DESIGN OF A RESOURCE MANAGEMENT SYSTEM FOR FOG DRIVEN IOT APPLICATION

A Thesis
Submitted in partial fulfillment of the requirements for the

award of the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

By

Heena Wadhwa
41700045

Supervised By
Dr. Rajni

Transforming Education Transforming India

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

2021

DECLARATION

I hereby declare that thesis entitled "Design of a Resource Management System for Fog driven IoT Application" submitted by me for Degree of Doctor of Philosophy in Computer Science and Engineering is the result of my original and independent research work carried out under the guidance of my Supervisor Dr. Rajni, Associate Professor, School of Computer Science and Engineering, Lovely Professional University, Jalandhar. This work has not been submitted for the award of any degree or fellowship of any other University or Institution.

Heena Wadhwa
School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab-144411, India
Date:

CERTIFICATE

This is to certify that the thesis entitled "Design of a Resource Management System for Fog driven IoT Application" submitted by Heena Wadhwa for the award of the degree of Doctor of Philosophy in Computer Science and Engineering, Lovely Professional University, is entirely based on the work carried out by her under my supervision and guidance. The work reported, embodies the original work of the candidate and has not been submitted to any other university or institution for the award of any degree or fellowship, according to the best of my knowledge.

Rajni

Dr. Rajni
Associate Professor
School of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab-144411 India
Date:

ABSTRACT

The group of real world physical devices like machines, vehicles and various “things” connected to the Internet is called as Internet of things (IoT). Currently, with the expansion of the Internet of Things (IoT) devices, the computing requires latency-sensitive support. The cloud computing paradigm offers several services to handle a large amount of data. However, the obstruction of the cloud computing framework is its inadequate flexibility and problem of accommodating the diverse requirements generated from an IoT-based environment. Cloud computing is emerging with the latest paradigms to ensure that the connected heterogeneous system can achieve High Performance Computing. The fog computing paradigm provides solutions to real time applications that require computational support near the edge devices. These latency sensitive applications execute all tasks in the fog-cloud environment which is also known as fog assisted cloud computing. The fog assisted cloud computing provides better Quality of Service (QoS) to the Internet of Things (IoT) applications.

The huge amount of data transmitted by the IoT devices results in bandwidth overhead and increased delay. However, Due to the increase in the Internet of Things (IoT) devices, traditional computing systems are shifting to fog computing. Fog computing has become a growing methodology to handle the network, resources and energy related issues of the traditional cloud computing approach. Fog computing is a viable high performance methodology and deals with resources management, resource provisioning, task scheduling, and load balancing to maintain the system’s high performance. In fog computing, resource management is required to improve the performance of the system. In order to access the resources needed at the right time, the resource management system should find out the available resource and allocation of these resources. Therefore resource management has become a key search area in fog computing and efficient resource utilization is required to ensure the better performance of the system.

Furthermore, many of today’s requirements prefer diverse geographic distribution of resources and near to the end device location. Hence, the new fog computing paradigm provides some innovative solutions for real-time applications. Resource scheduling and utilization are key challenges to handle resource management efficiently. To achieve the set objective, a comprehensive literature review on the existing resource scheduling has been done. A comparative study of existing approaches has been done and find out the gaps has been analyzed.

The fog computing framework's prime agenda is to support latency-sensitive applications by utilizing all available resources. These services include data storage, exploration, and analysis. Due to the increase in the Internet of Things (IoT) devices, traditional computing systems are shifting to fog computing. Resource utilization is a complex task that is often compromised due to the non-availability of required resources on the fog layer. The dynamic nature of fog layer resources depends on the users' requirements for resources. Due to limited resources available at the fog layer, a resource utilization policy is required to ensure efficient resource usability. Therefore, it is significant to allocate the resources and scheduling tasks on the fog layer. Initially, tasks are considered from IoT devices and calculate the requirement for resources. The allocation of resources are done through the Zero Hour policy. This policy represents the allocation of resources from the fog layer to the highest priority task for execution.

Moreover, a technique for resource allocation and management is proposed to ensure resource utilization at the fog layer. This technique is named as TRAM model. In this proposed model, tasks from the IoT devices are clustered based on the priority and deadline by implementing Expectation Maximization (EM) clustering to reduce the computational complexity and bandwidth overhead. These clusters are generated based on the intensity level of these tasks and categorize the tasks into high, medium and low intensity. The availability of resources is calculated and a prepared list of resources(LoR). TRAM effectively minimizes execution time, network consumption, energy consumption and average loop delay of tasks. OSCAR model is designed to schedule the clustered tasks by implementing Heap Based Optimizer (HBO) based on the QoS and Service Level Agreement (SLA) constraints. The QoS parameters are used to check the performance of the proposed algorithm. The allocation of resources performs the distributed resource management to the tasks based on several multiple constraints. The Categorical Deep Q Network (C-DQN), a Deep Reinforcement Learning (DRL) model, is implemented for this purpose. The dynamic nature of tasks from the IoT devices is addressed by performing preemption of tasks using the preference ranking method. The high priority task with a shorter deadline replaces the low priority task having a longer deadline by moving it into the waiting state.

Due to limited resources available at the fog layer, a resource utilization policy is required to ensure efficient resource usability. Therefore, it is significant to allocate the resources and scheduling tasks on the fog layer. The proposed model is experimented with in the iFogsim simulation tool and evaluated in terms of average response time, loss ratio, resource utilization, average makespan time, queuing waiting time, percentage of tasks satisfying the deadline and throughput. The obtained simulation results are compared with other existing techniques. It has been obtained from the result graphs that our proposed approaches outperform the existing approaches in terms of all considered QoS parameters.

ACKNOWLEDGEMENT

It is a pleasure for me to thank all those who have helped me accomplish this PhD thesis. Firstly, I wish to express my deepest gratitude to Dr. Rajni for guiding me throughout this research work. My supervisor has been a continuous source of knowledge, inspiration, motivation and encouragement during the entire course of this research work.

A special thanks to the management of Lovely Professional University to support me in the best possible manner and facilitate me in balancing my work and my research. The doctoral programme of LPU has made it possible for me to pursue my dream of study and upgrade my knowledge.

Special thanks to Examiners of end-term reports and reviewers of journals who vetted my submissions and gave valuable comments to improve the work further.

I take this opportunity to express my gratitude to all my teachers who have shaped me and have contributed immensely to my knowledge and skill development since childhood.

I wish to express my profound gratitude to my parents and all other members of my family. Their love, support and unshakable faith in me provide strength to succeed in all the goals of life.

I am also thankful to my husband, Ashish Wadhwa, who has offered full support to me during the entire period of my research work. I would like to thank all my colleagues Dr. Amanpreet Kaur for supporting and guiding me. Furthermore, I would thank my friends Mandeep Kaur and Kanika Jindal for their endless support. Finally, I would like to thank every person who has, directly and indirectly, helped and motivated me in this arduous task.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IoT	Internet of Things
EM	Expectation Maximization
HPO	Heap Based Optimization
LoR	List of Resources
HPC	High Performance Computing
FC	Fog Computing
QoS	Quality of Service
QoE	Quality of Experience
EST	Earliest Start Time
EFT	Earliest Finish Time
DLS	Dynamic Level Scheduling
BaTS	Bag of Tasks
CS	Consensus sensing
MPSO	Modified Particle Swarm Optimization
DRL	Deep Reinforcement Learning
FCFS	First Come First Serve
FSM	Fog Service Manager
MEC	Multi Access Edge Computing
MCC	Mobile Cloud Computing Architecture
VM	Virtual Machine
RNC	Radio Network Controller
MACC	Mobile ad-hoc Cloud Computing
TRAM	Technique for Resource Allocation and Management
OSCAR	Optimized Task Scheduling and Preemption
SLA	Service Level Agreement
C-DQN	Categorical Deep Q Network

CHAPTER 1

Introduction

Fog computing is to enhance the proficiency of processes and diminish the transportation of data from cloud to endpoints. There are numerous constraints in the traditional methodologies which show the need for a new computing paradigm for all real-time processing applications and reduce bandwidth consumption. Fog computing provided computational, data processing and storage services to the IoT applications or users.
The Fog computing enhance its services to cloud computing by local processing the data instead of adding extra burden on the cloud by transferring all data and information to the cloud. In this manner, fog computing enhances the performance of the system and enhances its efficiency. Although numerous constraints show the need for a new computing methodology for all real-time processing applications and reduce the bandwidth consumption. The new inventions and developments for the design of fog computing framework have occurred in the hybrid environment. and achieve high performance computing.
This chapter gives a high level view of the thesis. It discusses the need and basic principles of fog computing. It describes the integration of IoT devices and fog computing. In this chapter, the key issues have been discussed and provide the solutions to the problem that occurs during the implementation of the cloud computing paradigm. It also motivates to propose a resource scheduling and utilization based framework for the fog computing system. It concludes with a consideration of the rest of the thesis' organization as well as its contributions.

1.1 Fog Computing: Overview

Fog computing, a term introduced by CISCO in 2012 [13]. It is a new paradigm, that can be implemented by using various sensors, wearable devices, smart gadgets and vehicles. In this paradigm, the processing of jobs and computational tasks should be performed in distributed manner [69]. Instead of generating a centralized data center, multiple devices are used in the network. The calculation of latency starts from the end user to the cloud, reducing the bandwidth utilization and latency in the network. Fog computing refining data that is produced by the sensor and transferring to the cloud only after refinement. This paradigm offers various facilities such as faster communication, power of execution, monitoring and analyzing IoT services [34]. Information Technology (IT) plays a vital role in everyone's life. Advancement in IT has completely transformed the lifestyle of mankind. Various sectors such as healthcare, agriculture, co-operate banks, entertainment, and many other sectors have been impacted by significant IT resources, i.e. storage, computation power, and network bandwidth. The demand for these IT resources has increased day by day. Due to the growing demand for IT resources, various computing technologies such as utility computing, parallel computing, grid computing and cloud computing have been developed. Among these computing technologies, cloud computing allows users to perform services on-demand basis and pay for these services as per the usage. Cloud Computing has enabled on demand services to the computing resources.
Internet of Things (IoT) is growing in various private and industrial spaces, it requires changes in the existing approach. Cloud computing is considered as the pillar of all IoT services and it executes all IoT services in a centralized cloud. However, Due to a lack of location awareness, excessive latency, and the lack of geo-distributed data centers adjacent to IoT devices, cloud computing is unable to meet the stated requirements. Cloud computing may be the attainable solution to satisfy the requirements of distributed IoT based applications due to the issue of latency [15].
Information systems for IoT applications with a centralized global approach, in which IoT devices rely on remote management systems [15]. But there is a drawback in the model in terms of agility. In many real-time applications such as environmental analytics, ambient assisted living and healthcare related applications, users want quick responses. Even after the mobile internet speed has improved, the latency is quite high in the distant centralized model. To handle this problem, fog computing provides data filtering with computers available at local data centers and end user applications situated at the edge of the network

of IoT system [13]. All the resources used for fog computing are known as fog nodes, which are connected with the edge network via the internet [77]. Some examples are set-up boxes, industrial integrated routers and wireless access points, with virtualization technologies, which permit clients to install software onto them [91].

1.1.1 Integration of IoT and Fog Computing

The integration of IoT and fog computing often depend upon a distributed networking system to collect data from geographically distributed sources, such as sensors and datacentres [77]. Due to the high latency rate, the cloud is not a feasible approach to achieve the requirement of distributed applications. An information system was designed in which IoT devices depend upon the centralized datacentres [85]. This system was constructed for integrated IoT applications. The combination of IoT and fog computing often depend upon a distributed networking system to collect data from geographically distributed sources, such as sensors and datacentres. It delivers more efficient and cost effective IoT services. Data processing tasks can be performed by IoT devices to simplify by reducing the installation cost and integration for complex data processing.

One of the significant features of fog computing is the capability to execute large-scale sensor networks, which is a problem with many IoT devices. Various manufacturers invented a number of IoT devices and sensors and it is a complicated task to select the best possible components. However, each IoT application has different configurations and requirements. The striking feature of IoT devices is the performance in case of change in their workflow composition.

The whole business scenario gets affected by the next smart generation technologies. The IoT can be represented as individually identified smart objects and devices [26]. The IoT provides suitable solutions for a number of applications e.g. waste management system, smart traffic light system, logistic control system, emergency services and industrial control [75]. Smart healthcare devices and wearable sensor are the two most attractive areas for IoT. According to T. Zhang and M. Chiang [33], Many issues can be resolved by fog computing as defined in Table 1.1

Fog computing enlarges cloud computing performance and with more flexibility to the end devices level of the main network. It also shared processing features to create a viable solution that could be expanded.

Table 1.1: IoT Challenges and its Solution With Fog Computing

Sr.No.	IoT Challenges	How Fog can resolve the issues
1	IoT security Challenges	A fog system can 1) perform malware scanning functions and observe the security status of nearby devices. 2) behave as a proxy to update software credentials and detect threats in a timely manner.
2	Latency Constraints	The fog performs different computation tasks, which are the ideal method for time-sensitive data.
3	Network bandwidth constraints	Hierarchical data processing can be enabled by fog computing, allowing the transfer of data from the cloud to IoT devices. Data processing occurred if applications, network and computing resources are available on demand.
4	Uninterrupted Services	fog computing can execute autonomously to confirm uninterrupted services even it has some problem with a network connection.
5	Resource constrained Devices	fog computing can reduce device complexity, life cost and power consumption when some operations cannot be uploaded to the cloud.

1.1.2 Evolution of Fog Computing

Fog computing(FC) is a developing methodology that enhances traditional cloud computing(CC) facilities to the network endpoints to provide low latency via spatial distribution [85] [141]. The devices involved in distributed computing communicate through a message passing interface and support decentralized models of systems where all computational tasks are performed through various network devices. Various novel computation paradigms have emerged in distributed computing. Seven recognizable phases has been represented in Figure 1.1.

The first phase is mainframe computing, which uses batch processing. The mainframe environment was appropriate for the study of the impact of technology integration capability [2]. In the early 1960s, cluster computing was conceptualized. The concept of virtualization

has started in the late 1960s. Grid computing and Utility computing [8] [7] has emerged as the computing paradigm in the 1990s, where computational decisions are made jointly by a group of connected computers to create a grid. Utility computing comes before the cloud computing paradigm.Cloud computing [16] has become popular in the early 2000s. Fog computing comes into the picture in 2012 as represented in Figure 1.1, which includes computation over end devices like mobile phones, sensor boards and control systems.
Fog computing methodology could provide quicker access for end users. Therefore, the edge capacity of an application supported the computing capacity of cloudlets and worked with the cloud to serve a large variety of applications [42]. Cloudlets are small computing capacity nodes at the base station of the users that work along with the cloud and fog to provide facilities to a large scale of applications. All the applications related to fog computing are developing in such a manner that high-performance computing(HPC) achieves in interconnected systems [139].

In these interconnected systems, when devices and users switch from one point of access

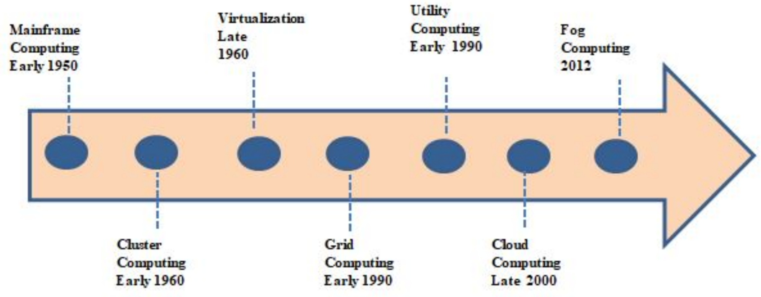

Mainframe Computing Early 1950
Virtualization Late 1960
Utility Computing Early 1990
Fog Computing 2012
Cluster Computing Early 1960
Grid Computing Early 1990
Cloud Computing Late 2000

Figure 1.1: The Evolution of Fog Computing

to another, all the data and processing related to the computer of each user often move [110]. The movement of data can help users to access their data in critical conditions easily. There are numerous Sensitive cases such as healthcare and transport systems where delays can lead to dangerous conditions [76]. For all time centric applications, fog computing methodology provides access to resources within a short span of time. The management of resources to enhance the utilization to achieve maximum performance with less operational cost. Resource utilization is vital for numerous reasons such as resource management, cost and response time. However, it is very challenging to implement fog computing in a real-time scenario. The high volume, data velocity and variety complicate the processing of resources which can affect the utilization of resources [126].

1.1.3 Fog Computing Architecture

A fog computing architecture consists of three layers. In Figure 1.2 a basic overview of the fog computing framework is depicted. The first and topmost layer represents a cloud, the middle layer represents a fog layer and the bottom layer represents edge devices or sensors.

I Bottom Layer: In this framework, on the bottom layer, IoT devices and actuators work to handle user requests. All the installed IoT devices will perform data collection.

II Lower middle layer: The lower middle level handles and manages data where the fog devices perform the bulk of the processing. However, a fog device is a combination of processing elements and application modules. The individual fog device has microdata storage to save a small amount of data in the form of tuples. All the tuples keep the record of every host. This lower level middle level is controlled by the upper-middle level, which works as a fog broker. Fog broker represents a combination of host control node and cloud-fog control middle layer.

III Upper middle layer: In this upper middle layer level, the fog broker manages all fog devices. It is used to manage all scheduling policies and communication between the layers in this framework. The connection type can be wired or wireless, which can affect throughput and speed. In this case, a wireless connection was established between endpoint, servers and fog devices. Resource utilization policy is implemented on the middle layer so that the implementation of the SRT framework can represent through multiple fog nodes and the cloud. This policy can help to resolve several issues related to latency, mobility, scalability and decentralization management [105].

IV Top most layer: All the cloud level services exist on the topmost layer. These services are only accessible through fog devices. This layer is used for heavy storage and processing.

As shown in Figure 1.2, the end-user can only contact the top layer through middle layer fog devices. The topmost layer is used for the permanent storage of data and outcomes for applications. In this framework, fog devices can communicate with each other, but there is no controlling node to keep the status of fog devices. There is a need for a controlling node

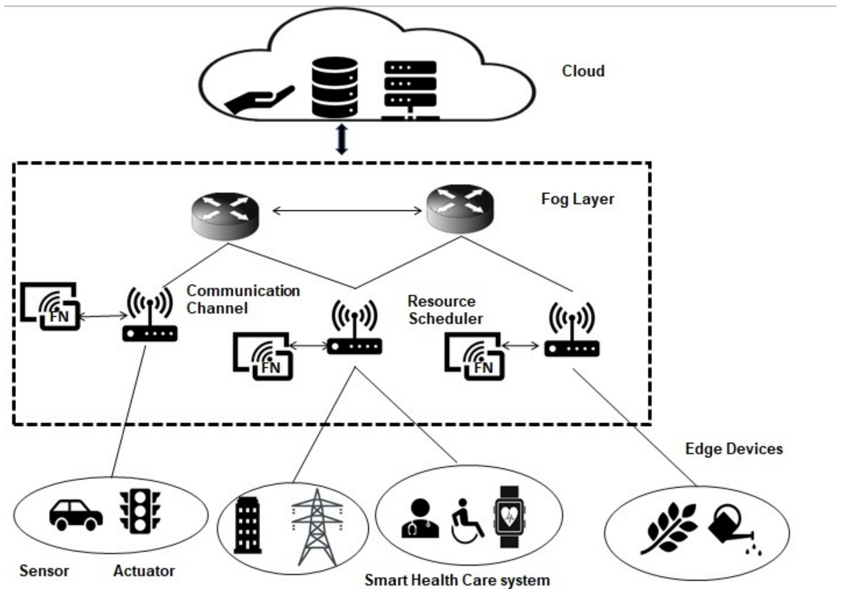


Figure 1.2: The Layered Fog Computing Architecture

in this framework to maintain all scheduling policies and to ensure the proper utilization of all provisioned resources. In the proposed work, the processing of large files can be done within less time span in an emergency. Along with this, computations can be rolled back to the normal working state.

1.2 Need of Fog Computing

Fog computing is a revolutionary technology of cloud computing that has overcome cloud computing's problems. Low latency, high volume of data and location of data centers are the main problem related to cloud computing. The cloud and fog have a correlation, which helps in many areas such as analytics and data management.
Table1.2. represents the comparison between fog computing and cloud computing basis on some parameters. Cloud computing builds up of two layer platform, where the bottom layer consists of edge devices where data collection tasks are performed via sensors. The upper layer performs analytic activities to provide services to the users. Fog computing permits faster communication and data processing as it is nearer to the data source. It reduces the risk of data loss during traveling on the network. When compared to a cloud system, a fog

system will have fewer computational resources such as memory, processor, and storage, but these resources can be extended on demand.

Table 1.2: Comparison Fog Computing and Cloud Computing

Parameters	Cloud computing	Fog computing
Server nodes location	Within the Internet	At the edge of the local network
Server nodes location	Within the Internet	At the edge of the local network
client and server distance	Multiple hops	Single hop
Latency	High	Low
Delay Jitter	High	Very Low
Security	Undefined Within the Inter	Defined
Awareness about location	No	Yes
vulnerability	High Probability	Low Probability
Geographical distribution	Centralized	Distributed
Number of server nodes	Few	Very Large
Real time interactions	Supported	Supported
kind of last mile connectivity	Leased lines	Wireless supported
Mobility	Limited Support	Supported

1.3 Fog Computing Key Issues

Fog computing has three key areas based on its functionality:

- **Resource Management :** IoT systems require resource management policies to assure QoS, reduce energy waste. There is a need for an evaluation environment to explore different resource management and scheduling strategies to stimulate innovation and progress in fog computing that enables real-time analytics. In many cases, it is too costly and does not give a testable and controllable environment for real time

IoT applications. In fog assisted cloud framework, the cloud will work with the collaboration of fog nodes.

- **Data Management :** The raw data generated by sensors is pre-processed on the local fog layer in fog computing. After pre-processing, the cloud receives relevant and efficient data with decreased volume and velocity. This information is used for global processing and storage. As a middleware, fog computing may process raw data gathered from end devices and transfer it to the cloud layer for storage. In conclusion, the fog layer decreases the quantity of computing in the cloud layer by producing meaningful results from raw data and the cloud layer's network usage and the monetary cost of computing.

- **Security:** Fog is usually required to operate in more sensitive situations, where they can provide the best solution to the client's needs and often wherever users want them. The proximity of fog nodes and end-users enables the capacity to handle new security challenges. On behalf of endpoints, fog systems can run security monitoring, threat detection, and threat protection locally. Fog nodes can also help manage and update end users' security credentials, which eliminates the need for all endpoints to communicate with remote cloud for such functions.

1.3.1 Resource Management

Resource management includes resource provisioning, allocation, estimation and scheduling services in fog computing. Resource estimate supports the allocation of computational resources according to rules, allowing for allocating needed resources for computing and the achievement of the desired QoS. The policies of resource estimation were developed in aspects of user characteristics, experienced Quality of Experience, features of service accessing devices [21] [20].

In fog computing, resource allocation should be done so that resource utilization is maximized and computational idle time is reduced. A balanced load and efficient task assignment on various components is verified to achieve the maximum utilization of resources. In fog computing, resource scheduling and resource allocation strategies have been implemented to manage the load on end devices and fog nodes [40].

This can enhance the QoE and handle the overhead on both sides. In the fog-cloud environment, a methodology for job allocation has been suggested that controls latency and power consumption. All the fog resources are heterogeneous and resource constrained,so coordination among various fog resources is very important for these frameworks. The existing fog nodes on the fog layer handle distributively deployed large scale applications. In such cases, it is not easy to attain desired performance without proper coordination of all resources. A directed graph-based resource coordination approach has been developed for fog resource management [25].

The systematic strategy to assigning available resources to required cloud customers virtually is known as resource provisioning. These resources should be allocated to the virtualized cloud environment's applications [13]. The order and timing of resource allocation is a crucial factor in achieving efficient resource allocation. The advantage of resource allocation is that it eliminates the need for the user to increase their hardware and software systems. Aggarwal et al. [32] discussed the issue of over-provisioning and under-provisioning and proposed architecture to remove the problem in fog environment.

The fundamental goal of scheduling is to decide which process to execute in the next go by implementing a set of applicable processes. Scheduling is the best use of processing time and proper allocation of resources to programs. The scheduling is performed to manage makespan, workload, cost, VM utilization, reliability awareness, energy consumption and security awareness [62]. Scheduling techniques can also assist in managing latency, load and duplication processes in fog computing. Scheduling algorithms and load balancing algorithms are used in the area of cloud computing as well as in fog computing. Fog provider can perform collaboration between cloud node and fog node for processing of offloading applications [36]. In the heuristic environment, scheduling remains a big issue in fog computing. The researcher has used task scheduling to align all processors to the scheduler to reduce the makespan of the schedule. Earliest time first (ETF) Dynamic level scheduling (DLS) and Heterogenous algorithm were used by Pham [36]. Task priority is determined with the help of task graphs and processor graphs algorithms. Two parameters Earliest Start Time (EST) and Earlier Finish Time (EFT), provided the methods are compared with nodes information, which nodes are available for new tasks.

1.3.1.1 Scheduling of Resources

The scheduling technique is used to reduce the overall execution time of a job. A scheduling algorithm helps in proper task scheduling and resource utilization. In fog computing technology, scheduling is a new concept, and there are very few researches done in this

area. Scheduling mechanism plays a vital important role in fog computing. Fog computing, similar to cloud computing, uses resource allocation to assign available resources to clients through the Internet. Scheduling is a challenging job in fog due to the availability of resources vary dynamically scheduling policy also helps in the effective and efficient utilization of virtualization machines. Fog computing contains a pool of virtualized computing resources near the user's end for fast computation, storing sensitive information and knowledge over the internet.
An effective scheduling is required to increase the usage of all resources and benefits to the providers of fog. Different scheduling approaches have been used to achieve better performance. These approaches also help to attain reduced latency, more energy efficient, active network, and geo-distribution.

1.3.2 Data Management

Fog computing paradigm permits processing, storage and networking services to transmit data at the cloud and IoT region. In fog computing, the data generated by IoT devices are pre-processed on the fog layer. When the processing is done on the fog layer, it can reduce the volume of the data. The users must give flexibility and dynamically relocate storage, processing, and control functions across various entities. As a result, the fog layer reduces the amount of work required in the cloud layer by obtaining results from raw data and minimizes the amount of work needed in the cloud layer. Fog computing performs three types of communication for the data transfer process.
All the IoT device tasks are consolidated and sent to the fog server responsible for effectively managing tasks. The data transfer in the fog-cloud environment is represented in a multi-fold framework. The framework is represented in Figure 1.3. Each layer of the framework is defined with its functionalities. The top layer is used for bulk data storage and performs all high computational tasks. On this layer, the user application can be customized and schedule resources to achieve better performance. On the middle layer, fog nodes are implemented for real-time data processing and handle all sensitive data resources. Data collection is done through sensors on the bottom layer and sent to the upper layers for processing. a) fog to cloud b) fog to fog c) fog to IoT devices
a) Fog to cloud: The interface between fog and cloud layer is required to enable fog to cloud and cloud to fog association. This association provides many services and functionalities. Some functionalities are like

(i) supervised or managed at fog layer with the ability of cloud computing.
(ii) Transfer of data for processing and comparison
(iii) cloud can help to decide the schedule of fog nodes
(iv) all the services of the cloud can make its availability through the fog to its end users. It is imperative to track which data and services should be transferred to the cloud through the fog layer. The response of fog and cloud towards that information depends upon the consistency and granularity of data.

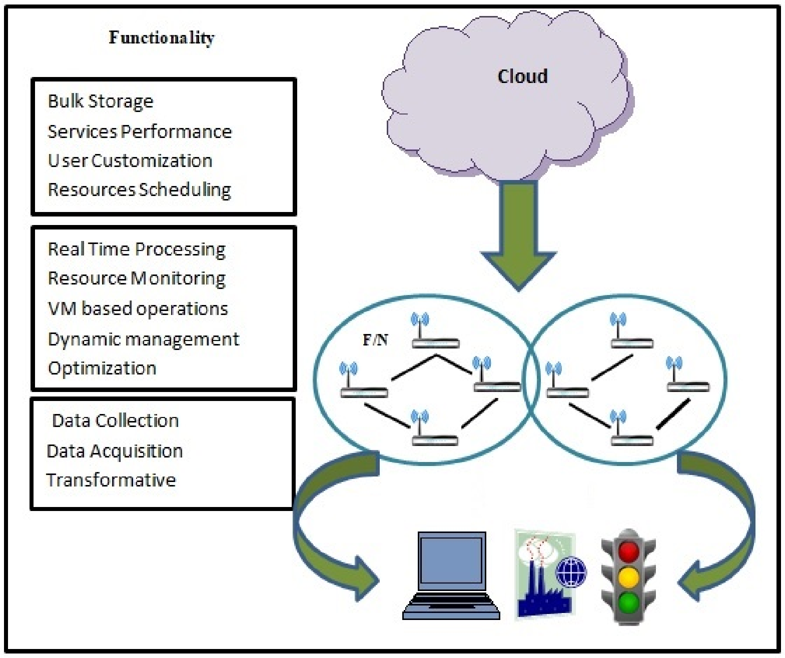


Figure 1.3: Multi-fold Framework

b) Fog to fog: A pool of resources must exist in fog nodes to support each other and to perform its functionalities. All the resources, such as data storage and computational processing, can share by deployed fog nodes. It can prioritize nodes functionality for users' applications. Several fog nodes might also perform their task together with the backup provision of each other.
c) Fog to end user/IoT device: The Internet of Things (IoT) represents an environment that involves objects such as computing machines, smart healthcare devices, vehicles, home appliances etc, connected via heterogeneous networks. Fog computing provides services to IoT devices such as smart devices and sensors. Its services are available to the widely

distributed structure to support smart devices.
In recent years, fog computing is acquiring attention to support time-sensitive applications in smart Internet of Things IoT services, including smart healthcare and smart traffic management. The recent market research reports presented the market rate for fog computing will increase up to $18.2 billion by 2022 [91]. This new paradigm is used in various fields but the topmost five domains will be agriculture, smart healthcare, industry, smart transportation and energy [81].
For the communication between users and fog or IoT interface to fog services, IoT access for fog services is required in a user friendly environment. All the resources can be used efficiently and securely. In the Figure 1.2., explains the interaction of fog with IoT and Cloud both. It is visualized that the top layer represents cloud/fog communication and on the right side of the picture it is pointed out that the communication between the fog layer with IoT devices

1.3.3 Security

In the fog computing paradigm, constructed network become subject to spoofing attacks due to exposing nature of wireless activities and between fog nodes and end users [84]. The protection of end users and fog nodes poses a significant challenge in the deployment of a real-world situation. Security awareness is used to identify harmful behaviours that might compromise the functioning or confidentiality of a real time fog system. To provide the security, Intrusion Detection Systems (IDS) are important system components that intend to prevent networks from malicious activities that may interrupt with the network and influence transmitted data. In the cloud computing, the cloud services managed by the service providers. whereas in the fog computing paradigm various options are available by controlling cellular base station and establish fog environment with the existing infrastructure.

1.3.3.1 Authentication

Trust based security solutions of fog nodes have been focused by industry and academia. There are several trust-based models and resource access control mechanisms used to provide security to the network devices. Internet service providers, cloud service providers and end users can create their own fog infrastructure by expanding their cloud services and convert the local private cloud into fog. These kind of expansion can complicates the authentication and trust situation of fog. Damini et al. [6], proposed a system for resource selection using a distributed polling algorithm for checking the reliability of resources before downloading. There are some special hardware base trusting model to provide trust utility in fog computing applications such as Trusted Execution Environment(TEE), Trusted Platform Module(TPM) and Secure Element(SE).

