fog computing, and various application fields may benefit from the priority-based approach.

In [45], Murtaza et al. contributed by adding streamline moment variable IoE requests. This method specifically studied the different IoE service request types and proposed an ideal technique for allocating the best available fog resource in accordance. They thoroughly assessed the effectiveness of the suggested strategy via simulation and its accuracy using rigorous checking. In terms of both energy efficiency and service quality, the findings of the examination are encouraging (QoS).

In [47], Mahmud et al. proposed a context-aware approach to application request allocation in the fog pedestal milieu is needed to deal with the problem of varying data sensing frequencies from different industrial Sensing devices; a problem made more complicated by the sheer volume of data generated by these sensors. The new policy's performance is compared to the current placement policies in real-world and simulated Fog scenarios. The experiment's findings demonstrate that, compared to alternative placement rules, this method delivers an overall 16 percentage point reduction in service delays, network easing, and compute load control.

In [48], Bahnasawy et al. recommended a brand new algorithm that was going to be known as the Deadline Aware Resource Allocation (DARA) algorithm. The method is contrasted with the DRAM algorithm, which stands for the Dynamic Resource Allocation Method. The simulation results demonstrated that the suggested algorithm offers superior performance regarding the overall cost, the amount of resources used, and the total amount of money spent.

In [49], AI-Tarawneh et al. presented a bi-objective Strategy for deciding where to deploy applications in fog computing settings. The proposed method tries to install software applications on the base fog devices in a way that considers the criticality of the applications involved and any security criteria that must be met. The suggested algorithm's capacity simulation findings show that optimizing application location in fog computing environments concerning performance metrics and efficiency is possible.

In [50], Pham et al. presented the Price-aware Scheduling heuristic as a scheduling method. The key objective is to find a middle ground between the required cost of using cloud resources and the speed with which applications may be executed. This tactic aims to improve the schedules generated by the Cost-makespan-aware Task scheduling to conform to any time limits or quality of service standards set by the system's users. The experimental findings demonstrate that their scheduling methodology is more cost-effective and produces more incredible performance than other methods.

In [51], Caminero et al. provided a network-aware scheduling technique to select the most appropriate fog node to execute an application within a specific time. The comparison demonstrates that their approach is the only one that, with specific settings in some of the analyzed conditions, can execute all of the submitted tasks within their deadlines, producing an optimal solution. Only their proposal can complete all the work by the deadline.

In [79], Maiti et al. said they have considered accesses as possible people who may benefit from a fog node being deployed in their vicinity. This work optimized the number of fog nodes deployed to reduce the overall delay caused by data aggregation and processing. Compared to traditional approaches to Internet of Things data processing in traditional cloud environments, their findings indicate that doing so with a Latency in the Internet of Things (IoT) network might be reduced with the proper placement of fog nodes.

Several open research issues and intriguing future paths are mentioned as potential focus areas for more study in fog computing. The scheduling issues and challenges arising due to user mobility in the fog-cloud system's hierarchical environment should be considered. There should be a need to take different scenarios of user mobility at

edges based on their requirements. The response time sensitivity of their requests was also considered for allocating the service requests of devices located at the cloud edge to the most appropriate actuators.

Table 2.3: Research Gaps Related to Mobility-Aware Scheduling Schemes

References	Features	Research Gaps
Kaur et al. [40]	The task scheduling-based approach only focuses on energy consumption minimization.	RG2
Wadhwa et al. [41]	A novel scheme for the proficient allocation of resources has been developed but does not take care of the mobility-aware environment of fog computing.	RG2
Kochovski et al. [42]	Mobility algorithms focus on service placement specific to construction projects.	RG1,RG2
Science et al. [43]	A data offloading-based scheduling mechanism only takes into account the work's deadline.	RG2
Choudhari et al. [44]	The proposed task schedulers significantly lower the total cost and overall response time.	RG2
Murtaza et al. [45]	This scheduling scheme allocates the best available fog resource based on its accuracy and rigorous checking.	RG2
Nguyen et al. [46]	The proposed scheduling strategy considers only execution time and cost.	RG1,RG2
Mahmud et al. [47]	A context-aware approach to request allocation minimizes service delays and balances the load.	RG2
Bahnasawy et al. [48]	A Deadline resource allocation algorithm minimizes overall cost but is silent about the other essential parameters, i.e., User Mobility and location prediction.	RG2
AI-Tarawneh et al. [49]	The proposed strategy provides efficient service placement, but the user's mobility is not considered.	RG2
Pham et al. [50]	A price-aware scheduling strategy has been presented to minimize cost only	RG1,RG2
Caminero et al. [51]	The proposed scheme is based on a network-aware scheduling technique, and only the network's current stage was considered.	RG3

2.1.4 Quality of Service (QoS) bases schemes

A literature review was provided in this section emphasizing QoS approaches established in academia for Internet of Things applications. This work also reviews and explores current trends in research. An in-depth look into the relevant literature will precede the presentation of background information on the Internet of Things,

quality of service measurements, and essential enabling technologies. This section also indicates that the most frequent quality of service measures to consider are latency, reliability, throughput, and network use. The examined studies took into account the indicators that were relevant to their respective provisioning strategies. The researcher concluded that the primary emphasis of academic research has been placed on application-specific QoS provisioning strategies.

In [52], Wang et al. suggested a qCon framework for managing network resources for containers that are QoS-aware to control the amount of outgoing traffic in fog computing. While developing a lightweight framework, qCon strives to offer load balancing and capacity shaping to meet containers' varying performance needs. For this reason, qCon enables the simultaneous application to multiple containers of the three scheduling rules: Minimum and maximum bandwidth reservations and proportionate share scheduling are all used.

In [53], Shaheen et al. suggested a simple structure that uses the idea of a fog head node, which monitors additional fog nodes regarding user registrations and location awareness - The recommended LAFF offers a precise location-aware method that consistently meets QoS. This suggested framework surpasses IFAM and TPFC, which target IoT applications when RAM and CPU use are considered. The framework is more lightweight since it uses 8.41% less RAM and 16.23% less CPU power than TPFC and IFAM.

In [54], Aujla et al. suggested an ensemble QoS-aware traffic flow control technique in SDN's built architecture. The proposed method operates in three stages: the first stage involves designing a linear ordering scheme to remove dependencies on incoming packets; the second stage consists of creating an application-specific traffic classification scheme; and the third stage involves designing a queue management scheme to schedule the flow of traffic efficiently. An experimental environment assesses that the suggested method performs well regarding various QoS factors.

In [55], Yao et al. tackle this joint optimization issue, which is presented as a mixed integer nonlinear programming (MINLP) problem. This problem aims to reduce the system cost (VM rents) while simultaneously ensuring QoS criteria. After that, a method for approximating the issue is suggested as a solution. The results of the simulation show the performance of their proposed approach.

In [56] Kan et al. addressed the critical challenges of offloading decision-making and resource allocation in MEC systems, including radio and computing resources. Their work is intended to improve QoS and outcome; this work models the QoS experienced by end users using a cost function that they devised independently. This allows us to establish a cost-minimization issue. Based on the numerical findings, their suggested approach significantly improves quality of service (QoS) while gaining a considerable performance advantage over competing systems.

In [57], Hosen et al. presented a Proxy-based clustering routing (QACR) for route selection. To balance the energy consumption of nodes, a cluster's number of CHs is adaptive and varies based on cluster condition. The QACR has been evaluated through simulations for different circumstances. The obtained findings demonstrate that, in comparison to the current protocols, the QACR enhances the quality of service concerning packet delivery rate, network latency, and throughput lifespan.

In [58], Yousefpour et al. introduced a FOGPLAN-based framework for dynamic fog service provisioning that is QoS-aware (QDFSP). This is done to fulfill the quality of service and low latency needs of applications while keeping costs minimal. They addressed the QDFSP in one instance by presenting a potential formulation of the issue (in the form of an optimization problem) and two effective greedy methods for solving the issue.

In [59], Tuli et al. discussed a performance engineering technique that is being built as part of the COSCO framework for improving the quality of service in edge/fog/cloud computing by use of artificial intelligence and coupled simulation.

In [60], Bidi et al. made their first contribution by proposing a fog-based fault-tolerant architecture. While effectively and quickly responding to user demands for service composition, this architecture overcame the scalability and reliability concerns common in smart cities. These metaheuristics were based on artificial bee colonies, genetic algorithms, and particle swarm optimization. The findings that were achieved based on the produced prototype of their proposal suggest that their proposed strategy is preferable to other current options for service composition. Specifically, the comparison was made using these results.

In [61], Guevara et al. presented a set of CoS for use with fog applications. This set includes the QoS criteria that most accurately represent fog applications. An

illustration of the implementation of this technique may be found in the evaluation of classifiers concerning their effectiveness, precision, and resistance to noise. Here, Fog computing relies heavily on distributed computing. In addition, the decision-making process for the fog scheduler may be simplified by categorizing the many uses for fog computing.

In [62], Guo et al. explained that the software-defined network (SDN) technique, which is renowned for its adaptability and elasticity, has been incorporated into the Internet of Things (IoT). This technique can extract information from historical traffic needs by interacting with the underlying network environment. Extensive simulation studies have been carried out regarding several different performance indicators for the network, and the results have shown that this DQSP has excellent convergence and a high level of efficacy

The researchers have provided a straightforward but all-encompassing technique for bringing multi-part IoT applications to fog networks while maintaining QoS. This section looks at several techniques that may be used to determine whether an application is suitable for deployment in a fog environment. The researchers also analyzed how fog computing may enhance network resilience and delay analysis experienced by network traffic.

This review also concluded that using fog computing makes the edge networks more robust to difficulties in the core network and prepares the network for a better response time if interactive requests are made. This section's articles investigated various QoS assurance techniques: administration of services and resources, communications, and applications. Regarding the chosen methodologies, the study illustrated their benefits, drawbacks, tools, assessment kinds, and QoS aspects. Based on the papers examined, the outstanding topics and difficulties that need additional study and research in QoS-aware techniques in fog computing are illustrated in Table 2.4.

Table 2.4: Research Gaps Related to QoS Aware Schemes

References	Features	Research Gaps
Hong et al. [52]	A framework (qCon) has been projected for controlling outgoing traffic only in fog computing.	RG3
Shaheen et al. [53]	A location-aware method that consistently meets QoS requirements but does not explore the dynamicity of end-user	RG2, RG3
Aujla et al. [54]	A QoS-based traffic flow mechanism has been described. Packet transmission time analysis has not been introduced.	RG3
Yao et al. [55]	The projected scheme has reduced the system cost while ensuring QoS criteria but didn't explore the other essential parameters, i.e., latency, associated overheads, and availability.	RG3
Kan et al. [56]	The projected scheme tackles the issue of cost minimization only, specifically designed for the mobile edge computing paradigm.	RG3
Hosen et al. [57]	Proxy-based clustering routing (QACR) enhances QoS concerning packet delivery rate, network latency, and throughput. The reliability of packet flow should also be ensured by employing some fault tolerance mechanisms.	RG1, RG3
Yousefpour al. [58]	A dynamic fog service provisioning framework has been devised to minimize the cost and latency requirement.	RG3
Tuli et al. [59]	The proposed strategy improved QoS in cloud/edge/fog and did not explore the application dynamicity and	RG3
Bidi et al. [60]	Fog-based fault tolerant architecture is specifically designed by mapping to a generalized traveling salesman problem, not exploring other IoT fog environment issues.	RG1, RG3
Guevara et al. [61]	The analytics-based approach did not explore fog-based application dynamicity.	RG2, RG3
Guo et al. [62]	A Software Defined Network(SDN) - Internet of Things (IoT) architecture for improving the routing strategy and assuring QoS. Not specific for fog environment.	RG3

2.2 Summary

The high dynamic and heterogeneous nature of devices at the cloud edge causes failures to be a popular event, making fault tolerance indispensable. Most early scheduling and fault-tolerant methods did not highly consider time-sensitive requests. This increases the possibility of delays in serving these requests, which causes unfavorable impacts. As projected in Table 2.1, various fault tolerance schemes

depicted disclosed that no schemes or policy in the published works paid closer attention to the reliability of the fog node, which is responsible for communication between IoT devices and cloud infrastructure; significantly less work has been done in the field of fault tolerance so there is a need to propose a framework for providing a fault tolerance service in the fog-cloud environments. In [42][43][44][45], it was depicted that the scheduling issue of the request of IoT devices arises due to the user's mobility in the hierarchical environment of the fog-cloud system. There should be a scheduling method for allocating service requests of devices located at the cloud edge to the most appropriate actuators. It can also classify the requester devices according to the response time sensitivity of their requests and scenarios of user mobility at edges. The work in [60][61] is intended to improve QoS and outcome; this work models the QoS experienced by end users using a cost function optimization. This literature review disclosed the issue of cost minimization.

CHAPTER 3

CR-BM-BASED FRAMEWORK FOR FAULT TOLERANCE

3.1 Introduction:

The exponential increase in Internet of Things (IoT) technology has led to digitizing all aspects of life. The number of connected devices is expected to reach 50 billion devices. IoT devices range from household devices to industrial, autonomous transportation, smart cities, and environmental monitoring sensors/actuators. All these devices must be connected to the internet to meet their prospective. With its services (platform, infrastructure, and software), cloud computing can be a promising option for IoT devices.

However, cloud computing cannot handle the massive growth. Contacting cloud data centers is considered to be expensive and raises network bottleneck congestion issues. Furthermore, the delay tolerance of IoT devices varies depending on how critical they are. For example, healthcare and traffic-controlling IoT devices require lower response times than household IoT devices. At this level, the need for a decentralized computing architecture has emerged. In 2012, Cisco introduced the term fog computing as an expansion of cloud computing, which mainly focuses on bringing data processing geographically closer to the data source. Open Fog consortium defines fog computing as "a horizontal system-level architecture that distributes computing, storage, control, and networking functions closer to the user along a cloud-to-thing continuum." This augmentation allows the distribution of the workload over widespread computing resources to reduce response time and bandwidth consumption, resulting in a higher quality of service.

One of the primary motivations for implementing fog computing is to provide an acceptable level of QoS in cloud computing platforms for location-aware, latency-responsive applications. Furthermore, fog computing needs QoS to assess and monitor the services appropriately supplied. Some implemented QoS measures in cloud computing did not apply to fog structure due to the diverse properties of fog computing, such as device diversity, geographic spread, and mobility [62]. Existing research indicates that in future QoS-based fog settings, QoS characteristics can be

represented as execution and response time, resource utilization, security, execution cost, trustworthiness, energy convention, availability, scalability, & throughput must be maximized.

Although QoS is a significant problem in fog computing, there has been no systematic or universal study on its fundamental techniques. As a result, this work undertook thorough research to identify, taxonomize, and systematically compare previous studies concentrating on QoS in fog computing. Briefly, the following are the study's primary contributions:

- Providing a thorough and systematic examination of the available QoS providing options in fog computing.
- Providing a technical categorization for QoS-aware techniques in fog computing.
- Introducing the current difficulties around the Quality of Service (QoS) issue in fog computing.
- Identifying primary focus for future research and development on QoS-aware methods.

3.2 Fault tolerance:

When a computer system or network device has fault tolerance, it may continue to function normally despite a bug or a series of bugs in the code that runs the system. Several reliable measures are included in the fault tolerance to forestall malfunctions like this. The availability and reliability of a fault-tolerant system are maintained even if one or more faults or failures occur in the system's components, allowing for the provision of the service at hand. Tolerant bug systems are those that can continue functioning despite the presence of an error. It might help to start with a description of error, as the mind often associates it with malfunction and failure. However, there are three key distinctions[63].

3.2.1 Failure

If the system's misbehavior allows it to adequately fail at least one of its capabilities, then it is malfunctioning and has failed to perform as planned. It turns out that a flaw in the system is to blame for the breakdown. Therefore, the issue is physical or results from a failed piece of hardware or software. A flaw in the system that allows errors to

occur is called a bug. It is possible for software, virtual computers, and even hardware to malfunction, fail, or have flaws. There has to be a way for the system to recover from the error and keep running normally. The fault line is shown in Fig. 3.1.

Figure 3.1: The path to failure

3.2.2 Fault types

The hardware, virtual machines, and application layers make up the three tiers of the cloud platform. Every one of them is broken. These errors may occur when the software runs on any layer of hardware or virtual computer. Therefore, the proper course of action should be taken, given the nature of the failure. In cloud computing, errors may be divided into several types.

Network problems: Network faults are errors that involve networks. This error happens when the data is not intended for whatever reason, such as packet loss, a closed date, a failed destination, a broken connection, etc. Hardware flaws, such as CPU crashes, memory crashes, crash failures, etc., are physical flaws.

Media fault: Errors brought on by a lack of communication tools.

Processor issue: An operating system malfunction caused a fault in the processor.

Process faults: are errors brought about either by a shortage of resources or software flaws.

Service termination fault: The application still needs to access the resource even after its expired service life.

Transient: This kind of defect only ever emerges once and lasts a very long period before dissipating when the measures are taken. For instance, a network communication from the origin to the destination may initially be unable to reach its intended recipient but eventually succeeds.

Alternate: faults are distributed alternately and again. These failures are not good, mainly because each component failed or malfunctioned concerning the other components, such as a bad connection.

Stable: Even when broken systems are fixed or, in certain circumstances, entirely replaced, this form of failure persists in the system.

3.2.3 Fault tolerance types

Hardware and software fault tolerance are the two categories into which fault tolerance may be divided. Fig. 3.2 displays this group:

3.2.3.1 Hardware fault tolerance

For a computer system to be fault tolerant, it must be able to recover on its own if many random failures occur in its hardware components. Partitioning a computational system into many modules is a common theme in the approaches proposed for this study. There has always been a backup for every part of the system. As a result, if a problem arises with one module, the other module may go on usually. Fault-tolerant techniques consist of two varieties: error handling and dynamic recovery.

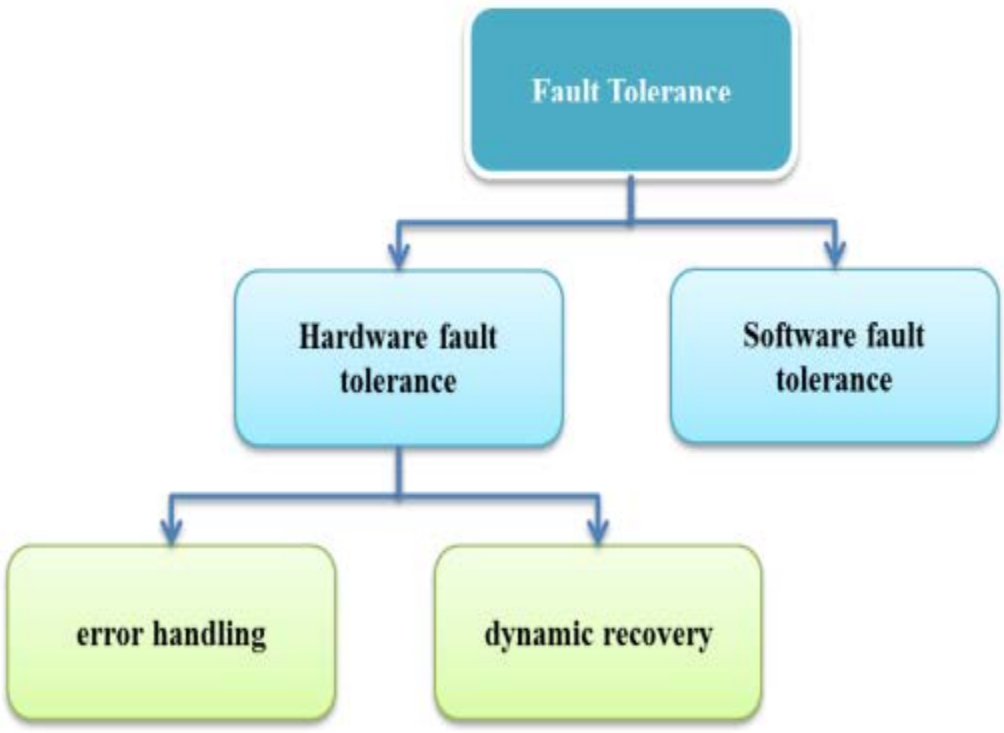

Figure 3.2: Types of Fault Tolerance

Fault coating is a structural redundancy method that removes defects in a collection of mixed components. A malfunctioning module's flaws are eliminated by voting on the output of many identical components that carry out comparable tasks. Dynamic retrieval: This method is only used when a single copy of the work or computation is

produced to execute. This method promotes self-repair. Additional spare components are employed to carry out backup activities, similar to a fault-coating strategy (preventive redundancy).