1.3.3.2 Rogue Fog Device

A rogue fog device that claims to be authentic in order to get end users to connect to it. A fog manager may be permitted to handle fog instances, but may create a rogue fog instance instead of a valid one. Han et al. [12]have introduced a monitoring method for preventing clients from connecting to rogue access points (AP). It is demonstrated that man in the middle attack, the gateway should be replaced or compromised. After that it can manipulate the requests from cloud and tamper the data to secretly launch further attack. This issue is difficult to handle with fog computing for a variety of reasons.

- Different trust management techniques are required in a complicated trust scenario.
- The continuous creation and deletion of virtual machines makes it difficult to maintain a blacklist of rogue nodes.

1.4 Research Motivation

Several challenges require to be handle in the area of resource scheduling and utilization. Resource utilization ensures the proper usability of resources and manages the resources dynamically. The motivation of this study is to find the optimal solution to ensure the effective utilization of resources located in the fog layer. The mentioned solution will also address various issues to improve the capability of fog nodes. The concept of using the resource scheduling within the fog layer motivated the development of a robust framework capable of processing real-time applications. The free accessible hardware resources of the network would be track and utilized to complete the task.
The integration of fog computing and task cluster can explore a novel way to manage resources for implementing real time services in a cloud-fog environment. Effectively managing resources in the fog layer decreases the latency and response time and improves performance. Thus, a framework based resource management technique can be designed

for prioritizing the tasks in a cloud-fog environment. In this environment, the rising trend of resource scheduling is coupled with other parameters such as energy consumption, latency and network consumption which as presented as QoS. Since the usage of IoT devices is growing rapidly, therefore a framework is required that can assign the resources to the tasks generated by IoT devices efficiently.

It also introduces new challenges which have been summarised up as follows:

- Cost: Nowadays, large networks are continuously being buffered with lots of raw data generated through devices that are circulating in different data centres. Therefore, bottlenecks have more susceptible to occur as the same resources are being utilized. These kinds of problems are less frequent to take place in the fog computing since less amount of data is being transferred throughout the centralized network [34]. This may be less expensive in terms of network bandwidth transmission.

- Network Expedience: The ability to process and analyze data closer to its original data source can dramatically reduce network latency [28]. This is a beneficial computational approach for the data execution, which needs low latency to respond to an action. IoT applications play an important role in effective computing in the banking sector, traveling and healthcare services with as low latency as possible.

- Security and Privacy: Security and privacy play an important role in the area of the latest applications. Since the data created by IoT applications are very sensitive and store the personal information of users. The fog computing layer can have security and privacy measures to prevent hackers from collecting sensitive data. It also includes details on information technology security protocols.

- Dynamic Scheduling of task: Scheduling tasks on a set of heterogeneous and dynamically changing resources is a difficult topic that necessitates sophisticated algorithms to deal with the task and resource dynamic behaviour.

  Due to all of these aspects, the goal of this research has been to address resource utilization and scheduling of tasks in the fog computing environment.

  The resource scheduling ensure that the distribution of resources among requested tasks to avoid the delay. If there is non availability of resources at the fog layer, it impacts the real time response and event detection. An efficient resource scheduling framework must identify the availability of resources so that task can be assigned for execution. Resource scheduling is required at the fog layer to achieve resource utilizaation, maintain system performance and avoid overload in the network.

The next section discusses the organization of this thesis

1.5 Thesis Organization

After providing an overview on fog computing and its architecture in chapter1, the rest of the thesis is organized as follows:
Chapter 2: This chapter provides the literature survey on fog computing and analyzes the work done by other researchers on techniques used in scheduling on fog based applications and problems related to tasks. A detailed description of the existing resource scheduling techniques and existing framework have also been discussed. The chapter further discusses the Problem Statement and contributions of the study. Chapter 2 is partially derived from:

- Wadhwa, Heena, and Rajni Aron. "TRAM: Technique for resource allocation and management in fog computing environment." The Journal of Supercomputing 78.1 (2022): 667-690. (Impact Factor-2.6)

  Wadhwa, H., Aron, R. Resource Utilization for IoT Oriented Framework Using Zero Hour Policy. Wireless Personal Communication 122, 2285–2308 (2022). (Impact Factor-1.6)

- Wadhwa, Heena, and Rajni Aron. "Fog computing with the integration of internet of things: architecture, applications and future directions." 2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). IEEE, 2018.

Chapter 3 describes the proposed Resource utilization framework for scheduling on Fog computing and the goals which it tends to achieve. In this chapter, the high level view of framework presents resource management using zero hour policy. The requirements are analyzed on the basis of QoS parameters. The working of framework has been described in the form of a flow diagram that explains the whole procedure of implementation of resource management technique in the fog layer. Moreover, this chapter represents the verification of the resource scheduling framework. The experimental setup, a simulation environment, and simulation tool for resource management present an experimental scenario.

Chapter 3 derives from:

- Wadhwa, H., Aron, R. Resource Utilization for IoT Oriented Framework Using Zero Hour Policy. Wireless Personal Communication (2021). https://doi.org/10.1007/s11277-021-08993-0

- Wadhwa, Heena, and Rajni Aron. "A Clustering-Based Optimization of Resource Utilization in Fog Computing." In Proceedings of International Conference on Advanced Computing Applications, pp. 343-353. Springer, Singapore, 2022.

Chapter 4 presents Resource Scheduling algorithm based on dynamic task preemption scheduling in the fog environment using heap based optimization(HBO). The task scheduling is implemented to utilize the resources of the fog nodes. This chapter provides a clear picture of the ways and methods for resource utilization for high medium and low intensity tasks in the fog computing environment. Finally, at the end of this chapter the Deep Reinforcement Learning (DRL) model is implemented to manage the dynamic nature of tasks. In this chapter, the performance evaluation criteria analyze the results based on test cases. This chapter provides the design and implementation details of the proposed algorithm. Chapter 4 derives from:

- Wadhwa, Heena, and Rajni Aron. "TRAM: Technique for resource allocation and management in fog computing environment." The Journal of Supercomputing (2021): 1-24.

- Wadhwa, H., Aron, R. Resource Utilization for IoT Oriented Framework Using Zero Hour Policy. Wireless Personal Communication (2021). https://doi.org/10.1007/s11277-021-08993-0

Chapter 5: This chapter represents the conclusion and discusses the future scope of the study.

1.6 Thesis Contribution

This Thesis contributes in the following ways:

- It analyzed the detailed literature of existing fog computing resource scheduling algorithms and the implementation of these algorithms in different areas. It provides a qualitative study of existing frameworks and their services.
- All the required QoS parameters have been identified to process resources scheduling and utilization.
- To address the challenges of resource utilization and scheduling in fog computing, a framework has been designed that offers resource management policies by considering the dynamic behaviour of tasks.
- The proposed policies have been designed based on expectation maximization clustering and heap based optimization. These policies address the challenges of resource scheduling in fog computing.
- It provides the qualitative comparative study of resource management approaches with respect to various parameters such as network consumption, latency, energy consumption, throughput and resource utilization.
- The experimental findings represent that the proposed algorithm improves resource utilization in comparison to the existing resource scheduling algorithms. It also shows better results for throughput, execution time and network consumption etc.
- The usability of the proposed policies, their validation have been checked using various parameters such as energy consumption, network usage, throughput, latency and loop delay.
- Statistical study of simulation results has been determined to assess the accuracy of the proposed algorithm.

Followings are the noteworthy contributions of this thesis:

- A Tri-fold task clustering framework is proposed using the concept of clustering. The proposed framework efficiently manages resources for high priority task clusters in the cloud-fog environment.

- In order to handle high priority task, zero hour policy is proposed that used to assign resources to the tasks as per Heap Based Optimizer(HBO).
- For the assignment of resources, the Technique of Resource Allocation and Management(TRAM) is proposed that prepare list of resources(LoR) and schedule the resources to reduce the network consumption and latency.
- For resource allocation, Categorical DQN(C-DQN) is used. It helps the task preemption to ensure high resource utilization.

CHAPTER 2

Related Work

The fog computing paradigm is an amalgamation of traditional technologies to assist massive data generated IoT environments. IT has emerged as an incredible technology that supports innovative applications, high performance in critical scenarios.
The deployment of fog devices at the fog layer involves the proper utilization of resources that are geographically and dynamically distributed in the system. The requirement of IoT is increasing day by day that has also increased the need for information storage, enormous power handling, and fast speed broadband systems that are required to process data stream. These requirements are now completed by fog Computing. Resource management and the efficient utilization is a fundamental task in fog computing. The resource management system offers resource provisioning and scheduling to support resource management decisions. This chapter discusses fog computing systems along with the description of resource management and resource utilization. It also discusses the key background information to facilitate a better understanding of resource scheduling in fog computing. A comparative study and analysis of the existing resource scheduling techniques has been done to understand better resource scheduling and resource utilization in the fog environment. Further, the existing framework and implemented scheduling terminology have been discussed to design the gap analysis.
The last section highlights the problem formulation which is related to the dynamic nature of tasks and resources. It also includes the latency related issues in real time applications. Further, list down all the objectives of the thesis after problem formulation.

2.1 Resource Management System

Fog computing helps to enhance efficiency and speed while minimizing the quantity of data sent to the cloud for processing, analysis, and storage. In fog computing, quality of service (QoS) characterizes a service's overall performance, particularly as experienced by network users. Several network service elements, including throughput, latency, resource availability, and jitter, are frequently evaluated when analyzing the quality of service.

It may take quite a long time to operate because it is stuck in long queues or chooses a less direct path to avoid delays. Excessive latency can sometimes degrade application performance. Due to the requirement for latency-sensitive applications and real-time data processing and routing, fog computing attempts to extend cloud services and utilities to the network edge [63].

In this new paradigm, computation is dynamically dispersed among cloud locations and network components based on Quality of Service (QoS) requirements. Despite numerous studies on cloud technology, the physical distance between cloud resources and end-users prevents the implementation of IoT services with specific requirements. There are a number of QoS parameters, which were introduced by different researchers. In a cloud-centric IoT, the delay caused by communication between end-points contrasts with the desired QoS on real-time services. Healthcare services may require immediate actions when abnormal scenarios occurred. Similarly, autonomous cars must be able to detect changes in the surroundings instantly. Real-time monitoring in manufacturing facilities and real-time navigation in traffic control systems are two more sensitive IoT scenarios. The QoSMonitor calculates network delay in relation to other system nodes and monitors the worker node's QoS features, such as availability and resource consumption [31]. C.-F. Lai et al. [144] proposed NAT based load balancing approach for resources in fog computing. The suggested load balancing system is ICE-based, and it employs a three-layer fog architecture with TURN servers at the network's edge that are monitored by fog node controllers. For analysis, the authors evaluated two parameters: maximum load and controller latency. It has been observed that maximum load has been applied to the nodes and obtained that the proposed method divides cloud load and prevent transfer server load to the users.

Furthermore, the successful deployment of real-time services, which necessitates a very short delay in allocating distributed resources, depends on the evaluation of the influence of regulating decisions on QoS. The control topology can significantly impact control decision latency, which can significantly impact QoS-aware service allocation.

2.2 Resource Utilization

There are Various terms are associated with resource utilization, such as resource provisioning and scheduling. For implementing the proposed work, resource utilization and resource scheduling terms are used for the framework. Resource utilization is a process that ensures the maximum usability of resources to efficiently complete the user requirement. High resource utilization can be achieved with proper resource allocation and split up all available resources among users. Resource utilization directly affects the cost and performance of any system. With the over-provisioning of resources, low utilization of all assigned resources can be achieved and it can also increase the cost of such a system whereas the under-provisioning resources and degrade the performance of system [72].

2.2.1 Resource Utilization in Fog Computing

In this section, a review has been done for existing resource utilization techniques in fog computing. Fog computing methodology has been presented to handle all the issues related to resource allocation for IoT applications.
In [9] researcher proposed BaTS (budget- constrained scheduler for Bag-of-Task applications). BaTS tracks the development of operations and re-configures the machines dynamically as per the requirements. They executed many tests with a performance-price ratio. Every test was performed on two separate clouds with the RR (Round-Robin) algorithm and the second with BaTS.
In [47] represented a cloud-based fog system. In this research, a simulation was establish based on discrete event specifications. The author has not used any specified load balancing algorithm to reduce the use of the cloud.

Yangui et al. [39]developed the PaaS based architecture for the cloud-fog environment. The fire detection application was developed and intimate with a robot to handle the situation. Taneja and Davy [63] introduced an algorithm for resource-aware module mapping. The developed algorithm selects suitable resources in the cloud-fog environment by considering all resource constraints such as network consumption, RAM and CPU. Benamer et al. [70] proposed an application placement algorithm to reduce latency. The implementation of this algorithm is also done in the cloud-fog environment. Two heuristic algorithms [73] [86] proposed for VM live migration and load balancing in the cloud-fog environment. The researchers used the best fit decreasing (BF) and min-min conflict optimizing algorithms.

In these algorithms, better results represent compared to the traditional algorithms.

Researchers in [87] represented an algorithm Energy-Efficient Task Scheduling (MEET) for homogeneous nodes to provide an energy-efficient solution. The total energy consumption was reduced with their selection policy and allocation of offloading time slots. The author in [104] proposed a greedy knapsack scheduling (GKS) algorithm for resource assignment in a fog network. Two case studies simulated the result of their study. The result of their proposed algorithm was better as compare to the FCFS, concurrent and delay-priority algorithms. A network-oriented scheduling methodology was implemented for container-related applications [105]. They have used Kubernetes based fog computing architecture and achieved 70% reduction in network latency. In [65], the author proposed a hybrid method in fog network for service orchestration. Two stages were introduced named South Bound and North Bound. In the South-Bound level, a choreography technique enabled automatic rapid decision making. On the other level, in the North-Bound centralized orchestration is implemented on both cloud and fog levels.

Zeng et al. [40] presented a scheduling algorithm along with image placement in fog computing. Embedded clients and fog nodes can perform all the computing tasks through storage servers. Storage server stored job image, which is shared with clients and fog nodes. The completion time can be minimized by scheduling all jobs. A dynamic resource allocation methodology was presented by Ni et al. [59] based on the completion time of any task and to improve resource utilization. A technique name Priced Timed Petri Nets(PTPNs) was used for the credibility of fog nodes and improved the QoS for users.

Pooranian et al. [61] presented a resource allocation algorithm to optimize energy consumption. It was heuristic based algorithm. The author proposed resource allocation methodology and name it as "bin packing penalty" where fog servers represent bin. All the virtual machines served based on frequency limitations and time. Another policy, named "penalty and reward policy," is used to represented optimize energy consumption.

A two level resource scheduling scheme was proposed by sun et al. [83]. These authors claim that they had achieved a short delay by allocating resources to various fog clusters. Different fog nodes were allocated to the same clusters and performed scheduling for the theory of improved non dominated stored generic algorithm-II. The implemented resource scheduling between fog nodes for multi objective optimization.

An integrated framework was proposed by [120] to achieve awareness about vehicular networks. They have been resorting to the OMNeT++ framework to check the flexibility. In this research, a blockchain based Consensus sensing (CS) application was designed to reconcile local information. Rafique at al. [103] proposed a novel bio-inspired hybrid algorithm for efficient resource management in fog computing. The mentioned work assigned the resources and managed them based on the demand of incoming requests. This work's main objective was to reduce the average response time and optimize resource utilization by scheduling the tasks optimally. The scheduler deployed between the devices and fog nodes was responsible for task scheduling. The task's inefficient scheduling was overcome by accomplishing the integration of Modified Particle Swarm Optimization (MPSO) and Modified Cat Swarm Optimization (MCSO). This approach was validated and the results showed that it has better accuracy in scheduling the tasks.

All the researches mentioned above have proposed a scheduling algorithm and did not address the dynamic behaviour of user requests in the evolving environment of cloud-fog. This has been found from the analysis of the current research that dynamic resource management is mainly being studied in the cloud computing field. Therefore, this study suggests a new resource provisioning and scheduling approach to dynamically managing the application.

A game theory based scheduling approach named fog Match was proposed in [119]. This research work was based on matching the IoT devices tasks to appropriate fog nodes to achieve minimum delay and effective resource management of the respective fog nodes. The matching is based on the preferences of both IoT devices and fog nodes. The mentioned approach introduced both centralized and distributed scheduling based on the requirement. This work was validated and the outcome illustrated that it had better performance in scheduling and resource management. However, the matching of fog nodes and the IoT devices is efficient in resource management, however each task has different deadlines which are not considered and this will increase the purpose of retransmission.

A task scheduling was implemented based on the ant colony system and combination of laxity in cloud-fog environment [109]. The laxity was used to determine task priority and the ant colony was implemented for scheduling. The cloud fog broker deployed between the cloud and fog layer was responsible for allocating tasks based on the task specifications. Initially, the requests from the IoT devices are decomposed into tasks and the computation estimation of the task was carried out to determine the task nature. Then the task was allocated for either fog or cloud. This method was tested and the output proved that it was efficient in task scheduling. The task allocation was performed effectively by the mentioned approach; however, the dynamic nature of the tasks is not considered, leading to a bottleneck problem when the number of tasks is increased.

The multi-level feedback queuing was proposed for task offloading based on deadline and priority in the fog computing framework [89]. Initially, the tasks were classified into three types based on the deadlines: high priority, intermediate, and low priority tasks. The virtual queue concept was implemented to queue the tasks and the tasks were implemented according to the priority. If the low priority tasks were not implemented for a particular period of time, then these tasks' priority will be incremented by one. This method was implemented and the output showed that it was efficient in task classification and scheduling. The mentioned deadline and priority aware task scheduling method was efficient in task classification and scheduling, however the energy efficiency of the process was not considered and it doesn't select the fog node based on energy and resource availability.

The integration of A3C learning and residual recurrent neural networks was deployed in edge-cloud computing environments to perform dynamic scheduling [137]. The tasks from the IoT devices were scheduled dynamically by implementing a resource management system. The RMS decided the scheduling based on the task's CPU, memory, bandwidth, expected completion time and deadline. The RMS consisted of Deep Reinforcement Learning (DRL) model to predict the next scheduling decision and the CSM will check the constraints and provide the possible migration and scheduling decisions. The loss values were used to update the parameters and the R2N2 was used to update the model parameters of the DRL model. The prediction of the next scheduling decision is efficient. It reduces the average response time of the process but the loss function of the upcoming scheduling task reduces the efficiency of the process.

The placement of applications in edge and fog computing environments was proposed in [123]. The weighted cost model for the resource hungry applications was implemented to reduce the response time and energy consumption. The applications are divided into batches for proper scheduling and a new application placement mechanism was proposed based on the Memetic Algorithm. The parallel tasking approach was implemented with the help of a lightweight pre-scheduling algorithm. The mentioned method has been experimented and the output showed that it reduced the time taken in scheduling.

A hybrid meta-heuristic algorithm was proposed to perform energy aware task scheduling in fog computing [124]. The tasks are constructed as a direct acyclic graph to exploit the dependencies of the tasks. The voltage and frequencies of tasks are observed to compute the task with minimal voltage, which will reduce the scheduling approach's energy consumption. The integration of Invasive Weed Optimization (IWO) and Culture evolutionary Algorithm (CA) was implemented to schedule the task optimally. The reduction of energy consumption was carried out by deploying Dynamic Voltage and Frequency Scaling (DVFS). The presented approach was validated and the result proved that it had better efficiency in reducing the energy consumption in task scheduling.

In this study [112], the author proposed the learning classifier system for efficient resource management and workload allocation in the fog-cloud computing paradigm. The power consumption of the edge to cloud nodes was balanced to manage the resources' availability efficiently. Two learning classifier systems were proposed named XCS and BCM-XCS to balance power consumption in the edge node. The edge node's workload, delay, and energy consumption was calculated to design an efficient resource management approach. The reinforcement learning solutions were carried out to allocate the task optimally and the BCM-XCS model was found to outperform the existing models in that area.

An energy efficient task scheduling method for optimal planning of IoT devices was proposed by Zhang et al. [94]. The existing problems on fog planning was overcome by implementing two integer linear programming models. The first model was proposed to reduce the CAPEX and OPEX costs involved in the planning and utilization of resources in the fog nodes. The second model was proposed to reduce the energy consumption of the fog nodes, thereby optimally allocating the tasks. The mentioned method was validated and it was found to be effective in reducing the overheads that emerged in the planning process.

Abbasi et al. [114] proposed mobility aware task scheduling for healthcare applications in a cloud-fog-IoT environment. The patient movements are monitored by the sensors and are transfer to the fog nodes via sink devices. The scheduling and allocation method includes two processes such as ranking process and the scheduling and allocation process. For the ranking process, the author proposed a weighted sum model algorithm. For scheduling and allocation, the author proposed the MobMBAR algorithm. The simulation result shows that the proposed method achieves high performance in terms of makespan and energy consumption. However, this algorithm does not consider a priority, thus reducing task scheduling and allocation efficiency. It increases retransmission.

Zhang et al. [116] proposed a task scheduling process for a fog cloud environment using Non-Dominated Sorting Genetic Algorithm II (NSGA-II). The main objective of this research is to optimize the execution time and cost. The proposed algorithm overcomes the issues of resource allocation and task scheduling. The simulation result shows that the proposed model achieves cost and makespan compared to the existing works. However, the allocation process does not consider energy consumption thus increase high energy consumption during task allocation, which reduces the performance of the proposed model.

Chen et al. [121] proposed a profit optimization method for task scheduling in the cloud environment. The proposed method includes cluster preprocessing, classification, profit matrix construction and optimal scheduling method. The proposed model includes three-layer architecture such as a bottom layer, middle layer and top layer. K means clustering algorithm is used to cluster the tasks and VMs thus reduces the searching time. Here, the k means clustering algorithm is used for task clustering, however the k means clustering algorithm needs to initialize the value of k thus reduces clustering accuracy. Finally, the simulation result shows that the proposed model achieves high performance in terms of reliability and time of task scheduling.

Pang et al. [102] proposed an EDA-GA algorithm for task scheduling in the cloud environment. The proposed model includes three components such as task manager, scheduler and resource manager. The resource manager has the responsibility of VM information and the task allocation is done by the scheduler using the EDA-GA algorithm. The main objective of this research is to reduce the completion time and capacity of load balancing. However, the task scheduling this process does not consider priority level thus increase retransmission and task completion time. Comparison of different scheduling techniques are represented in Table 2.1

Table 2.1: Comparison of Different Scheduling Techniques

Parameter	**Author (Journal name/ Impact factor)**	**Problem undertaken**	**Optimization technique applied**	**Goals achieved**	**Simula tion environ- ment**
Energy consumption and network usage	Rahbari and Nickray[62] (IEEE Conference)	Reduce time delay in allocation of Virtual machines and get fast results	KnapSOS algorithm was implemented with three phase:- Mutualism, Commensalism, Parasitism phase	Reduction in energy consumption by applying changes in original scheduling policy	iFogSim
Latency, Network congestion, cost and energy consumption	Gupta et. al[49] (Software: Practice and Experience) (SCIE-2.02)	To enhance the quality of services by performing resource fragmentation and reduce energy wastage	Power monitor- -ing and resource management services were used by iFogSim packages. Cloud only placement and Edge- ward placement strategies were checked different case studies	Meet QoS criteria and verified scalability	iFogSim

Table 2.1: Comparison of Different Scheduling Techniques

Parameter	**Author (Journal name/ Impact factor)**	**Problem undertaken**	**Optimization technique applied**	**Goals achieved**	**Simulation environment**
Time and cost	Agarwal et. al.[32] (International Journal of Information Engineering and Electronic Business (0.64)	Resource Provisioning in Fog computing	Create Fog data server with Fog server manager and perform Efficient resource allocation algorithm	Minimize overall response time and Cost	Cloud Analyst
Performance and Cost	Pham et. al[36] (IEEE Conference)	Task scheduling	Determine the task priority and choose suitable node to execute each task	Obtain Highest CMT(Cost makespan tradeoff) value	Cloud Analyst
Resource Estimation and management	Aazam and Huh[18][20] (IEEE Conference)	Service oriented resource management model	Resource estimation design with relinquish probability for calculating VRV(Virtual resource value) for new customers	Allocate resources depending upon the type of service and avoid resource wastage	Java and Cloud Sim

Table 2.2: Gap Analysis

Author name	Area	Technology used	Objective	Gaps in Study
Dastjerdi et al. [34]	• Resource allocation • Smart Traffic detection	• Hierarchical network topology used with cloudsim	• Incident detection on basis of congestion calculation and average speed calculation	• In case of Incident detection, no solution provided
Aazam M. and Huh EN [18][20][21]	• Resource utilization • Security measure	• Different scheduling algorithm • Cloudsim toolkit	• To reduce cloud burden • Better service provisioning by data preprocessing and data trimming	• Real time processing of data
Kabirzadeh et al. [52]	• knapsack-based scheduling • Hyper-heuristic scheduling	• Focused on job scheduling, energy consumption	• Reduced Energy consumption and simulation time	• There is no Scheduling approach followed for the IoT applications based on fog networks

continued on next page

Author name	Area	Technology used	Objective	Gaps in Study
Zhou et al. [[111]	• Resource allocation	•Pricing-based stable matching algorithm	• To improve performance • To reduce processing delay	• There is no security mechanism implemented •This technique can combine with machine learning approaches to handle more complicated scenarios.
Farahani et al. [71]	• Survey on IoT devices in healthcare and medicine	• Architecture of Fog node • Multilayer architecture of eHealth -Cloud	• A transition from the clinic-centric treatment to patient-centric healthcare • Design smart glove and smart eyeglasses	• Security of data • Comparison on basis of certain parameter

continued on next page

Author name	Area	Technology used	Objective	Gaps in Study
T. Gia et al. [19][48]	• Health care system	• Medical sensor • IPv6 Low-Power Wireless Personal Area Network (6LowPAN)	• Analyze and monitor real-time health data such as Electrocardiogram (ECG) and Electromyography (EMG) data.	• Early detection system Data filtering and Data compression technique
Skarlat et al. [37]	• Fog Orchestration Control Node	• Fog Colonies and micro data centres are used	• Time shared and space shared provisioning	• Real time testing of proposed methodology is missing
Juan et al. [96]	• Task scheduling	• Two different models, Temporary service model and long term service model were used.	• Energy balancing strategy to control transmission power by using multi- fog architecture	• The implemented work is difficult to implement with large number of user requests.

continued on next page

Author name	Area	Technology used	Objective	Gaps in Study
Rahmani et al. [80]	• Smart E-health Gateway UT-GATE	• Body sensor, Environment sensor used with different communication node (6LowPAN, wi-fi , BLE, Bluetooth)	• Design remote healthcare system Generate EWS(Early warning system)	• No notification send to Doctors and family member.
Goudarzi et al. [123]	• Resource Management	• Container based framework, NSGA2, NSGA3 and OHNSGA algorithm	• Implement custom scheduling policies and low overhead communication	• Security of resources on the fog layer, Performance of available resources
Zhou et al. [143]	• Vehicular fog computing	• Contract theory based vehicular computational resource management and task offloading mechanism	• Enhance resource utilization and boost the Performance of task offloading delay	• Task parameters do not consider in the study such as task size, distance and energy state

2.2.2 Existing Frameworks in Fog Computing

There are implemented frameworks in the fog computing paradigm with the integration of IoT devices and the cloud.
Liu et al. [54] proposed a framework for the reduction of latency of the resource allocation.

Table 2.3: Gap Analysis Based on Existing Evaluation Factors

Author Name	**Latency**	**Reliability**	**Resource Management**	**Security**	**Cost/Energy Consumption**	**Provisioning**
Dastjerdi et al. [34]	✓	✗	✗	✗	✗	✗
Aazam M. and Huh EN [18][20][21]	✓	✗	✓	✗	✓	✓
Kabirzadeh et al. [52]	✗	✗	✗	✗	✓	✗
Zhou et al. [111]	✓	✗	✓	✗	✗	✗
Farahani et al [71]	✓	✗	✓	✗	✗	✗
T. Gia et al. [19][48]	✗	✗	✗	✗	✓	✗
Skarlat et al. [37]	✓	✗	✗	✗	✗	✓
Juan et al. [96]	✓	✗	✗	✗	✓	✗
Rahmani et al. [80]	✗	✓	✗	✓	✓	✗
Goudarzi et al. [123]	✗	✗	✗	✗	✓	✗
Zhou et al. [143]	✗	✗	✓	✗	✗	✗
Yousefpour et al. [110]	✓	✓	✗	✗	✗	✓
Adhikari et al. [89]	✓	✓	✓	✗	✗	✗

In this framework, vehicular Adhoc networks (VANET) application was represented to transfer a large amount of data through communication channels. They have resolved resource allocation and task scheduling problems by MU-MIMO channels where the data is converted into chunks and transfers. They examined an application scenario and carried out resource optimization by identifying the problem and resolving it with a genetic algorithm.

A lightweight framework named FogBus was developed by Tuli et al. [138] for integrating IoT-enabled systems. According to application demands, fog and cloud infrastructure integrate both edge and central resources. The designed framework was implemented a blockchain and authentication process to secure sensitive data. The representation of the working framework was computed with a finger pulse oximeter for Sleep Apnea analysis. FogBus integrally supports dispersed application execution. There were no policies designed for dynamic resource management and migration of applications during execution time.

Rathee at al. [133] defined a reliable technique that uses the tidal trust algorithm to compute the Trust Value and Trust Factor (TV/TF) and successfully identifies authentic FN and IoT devices. The Social Impact Theory Optimizer (SITO) was implemented on the fog layer to calculate trust values in the proposed framework. In their study, they detected the malicious nodes based on predefined values. The framework was tested on various parameters and the NS2 simulator was used to create a virtual fog environment. In their study, the dynamic nature of IoT devices was not considered for the proposed framework.

A framework was designed, which is called foggy by Yigitolglu et al. [67]. This framework manages automated IoT application deployment in fog computing environments. The foggy framework consists of components namely: version control server, container registry, orchestration server, node, and continuous integration tool. The designed framework has not been implemented for real-world IoT applications. Zhang et al. [68] designed the Hierarchical Game Framework to handle resource allocation challenges in fog computing. Apply the Stackelberg sub-game for the interaction between Data service operator(DSO) and Authorized data service subscriber(ADSS), moral hazard modeling for the interaction between DSOs and FNs. This study doesn't evaluate the response time and cost.

Tuli et al. [138] proposed the FogBus framework and develop healthfog architecture. In healthFog, worker nodes contain sophisticated deep learning models to process and analyze the input data and generate results. They used data set of heart patients from the UCI Machine Learning Repository. They have not considered the priority and availability of resources.

Adhikari et al. [89] proposed a deadline and priority-aware task offloading in fog computing framework leveraging multi-level feedback queuing. Initially, the tasks were classified into three types based on the deadlines: highly-priority, intermediate, and low priority tasks. The virtual queue concept was implemented to queue the tasks and the tasks were implemented according to the priority. If the low priority tasks were not implemented for a particular time, then these tasks' priority will be incremented by one. This method was implemented and the output showed that it was efficient in task classification and scheduling.

Lin at al. [127] hybrid deep learning framework to optimize production system. In the proposed framework, visual sensors are embedded to detect the defective product and calculate the degree of deficiency. This method represents the reduction in the workload on the cloud layer. In the Table 2.4, a comparison of various studies and limitations have been discussed.

Table 2.4: Comparisons of Existing Fog Computing Frameworks

Authors	Problem Defined	Limitations
Benamer et al. [70]	This study presented the solution for Latency- Aware Module Placement Problem (LAMP) in a Cloud-Fog environment. To find the placement decision for each module.	The number of used IoT devices are less. It was handling the latency without considering the priority of modules.
Rahbari et al. [104]	A greedy knapsack scheduling (GKS) algorithm was proposed for resource assignment in a fog network. The result of their proposed algorithm was better as compare to the FCFS, concurrent and delay-priority algorithms.	The module list was sorted as per the lowest time order which can be changed during the execution. There was no fault tolerance policy available at the fog layer.
Bonadio et al. [120]	A blockchain based consensus sensing(CS) application was designed for the reconciliation of local information. They have been resorting to the OMNeT++ framework to check the flexibility.	A privacy mechanism is required to protect the user data from tracking at the fog layer. There is no resource management scenario implemented at the fog layer.
Rafique et al. [103]	The resource management and assignment were based on the users request. The scheduler deployed between the devices and fog nodes was responsible for task scheduling. This work's main objective was to reduce the average response time and optimize resource utilization by scheduling the tasks optimally.	The mentioned approach does not consider the dynamic nature of the fog nodes of being varied with the tasks and this will affect the efficiency of the mentioned approach when the number of tasks is increased.
Tuli et al. [138]	The designed framework was implemented a blockchain and authentication process to secure the sensitive data. FogBus integrally supports dispersed application execution.	There were no policies designed for dynamic resource management and migration of applications during execution time.

2.3 Classification of Architecture

It has been discovered that the architecture structure generated by various researchers, as well as the number of phases in it, are dependent on the type of the application, the objective for which it is being created, the required update in the existing one, and the level to which it will be deployed.There is wide variety of architecture are associated with the fog computing paradigm. The combined review of research article falling in different categories is represented in Figure 2.1.

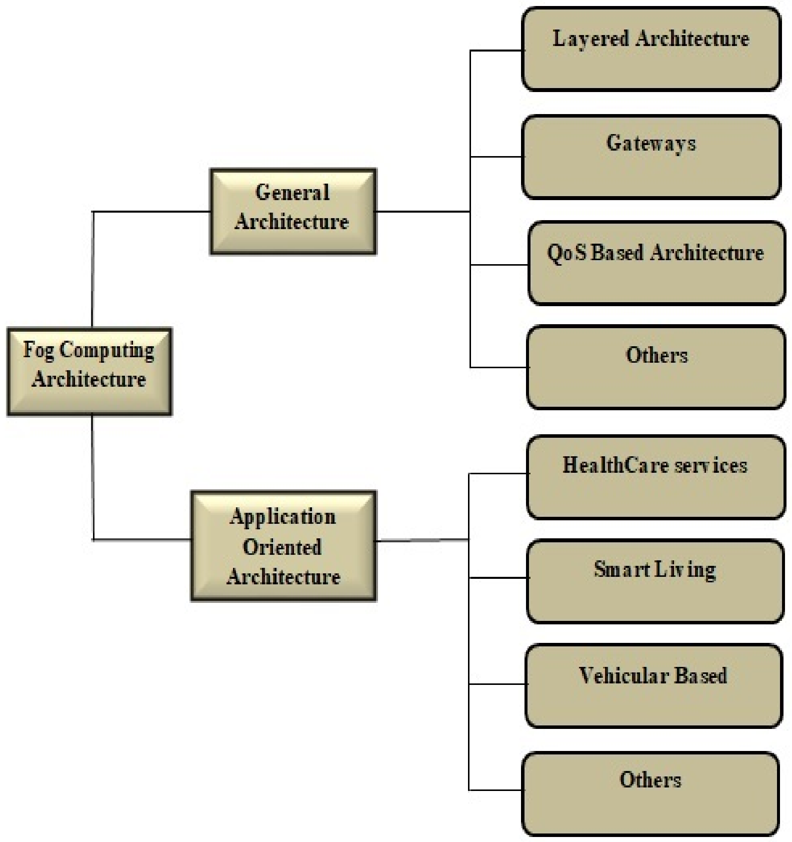


Figure 2.1: Classification of Architecture

It is found that the large quantity of architectures of fog computing is mentioned in the literature. The presented architectures are divided into two categories and it is found that structure of architecture developed by various researchers is depend upon the nature of the application. Two categories are specified as general architecture and application oriented architecture. The details of different categories are mentioned in tabular form 2.5.

2.3.1 General Architecture

Bonomi et al. [13] have discussed basic three layer architecture along with the applications of fog computing whereas Dastjerdi et al. [34] represented the five layer reference architecture to represent that fog computing uses sense-process- actuate process. Sarkar et al. [29] also describe 3 tier architecture to analyzed the suitability of fog in IoT establishments through mathematical modeling. Chiang and Zhang [33] have discussed the basic architecture and represented different opportunities related to fog computing. Puthal et al.[78] have described security attacks and their solutions on each layer of architecture. Mahmud at al. [77] mentioned the challenges about fog environment along with the basic architecture. Munir at al. [58] have proposed an architecture to increase in the performance using QoS parameters such scalability, localization accuracy and efficiency. The proposed architecture named as an IFCIoT architecture to ensure the high performance and presented reconfigurable fog nodes that adjusts to the workload.

Donassolo et al.[92] have proposed a novel approach for orchestration for IoT applications known as FITOR. They also addressed the issue of service provisioning by developing O-FSP, a novel technique for optimising the location of IoT application components. Femtolet architecture was developed by Mukherjee et al. [98],using a 5G device-based fog network to reduce delay and energy usage. The simulation of proposed method was designed on Qualnet 7. Mohammad and Huh [21] have proposed a smart gateway-based architecture for fog computing. Lee et al. [35] have given a conceptual model that is a gateway-based fog computing model for wireless sensor network (WSN's). Gope et al. [93] have consider security as major concerned for fog computing architecture. They have presented device to device service level architecture and introduced authentication protocol for communicating between architecture layers.

2.3.2 Application Oriented Architecture

Tang et al. [30] have proposed a distributed hierarchical Fog computing architecture that combines the huge number of services and infrastructure components to be implemented in smart city setups, in the future. Naranjo et al. [99] have proposed FOCAN. It is an

Table 2.5: Details of Different Categories of Architecture

Architecture class	**Sub categorization**	**Author**
General architecture	Layered architecture	Bonomi et al. [13], Alli et al. [117], Chiang and Zhang[33], Dastjerdi et al. [34], Puthal et al.[78], Sarkar et al. [29] Mahmud et al. [77], Yousefpour et al. [110]
	Gateways	Mohammad and Huh [21] Lee et al. [35]
	QoS based architecture	Munir et al. [58], Alturki et al. [90] Donassolo et al. [92], Maiti et al. [97], Wang et al. [108]
	Others	Byers et al. [44], Gope et al. [93]
Application oriented architecture	Healthcare Services	Rahmani et al. [80], Dubey et al. [46]
	Smart living	Tang et al. [30], Naranjo et al. [99], Tang et al. [106]
	Vehicular based	Datta et al. [23], Chang et al. [45], Ning et al. [101]
	Others	Sun et al. [38], Hao et al. [50], Gazis et al. [24], Byers et al. [44], Tortonesi et al. [107] Luo et al. [96]

architecture that allows application to compute and communicate with each other in a smart city scenario.

Rahmani et al.[80] and Dubey et al. [46] have proposed IoT based system for health care industry. The developed system supported immediate processing, storage, computations ans transmission of data. Soumya et al. [23] have proposed fog computing architecture for vehicles which are connected with M-M gateway. Chang et al. [45] developed fog computing environment using WiFi access point, routers and hubs. They have also discussed its deployment over different platforms such as vehicular Smartphone platforms. Ning et al. [101] have designed traffic management scheme for smart cities. The proposed architecture works collaboratively in the sharing of load of the network.

Sun et al. [38]have created an edge-IoT architecture that can significantly reduce traffic, as well as the volume of traffic and end-to-end delay among IoT devices. Hao et al. [50] have created a flexible architecture that may be deployed based on the needs and demands of the environment in which it is used. They have also talked about the WM-Fog computing framework, which incorporates the software architecture. Gazis et al. [24] have proposed an adaptive operation platform which works as per the operational requirement of the industrial process. Tortonesi et al. [107] have demonstrated the use of custom designed service and information models to aid in the comprehension of various fog services. Luo et al. [96] have presented a multi-cloud to multi-fog architecture based on resource unit containers and their implementation. This architecture's key feature is that it leverages containers as resource units to reduce request response times.

2.4 Fog Computing for Real Time Applications

Fog computing can be utilized in any latency-sensitive application, such as health care, urgent services, and cyber-physical systems. Here are some examples of fog computing applications.

The majority of researchers are interested in fog computing applications such as health care. In recent years, a wide range of studies on health monitoring, detection, diagnosis, and visualization have been offered [22]. Cao et al. proposed FAST, a fog computing enabled distributed analytic system to track the fall for stroke migration by implementing a fall detection algorithm. They have merged the proposed algorithm into a fog-based distributed fall detection system. This method spread the analytical across the network by separating the detection responsibility between the edge devices and the server.

In this paper rahmani et al. [80] prepared a guide tool Early Warning System(EWS) for hospitals. This tool is used for Estimating the degree of Illness and predicting the risk. They have done their research on physiological parameters like respiration rate, oxygen saturation, temperature, systolic blood pressure, and heart rate. They implemented smart E-health gateways to create a reliable, energy-efficient and scalable system using sensor nodes and smart gateways remote server. The IoT based health monitoring system was constructed by using a different integrated module.

Farahani at al. [71] proposed a transition from traditional clinic centric treatment to patient oriented treatment. They have explained mobile clinics, telemedicine, and smart homes to highlight the importance of IoT in the healthcare business. They developed smart eyewear for heart rate monitoring and a fog node based on Qualcomm's Snapdragon 410 quad-core CPU. Wi-Fi, Bluetooth, HDMI, SPI, and other connections are also included on the board. The embedded fog computer (Edison or Raspberry Pi) serves as a data processing platform as well as a gateway to the cloud.

Aazam et al. [20](Aazam and Huh, 2014) developed a layered fog node architecture that allows the processing of local service requests. In fog nodes, smart gateways and a data encryption layer have been implemented. They developed a layered architecture for a smart network and smart gateway. To manage the system, they employed a variety of physical nodes, wireless sensor networks, virtual nodes, and virtual sensor networks.

In follow up papers, Aazam et al. [20] [21] developed an architecture based on a similar concept and working in the fog computing environment. The concept consists of three layers: the cloud, the fog landscape, and IoT devices, each with its own set of control nodes capable of arranging fog cells. The authors used an approach of organizing micro DCs in the smart gateway. Also provided is a thorough theoretical resource management model. Future resource needs are forecasted using different types of access devices, relinquish probability derived from past access data, pricing models, and service types. This resource management approach was distinguished for its ability to adapt to changing fog conditions. In their further work, Aazam et al. presented an enhancement to the theoretical resource management model in terms of specification of usage and QoS in the context of heterogeneous IoT devices. There has been a lot of work put into building IoT-based remote health monitoring systems.

Gia et al. [19] proposed a method for continuous health monitoring based on a customized 6LoWPAN. Through a secure network, the technology allows for remote and real-time ECG monitoring. In the next paper, Nguyen et al. [48] provided an improved real-time and remote health monitoring IoT system with the adoption of an energy-efficient sensor node. The sensor node was designed based on a customized nRF protocol. In this research, the author tried to reduce power consumption.

Smart cities architecture is proposed by [82] They presented an overview of the smart campus project at IISC, Bangalore and designed a data driven IoT architecture. The IoT architecture was divided into two parts. One of the IoT fabric's functions is to control hardware, while the other is to serve as a decision-making platform. Smart power metres and a water level sensor were added to operate the water pumps. The installed system used to check the depth of water. For high level communication Zigbee (IEEE 802.15.4 based) protocol is used. LoRa WAN technology is used for long ranges of a kilometre. One portal is also used to verify entries for several locations.

Tang et al. [64]proposed a four layer architecture system for smart cities to monitor hazardous situations. They have used a fiber optic sensing network to detect physical changes in the environment and identify the threat using a machine learning algorithm. A pipeline prototype is prepared and implemented for event recognization. The designed system sends the processed data to the cloud, and response time is very less in a hazardous situation. This system may be used for the smart city monitoring system. The description of smart cities is

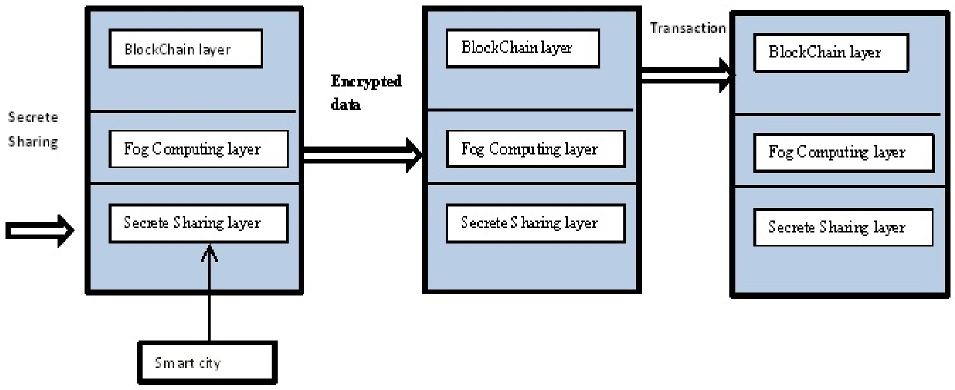


Figure 2.2: Smart City Framework
[118]

dissimilar for different researchers [14]. The smart city application has increased the usage of smart city applications. Hence the security of this collected data and protection from numerous threats is a critical task. Thus data integrity for all smart city applications and protect the data from external attacks during transmission is compulsory for all applicants. Many parameters impact the quality of data, modification of data, and transmission. There are two approaches to evaluated data quality [118]. Smart city framework is represented in Figure 2.2, which is described by [118] in his study.

A proactive approach can protect the data if it is updated within the defined appropriate rate. The second reactive approach is used to maintain the database quality. In this research [118] a framework is designed to support the integration of data that is collected from smart cities. In their research, three technologies have emerged in their framework. However, the proposed technique provides a data integration technique for sensitive data. In Figure 2.2 represents the working of the smart city framework, which represents the collection of sensitive information on the secrete sharing layer, and the second level represents the transfer of encrypted data to the fog layer. After performing these two-layer function data transfer

to the third blockchain layer. Blockchain is a developing technology that has chronological attached blocks, including records that are repeated in each node of an existing network. The network used in this blockchain is a peer to peer network. This technology is mainly used in recording commercial transactions.
Blockchain is an amalgamation two different technologies such as cryptography and a shared database, which permits several parties to have the transactional operational simultaneously through and continuously maintained digital shared records. In this kind of technology, transactions occur in a peer to peer system.

Microsoft HoloLens, Google Glass, and Sony Smart Eyeglass are just a few examples of prominent technologies and projects that may be utilized in real-time applications. To analyze video streaming, real-time applications often require a lot of bandwidth for data transfer and a lot of processing power.They have designed ENORM-A Framework For Edge NOde Resource Management [66].

2.5 Resource Scheduling Algorithm

Scheduling is the best use of processing time and proper allocation of resources to programs. The primary role of the resource scheduler is to choose which process to execute in the next go by implementing a set of applicable processes. In fog computing, resource estimate supports the allocation of computing resources according to certain criteria, ensuring that adequate resources are available for future computation.

Resource estimation policies built in terms of user characteristics, experienced QoE (Quality of Experience), features of service accessing devices [21] [20].In fog computing, resource allocation should be done in such a way that resource usage is maximized and computational idle time is reduced. A balanced load on various components is achieved more precisely. A scheduling-based workload allocation approach is presented to balance the computational load on fog nodes and client devices [40].

As a process, both parties' overhead becomes more reasonable, enhancing QoE.In the fog-cloud interaction, a task allocation paradigm has been developed that balances latency and power usage. Although fog resources are diverse and resource restricted, coordination among them is essential. Due to the decentralized aspect of fog computing, large-scale applications are often distributed among several fog nodes. In such circumstances, achieving the necessary performance will be difficult without effective fog resource coordination. A directed graph-based resource management paradigm has been proposed for fog resource management [25]. The systematic technique to providing available resources to needed cloud customers over the internet is known as resource provisioning. These resources should be assigned to the virtualized cloud environment's applications as efficiently as possible [13].

The arrangement and timing of resource distribution is a crucial factor in achieving efficient resource allocation. The advantage of resource allocation is that it eliminates the requirement for the user to increase their hardware and software systems.

Agarwal et al. [32] discussed the problem of over-provisioning and under-provisioning and proposed architecture to remove the problem in the fog environment. Scheduling is the best use of processing time and proper allocation of resources to programs. The main task of scheduling is to choose which process to execute in the next go by implementing a set of applicable processes. The scheduling objectives include cost, makespan, workload, VM utilization, energy consumption, reliability awareness and security awareness [62].

Scheduling techniques can also assist in managing latency, load and duplication process in fog computing. Scheduling algorithms and load balancing algorithms are used in the area of cloud computing as well as in fog computing. Fog providers can perform collaboration between cloud node and fog node for processing of offloading applications [36].

In the heuristic environment, scheduling still remains a big issue in fog computing. Researchers have used task scheduling to align all processors to the scheduler to reduce the makespan of the schedule. Earliest time first (ETF) Dynamic level scheduling (DLS) and Heterogenous algorithm were used by Pham [36]. Task priority is determined with the help of task graphs and processor graphs algorithms. Two parameters Earliest Start Time (EST) and Earlier Finish Time (EFT) provided the methods are compared with nodes information, which nodes are available for new tasks.

Rahbari [62] implemented scheduling techniques in many areas and represent an improvement in energy consumption, network usage, and cost. Two different case studies were conducted for implementing scheduling algorithms. one case study is about intelligent surveillance through distributed camera networks [52]. The object and motion recognition in raw video streams by the camera. The second case study is an application based designed for elderly human activity detection. Cloudsim simulator with iFogsim packages were used to implement KnapSOS and Knapsack scheduling algorithm. Network usage, cost and energy consumption were calculated and compared with FCFS (First come and first serve algorithm).

Agarwal et al. [32] performed an elastic resource allocation method to achieve minimum response time and maximize resource utilization. In this study, a cloud-fog environment was created to handle all resource related requests. Fog server manager (FSM) was designed to check the availability of the processor. Fog server manager was created as a part of the fog data server.

Oueis et al. [27]designed algorithms for mobile computation offloading. Three algorithms EDC-PC, EDC-LAT, CS-LAT were designed for multi user scenarios. Figure

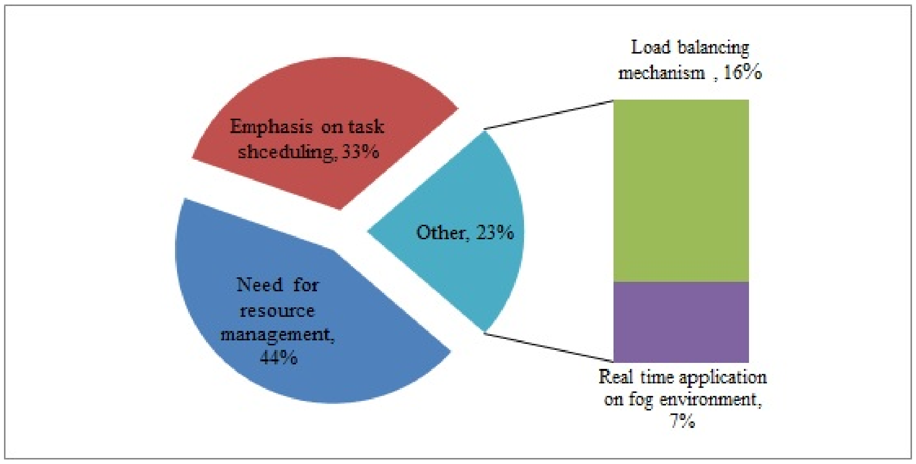


Figure 2.3: Distribution of Primary Studies

2.3, shows that out of total literature survey considered for the proposed problem, 33% of researchers have emphasized on task scheduling and it should given utmost importance in fog computing environment whereas 44% of researchers have point out that there is need of resource management and should be given primary status. The figure also shows the bifurcation of the 23% represents the other area consideration of fog computing, out of which 7% of researchers have addressed real time application on fog computing environment and 16% of the researchers have already given a scheme to ensure load balancing mechanism.

2.6 Fog Computing and its Related Concepts

In the fog computing paradigm, resources at the edge of the network (nearby to end devices) provide many benefits such as low latency and allows provisioning for user applications such as mobile data offloading. In this section, the application provisioning technique at the edge will be discussed. The difference between the related concepts of fog computing is represented in the Table 2.6. The fog computing related concepts are a)Cyber Foraging b)MEC c)Cloudlet d)Edge Computing e)Mobile Cloud computing f) Mobile Edge Computing.

I Cyber Foraging: Cyber foraging [4] was the first concept related to edge computing, but now recent and latest technology has been overcome with the concept of MEC and cloudlet. Balan et al. [5] redefined the concept of cyber foraging. It allocates a big computational job from one resource-limited device to one or more resourceful servers and returns its final results to the preliminary devices. In this method, the capabilities of nearby servers are exploited for resource limited mobile devices and connected to the internet through high-bandwidth networks.

All these servers are known as substitutes and perform computing. All the substitute servers help process a request such as a face recognition and access extensive volume data for the matching process. The substitute may match the database on its local disk and method on behalf of the mobile device.

II MEC (Multi-Access Edge Computing) is started in the year 2014. It is also known as mobile edge computing (MEC). This technology is focused on mobile networks and worked based on virtualization technology [41]. However, the scope of this concept has been expanded in march 2017. All the computational functionalities of cloud computing are providing at the edge of the network to achieve better QoS, and these types of technologies allow computation at the edge. MEC is also known as mobile cloud computing architecture (MCC) [11].In MCC, data storage and processing are performed outside mobile devices. The working of mobile edge computing is similar to fog networking, but it is mainly concentrated towards devices, whereas fog computing is focusing on the infrastructure [33].

III Cloudlets: Cloudlets use attests cloud computing methodology such as virtual machine (VM) based upon virtualization. There are many resources or clusters of servers located nearer to the mobile devices. They can execute on single or multiple VMs, due to which mobile devices can perform offloading for exclusive computing. When cloudlets are used with the face detection and matching process, it will perform all matching processes on VMs. Cloudlets can shrink and expand dynamically. Cloudlet can also behave like a full cloud at the edge and can work in an isolated form. Cloudlets can survive in a standalone environment.

IV Edge Computing: Edge computing paradigm enhances cloud services and extends older technologies like peer-to-peer networking, distributed data, and remote cloud services. In this technology, computation and storage are performed on the edge

Table 2.6: Attributes of Fog Computing Related Paradigms

Attributes	CC	FC	EC	MCC	MEC	MACC
Real time response	Relatively slow	Fast	Fast	Slow	Average	Moderated
Data storage capacity	Large scale data centres	Only sensitive data	Less amount of data	Mobile devices, Large amount of data	Small scale database with devices	Mobile devices, Small amount of data
Virtualization support	Highly support	Devices with virtualization	No support	No support	Devices with virtualization	No support
Location	Very Far	Comparatively close	close	Far	close	Close
Architectural Design	Centralized / Hierarchical	Decentralized/ Hierarchical	Localized/ distributed	Distributed mobile devices with centralized cloud	Localized / Hierarchical	Distributed
Service provider	Cloud service provider	Cloud service provider and users	Local Businesses and network infrastructure providers	Mobile users and cloud computing	Radio Access network-based network infrastructure provider	Self-organized
Type of service	Global	Less global	Local	local	less global	local
Security	Undefined, provided along cloud services	Implemented on participant nodes	Provided on edge devices	Provided on mobile devices and cloud services	Implemented on equipment of edge network	Implemented on mobile devices
Applications	virtualized applications, distributed computing for large data sets	Real time applications, IoT, smart healthcare, smart cities and vehicle	Traffic control, video surveillance locally, video caching	Data processing collected by different sensors, social networking	Real health monitoring, Content Delivery,Video analytic	Computing for tragedy relief group, live video and Networking

devices or very close to the data sources Hu et al.[51]. The edge devices perform several tasks such as data processing, device management and decision making to decrease the traffic and network bandwidth between the cloud and end devices. The processor is used to generate edge computing services with less power consumption and provide better security. Edge computing reduces the problem of the bottleneck. They can perform intercommunication due to the interconnection established in the local network.

V Mobile edge computing: Mobile edge computing (MEC) mainly focuses on mobile computing and virtual machines. The authors introduce the edge computing layer in the fog layer, which contains many distributed telemetry stations [57]. In MEC, the main focus is on mobile networks and virtualization. As per the previous scope of extension aimed at providing cloud computing capabilities at the network's end point, the Radio Access Network (RAN). These abilities are provided by mobile edge computing servers which can be deployed at LTE macro base stations (eNodeB) sites, multi -Radio Access technology and 3G Radio Network Controller (RNC) sites.

VI Mobile cloud computing: Mobile cloud computing (MCC) is defined as a structure where both the processing and storage of data occur separately from the mobile device. Due to MCC, the mobile application user can use it on smartphones. NIST extends this definition to include mobile devices: cloud computing is the synergy between IoT devices, mobile devices, and cloud computing that enables data-intensive and CPU-intensive applications for IoT environments (NIST).

In MCC, mobile applications can be handled through adaptive offloading to be separated at execution time. MCC decides to run the selected computation-intensive application, and it increases the battery life of a mobile device. All the cloud-based services are highly available in MCC rather than mobile computing. MCC always depends on cloud services for effectively operating high computation services. MCC also has the same restrictions as mobile computing and cloud computing. First, a sharing pool of all computation resources may not be suitable for all applications as the generality of the device is desired. Second, both MCC and cloud computing required internet access for all applications and facilities through the network core by WAN connection. MCC is having some issues related to connectivity challenges.

VII Mobile ad-hoc cloud computing (MACC): It is a decentralized network build-up by nodes to generate dynamic and temporary networks through transport protocols. For networking, storage, and computing, clouds can be formed with ad hoc mobile devices. Mobile ad hoc cloud computing can be used in group live video streaming and an unnamed vehicular system. It is different from MCC and CC. MACC only needs mobile devices, whereas MCC needs an extensive database used for cloud computing. MACC used the technology of Bluetooth, Wi-Fi, and cellular protocol for communication. In Table 2.6, some attributes are discussed by [110] which are related to fog computing and its paradigms

2.7 Problem Formulation

Resource management is one of the crucial problems in the fog computing environment. Task scheduling is an important process in managing resources efficiently. However, the existing works focus on scheduling the tasks, the optimal schedulability of the tasks is not addressed.

The optimization techniques were utilized to perform proper scheduling of tasks [113][141]. The barriers involved in task scheduling of fog computing were eliminated by implementing the Modified Marine Predators Algorithm [113]. The ranking approach was utilized to determine the number of consecutive iterations required to achieve a better position than the current one. An improved firework algorithm for optimal scheduling of tasks in the fog computing environment was proposed by Wang et al. [141]. The tasks were clustered based on classification types and the available resources on the fog layer were also categorized according to the classification of tasks. The major problems encountered in these approaches are,

- The above work implemented marine predators algorithm for energy efficient task scheduling. However, the problem between delay and load of the task is not considered, leading to wastage of resources in the fog node.
- The marine predator algorithm proposed in the paper scheduled the tasks without considering the availability of resources in the fog node this will affect the performance of this approach.
- This approach for task scheduling using an improved firework algorithm has many constraints and one among that is the tasks are not allowed to be preempted. This will reduce the efficiency of this approach.

- The dynamic nature of the fog node of being varied with the tasks is not addressed and this will affect the performance of the mentioned task scheduling approach in IoT based fog computing.

The management of resources for the purpose of execution of tasks was carried out by Rafique et al. and Shardoo et al. [103] and [145]. The inefficient scheduling of tasks was overcome by integrating Modified Particle Swarm Optimization (MPSO) and Modified Cat Swarm Optimization (MCSO). This work assigned the resources and managed them based on the demand of incoming requests [103]. The three methods, such as Self Organizing Map (SOM) and auto encoder, were utilized for resource management [145]. The concept of "earliest deadline first" was implemented for scheduling the tasks. The problems of these approaches are as follows

- The bio-inspired hybrid algorithm does not satisfy the QoS and SLA constraints for proper resource management and efficient task scheduling, which means the system's performance is ineffective.
- The Modified Particle Swarm Optimization (MPSO) implemented in this approach is prune to converge prematurely, particularly during scattering, which will reduce the efficiency of the approach.
- The mentioned approach of two phase scheduling efficiently scheduling the task but the random allocation of task results in increased overload and retransmission, increasing the average response time.
- The two phase scheduling directs the task to the nearest fog. This will affect the efficiency of computation because the energy availability factor of the fog node is not considered.
- Since all the tasks are scheduled in the fog layer, the missing rate of the tasks will lead to retransmission of the task. This will further increase the response time involved in the scheduling of tasks.
- While sustaining the workload in a fog-based cloud system, it becomes necessary to keep the energy consumption of computing nodes at the fog layer.
- An effective framework is required to decrease the energy consumption of fog-based cloud servers, which will aid in the optimal usage of fog nodes.

2.7.1 Research Objectives

The main objectives of the proposed work are:

- To design a framework for resource utilization in fog computing.
- To prepare a novel scheduling algorithm to minimize response time, latency and total execution time.
- To test the proposed algorithm in the fog computing environment.

This chapter focused on exploring the existing resource scheduling techniques in the fog computing environment. It analyzed the existing frameworks which are implemented on fog computing methodology. It examined the existing resource management strategies and their implementations on the fog layer. The next chapter proposes a resource scheduling framework to address the issues identified in problem formulation and to accomplish the objectives of this research work.

CHAPTER 3

Proposed Resource Utilization Framework

The research work done in the domain of fog resource management and scheduling was discussed in the previous chapter. According to the study, fog computing resource management and task scheduling challenges have not been adequately addressed. In real-time applications, tasks require immediate resources to complete their processing in time. There is a need for a proper resource management system, and resource scheduling is a must in real-time applications. Resource utilization and scheduling framework have been developed to address resource management and scheduling issues proposed and designed in this chapter.

In the fog environment, the proposed framework provides resource utilization strategies based on QoS parameters and a scheduling algorithm. A four level framework has been proposed in which end users are connected to the upper layer that is a communication channel to transfer data to the layers above it. Cloud is at the top to provide centralized control and other services. The next section describes tri-fold task clustering model that categorizes the tasks according to the nature of tasks.

After the proposed framework description, Zero Hour policy is explained which described the allocation of resource. The fog broker manages all the resources and allocates resources to the tasks from fog layer or cloud level. The cluster of tasks are created and handled with an expectation maximization algorithm.This chapter analyzes the proposed framework and validates the results.

The experimental results have been discussed which are obtained after implementing Zero Hour policy on the individual task. The simulation shows the performance of resources in the fog computing system. There is a fog broker that monitors the resource allocation to the tasks and their utilization. The TRAM performance evaluation is measured through the processing cost, processing time and delay.

3.1 High Level View of Framework

The proposed resource utilization framework analyzes users' requirements and describes processes that contribute to resource allocation. It represents resource utilization in a heterogeneous environment and the communication between these devices in a hierarchical manner. The designed framework can assist user's requests and assign resources for better services.