3.2.3.2 Software fault tolerance

Similar to how hardware faults are handled, software faults (programming flaws) may be exploited using static and dynamic techniques. N-version programming, which employs static redundancy in the form of separate programs, is one of these techniques. They are all acting in the same way.

3.3 Fault Tolerance Policies in Cloud Computing

Following is a classification of several methodologies and tactics based on fault tolerance [63][64]:

3.3.1 Tolerable Preventive fault

The action-oriented fault is the tolerance policy; it prevents errors and failure by foreseeing them and substitutes the doubtful component. It finds the issue before it arises. Preventive migration, software rejuvenation, self-healing, and other methods are some of the strategies based on this strategy.

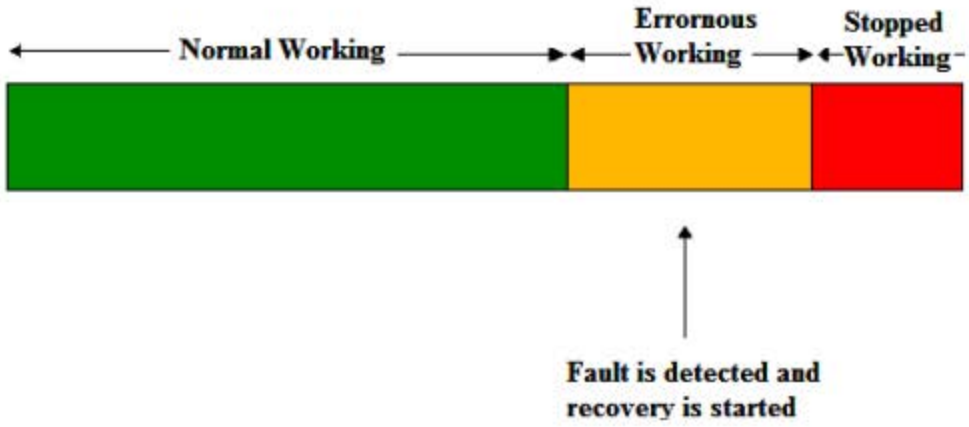

Figure 3.3: Timetable for a precautionary fault detection system

Software rejuvenation is a process that creates the mechanism for reloading regularly. This method facilitates a fresh start by returning the system to its initial condition.

Migration Prevention: Migration Prevention relies on the feedback loop control system. The application is constantly watched and examined.

Self-repair: An excellent task may be broken down into many components.

Performance is raised by using this technique. This functionality automatically controls application sample flaws when several instances of an application are operating virtually on various devices.

3.3.2 Tolerable reaction fault

A responsive fault tolerance policy tries to lessen failures when they occur. The system gains additional power using this technique. This policy provides the foundation for various strategies, including inspection/restart, function reassignment, workflow release, managing a user's unique behavior, retrial, labor migration, S-Guard, etc.

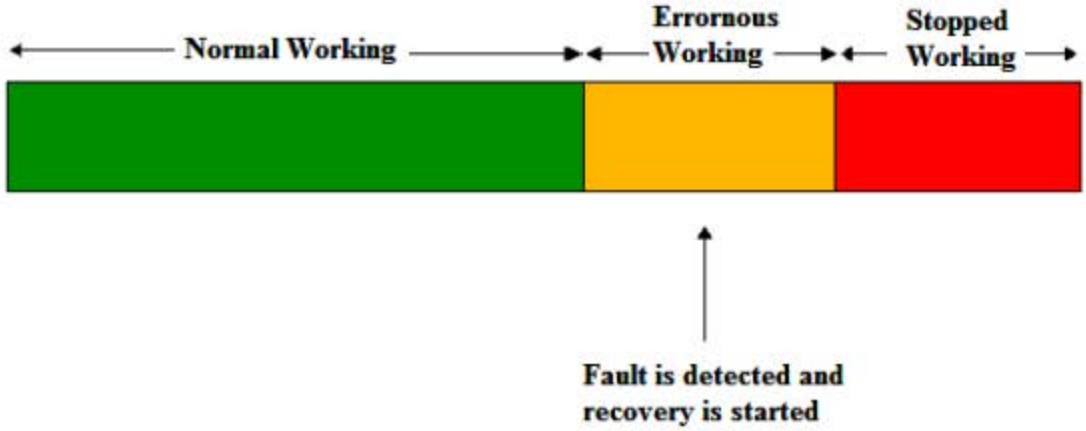

Figure 3.4: Timeline for a reactive fault detection system

Inspection: A valuable process for addressing errors, a method for handling big, lengthy jobs. After each system modification, a thorough inspection is done using this procedure.

Work migration: Sometimes, a task cannot be completed on a particular computer due to various factors. One may move the task to another machine if one computer fails.

Duplication: it is the verb form of copying. Various activities are repeated and implemented to succeed and get the desired outcome on multiple resources. Check out safety bundles blocking instructions that don't have safety features in this situation.

S-Guard: less interference with regular processing flow. It is based on rollback and work retrieval.

Retry: It will re-implement and retrieve a task in this situation. The simplest method to reuse inside a single source is this one.

Resubmitting a task: When a task fails, that fact is established. In this instance, the task is again delivered to the same source for execution at runtime.

3.3.3 Adaptive fault tolerance

Depending on the range of control inputs and present location within its space, the tolerance of an application failure necessitates a change. The instructions are automatically adjusted by adaptive tolerance to the status control. Preserving the validity of crucial modules while adhering to any time and resource limitations offers the most modules and resources for redirection.

3.3.4 Fault-tolerance Models in Cloud Computing:

Large-scale data centers have been established due to increased service demand and rising resource consumption. High performance was formerly considered the primary factor in data center architecture. Data center failure is prevalent in today's era of computing, which may be partially attributable to the massive amounts of data kept, given the rise of cloud computing-supported data centers for storage purposes and the need for cloud services. Different degrees of admittance might be necessary for each application or each data item as the size of the data increases and access to it becomes more challenging. To ensure robustness and reliability in any system, fault tolerance is used. The strategies may be categorized into proactive and reactive, depending on the fault tolerance rules and processes.

When failures do occur, reactive fault-tolerance policy aims to minimize them. It may be separated into approaches for handling errors and treating faults. Error processing is used to get rid of mistakes in calculations. The goal of error treatment is also to stop mistakes from resurfacing.

AFTRC is a fault-tolerance paradigm for real-time cloud computing applications based on the idea that real-time systems may benefit from the computational power and scaled-up virtualized cloud setting for improved real-time request execution. In the suggested paradigm, the system actively tolerates errors and bases decisions on the dependability of the processing nodes.

LLFT: As a service provided by the cloud's owners, the low latency fault-tolerance (LLFT) middleware in the projected paradigm offers distributed applications implemented in the cloud computing environment. This paradigm is predicated on the notion that one of the critical difficulties with cloud computing is ensuring that

applications operating there don't interrupt the user's experience. To safeguard the application against defects, this middleware duplicates the application using a semi-active replication or semi-passive replication technique.

FTWS: A suggested model called FTWS includes a fault-tolerant workflow scheduling method that uses task duplication and resubmission depending on the priority of the jobs in heuristic matrices to provide fault tolerance. This model is built on the idea that a workflow is a collection of activities that are completed in a specific sequence according to data and control dependencies. Planning a process that considers task failure in a cloud setting might be quite challenging. To fulfill the deadline, FTWS duplicates the tasks and schedules them.

FTM: is a model that has been put out to get around the problems with the on-demand service's current techniques. They provide a novel viewpoint on developing and maintaining fault tolerance to attain dependability and resilience. Using this specific technique, a user may declare and apply the desired fault tolerance without knowing how to do so. This FTM architecture may be seen as an amalgamation of many web services components, each with a specific capability.

CANDY: A thorough availability model is created semi-automatically from a system specification specified by a systems modeling language by the component-based modeling framework known as CANDY. This approach is founded on the idea that one of the critical characteristics of cloud services is a high availability guarantee, which is also one of the essential, challenging concerns for cloud service providers.

Vega-garden: This paradigm was developed for a cloud computing environment based on virtual clusters to address the usability and security issues infrastructure sharing brings.

FT-Cloud: A framework and its design for constructing the cloud applications are called FT-Cloud. FT-Cloud utilizes the component invocation organization and frequency to identify the component. There is an algorithm to compute fault tolerance stately automatically.

MAGI-CUBE: is a cloud computing storage design with excellent reliability and minimal redundancy. The system they build for writing and reading of files, maintaining metadata, and other tasks. Additionally, they created a file scripting and repair component that runs autonomously in the background. This paradigm is based

on the idea that the three competing elements of a storage system are high dependability, high performance, and cheap cost (space). A model of the Magi cube is suggested to provide this facility.

3.4 CR-BM-Based Hybrid Framework

The proposed system has utilized the reactive fault tolerance scheme for efficient fault tolerance and detection in the projected framework. The genetic Algorithm and Artificial Bee colony have been amalgamated to form the Bee- Mutation algorithm for proficient allocation of end-user requests. The subsequent attempts were made to realize the appropriate accomplishment of the projected fault tolerance and detection structure:

a. The checkpoints smooth the execution of the futile request and resume it from the crash position in the projected IoT-Fog-based system.

b. The replication scheme was engaged to produce user's request replicas on different resources until the whole assignment became futile.

c. The assignment of end-user requests was optimized using the Bee-Mutation algorithm.

3.4. 1 Preliminaries

3.4.1.1 Checkpoint

This method set up various checkpoints either at the application level or at the system level. The primary prerequisite is the knowledge of the failure occurrence in the system framework. The recovery from the crash commences from restarting the task where it left-off.

3.4.1.2 Replication

The method of fault tolerance that is most frequently used in storage devices like grid, cloud, and fog, where failure is more likely to occur, is replication. It is characteristically utilized to amplify the resource's accessibility and availability in a distributed storage system. Since the storage system is classically huge and composite, selecting replicas and placing them in a suitable position is the biggest replication problem.

3.4.1.3 Artificial Bee Colony The Artificial Bee Colony is an optimization algorithm inspired by honey bees' foraging habits in nature. The Eq. 3.1 yields the new neighbor solution

$\mathrm{Sl_{i,j}}$ as $Sl_{i,j} = Nl_{i,j} + \phi_{i,j}\left(Nl_{i,j} - Nl_{k,j}\right)_{(\mathrm{i}\neq k)\ (\phi_{i,j}\epsilon\ (-1,1)}$ (3.1)

The values of $\mathrm{Sl_i}$ and $\mathrm{Nl_i}$ were evaluated using the greedy selection method to update the solution with the higher fitness value. The solution is chosen based on the probabilistic model as indicated in Eq. 3.2.:

$$P_i = \frac{Ft_i}{\sum_{j=1}^{SS} Ft_j} \tag{3.2}$$

Where Ft_i, Ft_j, are the fitness measures of the i^{th} and j^{th} swarm solution [68].

3.4.1.4 Genetic Algorithm (GA)

A well-known optimizer that works with coded chromosomes is the GA. The GA typically uses the three important operators listed below [24]. The probability values are entered into the GA's selection operator, and the rank-based selection method is used to obtain the best fitness. However, the Artificial Bee Colony algorithm is used to carry out the recommended approach of the selection process. Two fitness values are used for the cross-over operator, and Eq 3.3 is used to calculate the cross-over probability ($\mathrm{CO_P}$) as,

$$CO_P = \begin{Bmatrix} if\, Ft_i \leq Ft_a, then\ K_1 \frac{Ft_{max} - Ft_a}{Ft_{max} - Ft_i} \\ if\ Ft_i > Ft_a, then\ K_2 \end{Bmatrix} \tag{3.3}$$

Where $\mathrm{Ft_i}, \mathrm{Ft_a}$ are the superior of the two solutions, and average fitness value are in solution population respectively, and $\mathrm{Ft_{max}}$, is the maximum fitness value. The K_1 and K_2 are the overall value that ranges from 0 to 1.

After the annihilation of the cross-over operator, the mutation operator is engaged over its output. Analogous to the cross-over, the mutation probability ($\mathrm{M_P}$) is predictable by using Eq. 3.4 as:

$$\mathrm{M_P} = \begin{Bmatrix} \mathrm{if\ Ft_j \leq Ft_a, then\ K_3} \frac{\mathrm{Ft_{max} - Ft_a}}{\mathrm{Ft_{max} - Ft_j}} \\ \mathrm{if\ Ft_j > Ft_a, then\ K_4} \end{Bmatrix} \tag{3.4}$$

Where $\mathrm{Ft_j}$ is the individual fitness of solution and K_3 and K_4 are the overall value that ranges from 0 to 1.

3.5 Proposed Methodology

Here, Fig. 3.5 depicts the suggested CRBM structure for deploying the service to locate the fault in the hybrid IoT-Fog environment. The proposed system integrates the cloud, numerous IoT devices, and fog nodes. In this case, the layer that gathers the data from the monitoring environment comprises IoT devices. With the aid of the service broker, this acquired data is also sent to the fog nodes for additional processing. This calculation makes the data available for the service to process further and reallocate in response to user demand. Securing the scalability and accessibility of data is a top priority, and this is done by effectively balancing the workload and allocating the resources according to the proper job schedule. When the fog nodes receive a request to transmit data, they process the available data before sending the user the response. The efficient fog node can be selected to process the request of IoT devices. The integrity of the data is also ensured by choosing the optimum fog node.

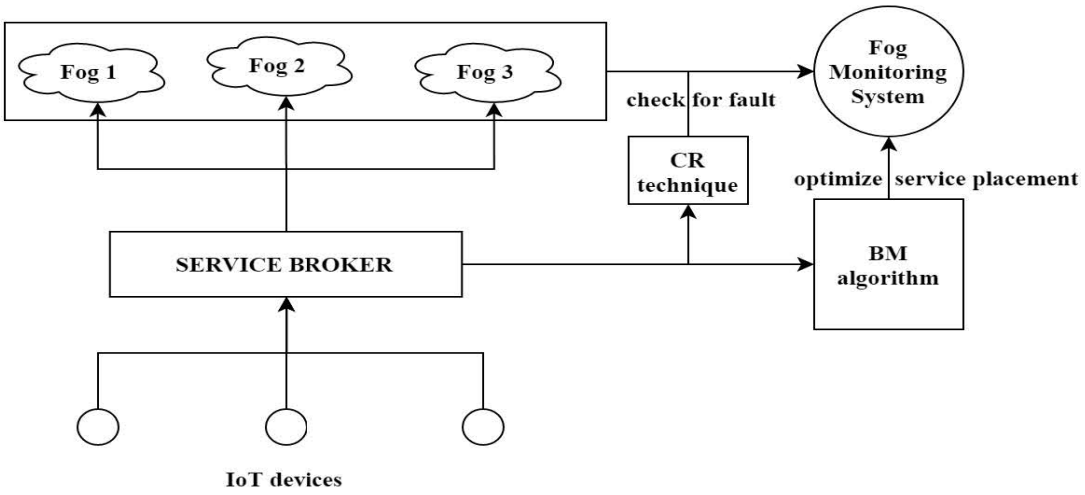

Figure 3.5: Proposed framework for service placement and fault tolerance

Therefore, there must be a fault-free or higher fault-tolerance system to attain faster response from available fog nodes. The projected system utilizes two methods to ensure improved fault tolerance. The checkpoints are established using the first method, and the existing fog nodes are replicated using the second method. The placement of a service profoundly depends on several QoS factors, including the use of the CPU, RAM, processing speed, and time. Other factors, such as response time and latency, can also be considered. The proposed CRBM framework uses the Bee-Mutation algorithm to optimize the QoS parameters. The fog nodes are strategically

positioned in the suggested design to provide services. The suggested paradigm comprises IoT devices and fog nodes with a cloud-based fog monitoring system. The following are the responsibilities of fog nodes:

a. IoT devices transmit data they have collected from the real-time environment to fog nodes for further processing and estimation.

b. Following processing and estimation, data is available for fulfilling user requests, and fog nodes reallocate resources in response to user demand.

c. After processing the request, the fog nodes revert the response to the user—the most efficient fog node among available fog nodes for handling the requests. The response is quickly processed without any data loss via the best node.

d. Following the fault tolerance framework, checkpoints are positioned at various periods during the processing of the fog nodes to assess their current state—use of replication and checkpoint to prevent the node from acting another way.

3.5.1 Diagnosing faults using the CR module

The proposed framework employs the CR module to carry out the problem diagnostics, as shown in Fig. 3.6. Two different defect detection models are implemented in this module. First, checkpoints are positioned periodically while processing fog nodes. These checkpoints maintain system functionality despite any unanticipated failures by analyzing the current status of each fog node in the framework. These checkpoints also connect to the resource server and scheduler in the fog monitor. Here, the resource server is used to freeze the process in any failure-like scenario and to continue the process from the neighboring checkpoint. At the same time, the scheduler offers the information about the fog nodes required for responding. Mechanisms used by the CR module lead to better time management and less resource use. Second, the proposed fault diagnostic uses a passive replica methodology as the replication method. This method involves the fog nodes forming onto the primary node after receiving the request for additional request processing. This strategy also installs the secondary or backup fog nodes in the background along with the primary fog nodes. These backup nodes operate constantly, and if the primary node fails, control is transferred to the appropriate backup node without pausing the current request processing. This replication technique combines the checkpoint strategy to ensure errors are handled correctly without interfering with

ongoing operations.

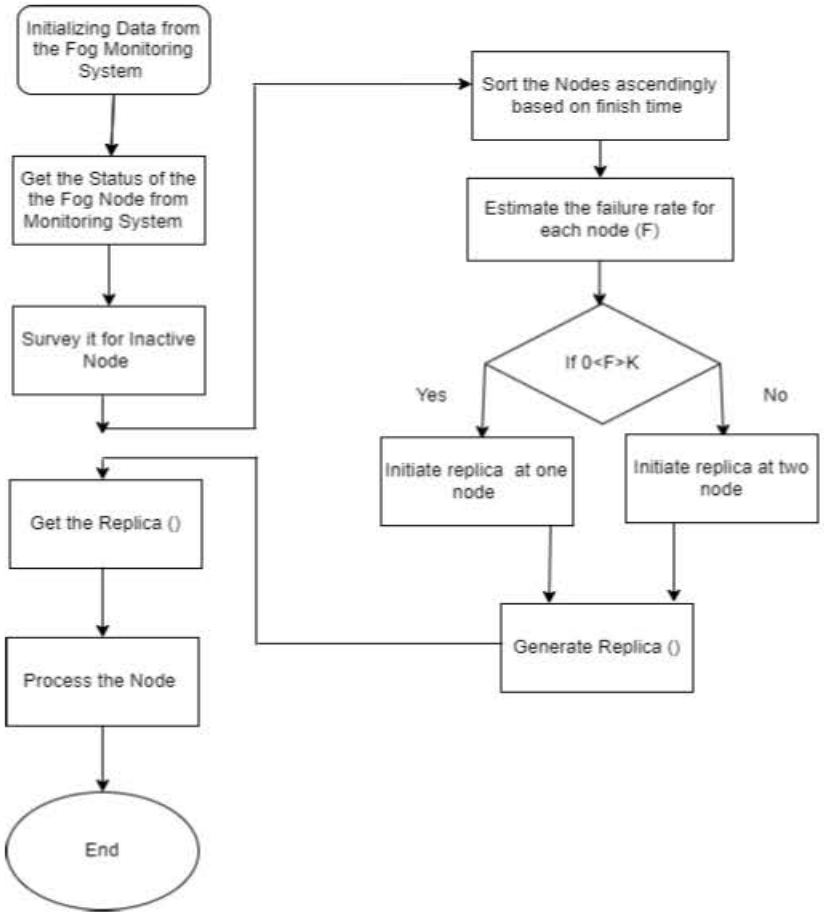

Figure 3.6 : Workflow for checkpoint & replication of proposed framework

3.5.2 Optimal service placement with BM

The placement of service to the efficient fog nodes is the next crucial factor in the projected framework. The bee-mutation method has been proposed for placing a service to the best fog node, as shown in Fig. 3.7. The CPU, memory, network bandwidth, system time, and processing speed of the fog nodes QoS metrics are input for the BM algorithm's evaluation. These input parameters are first given into the artificial bee colony algorithm, which uses Eq. 3.1 to produce solutions for the bee algorithm and Eq. 3.2 to calculate the probability fitness value. After obtaining the probability fitness value, the genetic algorithms cross-over operation is applied over the probabilities acquired to produce a new breed of values for each parameter of the fog nodes in Eq. 3.3. The mutation operator is conducted by replacing them with the values acquired using Eq. 3.4 with a mutation rate of 10% over the generated cross-overvalues. The suggested algorithms use an iterative approach with gradual expansion to get the desired results. All of the datasets use a certain amount of

iterations.