The high level view of the existing fog computing framework is shown in Figure 3.1. In this scenario, the hybrid approach of the cloud-fog paradigm is used where nodes on the bottom layers can be physically implemented devices and sensors for data collection.

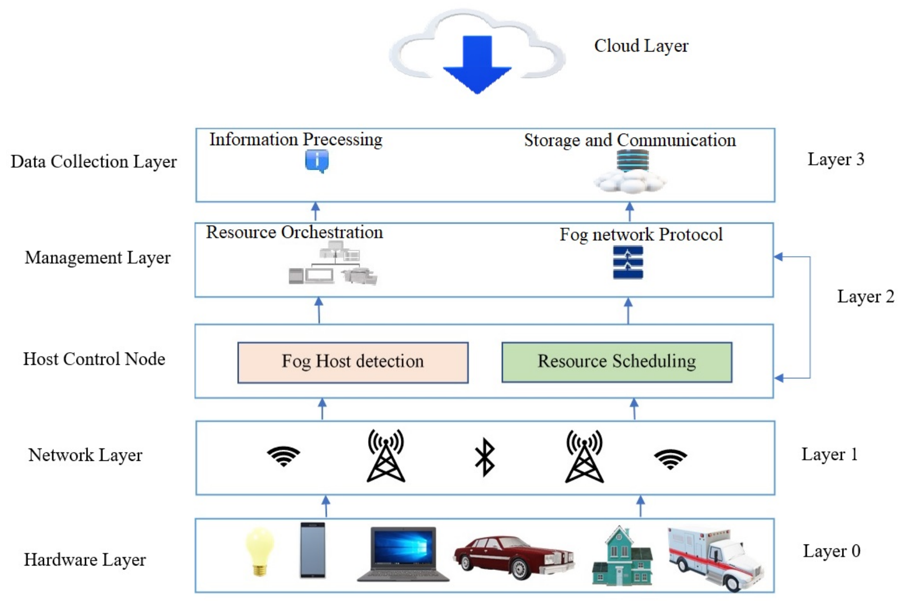


Figure 3.1: High Level View of Tri-fold Task clustering

The designed system simulates fog computing with numerous resources and IoT devices. When IoT devices request or send data back, many network resources would be consumed [119]. A fog network consists of different edge nodes with limited computing capability – these are often termed fog nodes. These fog nodes have storage as well as some limited computation facility.

A fog network consists of heterogeneous edge nodes which are handled by a fog server. Each edge node has its computing capabilities. A fog server selects one fog node for tasks that exist in the system. Each task will be assigned to the fog nodes for resource allocation.

As there are distributed systems, there are multiple users who are using them. During the application execution, tasks dynamically arrived and requested resources.

3.1.1 System Description

High level view of the system is represented in Figure 3.1. This view represents four levels of the framework, where the bottom layer is represented as layer 0. This layer transfers the user's request to layer 1, which works as a communication channel to transfer the data from edge devices to fog devices. After transfer data from layer 1 to layer 2, all the application requests manage and validate by layer 2.

All the fog resources that are the main component of fog computing exist on the host control node of layer 2. Each device has limited bandwidth, storage, processing capacity and memory. On the management layer, the decentralized decision system will be implemented and a number of fog servers are installed to provide resource orchestration services to all the resources. There are fog servers that handle fog devices and deal with several queues simultaneously.

In this four-level architecture, the topmost layer is the cloud layer works as a centralized

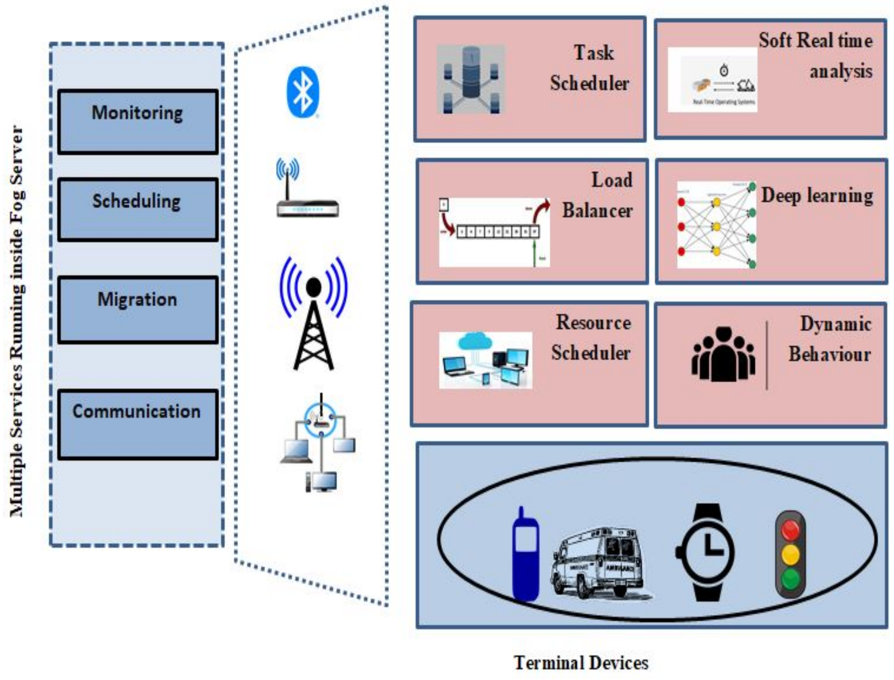


Figure 3.2: Application Workflow and Services Handle by Fog Server

system. After processing the data on the management layer, it sends data to the cloud layer.

3.1.2 Assumptions of the Study

This study will assume that the fog nodes in the network can transmit their locations via GPS or GIS. Moreover, the fog layer comprises intelligent devices capable of routing and transferring the data packets to the upper layer for computing and storing. The networking devices available at the fog layer can share resources among themselves. All the fog computing devices are capable of providing optimal support for terminal nodes. The more specific assumptions are:

- *Physical Security:* Service providers and users both have access to fog devices. These devices can be monitored, controlled and validated by the organizations through known best practices. Physical security is crucial to ensuring high-level hardware and software security for the components of fog-based region infrastructure.
- *Fog device integrity:* Clustering of tasks and resource allocation have been taken to avoid the issues of non availability of fog resources to regions.
- *Defined Policy:* The LoR (List of Resources) depends upon the MIPS, bandwidth and RAM of all virtual machines. The low-intensity tasks with longer deadlines can be preempted so that high priority tasks with shorter deadlines can be executed.

3.1.3 Applications and Services Workflow

Real-time IoT application handles task request processing and assigns fog servers as per their requests. The fog server provides multiple services to the terminal devices. The fog application workflow is represented in Figure 3.2.

As represented in the figure, terminal devices generate massive data that fog servers process that data through available resources. Fog server runs a computational process based on soft real-time analysis and deep learning techniques. It also supports the dynamic behaviour of clients and manages resources through a resource scheduler. All the mentioned techniques are used to provide numerous services such as real-time analysis, monitoring, scheduling, task migration and communication between devices.

3.1.4 Tri-fold Task Clustering

All the IoT device tasks are consolidated and sent to the fog server responsible for effectively managing tasks. The tri-fold task clustering is explained in Figure 3.3. Each layer of the

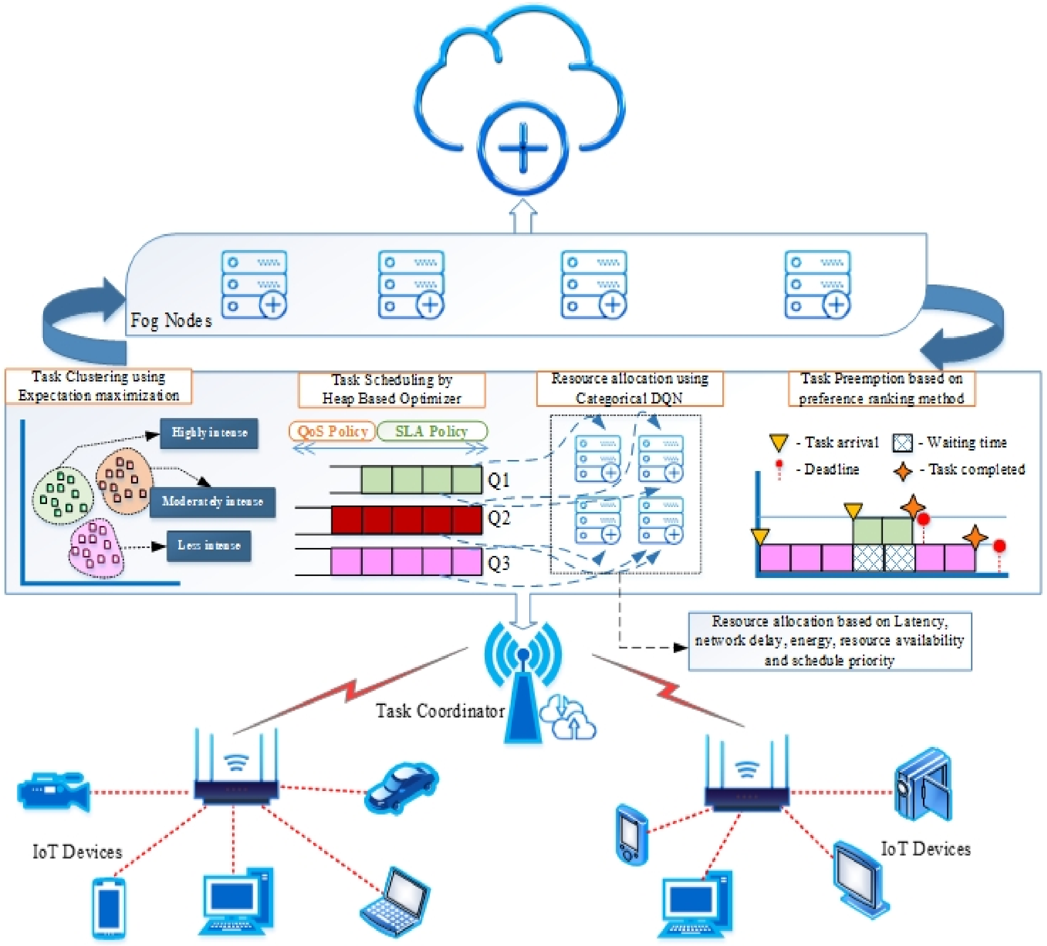

Fog Nodes
Task Clustering using Expectation maximization
Highly intense
Moderately intense
Less intense
Task Scheduling by Heap Based Optimizer
QoS Policy
SLA Policy
Q1
Q2
Q3
Resource allocation using Categorical DQN
Task Preemption based on preference ranking method
- Task arrival
- Waiting time
- Deadline
- Task completed
Resource allocation based on Latency, network delay, energy, resource availability and schedule priority
Task Coordinator
IoT Devices
IoT Devices

Figure 3.3: Overall Architecture of Tri-fold Task Clustering Model

framework is defined with its functionalities. The top layer is used for bulk data storage and performs all high computational tasks. On this layer, the user application can be customized and schedule resources to achieve better performance. On the middle layer, fog nodes are implemented for real-time data processing and handle all sensitive data resources. Data collection is done through sensors on the bottom layer and sent to the upper layers for processing.

The Tri- fold task clustering on fog nodes is represented in Figure 3.3, where four operations are performed by fog nodes. These operations are mentioned in Table3.1. Each operation perform some functionality to accomplished the research objectives. where all

Table 3.1: Goals of the Tri-fold Task Clustering Framework

Sr.No	Goals	Improvement criteria
1	Task clustering (EM clustering algorithm	Reduced Response Time High Throughput
2	Clustered Tasks Scheduling using HBO	Improve resource utilization Queuing waiting time reduction
3	Resource Allocation (C-DQN)	Less average makespan time High priority tasks are improved in resource utilization
4	Task Preemption	High Resource Utilization Optimized Deadline

the tasks received at the fog layer are classified into three categories.

I In the first category, the high intense tasks, the delay-sensitive activities need to be handled immediately.

II In the second category, the moderated intense tasks, which have soft deadlines. All the tasks have medium priority and soft real-time, which can be implemented through a zero-hour policy.

III The third category is about those tasks which are mainly storage based tasks. The task clustering is done by Expectation Maximization (EM) clustering, dividing the tasks into three categories.

- High Intensity
- Medium Intensity
- Low Intensity

3.1.5 Goals of the Proposed Framework

This research work is concentrated on the efficient management of resources of fog computing in the IoT environment. Figure 3.3 depicts the overall working of the proposed framework. Major Goals of the proposed framework along with the improvement criteria are represented in Table 3.1. These goals are described as follows:

- The implementation of this framework enables to categorize the tasks into three categories depending upon the task's nature.
- Providing scheduling policies and ensuring each task's execution in an atomic manner helps reduce the queuing waiting time.
- Providing efficient technique for resource allocation and management to handle high priority tasks. management to handle the task preemption.
- The implementation of framework contributes to achieve high resource utilization.

3.1.6 Framework Constraints

The proposed work aims to classify the tasks and schedule the tasks stochastically based on the resource availability of the fog nodes. The laid down research objectives have been achieved via the proposed resource scheduling framework, where efficient resource allocation and management techniques give the user better resource utilization. A realistic scheduling and resource allocation scheme should satisfy the following constraints as follows,

- Heterogeneity Task Constraints: All types of heterogeneous tasks are allocated and executed for early completion. All tasks are atomic, which means that all tasks must be executed without interruption
- Communication Constraints: All tasks must be executed without any latency and task transmission rate from the source fog node to the destination fog node and two tasks are assigned to fog at the same time.
- Storage Constraints: All tasks are serially executed to reduce the overlapping of storage issues and performed without any interruptions

This is accomplished by the following processes listed below:

Let us assume that there are a number of tasks $T = t_1, t_2 \ldots t_n$ are arrived from N number of users. These tasks require resources $R = r_1, r_2 \ldots r_m$ and task scheduling and resource allocation is accomplished by follows,

- **Task model:**Each task arrived from the users include m numbers and the i^{th} task is represented with a set of the attribute set $A_i = t_{id}, t_l, t_{cp}, t_{bc}, t_{sc}, t_{app}$ where t_{id} is the task identifier,t_l is the task length, t_{cp} is the computing power, t_{bc} is the bandwidth capacity, t_{sc} is the storage capacity and $t_app$is the applications requested in the task.

- **Resource model:** In a fog environment, resources are available in fog nodes and virtualization resources assigned to the particular task. Assume that m amount of resources in the fog node is available which represented by R_j and the attribute set of resource is defined as r_id,r_cp,r_bc,r_sc where r_id represents the resource identifier, r_cp is the resource computing power, r_bc is the resource bandwidth capacity and r_sc is the storage capacity of the resource.

3.2 Zero Hour Policy

Zero Hour policy intends to ensure the maximum usability of allocated resources at the fog layer [147]. This policy will understand user applications' current requirements and segregate the user's requests to ensure all tasks' execution without any delay. It facilitates the delivery of the resource to the highly intense tasks and ensures proper resource utilization. The following are steps to achieve resource utilization in the Zero hour policy

I The system will track all available resources at the fog layer to meet users' requirements. The resource scheduler assigns users' requests to the request manager. The request manager will transfer these requests to the cloud or fog service provider.

II The fog service provider allocates fog resources to the tasks as per allocation policy. The request manager allocates fog resources to the tasks as per the Zero hour resource allocation policy.

III In the third step, after completing user tasks, the free resources will add to the pool of resources and be ready for reallocation.

3.2.1 Mode of Operation

In the proposed framework, the upper layer contains processing units and large storage capacity. In the Figure 3.4, the upper layer communicate with the fog devices with the help of cloud-fog control middle layer. The host control node maintains information about resources, and updates resource status dynamically. The middle layer of the framework is responsible for all the computational and ensures resource utilization of various resources.

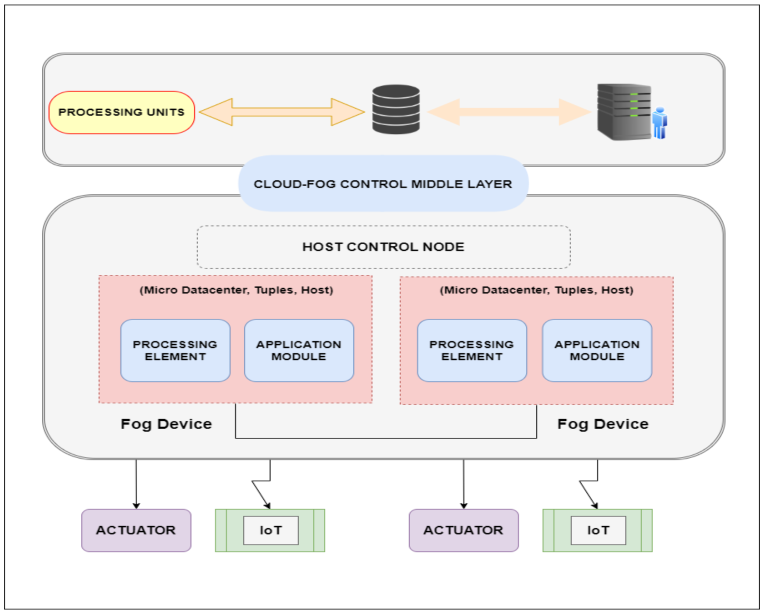


Figure 3.4: Fog Computing Framework

This layer contains processing element and application modules which behave as a small micro data centers. Data filtration, data analysis, and event processing are some of the data operations available in the main application module. Furthermore, the application module's execution can be completed with a notification and storage operation based on the results of the overall data activities. This layer is responsible for the overall control of all resources and manages the runtime environment. Furthermore, it can also synchronize all the resources based on the desired need of applications. This approach is used to achieve proper resource utilization for all the implemented fog devices.

In the Figure 3.5, host control node is a combination of resource scheduler and fog service provider. The users' requests communicate with the resource scheduler to ensure maximum resource efficiency. The resource scheduler received a set of requests from different devices and the resource manager manages all the received requests. A cloud is located at a distance from the devices and offers many complex services that are not time-bound. The terminal devices can access the fog node through different standards such as Bluetooth, Z-Wave, ZigBee and 6LoWPAN [53].

Each fog node performs a chain of operations, including monitoring, analyzing and communicating with other fog nodes. Once the fog node receives an event or request for the resource from edge devices, the data for that specific event will store on the fog device.

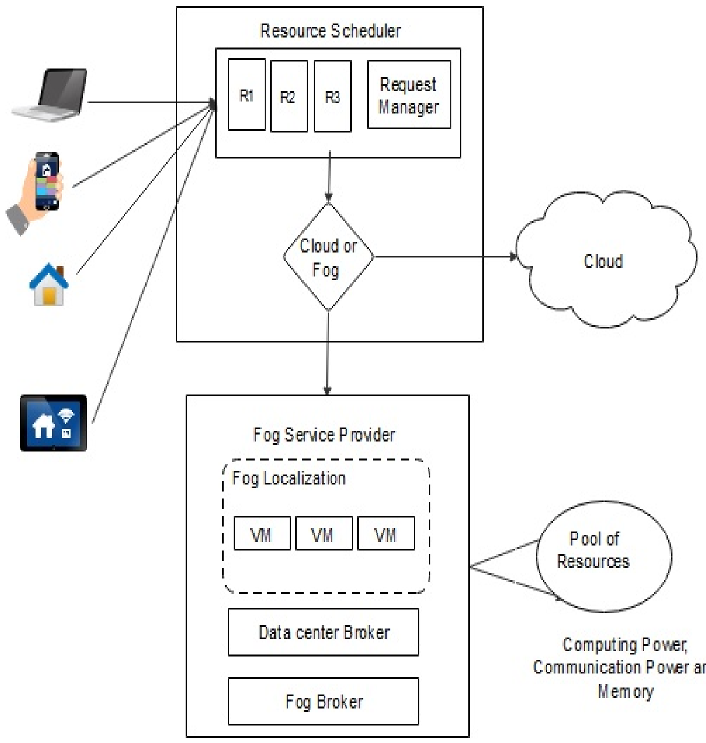


Figure 3.5: Host Control Node with Resource Scheduler

For the processing of resource related requests, fog nodes have two components: resource scheduler and resource manager.

The request manager checks the priority and requirements of requests. Resource scheduler tasks to allocate to the fog node or the cloud. All the directed requests towards the fog node must be accomplished within the specified deadline. For the completion of the designated task, the fog node must have enough resources to allocate by the resource manager. The task is considered as the smallest unit and cannot create a subtask.

All the fog devices are managed by a fog broker unit which has higher configurations as compared to other fog devices. The significant role of a fog broker is to monitor different resources units and datacentre broker and failure handling capacity. Resource units create a pool of resources that can complete real-time tasks. Each unit of resource pool has its own computing capacity, memory and communication power.The resource scheduler will allocate resources for that request and send information about that event to the cloud.

All the fog device, create one fog localization area. The fog localization area will perform the processing of requests after the appropriate resource allocation. Therefore, this fog localization area must have virtualization to run fog applications. During the execution, if a high priority task enters into the system then it will consider as a zeroth task for that system and woks as per zero-hour policy. As per this policy, it will consider the priority of users' requests and assign the resources as per the current need of applications.

Algorithm 1 Zero-Hour Policy

```
Declare: $r \leftarrow Route$ , $FD \leftarrow FogDevice$ , $PT \leftarrow ProcessingTime$ ,$P \leftarrow Priority$, C
for Method of Resource allocation $\boldsymbol{M_r}$ for all Fog nodes $\boldsymbol{N}$ do
    while $N$,r $\leftarrow \varepsilon \neq 0$ do
        $allocatedmodules \leftarrow r$
        $readymodules \leftarrow 0$
    end while
end for
if $P > priorsetlimit$ then
        $readymodule \leftarrow r$ AND $Q = readymodule$
else
        processing of r with set m
end if
for modules Q $\Longleftarrow readymodules$ do
if $req(PT_Q) \geq avail(PT_r)$ and $value(N_Q) \geq value(N_r)$ then
    set New (PT$_Q$) and $reduce(PT_r)$    else
        place Q in the list of allocatedmodules
    end if
    C $\longrightarrow set_time$
```

3.2.2 Design and Working Principle of Zero-Hour Policy

This resource utilization policy has been made based on users' requirements and considers this information as input. The user provides the deadline for the submitted tasks. The request manager calculates a list of required resources and the task's priority to complete the tasks. The resource manager checks the list of the available resources that can satisfy the need of users. In the proposed Zero-hour policy, resources are assigned based on the user's request. The specified time for the beginning of an operation or activity, particularly when a high priority task enters into the system it suspends all current activity, is known as zero hour.

As represented in algorithm 1, the resources recognized the most priority task as the zeroth position. These kinds of requests are mostly unexpected to the resources. In this research, when fog localization area is full of users' requests and high priority tasks arrive

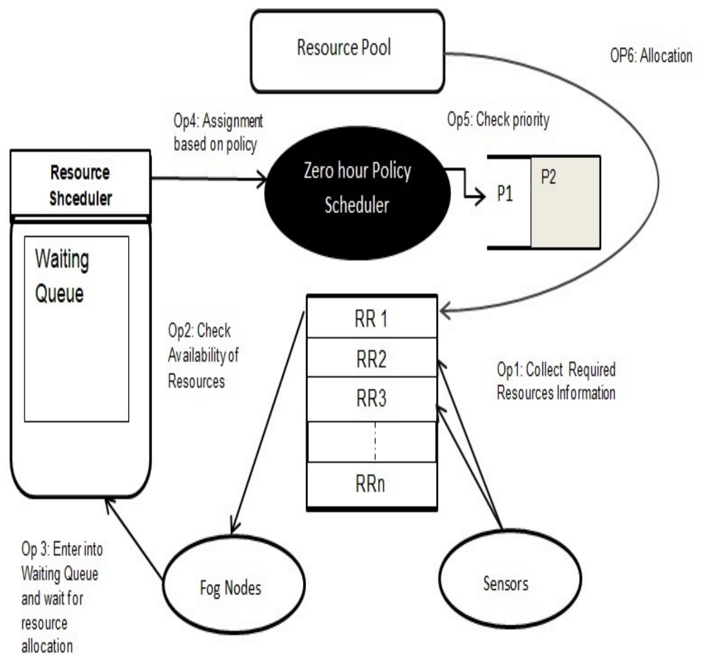

Resource Pool
OP6: Allocation
Op4: Assignment based on policy
Op5: Check priority
Resource Shceduler
Zero hour Policy Scheduler
P1
P2
Waiting Queue
Op2: Check Availability of Resources
RR 1
RR2
RR3
RRn
Op1: Collect Required Resources Information
Op 3: Enter into Waiting Queue and wait for resource allocation
Fog Nodes
Sensors

Figure 3.6: Working Architecture of Zero Hour Policy Scheduler

in the system is called zero hour has arrived. Moment of decision has occurred for all fog devices. Fog broker will give a response to that high priority task and allocate all required resources for that task.

Algorithm 1 shows how the Zero-Hour policy will be worked in the fog-Cloud environment. The input of this algorithm is the list of existing resources, available bandwidth and latency. Algorithm 1, illustrates that the method of resource allocation is represented as M_r for all fog devices N. Before this, the fog devices, processing time of requests and route is initialized. All the modules are processing through route r whereas the readymodule is empty.

The priority P is selection criteria that check the user request and select the best appropriate route. Moreover, the elected route details have information about available resources. Therefore, Processing time(PT) requires computing power by fog nodes compared with the available resource capacity. Hence, after comparing the available resources' capacity, new values are assigned to the resources to high priority tasks. The value and PTs are periodically changed. The output of this algorithm is reset the time quantum for that particular process.

The functionality of Zero Hour policy scheduler is presented in Figure 3.6. Six operations occurred, which are represented as Op1 to Op6. Each operation has a specific task to be performed. Op1 collects all data from sensors and creates a resource pool with the variable request of resources generated by users. Op2, check the availability of the resources with fog nodes. Op3, start entering each request into the waiting queue. The created waiting queue is handled by the resource scheduler and the working of the resource scheduler in explained in Figure 3.5. It also decides whether the request should assign to the fog nodes or cloud. Op4 When a request arrives from the sensor node with high priority or from nearby fog devices, it is sent to the Zero hour policy scheduler. A separate priority queue is created after this operation. Under Op5, the priority of existing and arrived request checks through this operation. After that, Op6 allocates resources to the selected request.

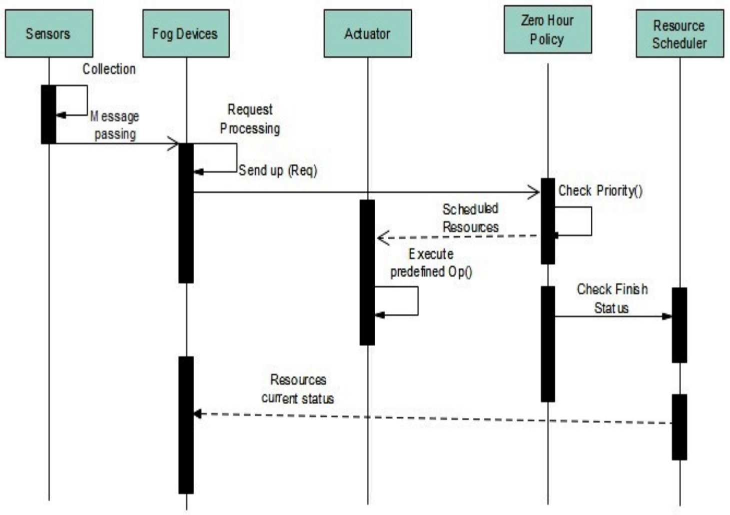


Figure 3.7: Sequence Diagram of Resource Scheduling and Execution

All the sequence of operations, such as scheduling and its execution represented in Figure 3.7. As shown in Figure 3.7, when a tuple is emitted a method collection() from one sensor, it is sent to fog devices via message passing(). A send up (Req) send to zero hour policy to check the priority of the request. This method checks the need for resources and requests to be processed on this layer or transfer to the top layer. After receiving approval, scheduled resources() operation execute the predefined operation and assign resources. After that, the zero Hour policy shares the finish status() to the resource scheduler to check the current status of resources.

3.3 TRAM: Technique For Resource Allocation and Management

Real-time IoT application handles task request and assigns resources as per their requests through fog server [148]. The fog server provides multiple services to the terminal devices. The fog application workflow is represented in Figure 3.2. As represented in the figure, terminal devices generate massive data that fog servers process through available resources.

Fog server runs a computational process based on soft real-time analysis and deep learning techniques. It also supports the dynamic behaviour of clients and manages resources through a resource scheduler. All the mentioned techniques are used to provide numerous services such as real-time analysis, monitoring, scheduling, task migration and communication between devices.

TRAM , a technique for resource allocation and management, is proposed to ensure resource utilization at the fog layer. TRAM implements through task clustering and resource allocation. As represented in Figure 3.8, the data or requests are generated through IoT devices, it directly passes to the fog layer and generates tasks. These tasks are processed as per EM algorithm and create clusters. The information about the high, medium and low intensity tasks are shared with fog server where the resource grading process will create LoR(List of Resources).

In the hybrid fog-cloud architecture, cloud platform is used for the large and long-term storage of application outcomes or analyzed data. The resources will allocate to the tasks for the execution. If the tasks belong to the low intensity and very less priority then they can be treated at cloud layer. High usability of the resources can be achieved with the proper allocation of resources. In the end, data and processing details are shared with the cloud for storage. This technique also deals with scalability, device mobility for an application. The connection between all the devices, along with the fog server will be wireless.

3.3.1 Mode of Operation

All the tasks from the IoT devise are integrated and cumulatively sent to the task coordinator which is responsible for the efficient organization of tasks. The task coordinator classifies the tasks into three classes namely (i) highly intense (ii) moderately intense and (iii) less intense. Figure3.9 provides the graphical representation of the classification process.

The highly intense represents the tasks that are delay-sensitive and need to be imple-

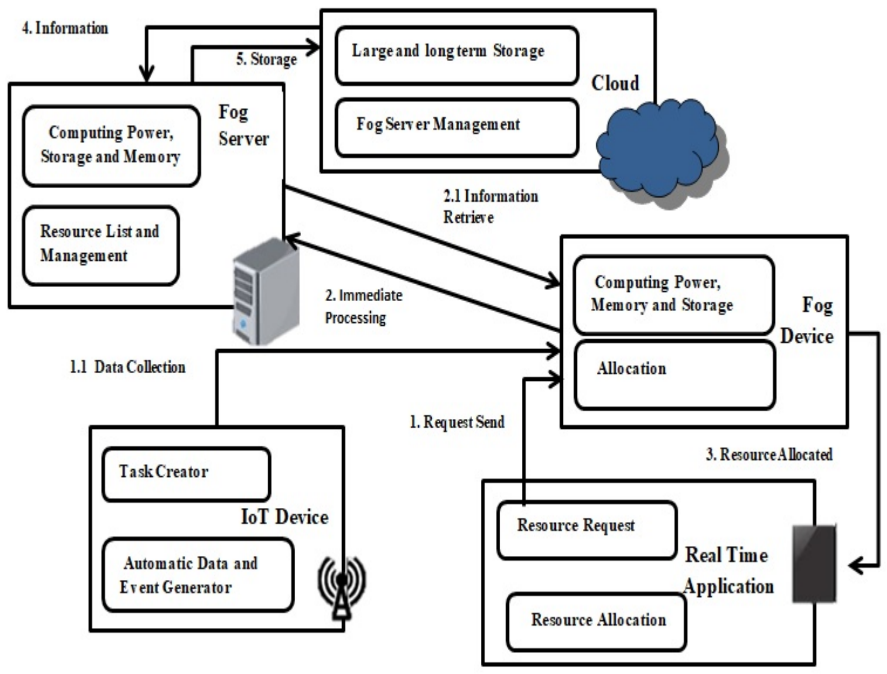


Figure 3.8: Technique for Resource Allocation and Management

mented immediately. The moderate intense tasks are the one which has a soft deadline and are given medium priority. The less intense tasks are mostly storage based tasks that do not have long deadlines.

The task coordinator clusters the tasks based on these three classes by using Expectation Maximization (EM) clustering. Through this process, the network overhead is eliminated which is raised due to the increase in the number of devices. This also reduces the computational complexity. Task clustering is the process of combining the fine grained tasks into coarse grained tasks.

The advantage of task clustering is that it minimizes the execution overhead and improves the computational granularity of task scheduling for resource allocation. Based on the task clustering, execution overhead is eliminated using small task clustering. This overhead reduces the cloud environment overhead on the whole by minimizing the traffic rate. The proposed work performs task clustering for fine grained tasks submitted by end users. For the first step, dependencies of tasks are computed by the probability function as follows,

$$P\left(T_i \mid T_j\right) = \frac{P\left(T_i \cap T_j\right)}{P\left(T_i\right)}, \quad P\left(T_i\right) > 0 \tag{3.1}$$

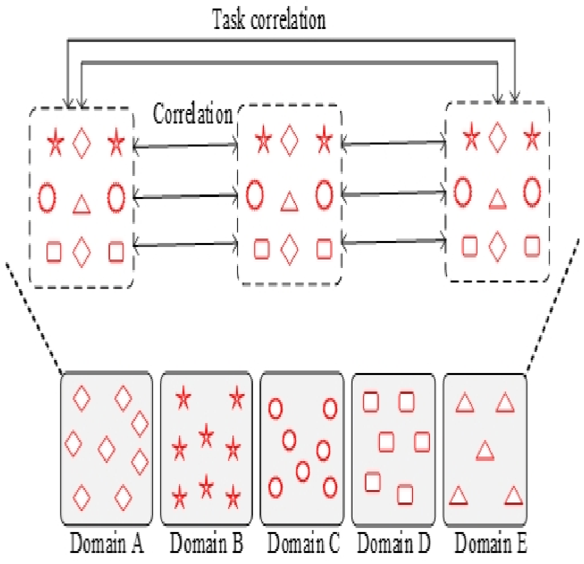


Figure 3.9: Task Clustering Result

WhereT_i represents the probability of task T_i that occurs the given task T_j, that dependency/similarity between the pair of tasks. In other words, it is called intercorrelation.

3.3.2 Task Clustering

Clustering of tasks improves the performance of individual tasks. Task clustering expressed using the EM algorithm. The proposed EM algorithm determines the Maximum Likelihood of the parameters of the probabilistic model. There are two steps utilized: expectation ϵ and maximization μ for optimum clustering.

Task clustering uses the Finite Gaussian Mixture Model and estimates the set of parameters for achieving the convergence value. The mixture of K probability distribution functions can be represented to one cluster and each task is respected to assign to the cluster through membership probability as represented in Figure 3.9 . Three steps are followed in the EM algorithm for task clustering that is mentioned below,

- Step 1 - Initialize Parameters: Mean and Standard Deviation is estimated for the input tasks i. This is implemented using the Normal Distribution Model.

- Step 2 - Refine Parameters for Iteration: Compute expectation ϵ and maximization μ for computing the possibility of membership to teach task and new membership possibility is computed in expectation and maximization.

- Step 3 – Assignment of Task to Cluster: Each task is assigned to the cluster based on the highest membership possibility. A result of EM for the task clustering is illustrated in Figure 3.10

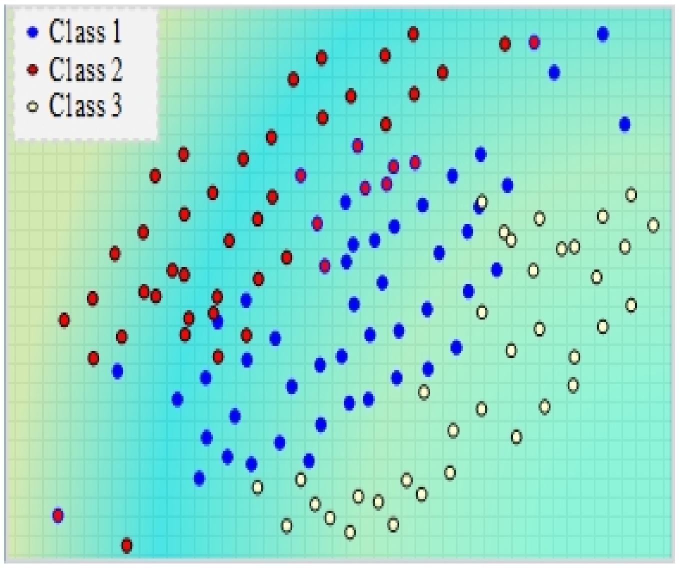


Figure 3.10: EM Clustering Result

The complexity of tasks were reduced by representing the tasks in the form of self-organized maps and were scheduled according to the deadline, but the tasks were directed to the nearest fogs without considering the resource availability and energy availability which led to retransmission of tasks[135].

The game theory concept was executed to perform matching tasks to the fog nodes by considering the preference list; however, this concept did not address each task's varied deadline [119]. Various existing works classified the tasks based on the deadline. They scheduled the tasks according to the tasks' priority pattern, which reduced the problem of missing deadlines and improved the process's efficiency. However, the tasks' dynamic nature was not considered, which limited the implementation of these works in real-time scenarios [109] [89]. The machine learning models were used to predict the nature of the upcoming task to perform computations at a very low latency but the dynamic nature of tasks restricted the performance of these works [137].

3.4 Expectation Maximization

The first step in identifying workloads with equivalent resource usage patterns is to perform a cluster analysis [131]. In the fields of statistics, pattern recognition, optimization, and machine learning, the clustering problem has been formulated in a number of different ways [3] [74]. The main concern is clustering (grouping together) data elements that are similar to one another. EM clustering is used in the proposed framework to build task clusters based on the intensity level.

The EM algorithm (Expectation-Maximization) is a popular and successful method for predicting mixture model parameters (cluster parameters and associated mixture weights) [1] [146]. The EM approach refines initial mixture model parameter repeatedly estimates in order to better accommodate the data and terminates at a locally optimal solution.

The maximum likelihood approach of evaluating design variables maximizes the likelihood that the model will generate the observed data. By calculating the negative log of the likelihood function, machine learning methods decrease an error function. The Expectation-Maximization (EM) algorithm is an iterative approach for calculating maximum likelihood estimates for probabilistic models with latent variables (variables inferred from observed data) [131]. The EM algorithm involves the following steps:

- Initialization: Using a random initialization, get an initial estimate of the model parameters.
- Expectation Step: Estimate the latent variable values while keeping the model parameters constant. values for the model parameters that minimize an error function. Maximization: Estimate the hidden variable values while keeping the model parameters constant.
- Termination: Steps 2 and 3 should be repeated until a convergence requirement is reached.

The EM algorithm using statistics iteratively finds the maximum likelihood of server for being in a particular cluster type. It determines parameters of probability distribution by obtaining maximum likelihood and then tries to maximize the obtained data log likelihood on the basis of the observations, revealing the cluster to which a server will belong. This process keeps repeating until reaching the case where the same points are being assigned to each cluster in consecutive rounds. Let $\{T = t_j | j = 1, 2, ...n\}$ be the set of n dimensional points to be clustered into a set of k clusters, $\{C = c_k | k = 1, 2, ...k\}$. Mathematically, EM clustering algorithm seek to minimize the squared error function.

$$J(C) = \sum_{k=1}^{K} \sum_{t_j \epsilon c_k} \left(\left| t_j \ - \mu_k \right| \right)^2 \tag{3.2}$$

where $\left(\left| t_j \ - \mu_k \right| \right)^2$ is a selected distance calculate between a data point t_j and the mean value μ_k of cluster c_k for all the tasks from their respective cluster centers.

3.5 Verification of Resource Scheduling Framework

Resource scheduling policies deal with the resources and the tasks. In latency, time, cost and throughput based resource scheduling, the resources are allocated to those tasks whose requirements such as deadline, latency and cost are fulfilled by the fog broker.

3.5.1 Simulation Model: iFogSim Toolkit

The proposed methodology is implemented in the simulation environment because the actual setup is less convenient to use. Hence, iFogsim [49] is used to determine the efficiency of the proposed methodology. The simulation environment offers an environment where experiments can repeat with numerous parameters. Experiments were conducted by increasing the time-sensitive task requests. For this research, synthetic and real time workload was used to explore the resource allocation scenario [130].

The simulation has been done with four different setups with different configurations by varying areas and sensor nodes. Initially, the topology is constructed to perform the scheduling of tasks. Each sensor node is connected via the communication channel to the fog server. The physical setup configuration of cloud devices, fog devices and end devices are represented in Table 3.2.

In set up 1, the number of the working area is one and the number of fog nodes implemented in that area is also one. However, the number of end devices is four. These devices will collect the task from users and transfer it to the resource scheduler.
In set up 2, two areas are evaluated with two fog nodes and four end devices.
Similarly, set up 3, is created with three areas, four fog nodes and each area is connected

with four end devices.
Final set up 4 is build up with four areas, eight fog nodes and four end devices.

Table 3.2: Physical Setup Configurations

Physical Setup Name	Number of areas	Number of Fog nodes	Number of end devices
Set up 1	1	1	4
Set up 2	2	2	4
Set up 3	3	4	4
Set up 4	4	8	4

The proposed policy is evaluated in a simulation environment. There are many simulators used by numerous researchers. Hence, a comparative study is done between different simulators available for fog computing. Cloudsim is implemented by Calheiros et al. [10] Cloudsim is an extensible framework that simulates the infrastructure of cloud computing and tests the performance of all services of cloudsim.

A fog simulator, iFogSim has been developed by Gupta et al. [49] on the basis of cloudsim in 2016. The authors listed that researchers and developers could use cloudSim to check the performance and improve or upgrade the system's services. iFogSim incorporates principles of two different technologies, fog computing and IoT.

This toolkit can calculate the latency, network usage, power consumption, and cost after implementing resource management techniques. The objective of cloud applications, RECAP is advancement in edge computing technology and to construct a mechanism for capacity provisioning[60].

In the previously designed simulators, a low level to QoS is retrieved. In RECAP, automatic reconfiguration and remediation are in a preemptive manner. The RECAP approach follows five different components for data collection, workload distributor, application modeller, simulator, and optimizer. The RECAP project will explain architecture in order to implement the ideas related to resource management, data analytics, and intelligent automation.

FogNetSim++ [79] that gives users different configuration possibilities to build large fog network. It facilitates researchers to implement scheduling algorithms and modified mobility models for fog nodes. A traffic management system is assessed to demonstrate the scalability and efficacy of the proposed simulator in terms of CPU and memory utilization.

Table 3.3: A Comparison Between Different Simulators

Authors	Tools Used	Year	Task	Language	Open Source	Mobile Node	Schedule algorithm
Gupta[49]	iFogSim	2017	Resource management	Java	Yes	yes	yes
Lopes[55]	MyiFogSim	2017	Resource management	Java	Yes	Yes	yes
Östberg[60]	RECAP	2017	Capacity Provisioning	C++	N/A	No	No
Qayyum[79]	FogNetSim++	2018	Resource Consumption	C++	Yes	Yes	Yes
Brogi[43]	FogTorch	2017	Resource Consumption	Java	Yes	No	No
Mayer[56]	EmuFog	2017	workload management	Python	Yes	No	Yes

Lopes et al. [55] defined MyFogSim extending the simulator for iFogSim to support mobile app migration policies. MyiFogSim manages appModules migration which supports the application's modules, similar to CloudSim methodology to handles the VM migration.

In Table 3.3 comparison between simulators is represented. iFogsim simulator is used to model the realistic fog environment. iFogSim is an efficient tool for simulation for all resource management techniques in the cloud-fog environment. For the implementations, predefined classes were modified to implement the scheduler. All the predefined classes include sensors class representing all edge devices and tuples class representing the communication between two nodes.

3.6 Experimental Scenario: Zero Hour Policy

In iFogSim, tuples form a communication connection between source to destination entities, inherited from the cloudsim simulator. A tuple is characterized by type and processing requirements. CPU length and network length are two attributes that specify the tuple type, whereas CPU length measures in a million instructions per second (MIPS) and network length are represented in bytes.

In Figure 3.11, six modules are represented, which perform six tuple processing. These modules named are sensor, fog device, object based detector, user interface, clouds based tracker and actuator and six tuple processing are know as raw Raw Data Collector (RDC), Process data (PD), Detected Object Information (DOI), Process Information (PI), Issued Instruction (I^2) and Action Against Instruction(AAI).

For the algorithm placement strategy, ModulePlacement and Controller Classes are used for all resources. All the properties related to tuples are described in Table 3.4. This research represents the need for such a policy that can automatically analyze the raw data collected from smart devices and review the result in such a manner that is useful to the end-users.

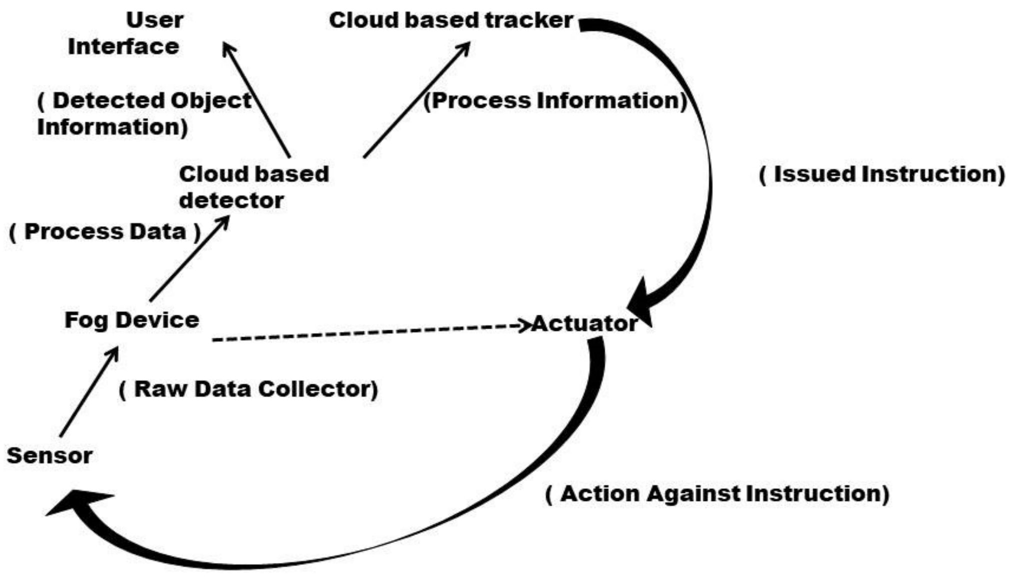


Figure 3.11: Application Model with Tuple Processing

Table 3.4: Tuple Type Description

Tuple Type	CPU length (MIPS)	Network Length (Bytes)
RDC	1000	2000
PD	2000	2000
DOI	1000	200
PI	1000	500
I^2	200	200
AAI	200	200

Table 3.5: Values of Parameters in the Cloud Fog Environment

Name	Level	UpBw (MB)	DownBw (MB)	CPU (MIPS)	RAM (MB)	Busy Power (Watt)	Idle power (Watt)
Cloud	0	1000	10,000	44,800	40,000	1648	1.332
Fog Server	1	10,000	10,000	2800	4000	107.339	83.4333
Fog Broker	2	10,000	10,000	2800	4000	107.339	83.4333
End Device	3	10,000	10,000	500	1000	87.53	82.44

3.6.1 Performance Metrics

The TRAM performance evaluation is measured through the processing cost, processing time and delay. Cloud and fog devices have varied processing power and cost. The hybrid computing environment is generally used for time-sensitive applications and always tracks these applications' processing time.

For example, we want to execute 900 million instructions and for this task, two nodes are available. The nodes are available with 500 MIPS with 100 Mbps and 1000 MIPS with 10 Mbps ethernet connection. If these two systems' performance checks and calculates the task completion time, the second system takes more time to transfer the required data. Even though the second system has a high processing capacity, it processes the task slower. The response time needs to be considered along with the task's finish time because if it is high, it might not finish the task within the prescribed time limit.

Table 3.5, represents the resourced configuration of iFogsim. Each fog device, fog server and end device has many parameters such as RAM, upper bandwidth, down bandwidth, busy power and idle power. Table 3.6, represents the host configuration used for the study.

The simulation shows the performance of resources in the fog computing system. There

Table 3.6: Host Configuration

Parameters	**Value**
Architecture	X86
Bandwidth	10,000B/S
OS	Linux
VM model	Xen
Time Zone	5
Cost	2
Cost per memory	0.10
Cost per storage	0.01

is a fog broker that monitors the resource allocation to the tasks and their utilization. In this study, the fractional Selectivity method was used by all the tuples. The experiment results are compared with CLC (cloud level computing) policy. CLC is associated with Cloud level resources and every computation is done at the cloud level. On the other side, Zero Hour policy represents the computing near the user device and assigns resources per the task's priority.

For comparing these two policies, delay, network consumption, execution time and execution cost are calculated. All the used notation for calculation purposes is represented in Table 3.7.

As represented in equation 3.8, the delay is measured in the simulator as the amount of time system has to wait before the assignment of task to the resources. The time when the request for the first service in the loop arrives $t_{S \cdot F}$ and the end time of services is represented as $t_{S \cdot E}$. In the upward direction, data transfer from the sensor node to the high-level node and downlink where the transfer of data forms the computing node to the actuators.

$$Lt = \sum^{R} \frac{Req\ (t_{S \cdot E} - t_{S \cdot F})}{R} \vee Req \in loop\ (S_F, S_E)) \tag{3.3}$$

In equation 3.9, $T_{Oi,'_{Ti}}$ is representing network consumption of devices from the original and targeted position of requests. NS_r is the total request size sent on the network. The total network consumption is calculated by adding the network consumption generated by

Table 3.7: Used Notations

Symbols	Definition
B	Bandwidth of Fog Nodes
E^x_{st}	Execution Start Time
E^x_{ft}	Execution Finish Time
F_{clk}	iFogSim Clock
H_{mips}	MIPS used by Host nodes
LU_t	Last Utilization Time
N	Fog nodes
NS_r	Total request size sent on the network
R_{mips}	Rate Per MIPS
$t_{s\cdot F}$	Start time of resources request
$t_{s\cdot E}$	End time of resource request
$T_{O_i,T_i'}$	Network consumption of devices from the original and targeted position of requests
T_c	Total Execution Cost
T_l	Total Task Length
T_I	Input size of Task
U_last	Last Utilization
V_mips	MIPS of Virtual machine

each request.

$$Network\ Consumption = \sum^{r(O_i,T_i)} \frac{\left(T^l_{O_i,T_i'} \times NS_{r(O_i,T_i')}\right)}{Time} \tag{3.4}$$

In the equation 3.10, execution cost is calculated based on the last utilization cost. The cost also depends on the MIPS of the host node and the time to execute the task.

$$Execution\ Cost = T_c + (F_{clk} - LU_t) \times R_{mips}(U_{last}) \times H_{mips} \tag{3.5}$$

The equation 3.11 and 3.14, represents execution time which can calculate by subtracting the task start time from the task finish time. However, E^x_{ft}, depends upon specific parameters such as task length, which is assigned to the readymodule. All the parameters used in equations are mentioned in Table 3.7.

$$Execution\,Time = E_{st}^{x} - E_{ft}^{x} \tag{3.6}$$

$$E_{ft}^{x} = \left(\frac{T_l}{N} \times V_{mips}\right) + \left(\frac{T_l}{B}\right) \tag{3.7}$$

Figure 3.12 to 3.15 represents the simulation results in various experimental settings. Four different setups were considered to check the performance of the proposed policy. The physical configurations of these setups are represented in Table 3.2.

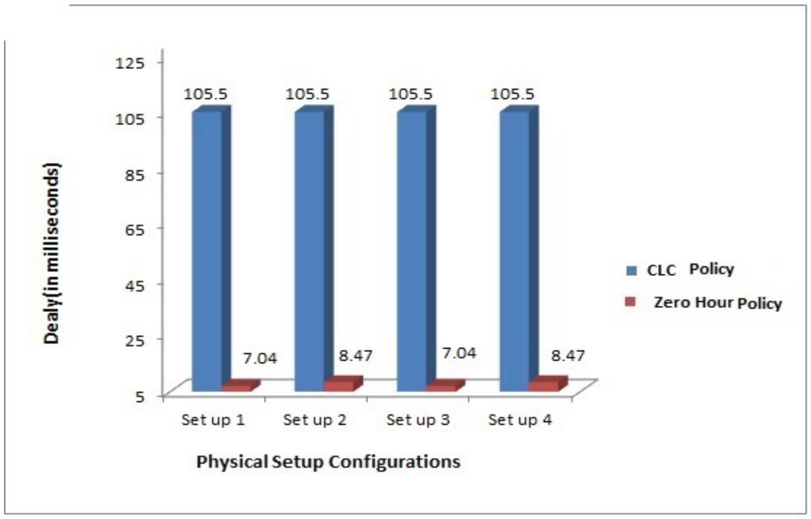


Figure 3.12: Delay in Control Loop

In Figure 3.12, delay in the control loop represents both policies. There is a significant difference in average delay for zero-hour policy for all four steup.

It represents that even with the increase in the number of areas and fog nodes, it reduces all loops' average delay. When the computing is done near edge devices, the delay time is less for all four setups. The overall average delay is reduced by ≈68% compared with CLC policy.

As represented in Figure 3.13, the Zero Hour policy can effectively reduce the unnecessary use of network resources. The network usage in the CLC policy is high, where the application consumes all the cloud resources. It can also reduce the performance of the application. Network usage follows the increasing trend for all setups because the number

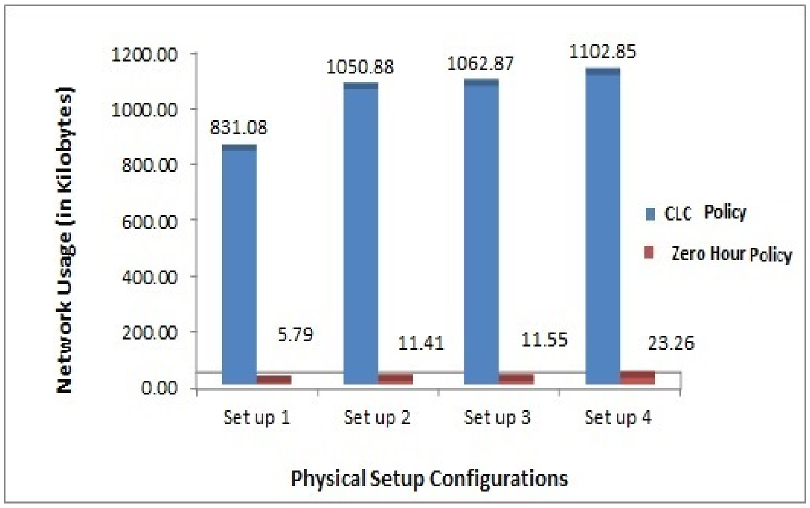


Figure 3.13: Network Usage

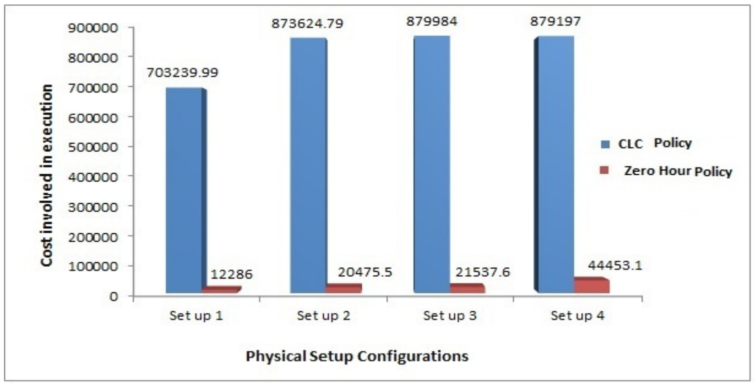


Figure 3.14: Execution Cost

of areas and fog nodes increases to collect more task requests. Network usage is calculated in Kilobytes.

The execution cost of CLC and Zero Hour policy is represented in Figure 3.14.

The cost is degraded when the application used fog resources instead of cloud resources. Total costs were reduced because execution time was less in the proposed algorithm. The proposed algorithm allocates resources to those tasks which are at zeroth position.

In Figure 3.15, the simulation time of different policies is measured for different configurations. For setup 1, the average processing time decreased by 55%. For setup 2, the average processing time decreased by 50%. For setup 3 and setup 4 it is decreased by

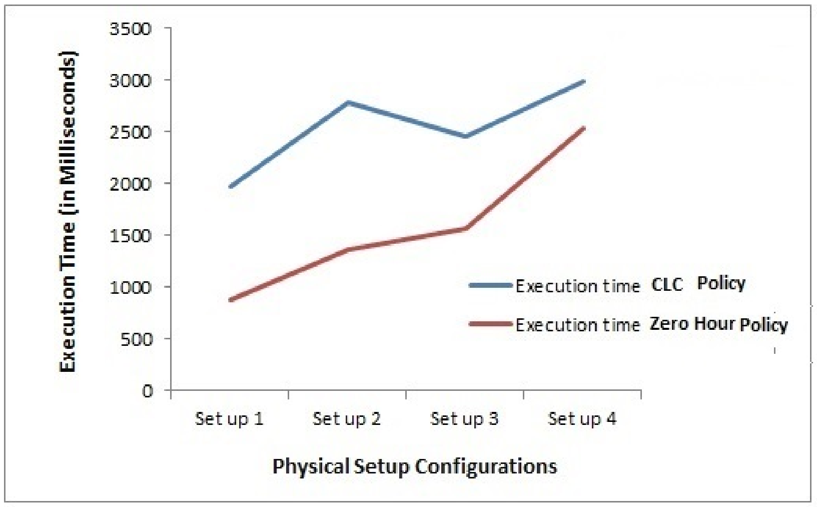


Figure 3.15: Execution Time

36% and 15% compared with CLC policy. Execution time decreased in zero-hour policy because it constantly allocates resources available at the fog layer to the task generated by end devices. Whereas in the CLC policy, tasks are directed to the cloud resources. Overall, the zero-hour policy performs better than the CLC policy in execution time, network usage, cost and delay.

3.6.2 Experimental Scenario: TRAM

The increased number of devices in the IoT environment exponentially increases the dynamic nature of the tasks. The task preemption is an effective process in dynamically adapting towards the computation demand of the tasks.

If the fog node having certain resources is computing a low priority task with a longer deadline and the high priority task that needs to be executed immediately arises, task preemption is used to enter the low priority task in the waiting state and implement the high priority task. However, preemptive tasks can be interrupted during execution so that they can be stopped during the execution.

For instance, in the Web Crawler application, when the preemptive task's resources are collided, tasks are reassigned to adjacent resources from collided point. Whereas non-preemptive tasks are uninterrupted and cannot be dismissed during their execution as Encoding, Rendering, Batch processing and Continuous Integration. When the resources of non-preemptive tasks are depleted, then these tasks are reassigned from the start.

This significantly reduces the average response time of the tasks. The dynamic task preemption is carried out using a preference ranking method, a multi-criteria decision-making approach. In decision making, priority is decided by the fog node and computed by the fairness, deadline and optimized response time. Furthermore, the study is focusing on the challenge of task granularity. This way of executing the high intense task between the low intense task is an efficient way of managing the resources. Moreover, errors in high priority tasks implementation require lots of computation and high value of resources. Hence performing preemption for low priority tasks and assign the resources to high priority ones increases the resource availability and reduces the task failure rates.

The simulation of our proposed TRAM model is carried out using the iFogsim simulation tool. The system configurations required for simulation of our model is presented in Table 3.9. Initially, the topology is constructed to perform the scheduling of tasks. The simulation parameters used in our model are provided in Table 3.8. Figure 4.6 depicts the network topology, simulation configuration and simulation results, respectively.

Table 3.8: System Parameters

IoT devices	15
Gateways	3
Task coordinators	3
Fog nodes	3
cloud server	1
Memory	10 MB
Bandwidth	1000 KBs
MIPS	1000
Capacity of RAM	10

Table 3.9: System Configurations

IDE	Netbeans 8.0
Type of topology	fully connected
development kit	JDK 1.8
Operating System	(Windows 7 ultimate [x86]-32 bit processor
language	Java
CPU	Pentium (R) Dual-Core CPU E5700 @ 3 GHz
Memory	4GB

3.7 Performance Evaluation Criteria

In this section,the performance evaluation criteria for the zero-hour policy based hyper-heuristic for resource scheduling have been defined. Four metrics, namely makespan, load, deadline and throughput have been selected and calculated using the equations described in Chapter 4 for evaluating the performance. The makespan, deadline, load and throughput are measured in seconds and MegaByte(MB) respectively.

3.7.1 Results and Discussion

To validate the proposed algorithm, hundred tasks have been considered. The simulation results have been presented using Rafique et al. [103] simulation model with the iFogSim discrete event simulation to test the performance of the proposed algorithm. In addition, a comparison of makespan, load, deadline and throughput of the proposed algorithm with existing heuristic algorithms such as NBIHA, MMPA, SJF and FCFS has been presented. The performance of the proposed approach is evaluated and investigated the effects of the different number of tasks. In all the experiments, a comparison for consistent and inconsistent parameters has been done.

3.7.2 Test Case 1: Energy Consumption

The fog devices energy can be determined by all hosts within specified time frame for execution. In the Equation (3.8), consumed energy by fog device is represented as ϵ_{FN}, current energy consumption is represented as ϵ_c. As per the resources consumption, τ_i is time of the last resource utilization whereas τ_c is current required time to complete the task and P is last utilized host power.

$$\varepsilon_{FN} = \varepsilon_c + (\tau_{c-}\tau_i\ )\ P \tag{3.8}$$

$$\psi = \min\left(\frac{A_m}{H_m}\right) \tag{3.9}$$

In the Equation (3.9),last utilization(ψ) is calculated where A_m is used for host allocated mips and H_m is total mips of all machines.

Figure 3.16, shows energy consumption by TRAM. The number of nodes involved in the simulation is represented along X-axis and energy consumed is represented along Y-axis. Energy consumed by FCFS, SJF, MPSO and proposed technique TRAM presented in

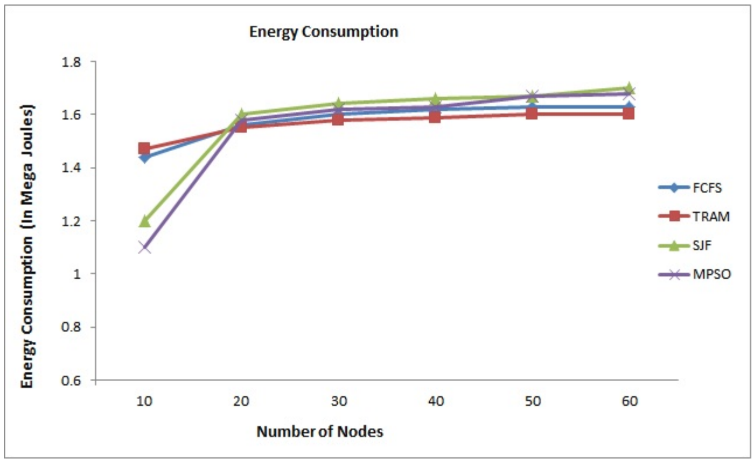


Figure 3.16: Energy Consumption

Figure 3.16. The energy consumption is high for the TRAM when the nodes count is 10, 20. However when the nodes count increases the energy consumption for the TRAM is low as compared with FCFS, SJF and MPSO.