The algorithm requires that the fitness functions in Eq. 3.2 can be computed for each iteration. The cross-over operator is then applied to two fitness values and the cross-over probability. This can be done by utilizing Eq. 3.3 and Eq. 3.4 to determine crossover mutation probability. Check pointing and replication algorithms are complicated by O(n) running time, and iterative bases bee mutation methods are O(n2) difficult.

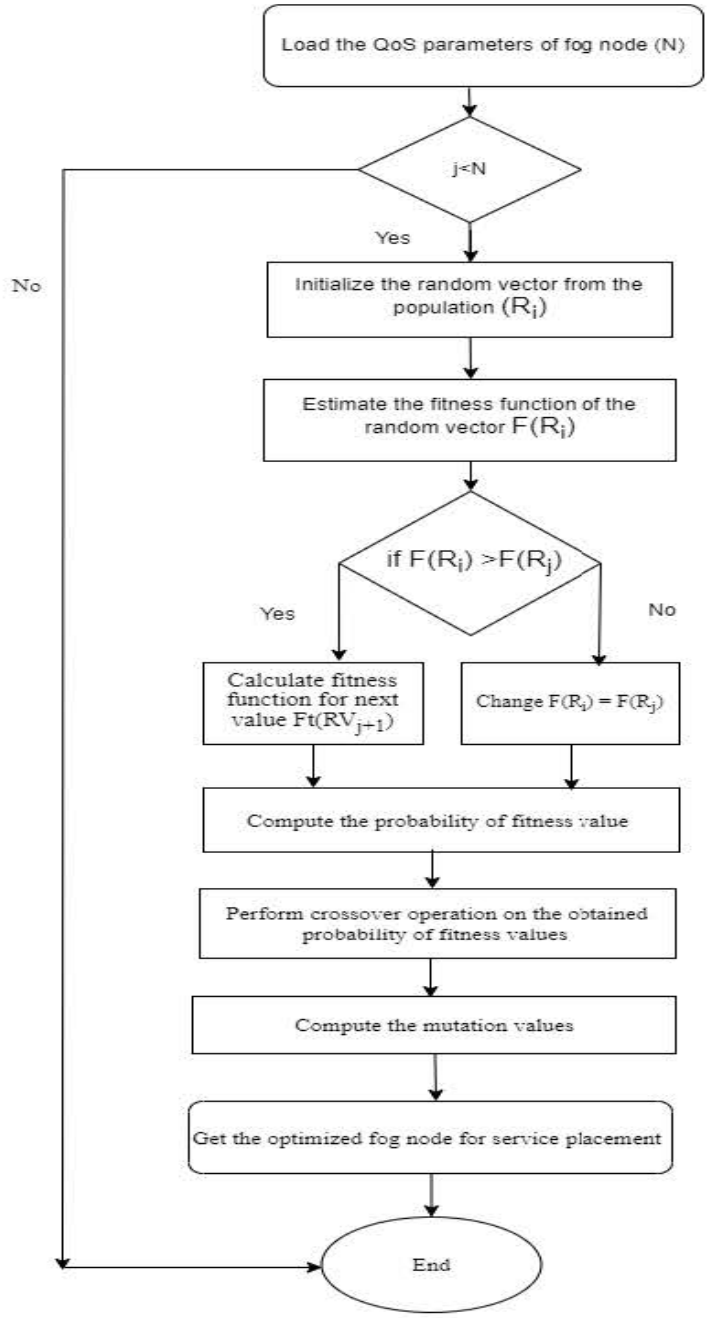

Figure 3.7: Workflow for Service Placement in CRBM

3.6 Result and discussion

The expanded version of iFogSim is used to simulate the proposed framework; iFogSim can be used on various operating systems, including Windows and Linux, and needs a minimum of 8 GB of main memory and an Intel Dual-core processor running at 1.87 GHz or above.

3.6.1 Performance Analysis

Several performance indicators have been used to assess the performance of the proposed hybrid CR-BM system. Five fog devices were considered for this framework; Fig. 3.8 provides a detailed analysis of the performance of the hybrid CRBM framework.

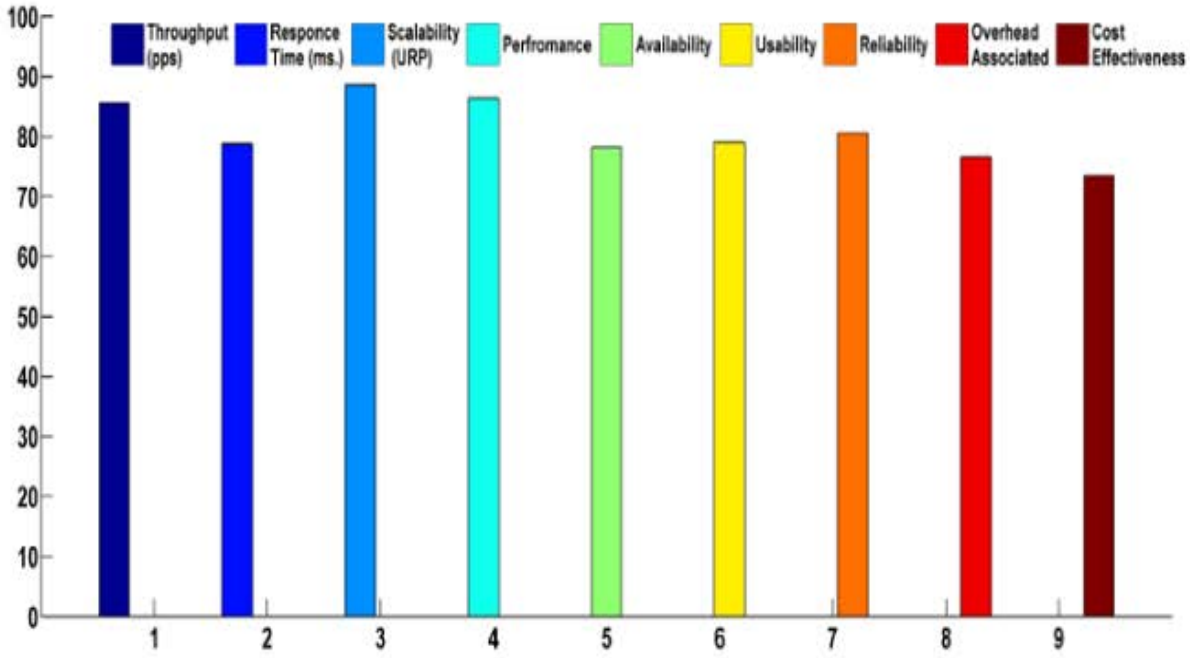

Figure 3.8: Analysis of the overall performance of the hybrid CRBM framework

Throughput is critical for proficient data transfer, particularly in the IoT-Fog network. The number of packets each device transmits in a second has been used to measure the throughput of the devices. The devices' collective throughput is around 85.6 packets per second. The response time indicates how long it takes the fog nodes to respond to a request from the end user. The response time is the total amount of time required for node deployment, make span, and request communication. The projected framework responds in roughly 78.8 milliseconds. Scalability in the IoT-Fog system refers to the capacity to handle a given number of user requests without

compromising fault tolerance. The average number of user requests handled (URP) among the five devices is 88.6. The proposed system has an overall effectiveness of 86.4%. The primary requirement for any IoT-Fog system is the availability of fog nodes to receive and process requests. Availability is the node's uptime as a percentage of its overall time. The availability of the proposed system is around 78.2%. The system's reliability considers its efficiency in producing accurate results, while its usability deliberates its efficiency in carrying out the procedure efficiently. The projected Checkpoint Replication Bee Mutation (CRBM) based hybrid framework has a usability and reliability rating of 79% and 80.6%, respectively. Any system that is deployed will undoubtedly have costs and other overheads. Therefore, the system needs to be efficient and have low overhead costs. The suggested framework has an average overhead of around 76.6%, and its cost-effectiveness is 73.4%.

3.6.2 Comparative analysis

Regarding execution cost and network utilization, the projected CR-BM-based hybrid framework has been contrasted with the already existing genetic algorithm in the fog-based system, as shown in table 3.1. Comparing the projected framework's execution cost to the cost of the existing genetic algorithm, which is $85042.20, as illustrated in Fig. 3.9.

Table 3.1: Comparison of proposed and existing IoT-Fog algorithms

Parameter	**Existing Genetic Algorithm [24]**	**Novel Bee Mutation**
Execution Cost of Fog Cloud	$85042.20	$82020.20
Total network usage (Mbps)	635550.0	626020.0

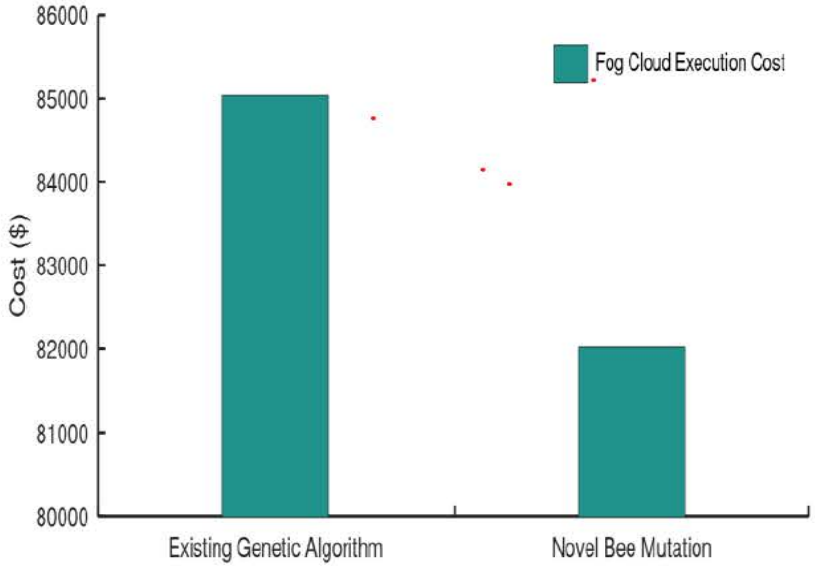

Figure 3.9: Execution cost of Proposed and Existing Algorithm

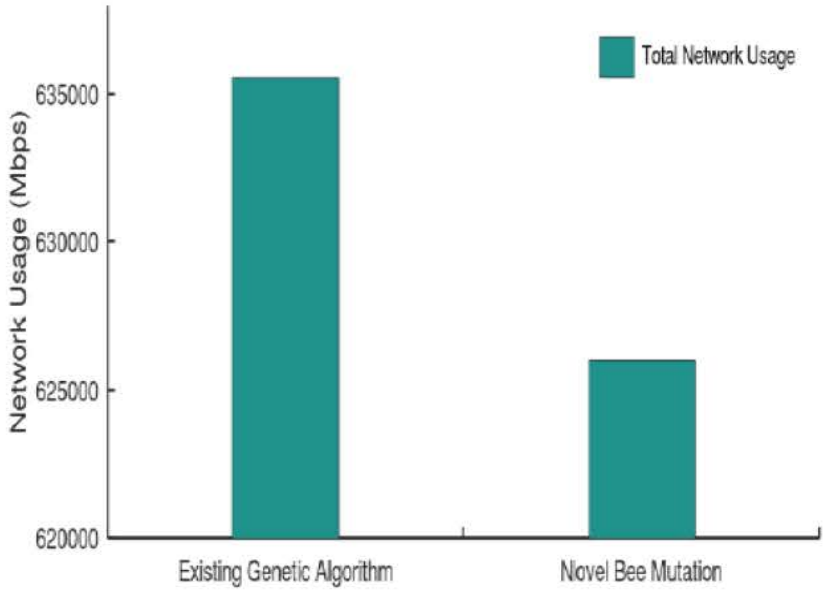

Figure 3.10: Total network usage proposed and existing Algorithm

In comparison to the suggested CR-BM-based hybrid framework, which obtained a total network use of 626020 Mbps, the existing Genetic Algorithm GA-based IoT-Fog framework had a greater total network consumption of 635550 Mbps. The network used for both strategies is displayed in Fig. 3.10, and Table 3.1 compares the proposed and current methods.

3.7 Summary

The efficient service placement and fault-tolerant QoS-aware hybrid CR-BM architecture have been developed. The projected model takes into account the combination of the well-known checkpoints and replication and Bee-Mutation approaches for the diagnosis of faults. In this case, the checkpoint technique starts the process from the pre-defined checkpoints established when there is a failure, whereas the replication technique avoids a delay in the checkpoint mechanism. As an outcome, the projected hybrid CR-BM framework is fault-tolerant. Placing the service strategically among the fog nodes is another feature of the proposed approach. A revolutionary Bee-Mutation method that combines the Artificial Bee Colony and Genetic method has been presented to place services in the best possible order. Throughput, performance, cost-effectiveness, and other performance indicators have all been used to assess the performance of the proposed CR-BM-based hybrid framework. Additionally, we have presented a thorough comparison between the proposed framework and the currently operational GA-based Fog system regarding execution costs and network use (Mbps). The suggested framework surpasses the previous model in the given results, utilizing the network at a rate of 626020 Mbps and costing $82020.20 to execute.

CHAPTER-4

MOBILITY-AWARE AND LATENCY-SENSITIVE-BASED SCHEDULING

4.1 Introduction

As the processing power and data transfer rates of mobile devices like smartphones, smart watches, and in-car entertainment systems grow, the computing demands of on-the-go consumers also rise. The usage of a distant resource, such as a large-scale computing capacity housed in a data center, is commonplace in modern software. Cloud services may store and process data and tasks offloaded from mobile or fixed devices, allowing them to provide additional app assistance.

With an increased emphasis on the Internet of Things (IoT), innumerable devices are spread and linked to the Internet, and creating and consuming data needs new levels of scalable resource management [70]. The data dynamic and heterogeneity from the predicted rapid proliferation of connected devices, known colloquially as Big Data, will need novel processing models and infrastructures to handle its three primary dimensions: data volume, velocity, & variety. One significant feature of this new age is that data production and consumption are extensively scattered at the network's edges (i.e., closer to or at end-user devices). While the cloud computing centralized data center architecture can manage many different kinds of applications and enormous volumes of data, its infrastructure and network connections to the edge are not built to handle the Big Data phenomena. Compute, and data management approaches that provide computing capability at network edges are the subject of significant study in this area. New distributed computing architectures that use edge capacity closer to data generation include mobile clouds, vehicular networks, and fog computing. 2 With data created at the edge, data creation and consumption may occur at various locations and times. Distinct applications may have different needs in this setting, particularly regarding reaction time. Currently, apps often depend on the cloud for data and processing support, which may be insufficient for low latency needs. Furthermore, application execution in cloud data centers does not account for user mobility, and data or processing of an application might occur in a geographically distant data center. In distributed computing, data processing may be

kept closer to the user, minimizing network traffic to data centers and improving application response times due to decreased network latency or delay[71].

4.1.1 A General Scheduling Framework:

Scheduling methods differ greatly depending on the kind of network, the intended use, and the environment, making direct comparisons difficult. One strategy for dealing with this complexity is to break out the scheduling challenges and compare them individually. This emphasizes the need for a comprehensive design to handle all expected networks and scheduling challenges. Here, we provide a framework to help with planning challenges and show the fundamental components of all scheduling algorithms to enable comparisons across existing approaches. There are five main parts to the framework, which are shown in Fig. 4.1. The many parts of this diagram represent different but equally important aspects of a scheduling algorithm. In the following paragraphs, we detail each part in detail [72].

4.1.2 Information Collection

The approach of information collecting that includes the fewest problems is known as the gathering of static information. According to this strategy, the person in charge of managing the network is the one who is accountable for instantly putting the necessary information into the system. For instance, in a long-distance static mesh network, the nodes are manually scattered throughout a geographical region, and then the neighborhood connectivity of each node is figured out. In addition to that, the network remains unchanging. This kind of network may sustain the transmission of exceedingly massive volumes of data. In Fig. 4.1, each fog node is responsible for a different coverage region, and a group of fog nodes is referred to as a location area (LA). The use of tier partitions improves the granularity of place descriptions. A tier comprises n mobile nodes governed by the middleware in the center of all tiers. In this example, middleware serves as a link between mobile and fog nodes. This work presents a proactive method of fog computing that uses the Hidden Markov Model to facilitate proactive fog service discovery & process migration—the suggested framework for mobility-aware systems.

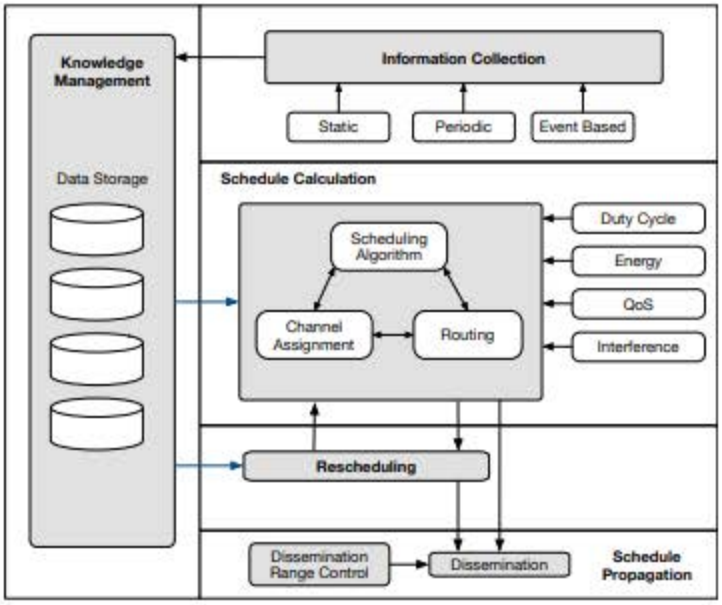

Figure 4.1: Comprehensive framework for scheduling in wireless networks.

Static: The most straightforward information collecting option is static information gathering, in which the network management complex codes the relevant information. For example, nodes are manually planted in a long-distance fixed wireless mesh network in a geographical region, and each node's neighborhood connectivity is predefined.

Periodic: Nodes in the network are kept up-to-date on any changes to the network state thanks to the scheduling node's regular transmission of control packets. Among the events that may convey a change in network state are beacon packets in mesh & ad hoc networks.

Event-Based: A node sends out an event packet to neighboring nodes when it notices a change in a monitored parameter. For example, an event message might provide a list of inaccessible nodes to highlight the inadequate timeline.

4.1.3 Knowledge Management

This part's primary function is storing relevant information for the scheduler in a format that scheduling algorithms can read and understand. If a mesh network's topology is represented as a graph or tree, with reachability values instead of coordination values, this would be an example of such a data structure. Depending on how far away their respective sources are from the scheduling node, there are three types of necessary knowledge.

Intra-Node: In this kind of timetable, nodes rely on their data to determine what needs to happen and when. Priority-based packet scheduling is one method used to organize data before it is sent across a network. In this scenario, scheduling may be based on local data collected at each node.

N-hop Neighbors: The scheduler in this category gathers network information from nodes and hops away from the scheduling node. Because obtaining information from faraway nodes is challenging in most circumstances, the value of N is minimal. Because one-hop neighbors are in the interference range of a scheduling node and possess crucial scheduling information, the knowledge of the first-hop neighbors (N=1) is often employed in the literature. A node may learn this information by passively listening to packets from surrounding nodes or actively exchanging control packets to modify its neighborhood database.