3.7.3 Test Case 2: Average Loop Delay

The represent end to end latency of all described modules is measured through control loop. The loop delay and processing time of CPU(T_p) is calculated through Equation (3.10).

$$\overline{T_p} = \frac{E_{Si} \times N + D_i}{N} \tag{3.10}$$

E_{Si} is the estimated service time time and E_i is the tuple ending execution time. Total number of executed tuples are represented as N and individual tuple is represented as ith tuple which is part of large tuple set T. By using Equation (3.11), calculated the execution delay of each tuple.

$$\{D_i\} = E_i - S_{i,} \quad \forall i \ \in T \tag{3.11}$$

In Equation (3.11), T represents existing tuple set. The comparison of application loop delays generated by the TRAM with the outcomes of implemented scheduling algorithm FCFS is shown in Figure 3.17.

Average loop delay determined in milliseconds using proposed technique TRAM and compared with the existing approaches such as FCFS, SJF and MPSO. Figure 3.17 shows

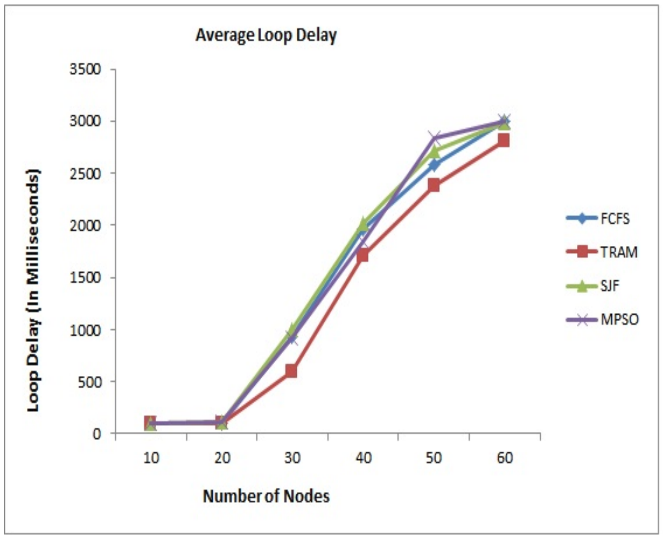


Figure 3.17: Average Loop Delay

the number of nodes along with the x-axis while the y-axis presented loop delay during the simulation. This demonstrates that it takes less time to transfer data from one device to another device by using the proposed technique. At the initial stage, each technique represents the same results, when the number of nodes increases the loop delay for TRAM is less as compared to the SJF, FCFS and MPSO.

3.7.4 Test Case 3: Latency and Network Consumption

In the this evaluation, N_w parameter is used for network usage. As the number of devices grows, network usage will also increase and it can develop network congestion. By dispersing the load on some other fog devices, TRAM helps to minimize network congestion.

$$Latency = \sum_{i=1}^{n} (Expected\ execution\ time - Actual\ execution\ time) \tag{3.12}$$

In Equation(3.13), network usage N_w is computed with latency L and network size n. Network usage is computer of ith tuple form complete tuple set T.

$$N_w = \sum_{i=1}^{T} L_i \times \eta \tag{3.13}$$

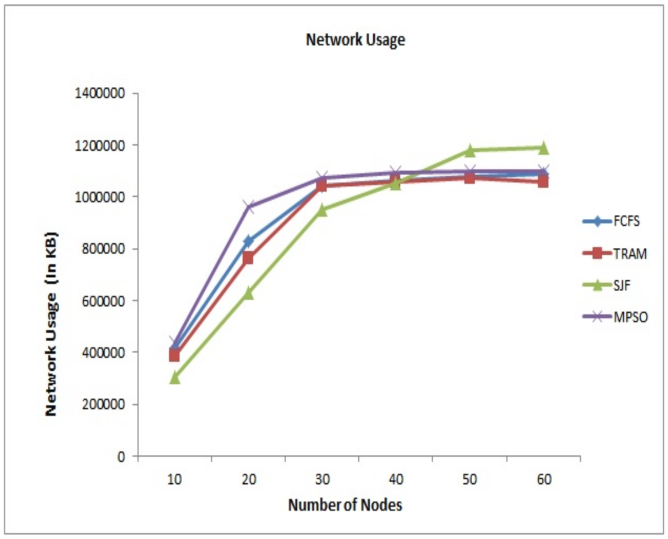


Figure 3.18: Network Usage

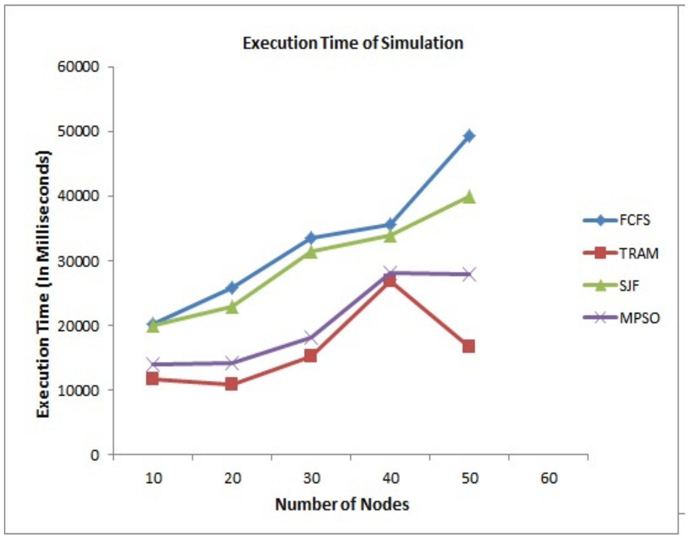


Figure 3.19: Execution Time of Simulation

Figure 3.18, represents the comparison of the proposed technique TRAM with existing approaches SJF, FCFS and MPSO. The number of nodes is presented along the X-axis and network usage is presented along the y-axis. In this evaluation, total execution cost is computed using Equation (3.14). The cost of applying TRAM method is calculated and compared with implemented techniques. Present cost, current execution time and updated last utilization time is represented as P_c, T_c and T_i respectively. The value of (ψ) is calculated through Equation (3.9). R is rate per mips and H_m is total mips of all host machines.

$$Cost = \sum_{i=1}^{Fn} P_c + (Tc - Ti) \times R \ \times \psi \ \times H_m \tag{3.14}$$

3.7.5 Test Case 4: Execution Time

In this fourth evaluation, the execution time for the proposed technique is represented for 60 nodes. The execution time of TRAM was compared with SJF, MPSO and FCFS. As fog infrastructures are dynamic and changes in the scheduling algorithm can also change simulation execution time. In Figure 3.19, execution time comparison is represented. The result represents that the proposed technique's execution time can save 48% of the execution time for the first four scenarios when the number of nodes is 40. The maximum number of nodes is 60, When the nodes extend, the execution time decrease upto 24%. However, in the last scenario 6, where the number of nodes is 60 it can save upto 60% time.

3.8 Applications of Proposed Architecture and Approach in Real Time Environment

3.8.1 Smart Traffic Light System

The proposed work can be used to implement real time application i.e a smart traffic light system to control the traffic of smart city. Sensor-based traffic control device that will be connected to the traffic signals. Due to the rapid increment in the number of vehicles in the metro cities, it has become necessary to take some initiative so that road congestion can control and make a path for emergency vehicles. The invention is aimed to create a traffic control system to avoid road congestion and reroute the emergency vehicle.

This system can be designed by implanting sensors at a distant location from traffic lights, which can directly be connected to the controlling system of fog nodes. This fog node can be associated with the traffic light system and the main server. The first sensor counts the vehicles and fog nodes will inform the nearby hospitals about the road congestion so that emergency vehicles can reroute the path. The actuators linked with traffic lights change the timing of signals to clear the road congestions. It works on the principle of IoT

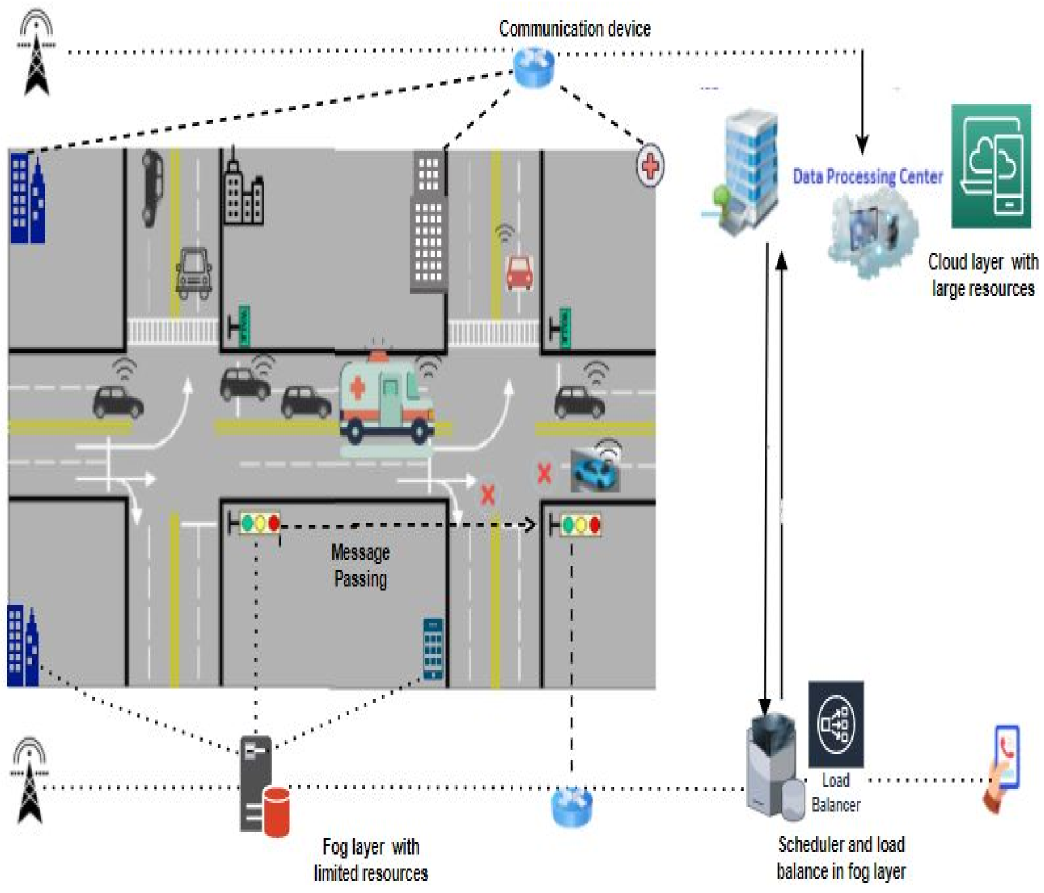

Communication device
Data Processing Center
Cloud layer with large resources
Message Passing
Load Balancer
Fog layer with limited resources
Scheduler and load balance in fog layer

Figure 3.20: Smart Traffic Light System

and is connected to the cloud to manage the traffic log. As represented in Figure 3.20, The proposed invention can used to reduce road congestion in metro cities and build smart cities. It also helps emergency vehicles to reach their destination on time, which save human life. This can also help in saving time of any person who stuck on signals and waits very long for their turn.

3.8.2 Smart Leakage Detection in Smart Cities

An architecture based on hierarchical distributed Fog computing is represented by Tang et al. [106] [30] to help the reconciliation of a vast number of designing smart devices and services in future urban communities. They had analyzed case studies of smart gas pipeline system which was developed by learning algorithms and fiber optical sensor to identify the risk in 12 distinct events. In the proposed 4-layer architecture, layer 4 was generated with sensing networks which were distributed at different public locations. Layer 3, included high performance computing nodes and these nodes are also known as edge devices. Layer 2, Contain intermediate nodes connected with a group of edge devices. If one section of gas pipeline is encountered with a leakage or a fire then Fog layer (layer 2 and layer 3) computing node will detect the risk and shutdown the gas supply in the area. Layer 1, Cloud computing database layer to provide monitoring services, long term analysis and pattern recognition to support decision making.

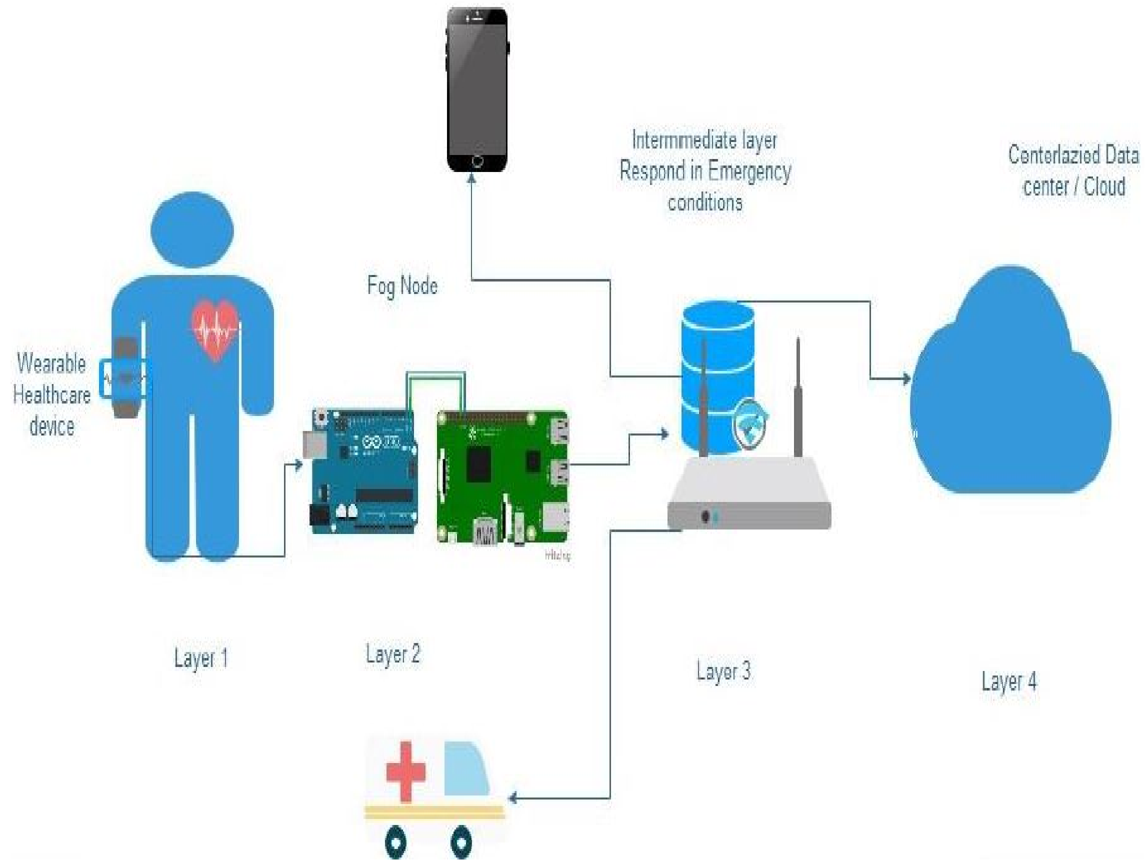

Intermmediate layer
Respond in Emergency conditions
Centerlazied Data center / Cloud
Fog Node
Wearable Healthcare device
Layer 1
Layer 2
Layer 3
Layer 4

Figure 3.21: Smart Healthcare Gateway

3.8.3 Smart Healthcare Gateway at the Fog Layer

Fog layer is extended to develop fog enabler by i) creating an arranged gateway network and ii) employing numerous abilities like acting as local repository for temporarily storage of data and combining it with different techniques. Local preprocessing is very essential part for creating smart e-Health Gateway. Figure 3.21 shows a theoretical architecture of a smart e-health gateway using a local database for data filtering and data processing: Fog computing having an significant feature that data can be processed locally to offer intelligence at the gateway [80].

In Fog computing, fog layer deals with vast amount of data generated by sensors and respond within short period of time. This is very important step in case of any medical emergency. Basic data analysis is performed before advance level processing. In the medical scenario, all bio signals collected from user's body are the main source for evaluating a patient health status. While getting input from sensors, noise can alter the quality of signal. Such noise is created by various reasons like inappropriate attachment of sensors to users' body. After receiving these digitalized signals via different communication protocols, some data filtering techniques may use to remove noise factor from signals. Before performing data analysis, smart healthcare gateway enables data compression and data fusion for improving high latency problem. In data compression, both lossy and lossless compression approaches are beneficial. For real-time applications such as ECG monitoring, lossless compression system is followed[17]. These techniques will ensure the high accuracy in the received signals. Data analysis technique checks the sensitivity of data and assists the fog layer to detect emergency situations. The system reacts faster to the emergency situations. In this paper, Rahimi et al. [80] proposed an approach for healthcare systems with some processing control at the end points.

3.9 Summary

This chapter discussed the proposed resource utilization based on the resource management policies. The detailed workflow of the Zero Hour policy and TRAM have been analyzed. Further, the mode of operation of resource allocation is discussed in the proposed framework. It has been observed that Zero hour policy helps in reducing the delay, network usage and execution time but increases the communicational and mobility overhead. In the next chapter, the proposed resource scheduling algorithm has been explained, covering all the framework's scheduling aspects.

CHAPTER 4

Resource Scheduling Algorithm

The previous chapter discussed in detail the design and working architecture of the resource management framework. In this chapter, a resource scheduling algorithm has been designed considering the intensity of tasks that are generated by IoT devices.

A resource scheduling algorithm has been proposed which is based on heap based optimization. The algorithm schedules the tasks and assigns resources for execution effectively in a hybrid environment. In the previous chapter, QoS parameter based resource policies have been discussed which is the idea behind the proposed algorithm.
This chapter initially presents the TRAM model for the resource allocation and management and discussed the working of the model through flow chart. Then the resource intensive task scheduling model represents the heap based optimization.Further, Multi constraint based resource allocation is allocated the resources in the markov decision process environment. The statistical analysis of simulation results of the proposed resource scheduling algorithm has been verified to check the system's performance. The implementation of the proposed algorithm is done via the iFogSim toolkit. The simulator selection is made after comparing the six simulators used by other researchers in their study.
This section discuss the dynamic nature of tasks. where thee high priority task executes immediately with task preemption and switch the the low priority task in the waiting state. The proposed algorithm reduce the execution time, latency and network consumption. Further, performing preemption for low priority tasks and assign the resources to high priority ones increases the resource availability.
Finally, the last section discuss the integration of optimal scheduling of clustered tasks using HBO and preemption of tasks dynamically by executing a preference ranking method based on QoS and SLA constraints achieves minimal makespan time and queueing time with a maximal percentage of tasks satisfying the deadline maximal throughput. To check the accuracy of the proposed technique, a comparative analysis is represented to validate the study from various other perspectives similar to the existing frameworks such as NBIHA, MMPA, SJF and FCFS.

4.1 An Efficient Solution for Resource Scheduling Model

Resource scheduling refers to the efficiently assignment of resources to the tasks. Resource scheduling techniques are required to manage the resources at fog layer. All the available resources are identified and assigned as per the tasks requirements. For the efficient scheduling model, the resources allocation should be done so that tasks do not have to wait long. Resource allocation is a method of scheduling and effectively allocating available resources to achieve resource utilization.

4.1.1 Availability of Resources for Allocation

In the proposed resource availability checking algorithm, compute the ranking of each resource available on the fog server. If the request resources are less or equal to the available resources, then the availability value is set to 1. If the value of requested resources is high from the available resources, the value is set to 0. The amount of data is also considered during processing. As represented in Figure 4.1, the resources are available on the fog layer and allocates to the tasks after evaluation.

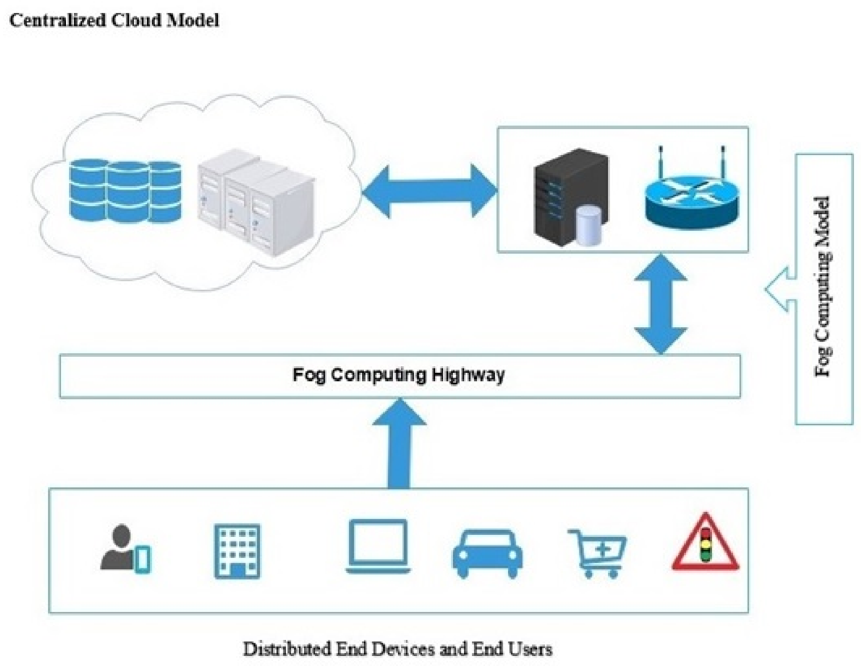


Figure 4.1: Resource Scheduling

Table 4.1: Used Notations

Symbols	Definition
a	Initialization of bandwidth and processing value
AssignResList	List of resource selected for assignment
B	Bandwidth availability
Bi	Bandwidth of every instances
Bm	Minimum Availability of bandwidth
Br	Requested bandwidth
CL	Cloud resources
D	Delay
Es	Start time of task execution
Fd	Fog device
Fs	Fog server
GradeResource	List of graded resources
Ltr	Required Latency
LoR	List of resources
P	Accessible Processing capacity
Pi	An individual instance processing power
Pm	Minimum processing power availability
Pr	Requested required Processing power
Pt	Processing Time(x=cl,fd,fs)
Parameters(Pp,Nb,Rt)	Parameters<Processing power,Network bandwidth and response time
Rr	Requested resources
RS	Store information of all available resources
Tr	Rank value of total resources
User Request List(Pp,Nb,Rt)	Tasks with required Processing power, Network bandwidth and Response time
Us	User task submission time
X	Transferred data volume

Algorithm 2, represents the resource ranking process in the cloud-fog environment. In this algorithm list of resources ***(LoR)*** is prepared with available bandwidth. All the symbols and notations used are presented in Table 4.1.

This algorithm prepares all available resources list which can assign to the users for the processing ***(*P*)*** with bandwidth ***(B)***. Initially, processing value and bandwidth values are set to 0. The minimum available bandwidth and processing of each virtual machine are represented as ***(Bm)*** and ***(Pm)*** in the fog server.

The input of this algorithm is all the available resources and tasks along with processing capacity***(Pp)***, bandwidth***(Nb)*** and latency***(Lt)***. This algorithm's output is to create ***LoR***(list of resources) as per the capacity of resources. The requested processing, latency and bandwidth are represented as ***(Pr)***, ***(Ltr)*** and ***(Br)*** respectively.

If the requested processing value is less than the ***(Pm)***, then the resource is ranked and add into LoR. In such a case where the task's requested processing value is equal to the ***(Pm)*** then calculate the rank by dividing the requested processing by the individual processing power of an instance.

The algorithm 2 executes in two sections. The first section will check the network bandwidth and processing power of available resources. The second portion of an algorithm will prepare the temporary list of resources and check whether the system is in a safe state. When the system is in a safe state, then this resources list assume as the final list of resources.

The algorithm starts with checking available resources in the list and check it are full or not. If the resource list can accommodate new resources then we will check the minimum available processing power. If the value of minimum available processing power is higher processing power available in the resources list then assign a new value for minimum available processing power. After evaluating processing power, the minimum bandwidth compares with the available bandwidth in the resource list. The network bandwidth is revised if the value is high than the available resource list.

In the second section of algorithm, network bandwidth compare with the amount of data required for data transfer, requested processing power compared with the minimum processing capabilities and prepare a list of temporary resources. If the requirement of tasks are completed then this list will consider as final list of resources(LoR).

Algorithm 2 Creating LoR in hybrid environment

```
Input: AssignResList< Pp,Nb,Rt >, a<Pp,Nb>,X, Pr
Output: LoR<Pp,Nb,Rt>
1 Pm←a.Pp Bm ←a.Nb RS [] ← $AvailableResource < Pp, Nb, Rt >$ if $RS[\,].number \neq null$ then
2 for $RS[]$ do
3
      if RS[i].Pp<Pm then Pm=RS[i].Pp
      end if
      if RS[i].Nb<Bm then Bm=R[i].Nb

  for $RS$ [] do
      if P$_m > P_r$ then
      $P_r[i] = RS[i].Pp/Pr$
      else
         Pr[i]=(Pr/RS[i].Pp)
         if B$_m > X$ then
      $B$r[i]=RS[i].Nb/X
    else
      Br[i]=(X/RS[i].Nb)
    if X>RS[i]Nb then
    R$_r[i] = RS[i](x/RS[i].Nb)$
    else
      R$_r[i] = RS[i]Lt$
    T$_r = P_r[i] + B_r[i] + L_t r[i]$
    end if

      LoR ←list of resources as per T$_r$ Check current status of tasks and resources Return
    LoR[ ] else
      Return Null
```

4.1.2 Illustration of TRAM

The next phase after developing the resource list is to provide services from the LoR by considering the demand of the tasks. Resource provisioning will be achieved in a hybrid and hierarchical manner.

The proposed technique reviews the LoR of fog devices and assigns all fog resources. In the unavailability of resources in the fog devices, all tasks will be assigned to the fog server or if the resources are not sufficient on the fog server and fog devices, then the proposed approach will allocate resources and provision them using cloud services. TRAM can provision resources through the cloud and fog server both.

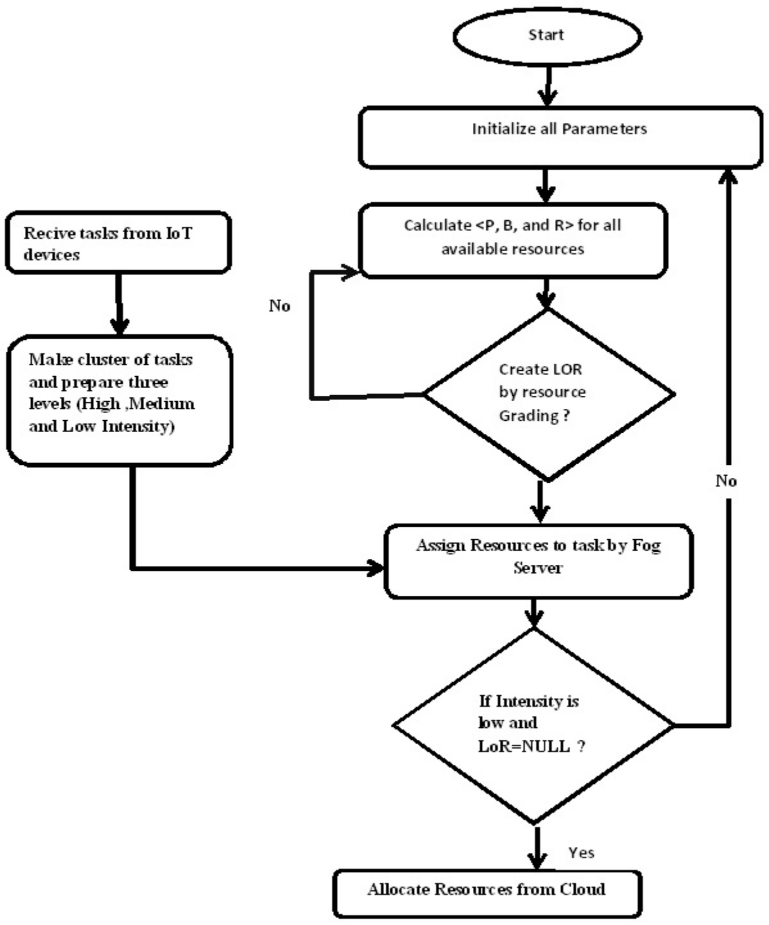

Start
Initialize all Parameters
Recive tasks from IoT devices
Calculate <P, B, and R> for all available resources
No
Make cluster of tasks and prepare three levels (High ,Medium and Low Intensity)
Create LOR by resource Grading ?
No
Assign Resources to task by Fog Server
If Intensity is low and LoR=NULL ?
Yes
Allocate Resources from Cloud

Figure 4.2: Flow Chart of TRAM

If all the above mentioned provisioned fail then a resource unavailability message will be generated. In this procedure, each resource has been identified at its assigned position in LoR. As represented in Figure 4.2, Each resource has three parameters Processing, Bandwidth, Response Time.

The working of fog servers in the TRAM represents in Figure 4.3. All the received tasks are divided into three categories such as high-intensity task, medium intensity task and

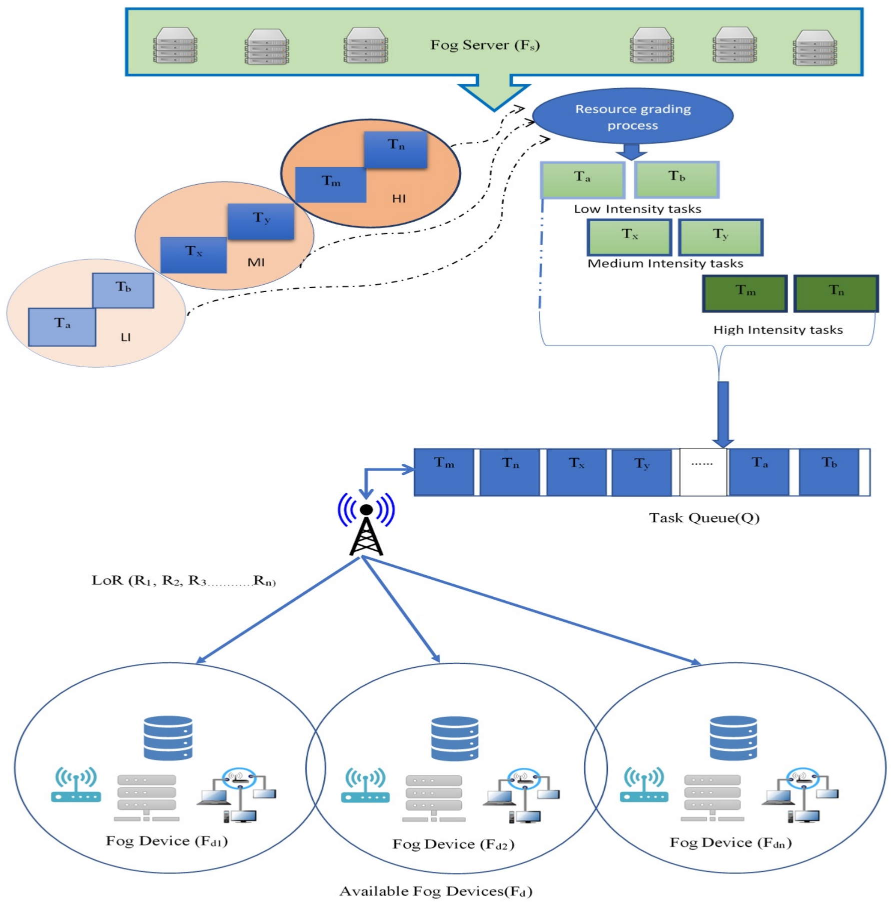


Figure 4.3: Intensity Based Resource Grading and Allocation

low-intensity task by using Expectation Maximization(EM) algorithm [74]. EM algorithm calculates the similarities among IoT tasks. High-intensity tasks are represented as Tm and Tn. Medium intensity tasks are represented as Tx and Ty and low-intensity tasks are represented as Ta and Tb. The EM algorithm is a probability clustering algorithm [74].