Network Wide Obtaining global network information enables the determination of an optimal schedule - although an NP-hard problem, global knowledge persistence is often short, especially in mobile networks. This kind of information management is often used in centralizing schedulers to ensure punctuality and reliability from start to finish.

4.1.4 Schedule Calculation

In the schedule calculation section, one or more scheduling algorithms take the information as inputs and produce a schedule for a subset of nodes in distributed scheduled for all nodes in centralized scheduling. Scheduling methods use the nodes' resources to carry out the scheduling process, regardless of the scheduler type. We evaluate the computational ordering of various algorithms and compare their execution times. In a perfect world, algorithms would have a logarithmic or linear computing order rather than a polynomial or exponential one.

4.2 Mobility in Fog Computing

This work looked at the fundamental idea of mobility and the challenges that are associated with it. The section may be broken down into three distinct portions: an overview, a low dynamic environment, and a high dynamic environment. In fog computing, the overview section is the foundational component for mobility. In this section, we have covered both systems for allocating time and managing transportation. Since scheduling is crucial, we must be flexible while maintaining the

highest service standards (QoS). Here, it explores vehicular networks' role in fog computing and the difficulties that come with it. The following two sections discuss the differences between low- and high-dynamics settings. To provide more nuanced details for readers and academics, these sections concentrate on human mobility & vehicular fog computing. To increase the system's level of consciousness, researchers are looking at the effects of movement in the fog zone. However, many existing studies neglect mobile devices and scheduling strategies in fog computing data centers in favor of optimizing server selection for mobility[73].

To improve QoE and network benefit, it is required to hone in on the calculation approach at both the mobile device and fog computing scheduling levels. Therefore, we provide a scheduling system and mobility management as two fundamental necessities for enhancing the quality of experience, the quality of service, and decreasing latency in fog computing[19].

4.2.1 Overview

Research on mobility in the fog zone is being conducted for adaptive, big data, and real-time urban surveillance tasks, including continuous target monitoring. Most existing studies, however, are geared at bettering how servers are selected in light of users' mobile devices. The scheduling algorithms employed in fog computing data centers and mobile devices are ignored in this research. Focusing on mobile devices & planning at the fog level is necessary to enhance experience quality and reap the advantages of network connection. Therefore, we provide two primary criteria, a scheduling system & mobility management, to improve fog computing's QoE and QoS while reducing latency.

4.2.1.1 Scheduling Mechanism

Fog computing results in decreased latency and cooperative efforts to cancel or diminish the effects of excessive traffic congestion in the leading network. Nevertheless, this may make it more challenging to manage the computation priorities, quality of service, and priority of low-delay application requests of each end user. The fog computing server's primary responsibility in the conventional scheduling system is communicating the offloading priority order to the end users[74].

It depends on the various local computing data, channel gains, and latency needs of

individual users. Because of the changing settings, the static scheduling technique can't be used directly with mobility in the case of multiuser fog computing systems. This includes channels that change over time and connections that come and go at random. As a result, the scheduling method became more challenging to implement with the introduction of dynamic environments. For example, what end users will do, when they will complete it, and where they will accomplish it to meet their quality of service and latency needs.

In addition, the scheduling algorithms used in fog computing must consider the mobility of data sources and sinks and intelligent and wearable devices. It is essential that scheduling consider users' geographic location when formulating resource allocation strategies to protect the financial benefits that come from fog computing's close proximity to end users. In light of what has been said thus far, the ever-changing nature of the surrounding environment drives us to devise an adaptive server scheduling system that, among other things, periodically reclaims the scheduling order and considers the input from real-time users. By using adaptive scheduling techniques, the users in the worse condition will be given greater offloading antecedences to meet the computing deadlines they have set for themselves. A design that takes into account mobility and gives offloading priority is also demonstrated. This method is applied to forecast users' mobility profiles and channels accurately. It will assist in overcoming the influence that movement has on the function and will re-determine the offloading antecedence. Second, the use of a resource reservation technique may result in an improvement in the operation of the server scheduling.

In addition, mobility management and traffic control are offered to improve the quality of experience (QoE) for users doing latency-tolerant jobs. The use of an intelligent cell association process accomplished the construction of this. Regarding edge caching, the mobility predication is included to optimize the movement of content caches placed at the edges. Caching at the network's periphery is recognized as a valuable contributor to maximizing the use of the backhaul restriction imposed by network concretion. The idea of caching on the edge may be extended to mobile edge computing, which allows for more flexible and context-aware caching decisions (MEC). It makes it possible for processing and storage resources to be located at the edge of mobile networks. MEC servers provide for the investigation and use of a large number of gathered radio access network context data, which may give an adaptive

caching method to users' context-aware information. Fog computing helps alleviate the strain on the network's backbone by decreasing lag time and encouraging collaboration. However, this might make it more challenging to prioritize low-delay apps and cater to users' preferences regarding quality of service, computation power, and other metrics. Under the classical scheduling system, the fog computing server is responsible for communicating with the priority order offloading to the end users. It depends on the various computational data, channel gains, and latency needs of the consumers. Due to the ever-changing nature of fog computing deployments with many users, a static scheduling approach cannot be used in tandem with mobility [75].

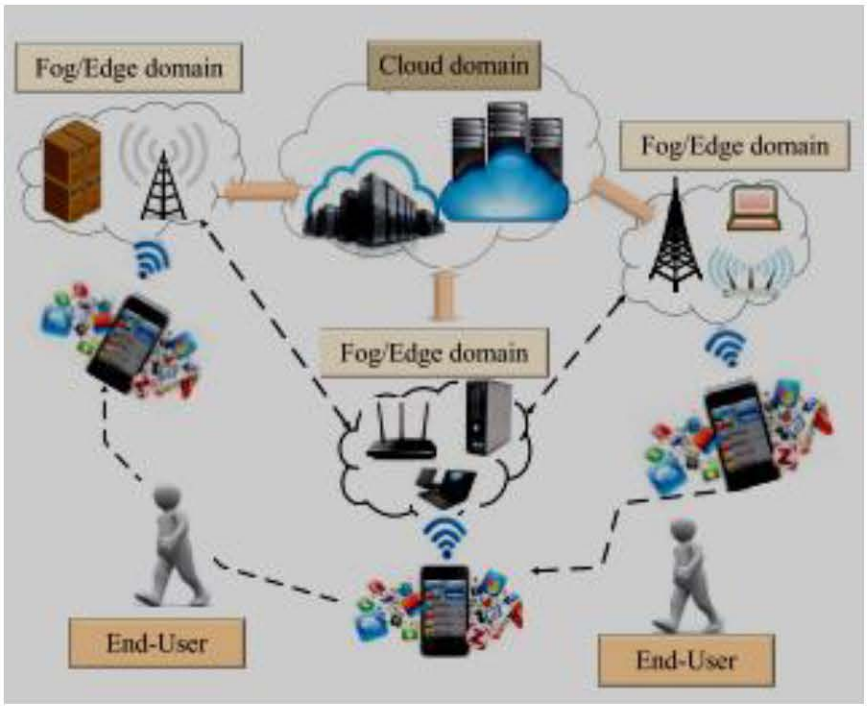

Figure 4.2: End-user mobility in the fog/edge computing scheduling process.

This entails not just intermittent connections but also channels that change over time. As a result, the introduction of dynamic environments has increased the difficulty of the scheduling process. One illustration is how users process the different requests to improve performance parameters. Because of this, proficient scheduling techniques in cloud computing with smart & wearable devices need to accommodate the data from diverse mobile sources and be submerged. The scheduling process should consider the end-user's locations while distributing resources to retain the advantages of fog computing a close range from the end users, as shown in Fig. 4.2. In light of the above debate, we feel compelled to develop flexible server request scheduling schemes, which retrieve the scheduling sequence at irregular intervals and include input from

real-time users. Because of adaptive scheduling, the users in the worst condition will be given earlier offloading antecedences to ensure they meet their computing deadlines. The priority offloading design that takes mobility into account is also discussed. The method is used to foresee individual users' mobility profiles and channels accurately. It'll aid in getting around mobility's impact and recalculating the offloading antecedence function. Second, an effective technique for reserving resources may provide more efficient server scheduling.

In addition, this work discusses mobility management and traffic control to improve users' quality of experience with latency tolerance jobs. The intelligent cell association technique was used in the making of this. Edge caching incorporates mobility prediction to facilitate the transfer of edge-based content caches. Edge caching is also recognized as a valuable tool for exploiting the asymmetrical limitation of a concrete network. For more thorough compromise and context-aware caching decisions (MEC), edge caching could be used for mobile edge computing. It makes it possible for mobile edge networks to have access to computing and storage facilities. By analyzing and using the vast amount of context data acquired from radio access networks, MEC servers may provide an adaptive caching mechanism for users' context-aware information.

4.2.1.2 Mobility Management

To provide a consistent computing service and ensure Quality of Service (QoS) for customers sensitive to latency, fog computing servers can reserve specific dedicated processing resources for mobile users. For example, the end user views video material when they wander from one access point (AP) to another. In this scenario, the video content is not lost[5]. The fog system must be capable of providing the end-user with the same video material from where it was left without causing any disruption to the service, as seen in Fig. 4.2. To accomplish this goal, architectural modules such as mobility aspects are necessary to ensure the end-users mobility management. Through the use of mobility handling algorithms, mobility may be carried out in a variety of choices. For instance, the algorithm for dealing with mobility may require the mobility aspects to copy the video and drive a copy to the terminal access point (AP) with fog capability if numerous end users are watching the same video while traveling in various directions. However, resource deracination also coincides with focusing on resource management strategies in the case of moving fog nodes.

On the other hand, users who are tolerant of delay can obtain on-demand provisioning from edge servers. This hybrid server need may improve the server scheduling to provide the maximum number of users with a high Quality of Service (QoS) while maximizing the income the servers generate. Nevertheless, there is a possibility that the fog computing computation offloading may fail owing to intermittent connectivity or/and rapid channel change in wireless channels. The failure is a catastrophe for applications that are responsive to latency and also for apps that require resources. As demonstrated, designing fault tolerance that considers mobility is crucial for fog computing systems, including fault prevention, detection of faults, and recovery from failure.

4.2.1.3 Low Dynamic Environment

Device-to-device communications, abbreviated as D2D communications, are considered a fresh paradigm with considerable potential for confirming proximity-based applications. These interactions occur underneath cellular networks and utilize the same spectrum as cellular users (CU). D2D fogging is becoming a new paradigm in the framework for mobile task offloading, and it does so by capitalizing on the benefits of direct-to-device connection. In this case, mobile users are functioning in a low-activity zone, and they can constructively share the computational and storage resources among themselves through the base station (BS). Furthermore, mobile device competence is perpetually advancing, and the multiplexing benefit may be used to advocate cooperative task execution for various applications. Despite this, D2D fogging is fraught with difficulties brought on by time-varying cellular positions and the lack of functional D2D connections. The cellular positions that change with time and the stochastic and capricious D2D connections are caused by the dynamic nature of the mobile users and the processing capacity of the devices they are using. Furthermore, the task offloading architecture should make a more robust effort to achieve network-wide optimum energy saving.

Device-to-device communications, abbreviated as D2D communications, are considered a fresh paradigm with considerable potential for confirming proximity-based applications. These interactions occur underneath cellular networks and utilize the same spectrum as cellular users (CU). It's becoming more common to use D2D fogging as a mobile task offloading framework, and it does so by capitalizing on the benefits of direct-to-device connection. In this scenario, mobile users work in a

sparsely populated location and may effectively pool their computing and communication capabilities via the ground station (BS). Furthermore, mobile device functionality is constantly improving, and the multiplexing advantage may be exploited to promote collaborative job completion execution for various applications. Despite this, D2D fogging is fraught with difficulties brought on by time-varying cellular positions and the lack of functional D2D connections. In addition, the framework for task offloading should give a better effort to achieve network-wide optimum energy saving for users' job performance. The next step, although certainly not the least important, is for the D2D fogging model to provide an acceptable incentive approach. It has a strong focus on user cooperation, and as a result, a more effective incentive approach may prevent excessive exploitation that undermines the users' motivation to work together[76].

When dealing with user mobility in a fog computing system, it is essential to make advantage of D2D fogging to build various D2D linkages. Because of these linkages, a user's calculation can be offloaded to other nearby users. This not only makes effective computing potentials possible but it also lowers the amount of energy needed for data transmission. This makes it possible for us to research users' mobility, which opens up new design concerns for fog and edge computing. For example, to take advantage of the benefits offered, D2D & cellular communications both provide the possibility of using the edge servers at the BSs, which have vast computational power, to handle data that requires a lot of processing. This will shorten the amount of time that the server spends performing computations.

4.2.1.4 High Dynamic Environment

Roadside units (RSUs) cloud servers behave as fog nodes, & automobiles serve as the data-generating layer for vehicular fog computing (VFC), used in dynamic settings, as shown in Fig. 4.3. They provide various mobile services, including ads, entertainment, and disaster relief efforts. VFC provides driving security, traffic effectiveness, and substantial convenience by transmitting important information. With the introduction of increasingly advanced devices & equipment, such as cellular networks & cloud computing, during the last 10 years, VFC networks and associated applications have seen substantial development. Along with the potential, several significant obstacles also surfaced, such as the sharp rise in the need for data transmission and processing power[77].

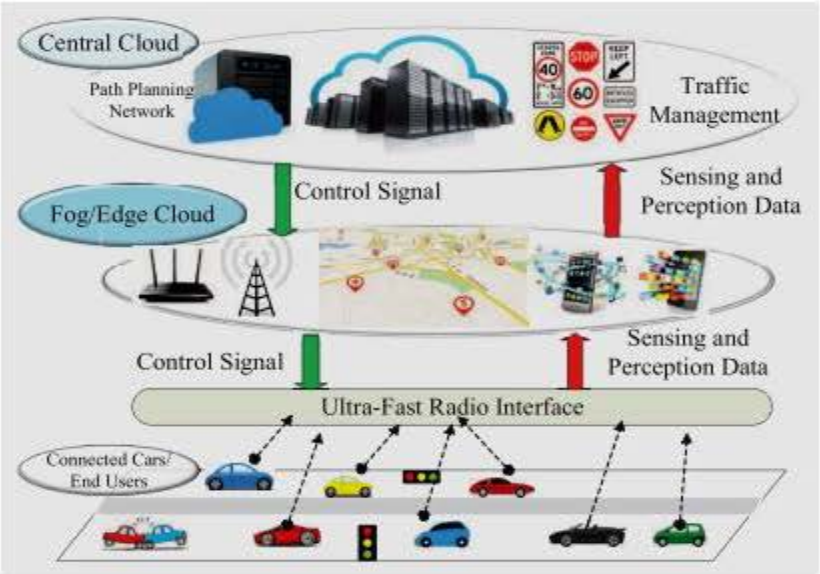

Figure 4.3: The vehicular fog computing mobility scenario.

Innovative applications that deal with composite data processing and storage procedures include self-driving technology and augmented reality (AR) techniques. Higher data transfer, calculation, and storage levels are needed, which poses significant problems to the current traditional vehicle networks. Therefore, VFC is garnering interest as data centers and better processing resources to meet this continuously growing need in data transmission and computing skills. Due to the fast-moving vehicles, the VFC is taken into account. However, there might be two different situations for vehicular fog computing: services and applications for stationary and moving cars.

4.3 Proposed Mobility Aware Framework

This architecture of fog computing is hierarchical, and the choice of the placement of processing and storage depends on the constraints imposed by the applications as well as the geo-location of the users. It can potentially deliver decreased latency and traffic congestion in the core network. These processes must include the data source and sink mobility in the fog.

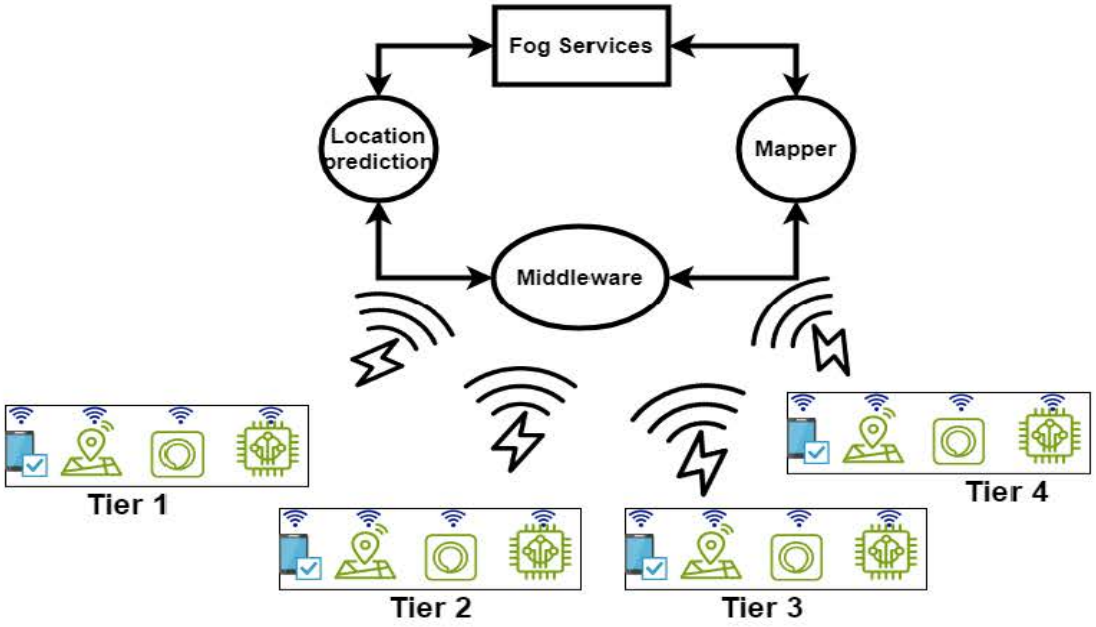

Figure 4.4: Proposed Mobility Aware System Framework

The scheduling of fog computing should consider users' location as part of the resource allocation algorithms to keep the benefits of fog computing close to consumers, as shown in Fig. 4.4. In addition, a user's actions determine the time and location of mobile devices and the quality of service limits. Therefore, it is impossible to obtain an accurate prediction of mobility if there is a lack of information or if the behavior of users is unexpected. As a result, scheduling in the setting of failed mobility prediction is an intriguing research challenge that calls for substantial study in fog computing. It is possible to construct scheduling models that capture mobility patterns, and it is also possible to design resource management algorithms that are more effective. However, the practical resource management system and scheduling algorithms for cloud computing are still in place. These issues are caused by the uncertainty and the fluidity of the resources utilized in fog computing.

4.3.1 Availability Prediction

The predictor of availability is the one who makes predictions about the registered MRP's dynamic availability.

4.3.1.1 Physical Availability Prediction

P1(t) denotes the hidden state at time t, and P2(t) represents the location visited at time t. P1(t) and P2 (t) also have specific dependencies that are conditional. In this situation, the value of the hidden variable P(t -1) determines the conditional

probability distribution of a hidden variable P1(t) at time t. However, at time t, the observed position P2(t) is dictated by the hidden variable P1(t). These probabilities must be computed for a particular observation series.

$$P(B)=\Sigma\, P(B \mid A)P(A) \tag{4.1}$$

Where PA is an unobservable variable representing the user's last known position PA= <P(AA),P(AB),P(BA),…..>. and the user's current location, PB, viewed at time t, is represented by PB= <P(BA), P(BB), P(AB),…..>. For training an HMM, we are given an output sequence, P1, and we must determine the optimal values for the probabilities at each state transition and the final output.

The mobile device Md's movement type estimates its physical availability. The nature of the motion may be used to determine which way the body will be moving. Using a hidden Markov model (HMM) for forecasting, we can determine where the mobile device (M_d) will be soon. The direction of Md's motion concerning Fn is used to infer the nature of the motion. Both forward motion (i.e., toward F_d) and backward motion (i.e., away from F_d) have been considered here. Because the present position of a mobile resource solely relies on its past locations, the Hidden Markov Model may be used to anticipate the movement that will occur [79]. Assumed to be embedded in the Fog layer, the GPS pinpoints the whereabouts of M_d. The Haversine formula calculates the gap between the M_d and the F_d. How far away is M_d computed as D, and R is the Radius, as shown in Fig. 4.5.

Where, $D=R\theta$ (4.2)

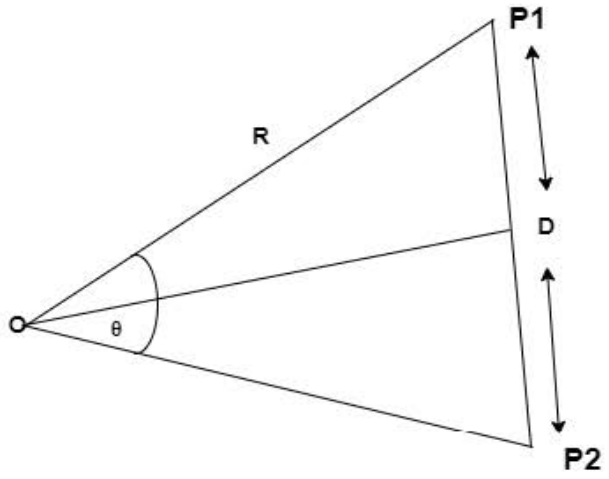

Figure 4.5: Mobile Devices Distance Calculation between two position

$$D = R\left(\sin^{-1}\frac{d}{2R^2}\sqrt{4R^2 - d^2}\right) \quad (4.3)$$

$$M_B = \frac{P1-P2}{T_{12}} \quad (4.4)$$

$$P_A = \frac{F_X A - P2}{M} \quad (4.5)$$

$$P_A F = \frac{P_A}{T_R T} \quad (4.6)$$

Distance D between the two different mobile device positions using Eq.4.2 and Eq. 4.3. P1 and P2 are the two different positions of mobile users. M_B is the user's mobility of two other locations, as shown in Eq. 4.4. T is the time taken to reach out from P1 to P2. P_A physical availability of mobile users can be calculated via Eq. 4.5. P_AF is the physical availability factor calculated using Eq.4.6. T_RT is the mobile user's time spent in a particular tier, as shown in Fig. 4.4.

4.3.3 Resource Allocation

The Cloud-Fog environmental paradigm in a computer system is dynamic and complicated; consequently, resource allocation might be difficult for providers. A technique based on prediction has been investigated to analyze the necessary resources for the FC system [78]. In this method, the mechanism for calculating the likelihood of giving something up has been used.[82] They developed their work using an allocation strategy, particularly for dispersing the workload within the context of hybrid cloud and fog systems. The computation of resource allocation may be more accurate thanks to examining the disparity between delay and power usage [83]. Applying a combined resource allocation model for various cloud and fog relationships and architectural configurations is possible. Based on the procedures

taken to appropriate the funds, these models can only anticipate the results for different FC uses. The assessment process may be advanced by contributing to developing the FC apps and determining the needed resources.

It is planned that the distribution of resources, which will be used to address the particular requirements of the FC setting, will be arranged in such a way as to accomplish the goals of the FC environment as a whole. Utilizing the capabilities of the resources efficiently and meeting the system's needs efficiently may help enhance the resource allocation process, which is necessary for conformance with the network delay and low latency management.

4.3.4 Mapper and Allocator

The mapping process is responsible for mapping procedures such as identifying mobile users, the system state, and the job required. The mobile user is identified by mapping the fog node ID from the proposed system. The job demand is mapped to the dynamically projected location of the mobile user, which is how mapping is accomplished. The allocator function receives notification of the fog node's or resource's availability only if the conditions for the mobile user are satisfied. Requests that are currently being processed and those that are waiting in line are both aspects of resource status, as shown in Table 4.1 in RQ-Running, and RQ_Wait_Queue. The resource states' potential policy traits have three essential information attributes: user requests in the execution state, requests in the waiting queue for execution, and requests that remain calculated using RQ_Remain in Table 4.1.

This information about the job credential is typically utilized to make predictions about the condition of the fog node. In its most basic form, the algorithm assigns jobs to fog nodes in the middleware depending on the processing capacity of these nodes in decreasing order. Conversely, it's possible that some tiers don't have access to a sufficient number of fog nodes. In this instance, the suggested method returns to the previous level, which is the point at which it was determined that adding one additional fog node would provide the desired effect of increasing the total amount of resources that are accessible. The system does not begin allocating the resource until after the optimum allocation has been finished, and it utilizes the bare minimum of the available resources for the next round of allocation rather than the maximum.

Table 4.1: Parameter for the resource state

Acronym	**Function**	**Formula**
RQ_Runing	Requests in execution state	RQ=nZ$_i$
RQ_Wait_Queue	Number of requests in the waiting queue	RQ=nWQi
RQ_Remain	A sum of remaining node * Expected time for executing requests	$RQ = \sum_{t=1}^{NRi} remaint(t) \times CPU(t)$

Algorithm 4.1 is employed for location prediction of mobile user's devices from the start location. Availability was predicted by generating the equivalent state probability in the Markov chain. Distance between the start to the current location was calculated via Eq. 4.2 and Eq. 4.3

Algorithm 4.1 : Location prediction

Input: Start location

Output: Mobile Device Location Information

```
1.  for Mdy ∈ N – {Mdx} do
2.    for  Md  ∈ Ti do
3.        for Pi(t) ← M * Pi(t-1) do
4.          Phy_avail  ();
5.          Phy_avai_fact ();
6.          Compute distance ();
7.        end for
8.    end for
9.  end for
```

Algorithm 4.2 is employed for mobile user's request allocation to fog device. The available fog nodes were arranged in descending order of their precedence, and precedence was assigned to each fog device as per their computing capability.

Algorithm 4.2 : Mobile device request allocation

Input: Mobile Device Location Information, fog nodes f_x .

Output: An efficient allocation of mobile device requests to f_x

1. Compute precedence among all the fog devices f_x
2. Arrange fx list in descending order of their precedence.
3. $f_x \leftarrow$ getstatus();
4. **for** $f_x \in N - \{ f_{xi} \}$ **do**
5. **if** precedence > f_x **then**
6. $R_q \leftarrow nW_{qi}$
7. $R_q \leftarrow nZ_i$
8. Update f_x list
9. **else**
10. Search f_x list
11. **end if**
12. **end for**

4.4 Performance Analysis

The projected mobility-aware scheduling technique was tested in a fog computing environment using the iFogsim toolkit because of its affluent set of simulation services that efficiently facilitate improving and assessing various scheduling algorithms for heterogeneous distributed computing environments, i.e., fog and mobile edge environments. This work has analyzed several characteristics, such as execution time, network bandwidth, overall delay, and the precedence-based scheduling mechanism provided by QoS parameters. A proposed environment was simulated with three tiers, each with 100 mobile devices and 20 fog devices reserved for resources. The system configuration for our experiment is set with a scheduling cycle of 100 milliseconds. To assign computing resources to mobile users, 20 fog nodes have been considered. Concurrently, mobile users submit requests to the cloud, with each request containing up to 200 MB of data. For each task, the delay was taken into account. Throughput is essential for efficient data transfer, especially in the IoT-Fog network. The amount of packets the device transfers in a second has been used to gauge its throughput. The devices' combined throughput is approximately 85.6

packets per second. The scheduling choice affects the overall network use according to the expected environment.

Figure 4.6: Analysis of Physical Availability Factor vs. Execution Time

As shown in Fig. 4.6, the physical availability factor declines, and the execution time increases. Based on its presentation, the suggested mobility-aware framework has also been evaluated by utilizing several performance measures. The projected framework consists of 20 fog devices and 100 ms time spam for calculating the execution time at three different criteria, i.e., Physical availability factor greater than 0.60, between 0.30 and equal to 0.60, and less than or equal to 0.30. The results also show that the execution time falls as the number of fog devices increases and increases when the number of fog devices decreases.

The projected algorithm's overall average response time is 78.6 milliseconds. The response time was when the fog nodes responded to a request. The response time is the total time spent on node deployment, communication, and request processing. Three scheduling algorithms have been compared based on response time and are shown in Fig. 4.7.

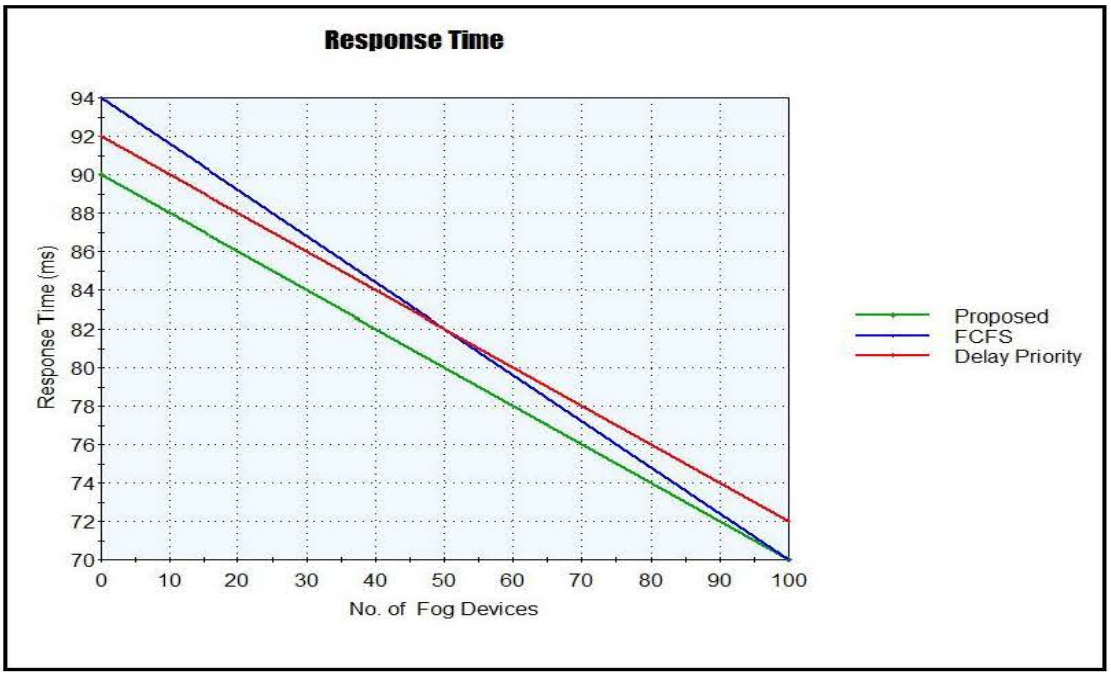

Figure 4.7: Analysis of response time of existing and proposed algorithms

The Proposed mobility-aware scheduling strategy also effectively reduced delays for various user requests. However, when more mobile users move and send requests to fog devices, the fog devices have to handle all mobile device requests. As a result, the overall delays decrease with the powerful fog device resources, as shown in Fig. 4.8.

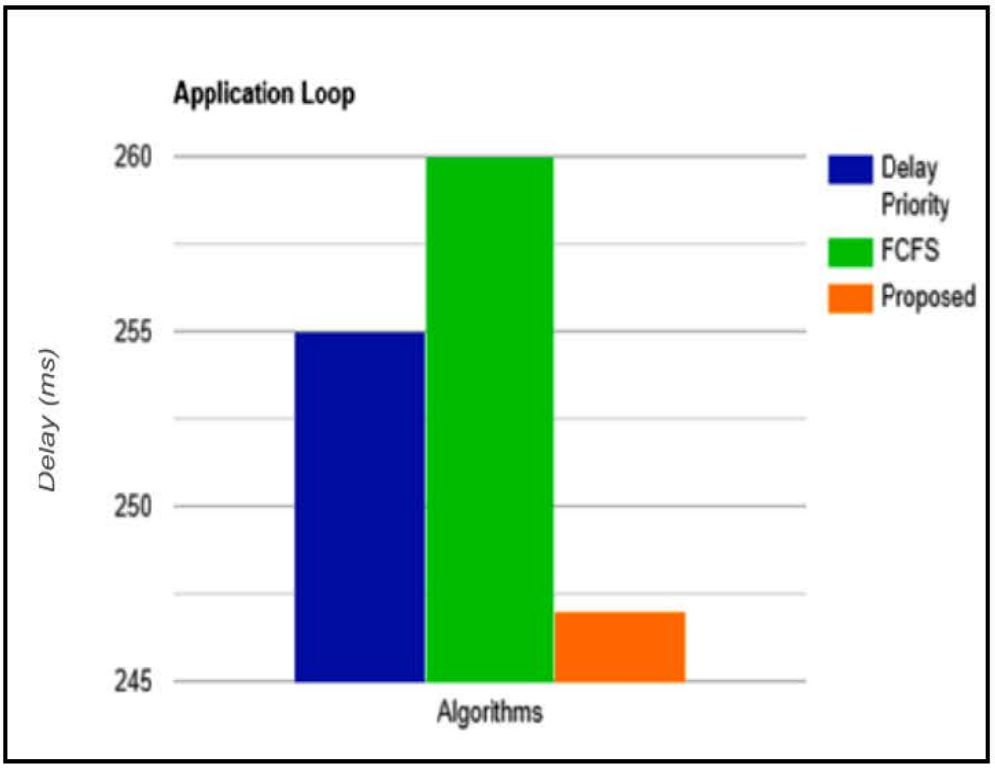

Figure 4.8: Analysis of delay of existing and proposed algorithms

In this case, with the adoption of the proposed strategy, fog resources also handle low-delay demand requests and decrease overall delays of the proposed framework compared to others. Fig. 4.9 depicts the total data transmitted over the network for each scheduling approach. The FCFS technique leads to a specific rise in network use, while the proposed strategy decreases network use. Delay-priority scheduling is more network-intensive and causes increased network usage as a result.

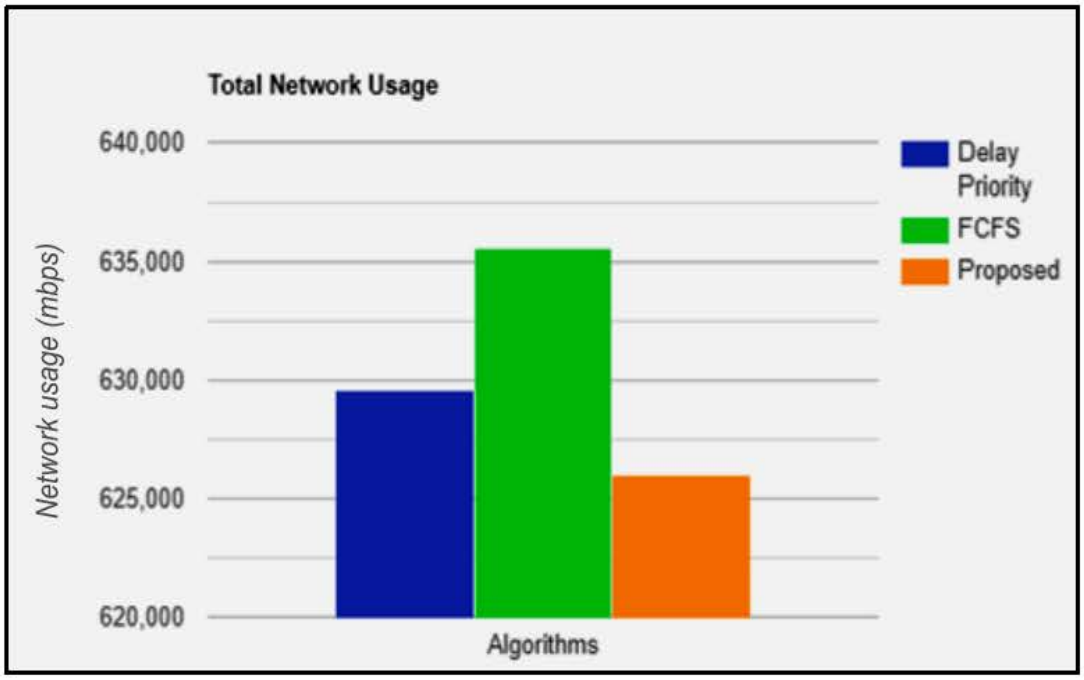

Figure 4.9: Total network usages of proposed and existing algorithms

4.5 Summary

This chapter introduced the mobility-aware and latency-sensitive scheduling strategy in the mobile device composition of a fog environment. The mobility-based scenario brings a dynamic computing demand at the network's edges. The mobility-aware scheduling policy anticipates the mobile user's location and distributes the requests to effective fog devices. The experiment's findings have shown how the execution time is impacted by physical availability. The result also concludes that if we increase the number of fog resources, then it will decrease the execution time and increase the cost. The proposed strategy's response time, delay, and network usage were also calculated, and compared with the existing strategy.

CHAPTER 5

PARTICLE BEE OPTIMIZATION FOR FAULT TOLERANCE

5.1 Introduction

Fog computing is a relatively new paradigm of computation that is spread. Cisco created it in 2012 to help with cloud computing and boost QoS (quality of service) to generate various supported IoT applications. The literal interpretation of the word "fog" reveals the characteristics of this meteorological phenomenon: when clouds are high in the atmosphere, fog descends to the earth and surrounds humans [84]. There are several definitions for fog that all center on the same concepts. One of these definitions is a novel computing archetype called fog computing that might be considered for further cloud computing reach at a network's periphery to lessen the cloud's load by carefully finding more compact heterogeneous devices like PCs, Gateways, Data Centres, Servers, etc. Fog Computing can sustain IoT applications in terms of time-responsive transmission and network bandwidth. The attributes of fog computing consist of [85]:

- Low latency and awareness of service quality mean that nearby edge devices handle the source data.
- Bandwidth reduction and reduction of data transmission time.
- The capacity for decentralized decision-making.
- Heterogeneity and compatibility between different communication technologies.

Fog computing is a kind of distributed computing that uses "edge" nodes to access cloud-like services. In fog computing, "*Fog*" means the same thing as everyday life. It is possible to encounter fog between clouds and the ground in the real world, and this concept is employed in fog computing to symbolize the region between clouds and the physical world. In other words, fog nodes are set up in the path that connects the cloud to the user's endpoint devices, as shown in Fig. 5.1.

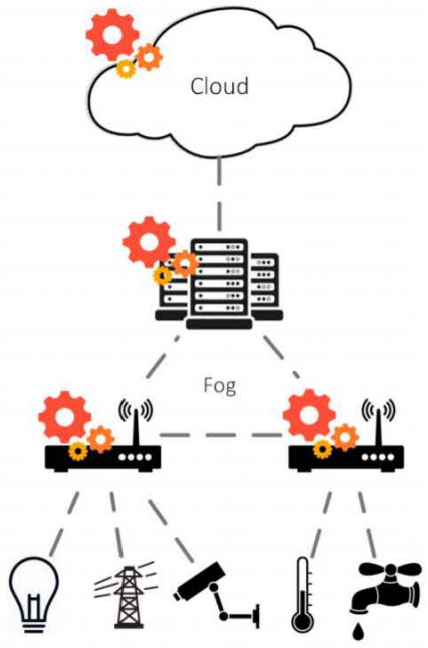

Figure. 5.1: Deployment of Fog in IoT Applications.

The central objective of this research was to develop a technique called "Fault Tolerance & Service Placement" for Fog Environments. The two exclusive strategies, Check Pointing and Replication and Novel Particle Bee Optimisation, are derived from hybridizing the original Ant Bee Colony and Particle Swarm Optimisation scheme. The employed methods provide numerous functions in a fog environment, including:

- Service for fog placement through managing mobile clients from numerous data centers in various regions.
- Using the nearest neighbour to reduce packet transfer time while performing transmission checkpointing.
- Enhanced efficiency during proxy replication.
- Network Optimisation Using Best Cost Path Determination.

In this chapter, the technique "Fault Tolerance & Service Placement" is implemented to address live server fault tolerance by simulating server operations when synchronizing it with a proxy server; checkpointing has three benefits: (i) Quality of Service Awareness (QoS) awareness, (ii) low latency, and (iii) reduced network bandwidth utilization.

5.2 Quality Of Service (QoS)

The foundation of Quality of Service (QoS) is based mainly on reliability, network latency, energy consumption, and throughput, all of which crucial services and imperative for fog are computing. In addition, it is also essential to consider the management of resources, utilization of power models, resource scheduling policies, and handling of power outages for ensuring QoS. The accuracy of the result or action might be impacted if any sensors malfunction for whatever reason [86]. Fog must uphold high reliability and strict QoS promises because it is designed to interact with latency susceptible systems. The latency consciousness requirement won't be met in such a case. According to Madsen et al., the accessibility of a variety of approaches and schemes collaborates with the fidelity of network connection and data to ensure correctness, which is essential for creating projects based on fog computing.