This algorithm is implemented on the lower level of the host control node(layer 2 of Figure 3.1). When the tasks are separated based on intensity, it will submit these tasks to the management layer. This management layer works as a distributed decision support and imitates the resource grading process to compute these tasks' requirements for resource allocation. All these tasks arrange in a queue as per their ranking. If the system has more than one high-intensity task, then the arrival time will be considered for the grading process. These tasks allocate to the fog devices, which are part of the existing LoR. This LoR was created by implementing an algorithm 2.

This TRAM has the following steps.

I Initiate each parameter

II Evaluate the value of processing power, response time and bandwidth of all available resources.

III Collect tasks from IoT devices

IV Divide the tasks into three categories High, Medium and Low intensity tasks

V Comparison of available resource with the requested resources processing power, amount of data transfer and latency.

VI Create LoR by the resource grading process

VII Change the minimum available processing capacity, bandwidth availability.

VIII Assign resources to complete the user's request and check that the resources must allocate to the high intensity task first then medium intensity tasks.

IX If the resources for the low intensity tasks are not available then assign these tasks to the cloud for execution

4.2 Optimized Task Scheduling and Preemption

The implementation of task scheduling was done to utilize the resources of the fog nodes. For energy efficient task scheduling, optimization algorithms was used however the delay and load of the tasks were not addressed which resulted in wastage of resources [142] [125]. The complexity of tasks was reduced by demonstrating the tasks in the mapform and scheduled according to the deadline. However, without considering the resource and energy availability, the tasks were directed to the nearest fogs, which directed to rechanneling of tasks [98].

The game theory concept was executed to perform the matching of tasks to the fog nodes by considering the preference list; however this concept did not address the varied deadline of each task [129]. The tasks were classified based on the deadline in various existing works. They scheduled the tasks as per the priority pattern of the tasks which decreased the problem of missing deadlines and improved the efficiency of the process. The dynamic nature of the tasks, on the other hand, was not taken into consideration, which restricted the implementation of these works in real-time circumstances.

However, the tasks dynamic nature was not considered which limited the implementation of the study in real-time scenarios [100] [88]. The machine learning models were used to predict the nature of the upcoming task to perform computations at a very low latency but the dynamic nature of tasks restricted the performance of these works [136] [128].

An application placement technique was executed to schedule resource hungry applications to increase the average response time by using a new application placement approach based on memetic algorithm [95] [132]. Several existing works implemented various techniques such as optimization and learning classifier systems to obtain optimal scheduling of tasks and had some limitations in achieving efficient resource management [115] [140] [122]. The stochastic nature of tasks is to be considered to manage the resources significantly and reduce the response time. Optimized Task Scheduling and Preemption (OSCAR) model is designed to overcome the limitations of network overhead.
The OSCAR model involved the following steps.

- Scheduled the clustered task using Heap Based Optimizer(HBO)
- Resource management in a distributed manner
- Multiple constraint based resource allocation
- Dynamic nature of tasks for preemption

4.2.1 Resource Intensive Task Scheduling

The scheduling of tasks is the primitive process in efficiently managing the fog computing resource in an IoT environment. The task scheduling model ensures the completion of tasks within stipulated time frame with all required resources. The clustered tasks are further scheduled based on the QoS and SLA constraints such as task size(S_T), task type(T_T), task deadline(T_D), task arrival time(T_A), device energy level(D_E), CPU required(C_R), memory required (M_R) and I/O bandwidth required (B_R).

The optimal scheduling of tasks is carried out by executing Heap Based Optimizer (HBO). This process of optimally selecting the task which needs to be processed first improves the efficient management of resources.

4.2.2 Heap Based Optimization

Heap based optimizer used heap based data structure to map the concept of corporate rank hierarchy. The first step of the HBO is initialization.

We initialize the input parameters$(S_T, T_T, T_D, T_A, D_E, C_R, M_R, B_R)$ and population, which is defined as follows,

$$I = \begin{bmatrix} y_1^t \\ y_2^t \\ \vdots \\ y_n^t \end{bmatrix} = \begin{bmatrix} y_1^1 & y_1^2 & y_1^3 & \cdots & y_1^d \\ y_2^1 & y_2^2 & y_2^3 & \cdots & y_2^d \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_n^1 & y_n^2 & y_n^3 & \cdots & y_n^d \end{bmatrix} \tag{4.1}$$

The next step is heap building which includes d-ary tree and can use a ternary heap for task scheduling. Let consider the heap is generated in an array and that is received from the parent node index which is calculated as follows,

$$P(i) = \left\lfloor \frac{i+1}{d} \right\rfloor \tag{4.2}$$

Where, p represents the parent node and represents the floor function that produces a greatest integer value that is less than or equal to the input. The parent node consist maximum of three child nodes which is defined as follows,

$$C(i, j) = i \times d - d + j + 1 \tag{4.3}$$

The depth of the heap tree is defined as follows,

$$D(i) = \lceil \log_d(d \times i - i + 1) \rceil - 1 \tag{4.4}$$

Where D represents the depth and represents the function of ceil that returns the minimum integer that is greater than or equal to the input. The next step is to calculate the colleague, which means the node level that returns the random selection of colleagues. The calculation of colleague is represented as follows,

$$\left[\frac{dd^{D(i-1)}-1}{d-1}+1, \frac{dd^{D(i-1)}-1}{d-1}\right] \tag{4.5}$$

The final process is heapify-up () that searches upward and detects the correct location to insert the new node. Figure 4.4 depicts the task scheduling process using HBO.

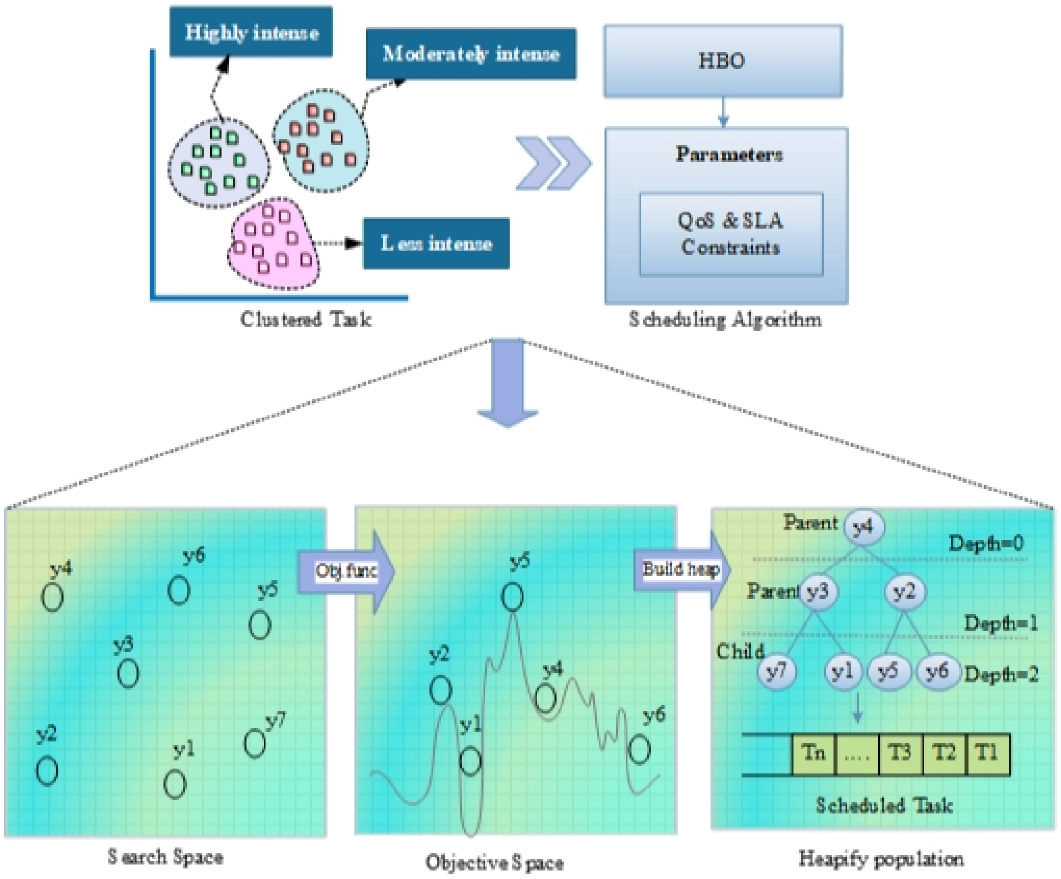


Figure 4.4: Task Scheduling

Finally, the input task is scheduled by following the above mentioned step.

Algorithm 3 represents the process of resource intensive task scheduling. The tasks are scheduled based on the parameters(S_T),(T_T),(T_D),(T_A),(D_E),(C_R), (M_R) and (B_R) which are considered for every task in the environment. Here, $P(i)$ represents the parent index

Algorithm 3 Resource Intensive Task Scheduling

```
   INPUT: Task (t) and population (I)
   OUTPUT: Scheduled task
   Begin
   Initialize population
   for each task (t) do
   Compute S$_T ComputeT_T$
Compute T$_D ComputeT_A$
Compute D$_E ComputeC_R$
Compute M$_R ComputeB_R$


while (iRoot and heap[i].Key<Heapt[P(i).Key) do
SWAP (heap[i],heap[parent(i)])
i←P(i)
end while
for  i←n do
Heap [i].Value←i
Heap [i].key←g(x)
Heapify up ()
end for
for r i← $n$ do$(downwardsto2)$
i←Heap(i).Value
P(i)←Heap[P(i).Value]
C(i)=Heap[colleague(i).Value
Position←y$_(P(i))RC_{,}y_(C(i))$
   for j←1 to D  do
P ← $Random()$
y$^k_{temp}$ ← $updatey^k_i(t)$
end for

      if  F(y$_{temp}$) < $F(\vec{y}_i(t))$ then$\vec{y}_i(t+1) \leftarrow \vec{y}_i(t)$
end
Heapify up()
end
end
return y(Heap[1].Value)
```

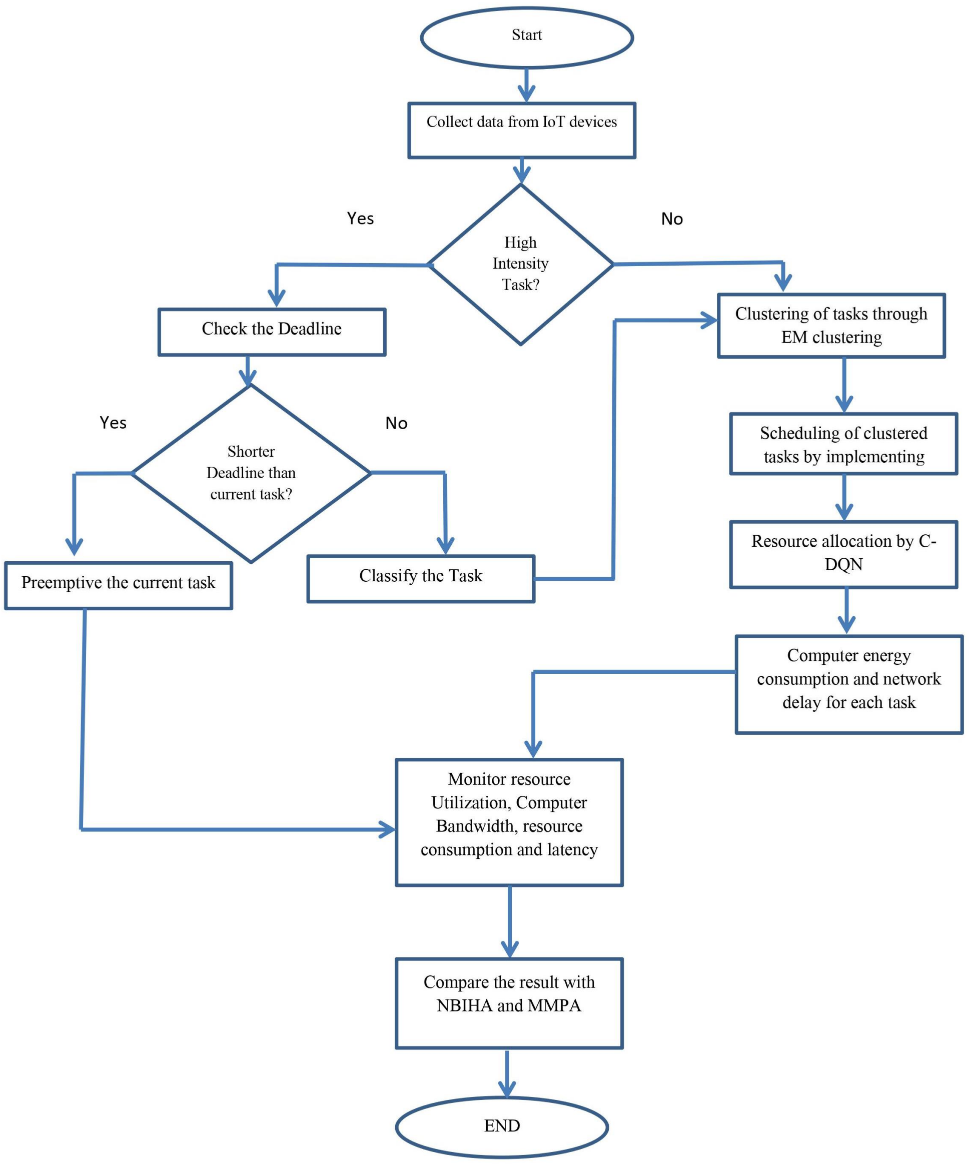

Start
Collect data from IoT devices
High Intensity Task?
Yes
No
Check the Deadline
Clustering of tasks through EM clustering
Shorter Deadline than current task?
Yes
No
Scheduling of clustered tasks by implementing
Preemptive the current task
Classify the Task
Resource allocation by C-DQN
Computer energy consumption and network delay for each task
Monitor resource Utilization, Computer Bandwidth, resource consumption and latency
Compare the result with NBIHA and MMPA
END

Figure 4.5: Resource Allocation Flow

and $C(i)$ represent the random colleague index and RC represent the position of a random colleague.

4.3 Multi Constraint Based Resource Allocation

Resource allocation is the process of tendering the available resources to compute the scheduled tasks. The allocation of resources is based on the parameters such as latency, network delay, bandwidth, resource consumption, execution time, resource availability, energy and number of tasks.

The DQN performs the allocation of resources in the markov decision process(MDP) environment. Once the resource allocation process is executed, the tasks scheduled in the prior process are computed with the available resources. The tasks are allocated based on the resources such as a load of the running task, makespan time of the required task, energy level of the node, resource availability etc.
Markov decision process is a generic mathematical problem that describes the optimal route subsequent decision. The agent (actor) decides on an action and moves to the next stage at each step of the sequence. Some rewards are accessible to get either positive or negative, depending on the present condition. A MDP is defined by:

- A set of states s ϵ S
- A set of action a ϵ A
- A transition function T(S,a,S') which generates a probability of landing from S to S' and took action a, i.e P(S'|S,a)
- A reward function R(S,a,S') which can represent as R(S) or R(S')

MDP has a Policies Solution which describe a strategy and rule specifying what action to adopt in each scenario.

The tasks are allocated by considering the status of the fog nodes, through this process the resources of the fog nodes are managed efficiently. Figure 4.5 illustrates the overall flow of the resource allocation process. The set of fog resources available in the fog nodes can be represented as follows

$$R = [r_1, r_2, r_3 \dots r_m] \tag{4.6}$$

For each fog node, resource constraints are represented as follows

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{bmatrix} \tag{4.7}$$

However, resource allocation and management usually manage the allocation and deallocation of computing and storage resources. Further, selecting the fog for the given task or suitable match of the resource allocation or the set of nodes is selected to distribute resources to fulfill the users' tasks QoS / application service requirement.

4.3.1 Deep Reinforcement Learning Model

The proposed categorical DQN provides the absolute solution for the incoming tasks by experience replay, freeze target Q-network, and clip rewards in the sensible range that are detailed in the following

- **Experience Replay:** Based on the agents experience, correlations are removed and take action a_t, store transition $s_t, a_t, r_t, s_(t+1)$ in replay memory D, sample random mini-batch of transitions $(s, a, r, s^{')}$ from replay memory D. Optimize MSE between Q network and Q-learning targets,

$$\begin{aligned} L_i(\theta_i) = E_{(a,s,r,s')\sim U(D)} \Big\{ r + \gamma \max_{a'} Q\left(a', s', \theta_i^-\right) \\ -Q(a, s, \theta_i)\}^2 \end{aligned} \tag{4.8}$$

- **Freeze Target Q-Network:** Q-learning target is going to fix for avoiding the oscillations and calculate the Q-learning target with respect to the previous set of parameters as follows,

$$r + \gamma \max_{a'} Q\left(s', a', \theta_i^-\right) \tag{4.9}$$

Optimize the performance of the MSE between the Q-learning agents and network by follows,

$$L_i(\theta_i) = E_{(a,s,r,s')\sim U(D)}\Big\{r + \gamma \max_{a'} Q\left(a', s', \theta_i^-\right) \\ -Q(a, s, \theta_i)\}^{2'} \tag{4.10}$$

Then periodically update the fixed parameters.

- **Clip Rewards in Sensible Range:**
  Based on the reward/value range, DQN clips the reward to -1 and +1, preventing too large Q-values. Ensure the gradient for well-conditioned. Based on the categorical DQN, the fog resources are allocated to the respective tasks as follows,

  - Utilization / amount of resources free
  - Distance / location of fog nodes
  - Response time / execution time / migration time
  - CPU, and memory availability / fluctuation behavior Energy availability of fog nodes

  Therefore, resource availability for executing the task can be disguised by follows,

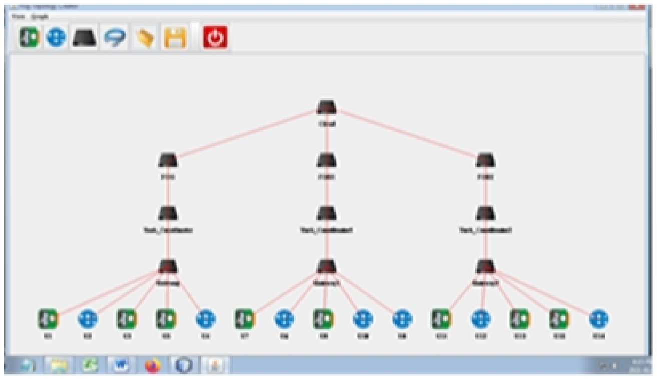
$$a^e = \left(MP - \left(r^l \times d^e\right) / MP\right) \tag{4.11}$$

Where a^e represents the resource availability, MP represents the measurement period, r^l represents the likelihood of resource loss for the given time period and d^e is the expected downtime from the resource of the loss.

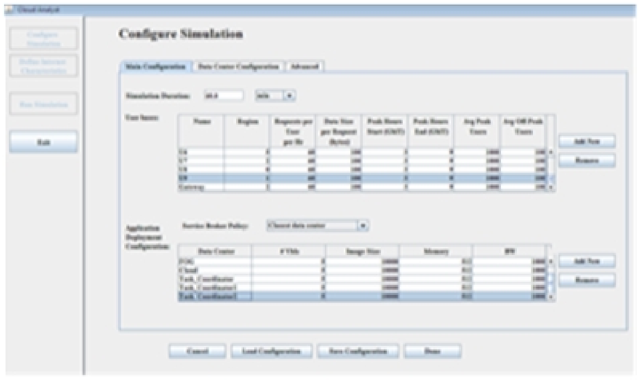
4.4 Dynamic Task Preemptive Scheduling

The increased number of devices in the IoT environment exponentially increases the dynamic nature of the tasks. The task preemption is an effective process in dynamically adapting towards the computation demand of the tasks.

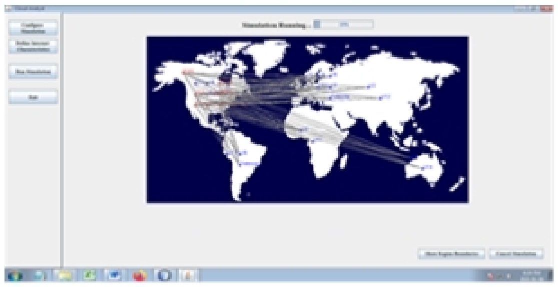
If the fog node having certain resources is computing a low priority task with a longer deadline and the high priority task that needs to be executed immediately arises, task preemption is used to enter the low priority task in the waiting state and implement the



(a) A



(b) B



(c) C

Figure 4.6: a) Fog Topology b) Configure the Simulation c) Running Simulation Cost Analysis of Different Workflows

high priority task. However, preemptive tasks can be interrupted during execution so that can be stopped during the execution. For instance, in the Web Crawler application, when the preemptive tasks resources have collided, tasks are reassigned to adjacent resources from collided point, whereas in the non-preemptive tasks are uninterrupted and cannot be dismissed during their execution as Encoding, Rendering, Batch Processing and Continuous Integration.

When the resources of non-preemptive tasks are depleted, then these tasks are reassigned from the start. This significantly reduces the average response time of the tasks. The dynamic task preemption is carried out using a preference ranking method, a multi-criteria decision-making approach. In decision making, priority is decided by the fog node and computed by the fairness, deadline and optimized response time. Furthermore, the study is focusing on the challenge of task granularity. This way of executing the high intense task in between the low intense task is an efficient way of managing the resources. Moreover, errors in high priority tasks implementation require lots of computation and high value of resources.

Hence performing preemption for low priority tasks and assign the resources to high priority ones increases the resource availability and reduces the task failure rates.

4.5 Comparative Analysis

In this sub-section, the OSCAR model is evaluated by comparing with several existing approaches such as MMPA (Energy-Aware Marine Predators Algorithm for task scheduling in IoT-based fog computing Applications) [113] and NBIHA (A Novel Bio-Inspired Hybrid Algorithm (NBIHA) for Efficient Resource Management in fog computing) [103] in terms of performance metrics such as, average response time, loss ratio, resource utilization, average makespan time, queueing waiting time, percentage of tasks satisfying the deadline and throughput.

4.5.1 Impact of Average Response Time

The response time is referred to as the measure of time taken to respond to a particular task. The response time can be computed as the summation of waiting time and service time. In Figure 4.7 the comparison of the average response time of the proposed OSCAR model and other existing models with respect to the number of tasks is provided. The average response time increases with an increase in the number of tasks.

Table 4.2: Simulation Parameters

iFogSim Configuration	Network Topology	IoT Devices	15
		Gateways	3
		Task Coordinators	3
		Fog Nodes	3
		Cloud Server	1
	Application Module	Memory	10 MB
		Bandwidth	1000KBs
		MPS	1000
		Capacity of RAM	10
	Fog Node	RAM (GB)	16
		Resource Cost	3.0
		MIPS	2800
		Memory Cost	0.05
		Bandwidth	100000
		Storage Capacity	11TB
		Storage Cost	0.001
	Cloud Server	RAM(GB)	40
		MIPS	44800
		Delay	1000ms and More
		Bandwidth	100000
	IoT Devices	Delay	1 ms
		RAM(GB)	4
		MIPS	1500

Table 4.3: Analysis of Average Response Time

Technique	No. of Task
MMPA	45.5 ± 5
NBIHA	25±3
OSCAR	9.75±2

The average response time of our proposed model is low than other models due to the proper allocation of resources and preemption of tasks. The resource allocation is

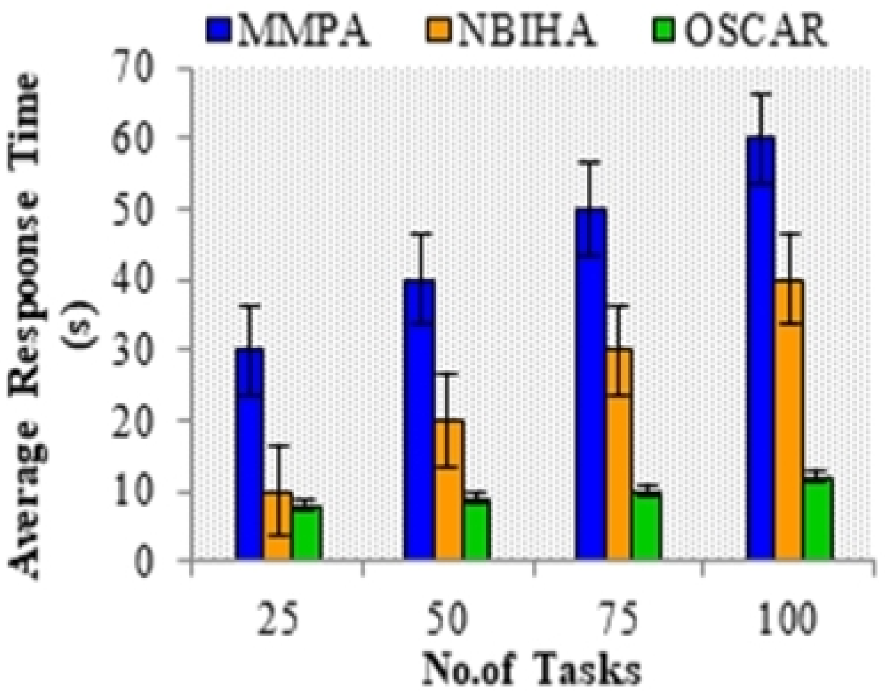


Figure 4.7: Number of Tasks vs. Average Response Time

performed by using C-DQN based on the factors such as latency, network delay, execution time, bandwidth, resource consumption, energy consumption, number of tasks and resource availability and the status of current tasks. Further, the preemption of tasks provides execution of tasks in a dynamic manner. The existing approaches lack in the adoption of preemption of tasks which increases the response time of those approaches. Table 4.3 presents the numerical analysis of the response time of the proposed model and other existing approaches with respect to the number of tasks. The average response time of our model is found to be 9.75 seconds whereas the existing approaches possess up to 45.5 seconds of average response time which is five times greater than the proposed work. From this, we can conclude that our proposed model is efficient to achieve the QoS and SLA of task completion.

4.5.2 Impact of Load Ratio

The load ratio is referred as the ratio of load occupied by the VMs in the fog layer. The load ratio is influenced by the proper allocation of resources in the VMs. The load ratio of our proposed OSCAR model and existing approaches with respect to the number of tasks is presented in Figure 4.8.

Table 4.4: Analysis of Load Ratio

Technique	No. of Task
MMPA	96.5 ± 4
NBIHA	94.5±3
OSCAR	92.5±2

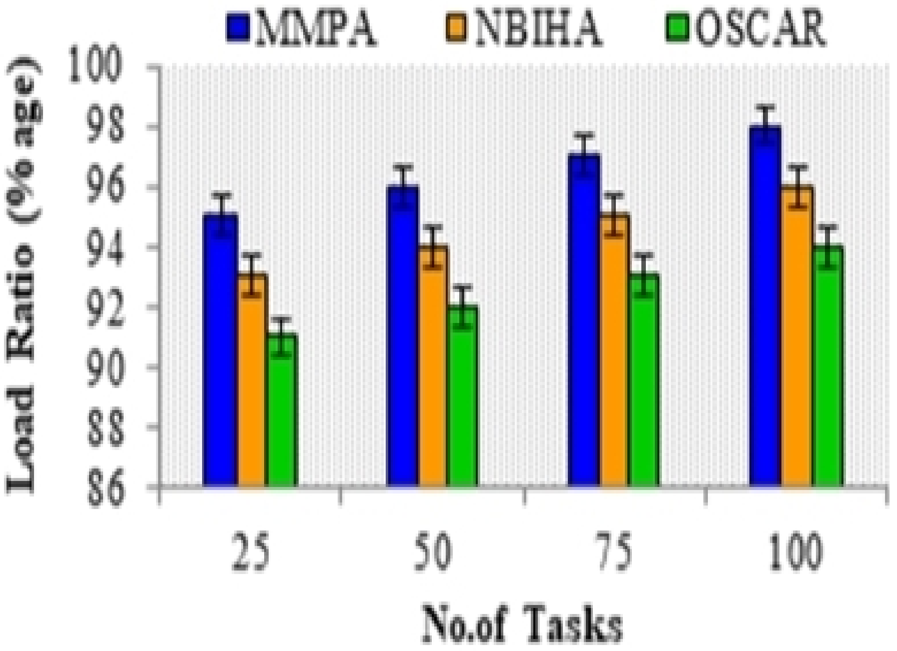


Figure 4.8: Number of Tasks vs. Load Ratio

The load ratio increases with an increase in the number of tasks. The load ratio of our proposed model is lower than the existing models due to the proper allocation of resources carried out by the C-DQN model based on several significant characteristics of tasks. The existing approaches lack consideration of task characteristics to allocate resources, thereby increasing load ratio effectively. The Table 4.4 provides the numerical analysis of the load ratio of the proposed model and existing approaches with respect to number of tasks. The load ratio of our OSCAR model is found to be 92.5% whereas the existing approaches possess about 96.5% of load which affects the execution of tasks resulting in increased delay.

4.5.3 Impact of Resource Utilization

Resource utilization means the utilization of available resources for the computation of tasks. The higher the utilization of resources the more efficient the approach will be. The comparison of resource utilization of our proposed OSCAR model and existing approaches in relation to the number of fog nodes is presented in Figure 4.9.

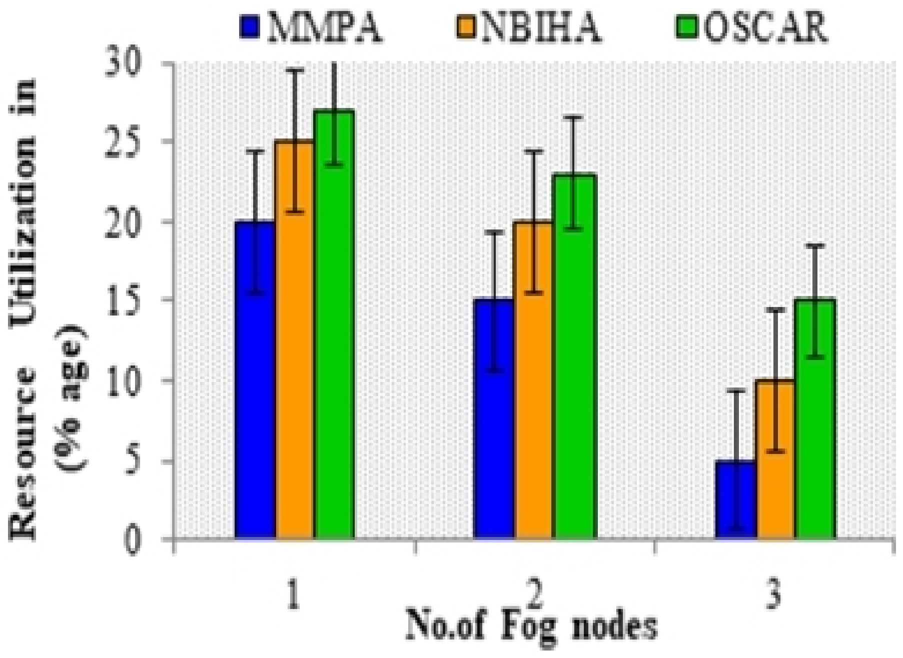


Figure 4.9: Number of Fog Nodes vs. Resource Utilization

Table 4.5: Analysis of Resource Utilization

Technique	No. of Task
MMPA	13.4 ±3
NBIHA	18.5±2
OSCAR	21.6±1

The distribution of tasks due to an increase in the number of fog nodes reduces the utilization of resources. The proposed model's resource utilization is higher than the existing approaches due to the proper management of resources in the VMs by using the C-DQN model. The existing approaches lack proper allocation of resources resulting in reduced utilization of resources. The numerical comparison of resource utilization of our proposed approach and existing approaches are presented in Table 4.5. The proposed work possesses 21.6% resource utilization per fog node whereas the existing approaches have only 18.5% of resource utilization. This study concludes that the proposed work is efficient in utilizing the resource resulting in reduced response time.