5.2.1 QoS in fog computing

Providing an appropriate service quality for location awareness and latency-receptive applications inside a cloud computing infrastructure is one of the most significant reasons for adopting fog computing. In addition, quality of service is necessary for fog computing to assess and monitor the services provided effectively. The QoS practices utilized in cloud computing do not apply to fog computing because of the unique properties of fog computing, such as the vast diversity of devices, geographical spread, and mobility. According to previous research findings, future QoS-based fog environments should prioritize optimizing quality of service (QoS) characteristics, including response time, resource usage, cost, execution time, dependability, energy consumption, availability, scalability, and throughput.

Delivering an appropriate degree of QoS is a crucial concern in fog computing. Some aspects of QoS in cloud computing are irrelevant in fog computing due to distinct factors, such as heterogeneity, mobility, and dispersion. Supporting real-time applications is one of the main goals of fog design. Fog-based systems take into account a number of QoS parameters for a successful system design [88].

Throughput: The highest desired service rate that the system can handle.

Deadline: The time a request must be submitted to be processed before it is considered late.

Response time: The interval between a user's request and the time it takes to get a response.

Resource utilization: The best use possible of a system's resources.

Cost: The price a service applicant must pay for calculation, communications, or data storage over a specific period.

Execution time: The time it takes for a program to run entirely.

Energy consumption: The quantity of energy a resource uses to provide the required service.

Reliability: A system's capacity to complete its needed functions under specified conditions and at specified times.

Availability: A system's capacity to guarantee that the requested resources are available and performing as intended.

Scalability: The capacity of a computer system is such that system performance is maintained with an increase in the number of service requests or resource applications.

Security: Safe approaches safeguard the data now accessible in the fog/cloud environment.

5.2.2 QoS in IoT

IoT's primary goal is to connect devices to the Internet. By building a network of interconnected objects, this objective is accomplished. The quantity of data gathered would rapidly rise as IoT devices proliferate. This rise is caused by the gadgets' capacity to provide many services simultaneously. Consequently, numerous elements necessary for user-side QoS prediction have been clarified. The Quality of Service (QoS) service, which prioritizes application traffic over a network, is also referred to as a quality assurance service of network connection. IoT networking relies heavily on quality of service (QoS) since it manages the network's operation and resources and provides a secure connection. QoS systems analyze traffic to control things like delays, bandwidth use, and lost packages. The Internet of Things and the services that it enables should prioritize the quick and effective delivery of data. Because of this, IoT has to provide various services, from which users may choose the most appropriate option depending on the Quality of Service requirements. Because

Internet of Things systems combine computation and communication with physical objects, a wide variety of criteria and metrics must be satisfied. To have an efficient and successful IoT system, each component must fulfill the Quality of Service standards[89][90].

Regarding the objects, the IoT devices' quality of service (QoS) may be influenced by power consumption, coverage, the ideal number of active sensors, sensor quality, data bulk, trustworthiness, and mobility. When taken by itself, any of the indicators discussed above run the risk of not being significant. However, there is much more to consider when considering the various devices employed for providing. For instance, the accumulated power consumption of hundreds of sensors that only use 0.9W each might significantly affect the amount of power used by the network. For communication, the quality of service provided by the network would comprise throughput, reaction time, availability, capacity, maintenance time, and jitter. Concerning computing, the data analysis programming models inside the cloud need quality service measures that fulfill throughput and reaction time requirements.

However, the QoS requirements for the cloud infrastructure layer may be broken down into four categories: CPU consumption, memory usage, network latency, and network bandwidth. With context to the applicability areas of the Internet of Things, the primary Quality of Service criteria shift depending on the application's field. For instance, an application that monitors a patient's health must satisfy requirements for confidentiality, security, exactitude, longevity, responsiveness, resilience, accuracy, dependability, and availability. However, time-sensitive applications regard low latency as the need with the greatest priority. In contrast, less time-critical applications, such as building automation, emphasize the usage of the network and the efficiency of energy use.

5.2.3 Qos requirements

Next, the apps that will run on fogs, especially those made feasible by implementing fogs, will be identified, and the service quality demands of fog applications, along with other needs, will be supplied. Additional requirements will also be shown[91][92].

Bandwidth: Users need to be able to set a minimum bandwidth requirement for an application's needs. This might call for a Guaranteed Bit Rate (GBR) requirement and

a Non-Guaranteed Bit Rate (NGBR) requirement for best-effort applications.

Delay sensitivity: applications that need to work instantly, such as facial recognition in large groups, need the assurance that delay limitations will not be exceeded.

Packet loss: Applications susceptible to data loss, such as those dealing with finances, may need lossless transfer services.

Reliability: Certain applications need failing fog components to be restored as rapidly as possible to carry out actions within the allotted amount of delay.

Availability: Consists of a measurement of the frequency with which, during the process's execution, users have access to the fog's resources of the application. On the other hand, applications and services whose execution may be delayed for a limited time don't insist on having access to materials at all times.

Security: Personal and vital information will be sent through applications. It is necessary to implement either information security techniques or mission-critical application software. In addition, As the future information superhighway, the fog network necessitates stringent measures to ensure the privacy of all user-generated content.

Data location: The end device may store data locally, using a Fog layer or an off-site Cloud storage facility. All three alternatives are viable.

Mobility support: Even for end users with high mobility, constant connectivity should be guaranteed. End devices provide processing power for applications like rendering, and continuous connectivity is essential to carry out all the processing. That is required. Connectivity is also critical for collaborative operations in separate fog devices or fogs.

Scalability: Because of user movement and the usage of applications or sensors may cause changes in the user count in a fog. Big data processing may need some time to handle data streams. Fog demand might vary. Hence, resource flexibility must be offered to meet such demands to make such applications profitable.

5.3 Proposed Methodology

Several services, including i) monitoring one fog node by another, ii) performing replication for a fog service, iii) identifying the list of neighboring nodes, and iv)

providing transmission checkpointing, can be supported by fault tolerance in fog computing. By replicating the server's operations and checkpointing when synchronizing it with a proxy server, the proposed technique "Fault Tolerance & Service Placement" addresses the live server fault tolerance and offers i) Quality of Service Awareness, ii) Low Latency, and iii) Reduce Network Bandwidth usage. The two exclusive strategies, Check Pointing and Replication and Novel Particle Bee Optimisation, are derived from hybridizing the original Ant Bee Colony and Particle Swarm.

In a fog environment, applied approaches provide various services, including.

- Service for fog placement through managing mobile clients from numerous data centers in various regions.
- Using the nearest neighbor to reduce packet transfer time while performing transmission checkpointing.
- Enhanced efficiency during proxy replication.
- Network Optimisation Using Best Cost Path Determination.

The mentioned suppositions are considered to make fault tolerant in fog computing utilizing IoT.

i) The configuration of fog sensor devices in the projected framework allows them to connect with more than one neighbor node directly or indirectly.

ii) Fog nodes manage node failure and carry out required communications.

iii) Covering numerous distribution zones with fog nodes raises the amount of fault tolerance..

iv) Localization of the input population will facilitate fault finding..

As per the consequence of the projected model, fog nodes offer services while taking into account the following factors: i) resource availability for allocation of services; ii) accessibility maintenance among various services; iii) the limitation of real-time services with periodic information and iv) timing and resource restrictions will not overlap the proxy replications [94]. The algorithm depicts the numerous actions taken during the tasks that are being presented.

The Novel Particle Bee Optimization algorithm initializes the input population and

locates the source and neighbors of the fog sensor. The optimal solution cost is fixed based on analyzing each transmission's completion time. As illustrated in Algorithm 5.3, the step is continued until service to each node is assigned. The framework of the technique mentioned above is shown in Fig. 5.2.

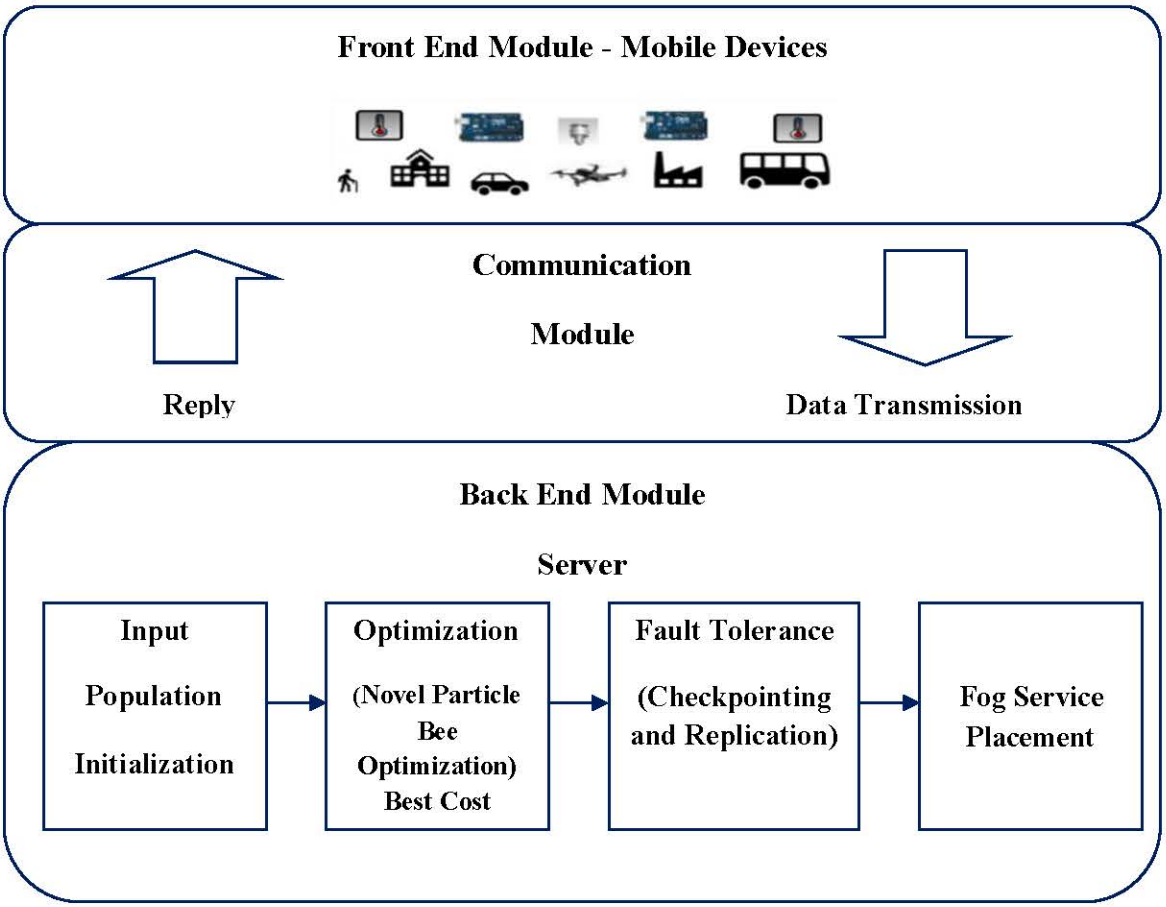

Figure 5.2: Optimized Framework for Fault Tolerance & Service Placement

For communication, the source device connects to the neighbor node. The neighbor node N receives the data packets and reverts the source for each successfully received packet. The acknowledgment contains the number of packets received and the turnaround times (minimum, maximum, and average). The iteration was repeated until every node identified the neighbor [93]. The central server makes device connectivity possible. Cloud data centers house a concentrated amount of computing and storage resources, which user devices can easily access[22].

The Fog service placements provide numerous advantages. The services provided to several client devices are attended to by both the fog server and its replication proxy server. The conduct of ants in nature serves as an inspiration for the optimization technique. An iterative process called novel particle bee optimization creates the most

affordable solution to an optimization challenge. Consequently, this converges the parameters of the optimum cost solution space through further iterations that use better results. The many functions of the suggested system are represented by the algorithms listed below.

Algorithm 5.1 Service Allocation

Input: Initialize input population.

Output: An optimized service allocation among fog sensor nodes.

```
1.    Begin
2.          Initialization: MNClient_ID;
3.          Fog Sensors == Enabled;
4.
5.     for Run((virtulr_gm_0)&&(virtulr_gm_1))
6.     for (virtulr_gm_0; MNclient_ID++)
7.    Actuator Signal==Enabled;
8.     end for
9.     for (virtulr_gm_1; MNclient_ID++)
10.          Actuator Signal==Enabled;
11.    end for
12.   end for
13.  end
```

Algorithm 5.2 Network Path Optimization

Input: Initialize the input population.

Output: The best solution cost calculated.

```
1.  Begin
2.   for (I==x;J==y; Next++;)
3.   {
4.     Next ()
5.    Until Best Solution(Best I, Best J);
6.   }
7.  end for
8.   Neighbour Path Optimization ()
9.   {
10.          Sensor Node ==Source;
11.          Estimate the cost of the Source Node;
12.         Sensor Node ==Neighbour;
13.         Estimate the cost of the Neighbour Node;
14.  }
```

```
15.          Iterated Until Complete Network Path Optimized ;
16.          Display the result for Best Cost();
17.  end // Neighbour Path Optimization.
18. end
```

Algorithm 5.3 Delay Calculation

Input: Initialization

Output: Execution time, energy consumption,
and loop delay have been calculated.

```
1.  Begin
2.      Show()// for Every Node
3.  {
4.      Execution_Time ();
5.      Loop_Delays();
6.      Execution_ Delay_for_CPU_Tuple();
7.      Game_State_Player ();
8.      Game_State_Global ();
9.      Energy_Consumed();
10. }
11. end
```

The algorithms above were implemented using the ifogsim simulation tool. The performance evaluation is contrasted with the two best achievements displayed algorithms, namely the genetic algorithm and the already projected novel gene selection bee mutation scheme.

5.3.1 Network Path Optimization

The source and neighbors of the fog sensor node system are found using the Novel Particle Bee Optimization algorithm, which also sets through initializing the input population. The optimal solution cost is fixed based on analyzing each transmission's completion time. Up till every service is assigned, the process is repeated. The neighbor node N receives the data packets, which notifies the source device for each packet it has received.

The acknowledgment comprises the minimum, maximum, and average turnaround times and the number of packets received. The iteration process was repeated until all the devices used the angel of arrival scheme of localization strategy to locate the nearest neighbor. Circles in Fig. 5.3 represent IoT devices and sensor nodes, while

rectangle boxes represent fog devices in multi-hop communication. Suppose N0 is the fog node. Nodes N21 and N22 of the sensor network are two hops distant from the fog device, while N11, N12, and N13 are one hop away. The first step is determining how far a fog device is from a sensor node.

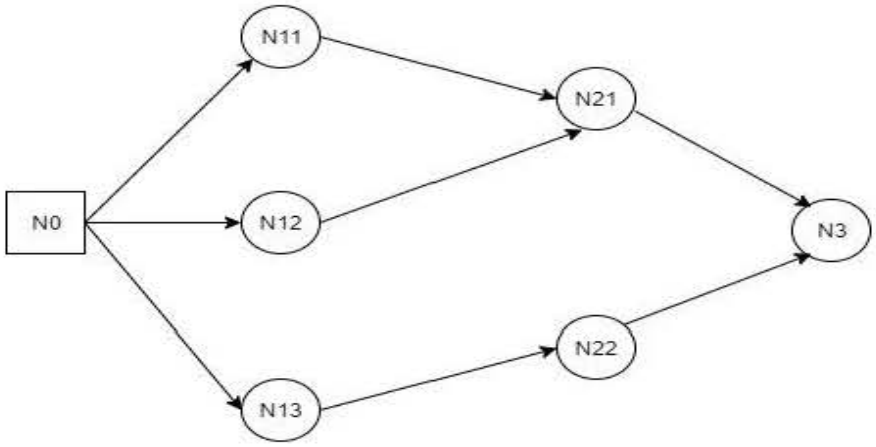

Figure 5.3: Multihop Communication Layout using Path Optimization

Let Dij stand for the distance between the fog device and the sensor node. Assume N0 and N21 are the two distinct nodes that can communicate with one another through routing node N11.

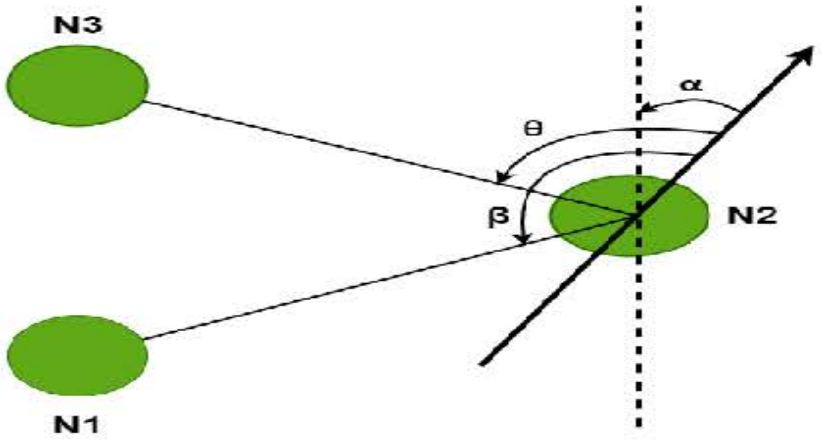

Figure 5.4: Fog Nodes distance calculation using Angle of Arrival (AoA)

In Fig. 5.4, the nodes N1 and N3 represent two different nodes communicating with each other through the routing node N2.

The approaching signal message from node N3 has the elevation and azimuth angle of arrival (AoA) of and, respectively. Consider DT_{ij} to be the distance between nodes N3 and N2. In the figure mentioned above coordinates of N2, the spherical coordinate system is (Radius D_{ij}, inclination α_{ij}, azimuth θ_{ij},) which can also be transferred into

the Cartesian coordinates of the proposed system.

$$X_{ij}= DT_{ij} \text{ Cos } \theta_{ij}, \text{ Sin } \alpha_{ij} \quad (5.1)$$

$$Y_{ij} = DT_{ij} \text{ Sin } \theta_{ij}, \text{ Sin } \alpha_{ij} \quad (5.2)$$

$$Z_{ij}= DT_{ij} \text{ Cos } \alpha_{ij} \quad (5.3)$$

$$CarT_{ij}= [X_{ij}.Y_{ij}.Z_{ij}] \quad (5.4)$$

The Approximation of the coordinates of N3 is to be anticipated through the coordinates of node N1, as shown in Eq. 5.1, Eq. 5.2, and Eq. 5.3 along the X, Y, and Z axes. The distance between the Node N1 and Node N3 is calculated using Eq. 5.5.

$$D_{ij}=\| CarT_{ij}-CarT_{i'j}\| \quad (5.5)$$

The distance between a hop node and a fog node higher than two hops can be calculated by computing the rotation matrix readings between the local coordinate systems of the two neighboring nodes. After obtaining the value of the rotation matrix Q_{ij}, the coordinates of the node N3 in the Local Coordinate system are determined.

$$CartT_{ij}= Q_{ij} CarT_{i' j} + CartT_{ij} \quad (5.6)$$

The service assignments in fog, which provide numerous advantages, come next. The replication proxy server of the fog server shares the idea of attending to numerous client device services. The actions of ants in nature serve as inspiration for the optimization technique. An iterative approach called novel particle bee optimization is used to develop the optimal cost resolution for an optimization problem. This converges the optimum cost solution space parameters through further iterations that use better results. The algorithms represent the many functions of the suggested system.

5.4 Results and Discussion

The mobile devices' transmission readings fixed with timestamps communicated from client to server are considered inputs for service placements. The device running Windows 10 and equipped with an Intel(R) Core(TM) i7-4500U CPU and 8GB of RAM. The proposed work implements a “Fault Tolerance & Service Placement” technique for the FOG Environment. As described in the preceding section, the method uses the revolutionary particle bee optimization algorithm and two innovative algorithms, checkpointing and replication technique.

Table 5.1 and Table 5.3 represent the transmission details of five devices of Device_ID 1-5. The IP 204.79.197.200 and TTL 121 are randomly assigned as per algorithms. The total number of bytes considered for transmission is 32 bytes. This is separated into four data packets, each with a minimum turnaround time of one millisecond, a maximum turnaround time of one millisecond, and an average turnaround time of one millisecond. All packets, however, are received with 0% packet loss. Throughput, response time, scalability, performance, availability, usability, reliability, overhead, and cost are the evaluation criteria for all active devices. The effectiveness of the check-pointing and replication technique is assessed. Concerning the devices, Device ID 3 has the highest throughput (93%), reaction time (62%), performance (94%), availability (97%), and reliability (85%) as depicted in the graph of the Fig 5.5. According to Table 5.5, Device ID 4 has the maximum throughput of 88%, reaction time of 72%, performance of 84%, availability of 61%, and dependability of 99%. The outcome is illustrated in Fig. 5.6 and Table 5., which depicts the Particle Bee Optimization-Best cost calculation. Here, the best cost is identified at iteration 49, with x- 3.0000937146270292 and y- 0.5000543228356094.

Table. 5.1. Packet Transmission details for Checkpointing

Device_ID	IP	No. of bytes	Time (ms.)	TTL	No. of Packets Sent	No. of Packets Received	Packet Loss	Minimum Turn Around Time (ms.)	Maximum Turn Around Time (ms.)	Average Turn Around Time (ms.)
1	204.79.197.200	32	1	121	4	4	0	1	1	1
2	204.79.197.200	32	1	121	4	4	0	1	2	1
3	204.79.197.200	32	1	121	4	4	0	1	2	1
4	204.79.197.200	32	1	121	4	4	0	260	267	262
5	204.79.197.200	32	1	121	4	4	0	1	8	3

This is separated into four data packets, each with a minimum turnaround time of one millisecond, a maximum turnaround time of one millisecond, and an average turnaround time of one millisecond. However, all packages are received with 0% packet loss. The checkpointing and Replication technique's performance is evaluated

based on the parameters throughput, reaction time, scaling, performance, availability, usability, dependability, overhead, and cost for all the active devices. Throughput is the most crucial parameter for more effectual data transmission, predominantly in the proposed IoT-Fog-based framework. Several packets sent by a specific device in a second were used to compute the device's throughput. The time it takes the fog nodes to react to a request is used primarily to calculate response time. Response time in this work also includes the accumulation of node deployment, communication, and make-span times for requests, as well as the capacity of the projected framework to handle user requests without impairing the overall effectiveness of the suggested method.

Table. 5.2. Packet Transmission details for Replication

Device_ID	IP	No. of bytes	TTL	No. of Packets Sent	No. of Packets Received	Packet Loss	Minimum Turn Around Time (ms.)	Maximum Turn Around Time (ms.)	Average Turn Around Time (ms.)
1	98.137.246.8	32	48	4	4	0	260	261	260
2	98.137.246.8	32	48	4	4	0	260	261	260
3	172.217.166.110	32	54	4	4	0	1	1	1
4	98.137.246.8	32	48	4	4	0	260	261	260
5	172.217.166.110	32	48	4	4	0	1	1	1

The primary obligation in this system is the fog node accessibility for receiving the approaching request and processing that request in the node's uptime to the node's total time. Usability is the statistic that pinpoints how many computing and storage resources are needed to complete the procedure effectively. On the other hand, the system's effectiveness might determine its dependability to produce precise results. In terms of the devices, Device_ID 3 has the maximum throughput (93%), reaction time (62%), scalability (76%), performance (94%), availability (97%), and reliability (85%). The outcomes are shown in the Fig. 5.4.

Table 5.3: Checkpointing Performance Evaluation

Device_ID	Status	Throughput	Response Time	Scalability	Performance	Availability	Usability	Reliability	Overhead Associated	Cost Effectiveness
1	Active	88	67	61	80	90	63	64	99	87
2	Active	84	76	63	64	66	80	79	98	73
3	Active	93	62	76	94	97	68	85	80	76
4	Active	78	63	73	64	93	93	79	64	95
5	Active	67	73	80	62	96	93	71	92	77

Table. 5.4. Replication Testing Evaluation

Device_ID	Status	Throughput	Response Time	Scalability	Performance	Availability	Usability	Reliability	Overhead Associated	Cost Effectiveness
1	Active	87	92	73	95	98	72	82	68	85
2	Active	83	67	92	95	92	98	61	88	75
3	Active	67	96	75	64	89	74	68	60	62
4	Active	88	79	81	87	61	92	99	93	73
5	Active	74	85	98	77	79	69	73	94	62

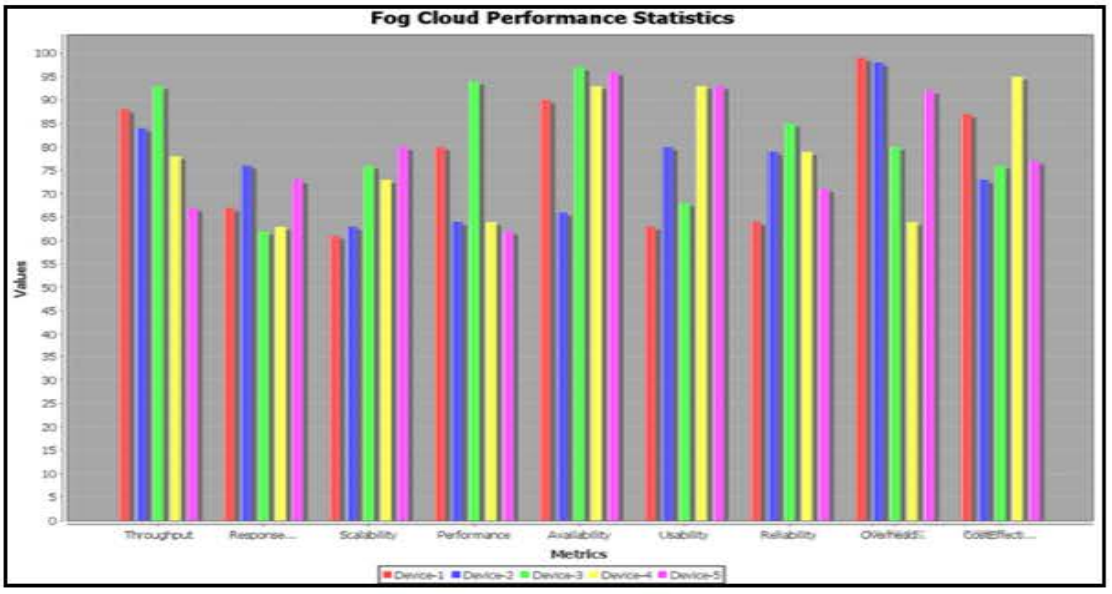

Figure. 5.5. Analysis of Proposed Framework Overall Performance Statistics

According to Table 5.5, Device_ID 4 has the maximum throughput of 88% with a response time of 79%, performance of 87%, availability of 61%, and reliability of 99%; overhead costs and cost-effectiveness are 93 and 73, respectively. The Particle Bee Optimization-Best cost calculation identifies the best cost at iteration 49, with x- 3.00 and y-0.500.

Table 5.5 Particle Bee Optimization-Best Cost Calculation

Iteratio	Best X	Best Y	Value
0	3.4515373536981553	0.6697843810097153	0.25318166066926995
1	2.5212136778952026	0.3479606909770237	0.12962068533529675
3	3.2409214817449	0.5641936583638616	0.019336250778752045
5	3.2409214817449	0.5641936583638616	0.019336250778752045
7	3.2409214817449	0.5641936583638616	0.019336250778752045
9	3.1661662016761607	0.5387507123923697	0.009133244544166903
11	0.009133244544166903	0.5387507123923697	0.009133244544166903
49	3.0000937146270292	0.5000543228356094	2.2116045281968125E-

The execution latency of the Fog device is shown in Table 5.8 for the player states Virtulr_game_0 and Virtulr_game_1. The actuator signals are enabled for each device in tandem with the game player, performing the corresponding reply action for each transmission as illustrated in algorithm 5.1. The energy usage of various devices according to algorithm 5.3 is shown in Table 5.6.

Table 5.6: Fog Device Tuple Execution Delay

Device	**Execution Delay (sec.)**
Player Game State	2.332
EGG	3.416
Concentration	0.185
Sensor	3.186
Global Game State	0.056

Table 5.7: depicts the energy consumption of various devices as per algorithm 3.

Devices	**Energy Consumption (kWh)**
Cloud	459.585
Proxy-server	250.299
Fog Device	250.299

Table 5.8: Comparison with the existing algorithm

Parameters	**Existing Genetic Algorithm**	**Novel Particle Bee Optimization**
Cost of Fog Cloud Execution ($)	85042.0	81042.2
Total Network Utilization (Mbps)	635550.0	612643.3

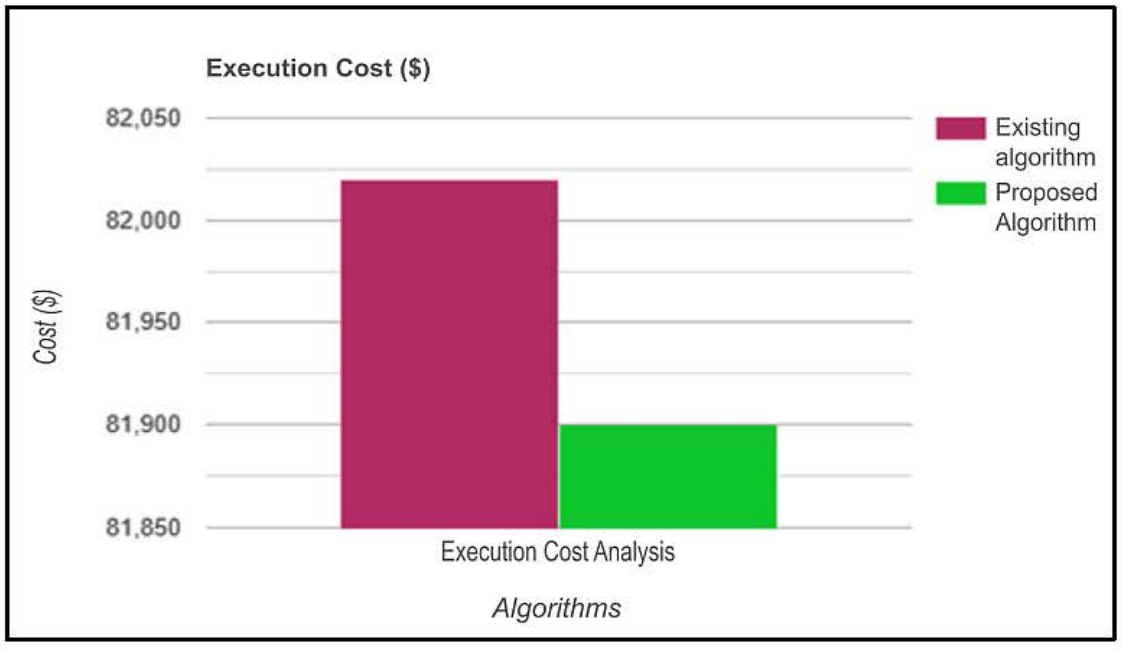

Figure. 5.6: Comparison of existing and proposed algorithms in terms of execution cost

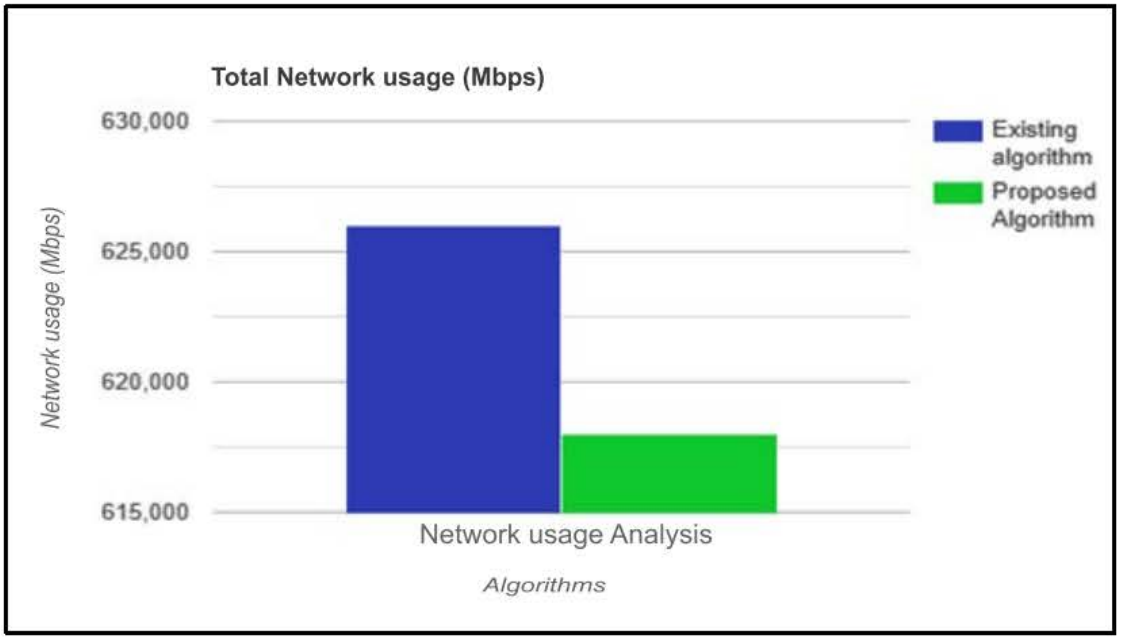

Figure. 5.7: Comparison of existing and proposed algorithms in terms of network utilization

Table 5.8 depicts a performance evaluation of the projected technique in contrast to the already existing Genetic Algorithm (GA) and suggested Bee Mutation scheme. The comparison results show that Novel particle bee optimization leads to reduced network usage of 612643.34 and better execution cost of fog cloud 84042.20 considering the other two existing algorithms depicted in Fig. 5.6 and Fig. 5.7.

5.5 Summary

This chapter projected the "Fault Tolerance & Service Placement" framework that handles live server fault tolerance by repeating server operations and checkpointing when communicating with the proxy server. This minimizes the network use of 612643.34 at a lower fog cloud execution cost of 84042.20 in the context of evolving fog technologies. Devive_ID 3 has the maximum throughput of 93%, reaction time of 62%, performance of 94%, availability of 97%, and reliability of 85% for the devices. Further study into fog service placements may be conducted to improve fog sensor devices' energy and resource efficiency.

CHAPTER 6
CONCLUSION & FUTURE SCOPE

6.1 Conclusion

The proposed approach included checkpoints and replication procedures, two well-known fault diagnostic methods, through a fog system monitor. The replication technique avoids the checkpoint mechanism's delay. According to the results, the suggested framework works better than the current model, using the network at 626020 Mbps for $82020.20. The checkpoint procedures start the process from the pre-defined checkpoints established at the failure occurrences, which makes the suggested CRBM method fault tolerant.

A mobility-aware service placement policy determines the mobile node's position and assigns users' requests to the best fog devices. It can also improve the response time and reduce the application loop delay. In this work, a mobility-aware scheduling mechanism has been given for determining the position of mobile devices and distributing their requests to fog nodes. The movement type, pattern, and direction are used to determine the mobile node's physical availability in the projected framework. The outcomes of the experiments have shown how physical accessibility affects execution time.

The proposed framework, "Fault Tolerance & Service Placement," addresses real-time server faults by duplicating the server's operations using a checkpoint while synchronizing it during a proxy server. The solution addresses live server failure tolerance by duplicating server actions and check-pointing when synchronized with a proxy server. It also discusses pioneering particle bee optimization and an efficient strategy for optimizing the optimal cost path utilizing the node localization scheme, i.e., angle of arrival. The suggested plan reduces network consumption by 618020 for an execution cost of $81900 less fog.

6.2 Future Scope

The projected fault tolerance-based framework has improved the overall performance of the fog environment, and this work can further be enhanced by reducing the overhead associated, response time, and execution cost. More realistic and sophisticated mobility patterns can be implemented based on real data sets.

REFERENCES

[1] Gokhale, P., Bhat, O., & Bhat, S. (2018). Introduction to IOT. "International Advanced Research Journal in Science, Engineering and Technology," 5(1), 41-44,2018.

[2] Sethi, P., & Sarangi, S. R. (2017). "Internet of things: architectures, protocols, and applications". Journal of Electrical and Computer Engineering, 2017.

[3] Lo, N., Flaus, J. M., & Adrot, O. (2019, July). "Review of Machine Learning Approaches In Fault Diagnosis applied to IoT System".

[4] Yi, S., Li, C., & Li, Q. (2015, June). "A survey of fog computing: concepts, applications and issues". In Proceedings of the 2015 workshop on mobile big data pp. 37-42, 2015.

[5] Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F., & Parashar, M. (2017). "Mobility-aware application scheduling in fog computing". IEEE Cloud Computing, 4(2), 26-35.

[6] Wang, K., Shao, Y., Xie, L., Wu, J., & Guo, S. "Adaptive and fault-tolerant data processing in healthcare IoT based on fog computing". IEEE Transactions on Network Science and Engineering, 2018.

[7] Brogi, A., Forti, S., Ibrahim, A., & Rinaldi, L. (2018, April). Bonsai in the fog: An active learning lab with fog computing. In 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC) (pp. 79-86).

[8] Hu, P., Dhelim, S., Ning, H., and Qiu, T., "Survey on fog computing: architecture, key technologies, applications and open issues," Journal of Network and Computer Applications, vol 98, pp. 27-42, 2017.

[9] Amjad, M.K., Butt, S.I., Kousar, R., Ahmad, R., Agha, M.H., Faping, Z., Anjum, N. and Asgher, U., (2018). " Recent research trends in genetic algorithm based flexible job shop scheduling problems". Mathematical Problems in Engineering.

[10] Ahmed, E., and Rehmani, M. H., " Mobile edge computing: opportunities, solutions, and challenges," Future Generations Computer System, vol. 70, Pages 59-63,2017.

[11] Ahmed, A., & Ahmed, E., " A survey on mobile edge computing," In Proc of 10th International Conference on Intelligent Systems and Control (ISCO), IEEE, 2016,pp. 1-8.

[12] Mach, P., and Becvar, Z., "Mobile edge computing: A survey on architecture and computation offloading," IEEE Communications Surveys & Tutorials,2017

[13] Mohamed, N., Al-Jaroodi, J., & Jawhar, I. (2019, January). "Towards fault tolerant fog computing for IoT-based smart city applications". In 2019 IEEE 9th annual computing and communication workshop and conference (CCWC) (pp. 0752-0757). IEEE.

[14] Wang, N., Varghese, B., Matthaiou, M., and Nikolopoulos, D. S., "ENORM: A framework for edge node resource management," IEEE Transactions on Services Computing, pp. 1-14,2017.

[15] Satria, D., Park, D., & Jo, M., "Recovery for overloaded mobile edge computing," Future Generation Computer Systems, vol. 70, pp. 138-147, 2017.

[16] Verba, N., Chao, K. M., James, A., Goldsmith, D., Fei, X., and Stan, S. D., " Platform as a service gateway for the Fog of Things," Advanced Engineering Informatics, vol. 33,pp. 243-257,2016.

[17] Liu, Y., Fieldsend, J. E., & Min, G. (2017). A framework of fog computing: Architecture, challenges, and optimization. IEEE Access, 5, 25445-25454.

[18] Baranwal, G., Yadav, R., & Vidyarthi, D. P. (2020)." QoE aware IoT application placement in fog computing using modified-topsis". Mobile Networks and Applications, 25, 1816-1832.

[19] Kimovski, D., Mehran, N., Kerth, C. E., & Prodan, R. (2021). "Mobility-aware iot applications placement in the cloud edge continuum". IEEE Transactions on Services Computing.

[20] Ghosh, S., Mukherjee, A., Ghosh, S. K., & Buyya, R. (2019). Mobi-iost: mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications. IEEE Transactions on Network Science and Engineering, 7(4), 2271-2285.

[21] Waqas, M., Niu, Y., Ahmed, M., Li, Y., Jin, D., & Han, Z. (2018). "Mobility-aware fog computing in dynamic environments: Understandings and implementation". IEEE Access, 7, 38867-38879.

[22] Alarifi, A., Abdelsamie, F., & Amoon, M. (2019). "A fault-tolerant aware scheduling method for fog-cloud environments". PloS one, 14(10), e0223902.

[23] Souza, V. B., Masip-Bruin, X., Marín-Tordera, E., Ramírez, W., & Sánchez-López, S. (2017, June). "Proactive vs reactive failure recovery assessment in combined Fog-to-Cloud (F2C) systems". In 2017 IEEE 22nd international workshop on computer aided modeling and design of communication links and networks (CAMAD) (pp. 1-5). IEEE.

[24] Skarlat, O., Nardelli, M., Schulte, S., Borkowski, M., & Leitner, P. (2017). "Optimized IoT service placement in the fog". Service Oriented Computing and Applications, 11(4), 427-443.

[25] Wang, N., Varghese, B., Matthaiou, M., & Nikolopoulos, D. S. (2017)." ENORM: A framework for edge node resource management". IEEE transactions on services computing.

[26] Verma, K., Kumar, A., Islam, M. S. U., Kanwar, T., & Bhushan, M. (2021). "Rank based mobility-aware scheduling in fog computing". Informatics in Medicine Unlocked, 24, 100619.

[27] Mahmud, R., Ramamohanarao, K., & Buyya, R. (2018). "Latency-aware application module management for fog computing environments." ACM Transactions on Internet Technology (TOIT), 19(1), 1-21 .

[28] M. Al-Khafajiy, T. Baker, H. Al-Libawy, A. Waraich, C. Chalmers, and O. Alfandi, "Fog computing framework for internet of things applications," Proc. - Int. Conf. Dev. eSystems Eng. DeSE, vol. 2018-Septe, no. February 2019, pp. 71–77, 2019, doi: 10.1109/DeSE.2018.00017.

[29] M. García-Valls, C. Calva-Urrego, and A. García-Fornes, "Accelerating smart eHealth services execution at the fog computing infrastructure," Futur. Gener. Comput. Syst., vol. 108, pp. 882–893, 2020, doi: 10.1016/j.future.2018.07.001.

[30] M. Verma, N. Bhardawaj, and A. K. Yadav, "An architecture for Load

Balancing Techniques for Fog Computing Environment," Int. J. Comput. Sci. Commun., vol. 6, no. 2, pp. 269–274, 2015, doi: 10.090592/IJCSC.2015.627.

[31] N. Mazumdar, A. Nag, and J. P. Singh, "Trust-based load-offloading protocol to reduce service delays in fog-computing-empowered IoT," Comput. Electr. Eng., vol. 93, no. October 2020, p. 107223, 2021, doi: 10.1016/j.compeleceng.2021.107223.

[32] H. K. Apat, B. Sahoo, P. Maiti, and P. Patel, "Review on QoS aware resource management in fog computing environment," Proc. - 2020 IEEE Int. Symp. Sustain. Energy, Signal Process. Cyber Secur. iSSSC 2020, no. July 2021, 2020, doi: 10.1109/iSSSC50941.2020.9358897.

[33] J. wen Xu, K. Ota, M. xiong Dong, A. feng Liu, and Q. Li, "SIoTFog: Byzantine-resilient IoT fog networking," Front. Inf. Technol. Electron. Eng., vol. 19, no. 12, pp. 1546–1557, 2018, doi: 10.1631/FITEE.1800519.

[34] J. P. Araujo Neto, D. M. Pianto, and C. G. Ralha, "An agent-based fog computing architecture for resilience on amazon EC2 spot instances," Proc. - 2018 Brazilian Conf. Intell. Syst. BRACIS 2018, no. Cic, pp. 360–365, 2018, doi: 10.1109/BRACIS.2018.00069.

[35] F. E. F. Samann, S. R. M. Zeebaree, and S. Askar, "IoT Provisioning QoS based on Cloud and Fog Computing," J. Appl. Sci. Technol. Trends, vol. 2, no. 01, pp. 29–40, 2021, doi: 10.38094/jastt20190.

[36] Z. Bakhshi, G. Rodriguez-Navas, and H. Hansson, "Fault-tolerant Permanent Storage for Container-based Fog Architectures," Proc. IEEE Int. Conf. Ind. Technol., vol. 2021-March, pp. 722–729, 2021, doi: 10.1109/ICIT46573.2021.9453473.

[37] J. Tong, H. Wu, Y. Lin, Y. He, Y. He, and J. Liu, "Fog-Computing-Based Short-Circuit Diagnosis Scheme," IEEE Trans. Smart Grid, vol. 11, no. 4, pp. 3359–3371, 2020, doi: 10.1109/TSG.2020.2964805.

[38] U. Ozeer, X. Etchevers, L. Letondeur, F. G. Ottogalli, G. Salaün, and J. M. Vincent, "Resilience of stateful IoT applications in a dynamic fog environment," ACM Int. Conf. Proceeding Ser., pp. 332–341, 2018, doi: 10.1145/3286978.3287007.

[39] M. H. Naim, J. M. Zain, and K. A. Jalil, "Providing Fault Tolerance Mechanism in Clinical Support System Through Fog Computing as Middleware", [Online]. Available: http://journals.uob.edu.bh

[40] M. Kaur and R. Aron, "FOCALB: Fog Computing Architecture of Load Balancing for Scientific Workflow Applications," J. Grid Comput., vol. 19, no. 4, 2021, doi: 10.1007/s10723-021-09584-w.

[41] H. Wadhwa and R. Aron, "TRAM: Technique for resource allocation and management in fog computing environment," J. Supercomput., vol. 78, no. 1, pp. 667–690, 2022, doi: 10.1007/s11227-021-03885-3.

[42] P. Kochovski and V. Stankovski, "Building applications for smart and safe construction with the DECENTER Fog Computing and Brokerage Platform," Autom. Constr., vol. 124, no. December 2020, p. 103562, 2021, doi: 10.1016/j.autcon.2021.103562.

[43] A. A. Science and A. Tsega, "DESIGNING A DEADLINE AWARE DATA OFFLOADING ALGORITHM FOR FOG," no. June, 2020.

[44] T. Choudhari, M. Moh, and T. S. Moh, "Prioritized task scheduling in fog computing," Proc. ACMSE 2018 Conf., vol. 2018-Janua, 2018, doi: 10.1145/3190645.3190699.

[45] F. Murtaza, A. Akhunzada, S. ul Islam, J. Boudjadar, and R. Buyya, "QoS-aware service provisioning in fog computing," J. Netw. Comput. Appl., vol. 165, p. 102674, 2020, doi: 10.1016/j.jnca.2020.102674.

[46] B. M. Nguyen, H. T. T. Binh, T. T. Anh, and D. B. Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based Bag-of-Tasks application in Cloud-Fog computing environment," Appl. Sci., vol. 9, no. 9, 2019, doi: 10.3390/app9091730.

[47] R. Mahmud, A. N. Toosi, K. Ramamohanarao, and R. Buyya, "Context-Aware Placement of Industry 4.0 Applications in Fog Computing Environments," IEEE Trans. Ind. Informatics, vol. 16, no. 11, pp. 7004–7013, 2020, doi: 10.1109/TII.2019.2952412.

[48] N. El-Bahnasawy, A. Elnattat, A. El-sayed, and S. Elkazaz, "Performance Enhancement of Fog Environment with Deadline Aware Resource Allocation

Algorithm," Menoufia J. Electron. Eng. Res., vol. 31, no. 2, pp. 107–119, 2022, doi: 10.21608/mjeer.2022.98856.1038.

[49] M. A. B. Al-Tarawneh, "Bi-objective optimization of application placement in fog computing environments," J. Ambient Intell. Humaniz. Comput., vol. 13, no. 1, pp. 445–468, 2022, doi: 10.1007/s12652-021-02910-w.

[50] X. Q. Pham, N. D. Man, N. D. T. Tri, N. Q. Thai, and E. N. Huh, "A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing," Int. J. Distrib. Sens. Networks, vol. 13, no. 11, 2017, doi: 10.1177/1550147717742073.

[51] A. C. Caminero and R. Muñoz-Mansilla, "Quality of service provision in fog computing: Network-aware scheduling of containers," Sensors, vol. 21, no. 12, pp. 1–16, 2021, doi: 10.3390/s21123978.

[52] C. H. Hong, K. Lee, M. Kang, and C. Yoo, "QCon: QoS-aware network resource management for fog computing," Sensors (Switzerland), vol. 18, no. 10, pp. 1–21, 2018, doi: 10.3390/s18103444.

[53] Q. Shaheen, M. Shiraz, M. U. Hashmi, D. Mahmood, Z. Zhiyu, and R. Akhtar, "A Lightweight Location-Aware Fog Framework (LAFF) for QoS in Internet of Things Paradigm," Mob. Inf. Syst., vol. 2020, 2020, doi: 10.1155/2020/8871976.

[54] G. S. Aujla, R. Chaudhary, N. Kumar, R. Kumar, and J. J. P. C. Rodrigues, "An Ensembled Scheme for QoS-Aware Traffic Flow Management in Software Defined Networks," IEEE Int. Conf. Commun., vol. 2018-May, no. October 2020, 2018, doi: 10.1109/ICC.2018.8422596.

[55] J. Yao and N. Ansari, "QoS-Aware Fog Resource Provisioning and Mobile Device Power Control in IoT Networks," IEEE Trans. Netw. Serv. Manag., vol. 16, no. 1, pp. 167–175, 2019, doi: 10.1109/TNSM.2018.2888481.

[56] T. Y. Kan, Y. Chiang, and H. Y. Wei, "QoS-Aware Mobile Edge Computing System: Multi-Server Multi-User Scenario," 2018 IEEE Globecom Work. GC Wkshps 2018 - Proc., no. 1, pp. 1–6, 2019, doi: 10.1109/GLOCOMW.2018.8644384.

[57] A. S. M. S. Sanwar Hosen et al., "A QoS-Aware Data Collection Protocol for

LLNs in Fog-Enabled Internet of Things," IEEE Trans. Netw. Serv. Manag., vol. 17, no. 1, pp. 430–444, 2020, doi: 10.1109/TNSM.2019.2946428.

[58] A. Yousefpour et al., "FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework," IEEE Internet Things J., vol. 6, no. 3, pp. 5080–5096, 2019, doi: 10.1109/JIOT.2019.2896311.

[59] S. Tuli and G. Casale, Optimizing the Performance of Fog Computing Environments Using AI and Co-Simulation, vol. 1, no. 1. Association for Computing Machinery, 2022. doi: 10.1145/3491204.3527490.

[60] A. Hosseini Bidi, Z. Movahedi, and Z. Movahedi, "A fog-based fault-tolerant and QoE-aware service composition in smart cities," Trans. Emerg. Telecommun. Technol., vol. 32, no. 11, pp. 1–20, 2021, doi: 10.1002/ett.4326.

[61] J. C. Guevara, R. da S. Torres, and N. L. S. da Fonseca, "On the classification of fog computing applications: A machine learning perspective," J. Netw. Comput. Appl., vol. 159, 2020, doi: 10.1016/j.jnca.2020.102596.

[62] X. Guo, H. Lin, Z. Li, and M. Peng, "Deep-Reinforcement-Learning-Based QoS-Aware Secure Routing for SDN-IoT," IEEE Internet Things J., vol. 7, no. 7, pp. 6242–6251, 2020.

[63] I. Martinez, A. S. Hafid, and M. Gendreau, "Robust and Fault-Tolerant Fog Design and Dimensioning for Reliable Operation," IEEE Internet Things J., vol. 9, no. 19, pp. 18280–18292, 2022, doi: 10.1109/JIOT.2022.3157557.

[64] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," IEEE Internet Things J., vol. 4, no. 5, pp. 1185–1192, 2017, doi: 10.1109/JIOT.2017.2701408.

[65] Plachy, J., Becvar, Z., and Strinati, E. C., " Dynamic resource allocation exploiting mobility prediction in mobile edge computing," In Proc of IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016, pp. 1-6.

[66] Roman, R., Lopez, J., & Mambo, M. (2 A. C. Caminero and R. Muñoz-Mansilla, "Quality of service provision in fog computing: Network-aware scheduling of containers," Sensors, vol. 21, no. 12, pp. 1–16, 2021, doi: 10.3390/s21123978.

[67] Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., and Nikolopoulos, D. S., "Feasibility of fog computing," ar[Xiv preprint arXiv:1701.05451.

[68] Canali, C., & Lancellotti, R. (2019). "Gasp: Genetic algorithms for service placement in fog computing systems. Algorithms, 12(10), 201.

[69] Guerrero, C., Lera, I., & Juiz, C. (2019). "Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures". Future Generation Computer Systems, 97, 131-144.

[70] Khosroabadi, F., Fotouhi-Ghazvini, F., & Fotouhi, H. (2021)." Scatter: Service placement in real-time fog-assisted iot networks". Journal of Sensor and Actuator Networks, 10(2), 26.

[71] Hosseini, S. M., & Arani, M. G. (2015)." Fault-tolerance techniques in cloud storage: a survey". International Journal of Database Theory and Application, 8(4), 183-190.

[72] Wang, M., Wu, J., Li, G., Li, J., Li, Q., & Wang, S. (2017, August). "Toward mobility support for information-centric IoV in smart city using fog computing". In 2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE) (pp. 357-361). IEEE.

[73] Taneja, M., & Davy, A. (2017, May). "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm". In 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (pp. 1222-1228). IEEE..

[74] Xu, Y., Fan, P., & Yuan, L. (2013). A simple and efficient artificial bee colony algorithm. Mathematical Problems in Engineering, 2013.

[75] M. S. U. Islam, A. Kumar, and Y. C. Hu, "Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions," J. Netw. Comput. Appl., vol. 180, no. December 2020, p. 103008, 2021, doi: 10.1016/j.jnca.2021.103008.

[76] A. R. Hameed, S. ul Islam, I. Ahmad, and K. Munir, "Energy- and performance-aware load-balancing in vehicular fog computing," Sustain. Comput. Informatics Syst., vol. 30, p. 100454, 2021, doi: 10.1016/j.suscom.2020.100454.

[77] M. H. Shahid, A. R. Hameed, S. ul Islam, H. A. Khattak, I. U. Din, and J. J. P. C. Rodrigues, "Energy and delay efficient fog computing using caching mechanism," Comput. Commun., vol. 154, pp. 534–541, 2020, doi: 10.1016/j.comcom.2020.03.001.

[78] D. Alsadie, "Resource Management Strategies in Fog Computing Environment-A Comprehensive Review," Int. J. Comput. Sci. Netw. Secur., vol. 22, no. 4, pp. 310–328, 2022.

[79] J. Bisht and V. Subrahmanyam, "Survey on Load Balancing and Scheduling Algorithms in Cloud Integrated Fog Environment," no. February, 2021, doi: 10.4108/eai.27-2-2020.2303123.

[80] F. M. Talaat, S. H. Ali, A. I. Saleh, and H. A. Ali, Effective Load Balancing Strategy (ELBS) for Real-Time Fog Computing Environment Using Fuzzy and Probabilistic Neural Networks, vol. 27, no. 4. Springer US, 2019. doi: 10.1007/s10922-019-09490-3.

[81] A. Chakraborty, M. Kumar, N. Chaurasia, and S. S. Gill, "Journey from cloud of things to fog of things: Survey, new trends, and research directions," Softw. - Pract. Exp., no. November, 2022, doi: 10.1002/spe.3157.

[82] Z. M. Nayeri, T. Ghafarian, and B. Javadi, "Application placement in Fog computing with AI approach: Taxonomy and a state of the art survey," J. Netw. Comput. Appl., vol. 185, no. March, p. 103078, 2021, doi: 10.1016/j.jnca.2021.103078.

[83] E. Badidi, "QoS-Aware Placement of Tasks on a Fog Cluster in an Edge Computing Environment," J. Ubiquitous Syst. Pervasive Networks, vol. 13, no. 1, pp. 11–19, 2020, doi: 10.5383/juspn.13.01.002.

[84] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach," J. Netw. Comput. Appl., vol. 201, no. January, 2022, doi: 10.1016/j.jnca.2022.103333.

[85] D. M. A. da Silva, G. Asaamoning, H. Orrillo, R. C. Sofia, and P. M. Mendes, "An analysis of fog computing data placement algorithms," ACM Int. Conf. Proceeding Ser., pp. 527–534, 2019, doi: 10.1145/3360774.3368201.

[86] P. Maiti, J. Shukla, B. Sahoo, and A. K. Turuk, "QoS-aware fog nodes placement," Proc. 4th IEEE Int. Conf. Recent Adv. Inf. Technol. RAIT 2018, pp. 1–6, 2018, doi: 10.1109/RAIT.2018.8389043.

[87] W. Saeed, Z. Ahmad, A. I. Jehangiri, N. Mohamed, A. I. Umar, and J. Ahmad, "A fault tolerant data management scheme for healthcare internet of things in fog computing," KSII Trans. Internet Inf. Syst., vol. 15, no. 1, pp. 35–57, 2021, doi: 10.3837/TIIS.2021.01.003.

[88] T. Djemai, P. Stolf, T. Monteil, and J. M. Pierson, "Mobility Support for Energy and QoS aware IoT Services Placement in the Fog," 2020 28th Int. Conf. Software, Telecommun. Comput. Networks, SoftCOM 2020, pp. 1–7, 2020, doi: 10.23919/SoftCOM50211.2020.9238236.

[89] F. Hoseiny, S. Azizi, M. Shojafar, and R. Tafazolli, "Joint QoS-Aware and Cost-efficient Task Scheduling for Fog-cloud Resources in a Volunteer Computing System," ACM Trans. Internet Technol., vol. 21, no. 4, 2021, doi: 10.1145/3418501.

[90] H. Abdah, J. P. Barraca, and R. L. Aguiar, "QoS-aware service continuity in the virtualized edge," IEEE Access, vol. 7, pp. 51570–51588, 2019, doi: 10.1109/ACCESS.2019.2907457.

[91] A. Ahmed, S. Abdullah, S. Iftikhar, I. Ahmad, S. Ajmal, and Q. Hussain, "A Novel Blockchain Based Secured and QoS Aware IoT Vehicular Network in Edge Cloud Computing," IEEE Access, vol. 10, no. June, pp. 77707–77722, 2022, doi: 10.1109/ACCESS.2022.3192111.

[92] S. Tuli and G. Casale, Optimizing the Performance of Fog Computing Environments Using AI and Co-Simulation, vol. 1, no. 1. Association for Computing Machinery, 2022. doi: 10.1145/3491204.3527490.

[93] Huang, H., & Zheng, Y. R. (2018). Node localization with AoA assistance in multi-hop underwater sensor networks. Ad Hoc Networks, 78, 32-41.

[94] Kanwar and A. Kumar, ``DV-Hop based localization methods for additionally deployed nodes in wireless sensor network using genetic algorithm " J. Ambient Intell. Humanized Comput., vol. 11, pp. 1-19, Mar. 2020.

List of Publication

1. Sharma, P., & Gupta, P. K. (2021). QoS-aware CR-BM-based hybrid framework to improve the fault tolerance of fog devices. *Journal of Applied Research and Technology*, 19(1), 66-76. https://doi.org/10.22201/icat.24486736e.2021.19.1.1493 **(Scopus) .**

2. Sharma, P., & Gupta, P. K. (2023). Optimization of IoT-Fog Network Path and fault Tolerance in Fog Computing based Environment. *Procedia Computer Science*, 218, 2494-2503. https://doi.org/10.1016/j.procs.2023.01.224 **(Scopus).**

3. Sharma, P., & Gupta, P. K. (2023, August). Novel Particle Bee Optimization based Framework for Fault Tolerance in Fog Environment. In *Journal of Physics: Conference Series* (Vol. 2570, No. 1, p. 012029). IOP Publishing. DOI 10.1088/1742-6596/2570/1/012029. **(Scopus)**

4. Sharma, P., Gupta, P.K. (2021). An Adaptive Service Placement Framework in Fog Computing Environment. *Advances in Computing and Data Sciences. ICACDS 2021. Communications in Computer and Information Science*, vol 1440. Springer, Cham https://doi.org/10.1007/978-3-030-81462-5_64. (**Scopus**).

5. Sharma, P., Gupta, P.K. (2022). An Efficient Mobility Aware Scheduling Algorithm. *Artificial Intelligence and Data Science. ICAIDS 2021. Communications in Computer and Information Science*, vol 1673. Springer, Cham. https://doi.org/10.1007/978-3-031-21385-4_30. (**Scopus**)