4.5.4 Impact of Average Makespan Time

The makespan time means the total time required for the completion time. It differs from the response time in which the transmission time is not considered. The makespan time is the summation of transmission time, system time and waiting time. The comparison of makespan time of the proposed approach and existing approaches is presented in Figure 4.10.

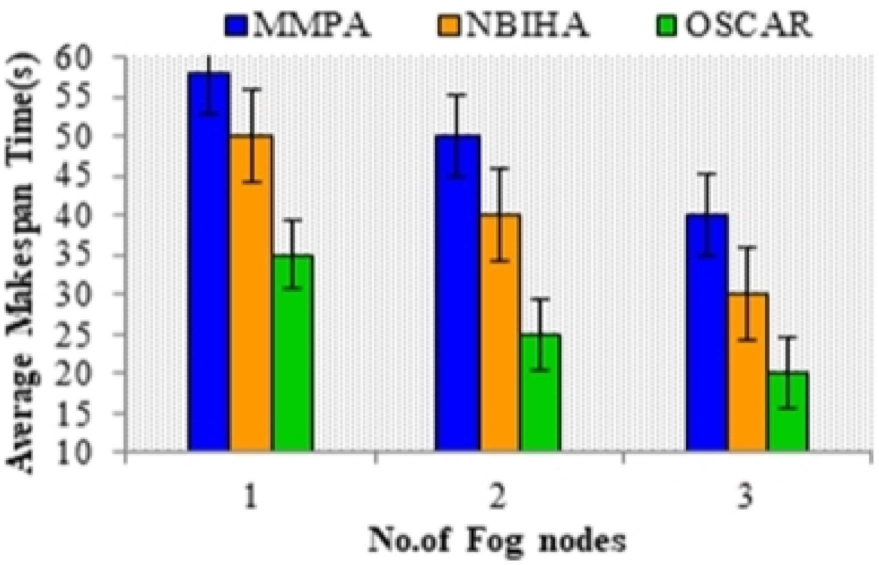


Figure 4.10: Number of Fog Nodes vs. Average Makespan Time

Table 4.6: Analysis of Average Makespan Time

Technique	No. of Fog nodes
MMPA	49.6 ± 4
NBIHA	40.3±3
OSCAR	26.6±2

The makespan time of our proposed OSCAR model is high due to the implementation of preference rank method based task preemption in which the completion of both low intensity and high intensity tasks is ensured within the deadline. The existing approaches lack in the proper execution of tasks within the deadline which affects the completion of tasks resulting in increased makespan time. Table 4.6 presents the numerical analysis of the makespan time of the proposed approach and other existing approaches in relation to the number of fog nodes. The makespan time of our proposed work is found to be 26.6 seconds whereas the existing approaches possess makespan time up to 49.6 seconds from this we can conclude that the proposed work outperforms the existing approaches in completing the tasks with reduced makespan time.

4.5.5 Impact of Queueing Time

The queueing time refers to the measure of time the task is placed in a queue before executing it. The queue time of the task must be less than that of the deadline to complete the task successfully. The queueing time of the proposed OSCAR model without preemption is compared with the existing approaches in relation to fog nodes is presented in Figure 4.11.

It is found that the queueing time of the proposed model without the preemption of tasks is high than the existing approaches due to the scheduling of tasks by implementing HBO based on the QoS and SLA constraints.

The existing approaches also implemented optimization based task scheduling but the

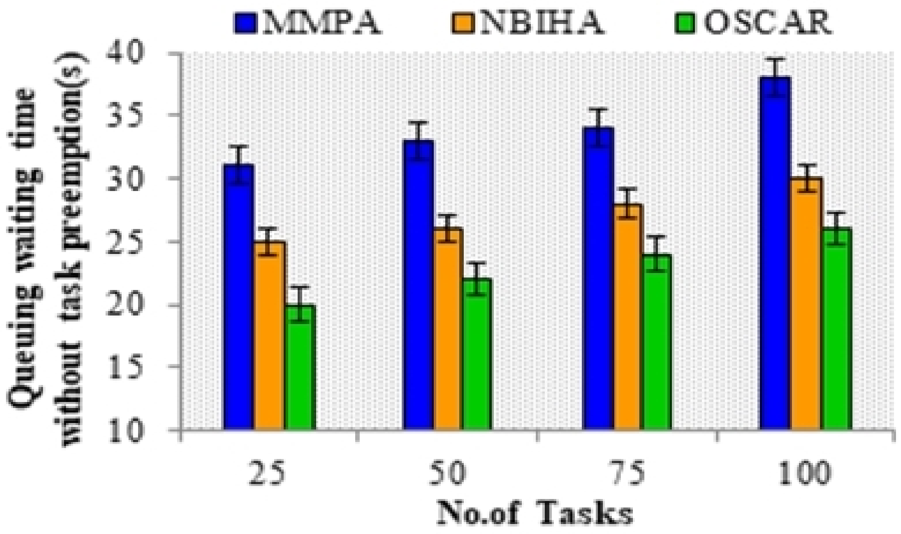


Figure 4.11: Queueing Time Without Task Preemption

Table 4.7: Analysis of Queueing Time Without Task Preemption

Technique	No. of Fog nodes
MMPA	34 ± 4
NBIHA	27.25±3
OSCAR	23.1±2

convergence of those approaches was not effective than the proposed approaches resulting in increased queueing time. Further, the clustering of tasks based on intensity initially reduces the complexity which supports the reduced queueing time of the proposed approach. The numerical analysis of queueing time of the proposed work without preemption is compared with that of existing approaches with respect to the number of fog nodes is presented in Table 4.7.

The queueing time without preemption was 23.1 seconds whereas the existing approaches possess high queueing time up to 34 seconds. This proves that even without preemption the proposed approach performs better than that of the existing approaches. The queueing time of the proposed approach with preemption is compared with the existing approaches in relation to the number of fog nodes is presented in Figure 4.11

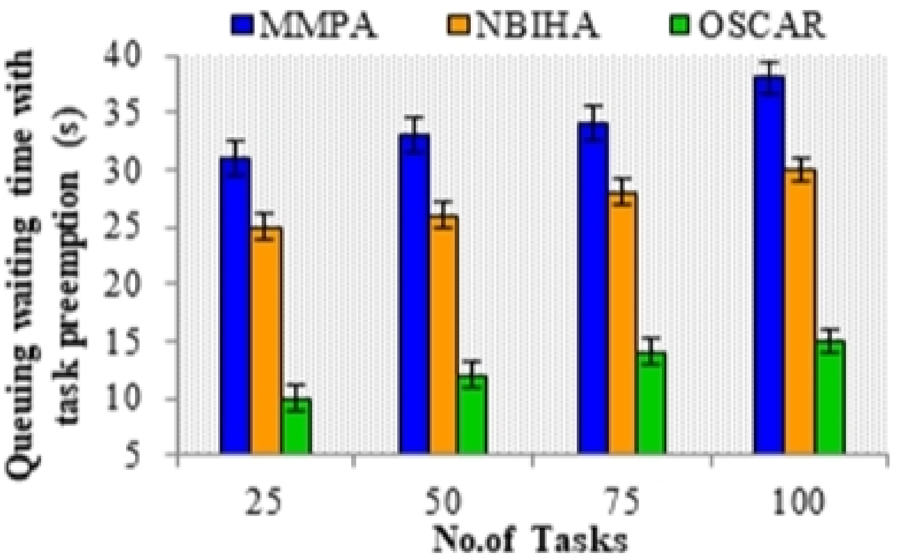


Figure 4.12: Queueing Time With Task Preemption

The queueing time of our proposed OSCAR model with task preemption is very low than

Table 4.8: Analysis of Queueing Time With Task Preemption

Technique	No. of Fog nodes
MMPA	34 ± 4
NBIHA	27.25±3
OSCAR	12.75±2

the existing approaches due to the execution of dynamic switching from low intense task having a longer deadline to the newly obtained high intense task having a shorter deadline.

By doing both the low intense and high intense tasks are completed resulting in an increase in the percent of tasks dissatisfied the deadline. The existing approaches lack consideration of the dynamic nature of incoming tasks from the IoT devices resulting in increased queue time. Table 4.8 presents the numerical comparison of the proposed OSCAR model with preemption and existing approaches in relation to a number of fog nodes. The queueing time of our approach with task preemption is found to be 12.75 seconds and the queueing time of our approach without queueing time is 23.1 seconds from this we can say that by performing preemption of tasks the queueing time of tasks can be reduced.

4.5.6 Impact of Task Satisfying the Deadline

This metric is used to measure the number of tasks completed successfully within the deadline. The proposed OSCAR model is compared with existing approaches in terms of tasks satisfying the deadline with respect to a number of fog nodes, presented in Figure 4.13

Table 4.9: Analysis of Task Satisfies the Deadline

Technique	No. of Fog nodes
MMPA	85.75± 3
NBIHA	92±2
OSCAR	96.75±1

The task satisfies the deadline of the proposed approach which is higher than other existing approaches due to the scheduling of tasks and dynamically preempting the scheduled tasks. The existing approaches perform any of these processes, affecting the proper allocation of tasks, thereby possessing more failed tasks.

The numerical analysis of tasks satisfying the deadline of the proposed model and the existing approaches in relation with the number of fog nodes is presented in Table 4.9. The

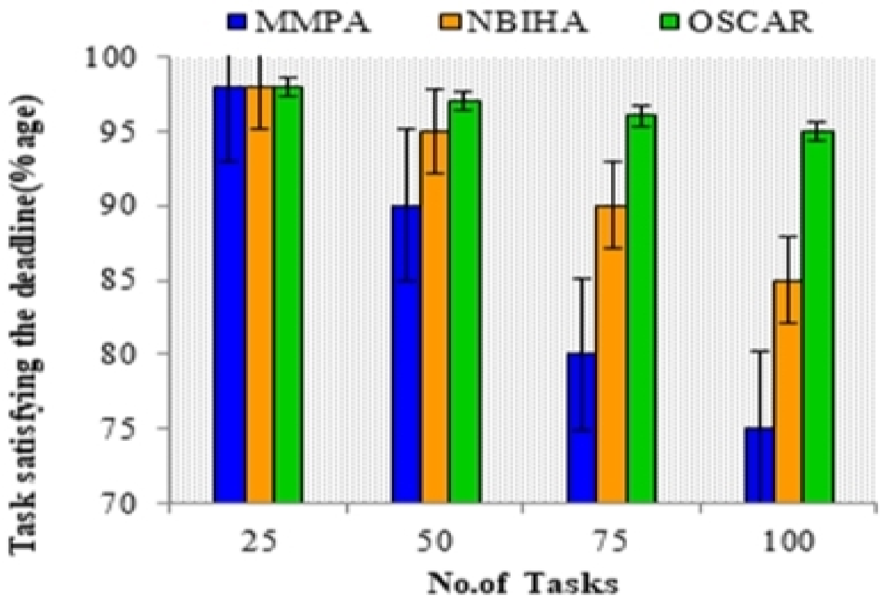


Figure 4.13: No. of Fog Nodes vs. Tasks Satisfying the Deadline

average number of tasks satisfying the deadline by implemented our proposed approach is 96.75% whereas the existing approaches possess up to 92%. The lack of consideration of the dynamic nature of incoming tasks increases the failure rate of tasks.

4.5.7 Impact of Throughput

The throughput is defined as the measure of the completion rate of a task by an approach within a specific time period. The throughput of the proposed OSCAR model and other existing approaches with respect to the number of fog nodes is illustrated in Figure 4.14 The proposed OSCAR model seems to have high throughput than the existing models due to the integration of both task scheduling and resource allocation. The task coordinator schedules the tasks based on intensity and the fog nodes allocate the resources required for the computation of tasks. By doing so, the utilization of resources becomes high thereby achieving maximum throughput. The existing approaches performed either of these processes which affect the execution of tasks within the deadline.

Table 4.10 presents the numerical analysis of throughput of the proposed approach and the existing approaches in relation to the number of fog nodes. The throughput of our

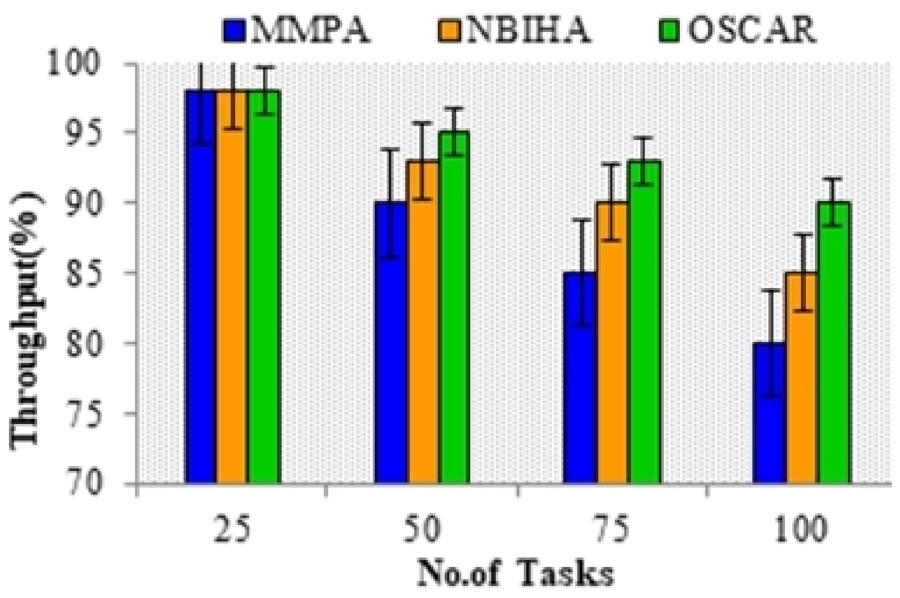


Figure 4.14: No of Fog Nodes vs. Throughput (Kbps)

Table 4.10: Analysis of Throughput

Technique	No. of Fog nodes
MMPA	88.75± 3
NBIHA	91.5±2
OSCAR	94.3±1

proposed approach is found to be 94.3% whereas the existing approaches possess up to 91.5%. It is concluded from the study that our proposed OSCAR model is efficient in completing more tasks within the simulation period.

4.6 Test Cases for Evaluation

This work provides four test cases to represents the experimental results of evaluated tasks. The considered tasks data set [134] are run on the iFogSim to evaluated execution time, cost, energy consumption and response time in the fog environment. The dataset describes the requirements of future crowd-based IoT applications. This dataset is developed for scheduling/rescheduling/monitoring algorithms to handle massively dynamic situations in crowd-based resource allocation. For simulation results, iFogSim has been checked the performance of the proposed technique. The proposed OSCAR technique is a combined form of the HBO and C-DQN model. The proposed technique's results are compared with existing techniques to verify the performance of the OSCAR model.

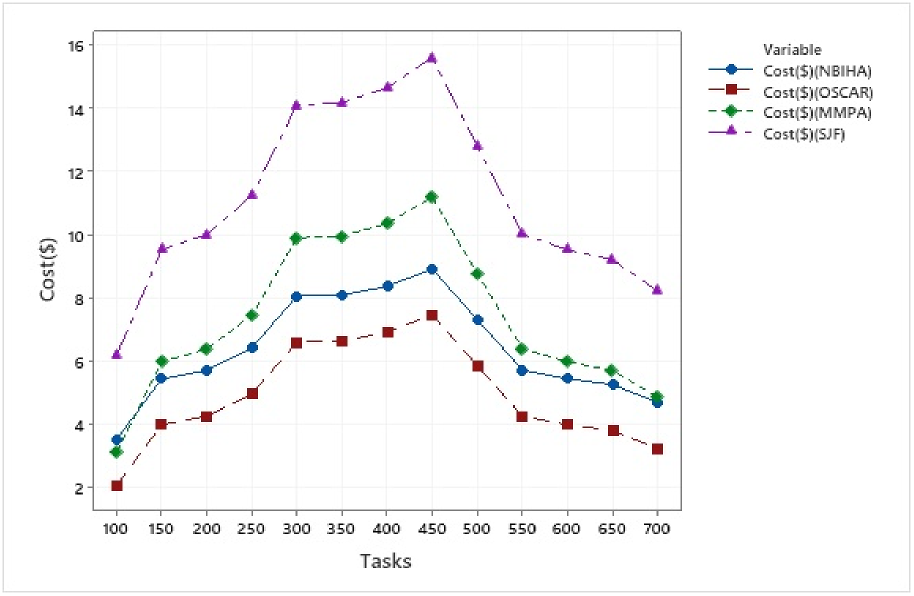


Figure 4.15: Analysis of Cost

4.6.1 Test case I: Cost Analysis

Different tasks are assigned to the fog nodes and their performance is analyzed. This work has considered 700 tasks for execution and compared the existing approaches with the proposed model.
In Figure 4.15 shows the number of tasks on the X-axis and the cost on the Y-axis. The results represent that with the increase of tasks, the implementation cost will also be increased. The proposed model reduced the cost and compared with the other techniques i.e., NBIHA, MMPA and SJF. The OSCAR model reduces implementation costs by approximately 29

4.6.2 Test Case II: Execution Time of Tasks

Execution time increases when a large number of tasks are parsed to the fog nodes. The fog layer required more resources to execute a large number of tasks. The proposed approach has been implemented and analyzed for all the tasks in the fog layer. It represents the effect of increasing the number of tasks while keeping the constant number of fog nodes. In this experiment, 700 tasks are executed with a varying number of resources. Figure 4.16 represents the execution time which is calculated by the time taken from the submission of tasks till execution is completed. The X-axis shows the number of tasks and the y-axis represents execution time in the graph. The results indicate that with increasing the number of tasks, the execution time decreases as compared to the other existing techniques.x The Figure4.16 shows that the execution time decrease by 33% in the same proportion as increase the number of tasks. This observation indicates that OSCAR has given good performance in comparison to NBIHA, SJF and MMPA.

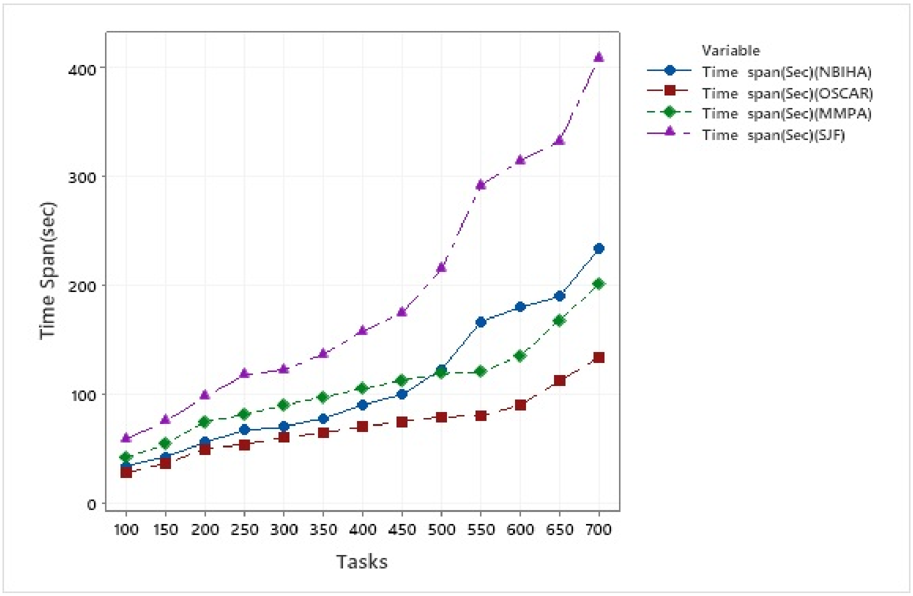


Figure 4.16: Analysis of Execution Time

4.6.3 Test Case III: Energy Consumption of Tasks

In this case, different tasks are analzed to compute energy consumption in the fog layer. When the tasks increases, energy consumption will also be increased. The Figure 4.17 represents the energy consumption in the fog layer, where the x-axis representing the number of tasks and energy consumption on the y-axis. The proposed approach reduced the energy consumption and compared the results with already existing techniques. OSCAR tries to reduce energy consumption by 7% .

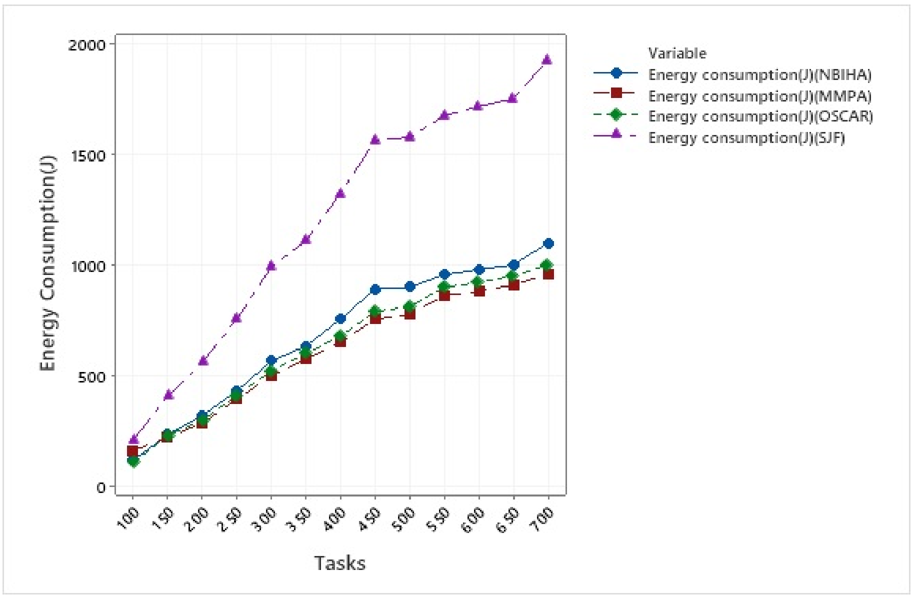


Figure 4.17: Analysis of Energy Consumption

4.6.4 Test Case IV: Response Time Analysis of Tasks

The response time is calculated when the task sends a request for resources allocated to the task as per the requirement. The proposed model mainly works to reduce the response time. The Figure 4.18 represents the response time analysis for 700 tasks. In this figure, the x-axis represents the tasks and y-axis represents the response time. The results are compared with the existing approaches such as MMPA, NBIHA and SJF. In the proposed approach, response time is reduced by 39% as compared to the MMPA approach.

4.7 Statistical Analysis

Resource utilization and scheduling framework can be verified by a formal language like paired t test. In the paired t test, a comparison of mean is done when the observation have been obtained in pairs. Paired T test is used to testing two sample means when their respective population and standard deviation are unknown. The paired sample t-test hypotheses are describe below:

- The null hypothesis (H_0) assumes that the mean energy consumption, response time and execution time are same. There is no difference in the mean (μ_d) and it is equal to zero.

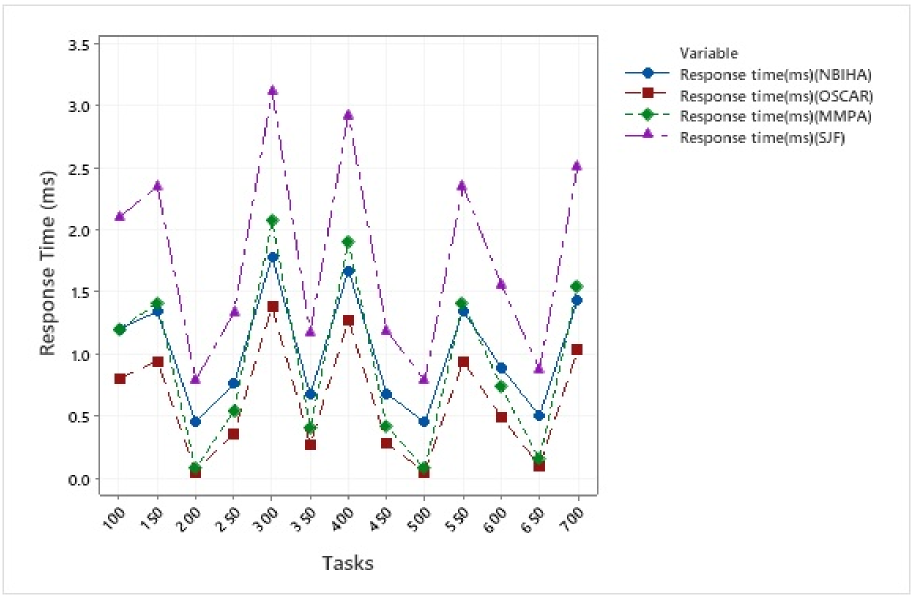


Figure 4.18: Analysis of Response Time

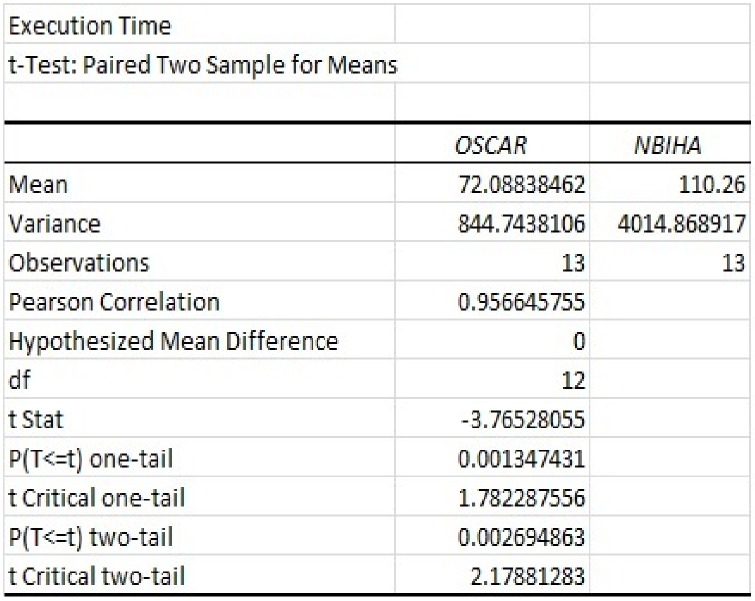
- The two tailed alternation hypothesis (H_1) assumes that mean difference (μ_d) is not equal to zero.

The purpose of paired t test to find out the statistical evidence that the means difference between paired observation is significantly different from zero. In the table, the obtained p value to the given test t is less than α.

The p value for the energy consumption is $p < 0.05$ represent that mean difference between OSCAR and NBIHA approach is statistically significantly different from zero. The obtained values revealed that the energy consumption, response time and execution time in the NBIHA is greater than the proposed OSCAR technique. In Figure 4.19, P(T<=t) two tail(0.002694) gives the probability that the absolute value of the t-statistic(-3.765) is less than the critical t values (2.178). Since the p- value is less than alpha 0.05, the null hypothesis is rejected that there is no significant difference in the sample means.

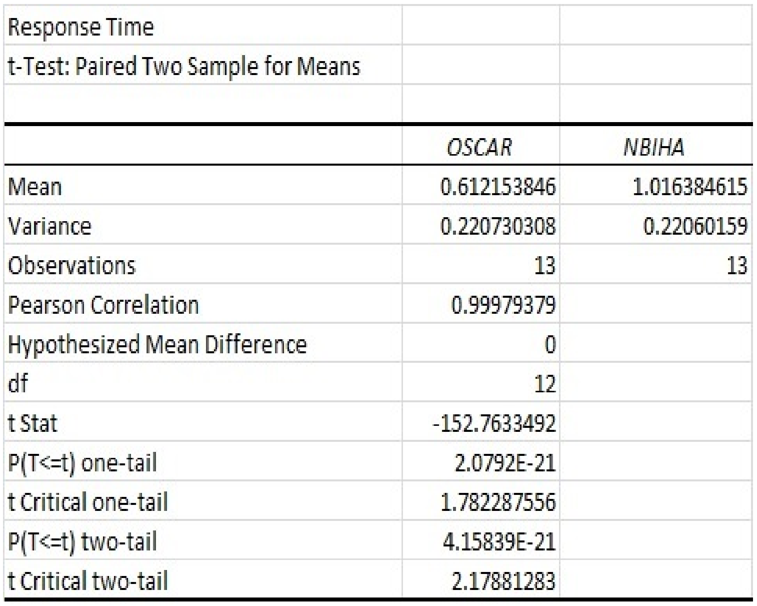
In Figure 4.20, P(T<=t) two tail(4.15839E-21) gives the probability that the absolute value of the t-statistic(-152.76) is less than the critical t values. Since the p- value is less than alpha 0.05, the null hypothesis is rejected that there is no significant difference in the sample means.

In Figure 4.21, P(T<=t) two tail(0.000123) gives the probability that the absolute value of the t-statistic(-5.56619) is less than the critical t values (2.178). Since the p- value is less than alpha 0.05, the null hypothesis is rejected that there is no significant difference in the



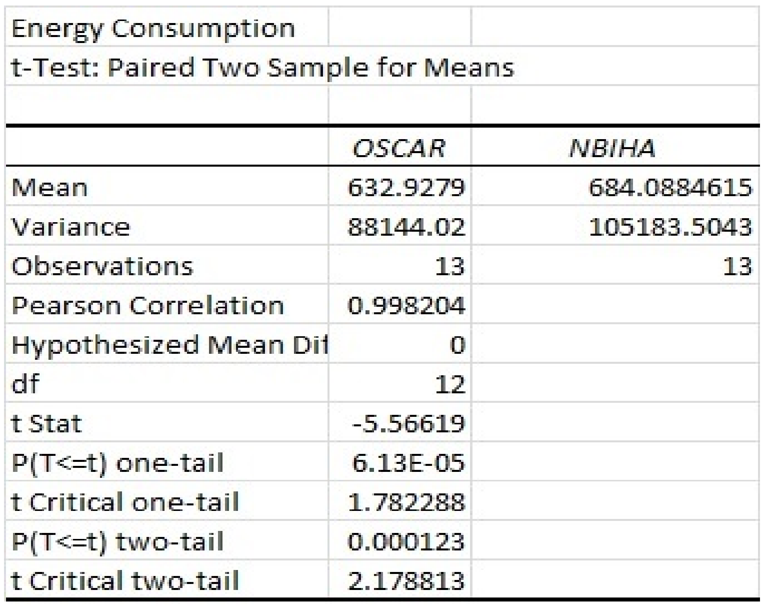
Execution Time		
t-Test: Paired Two Sample for Means		
	OSCAR	*NBIHA*
Mean	72.08838462	110.26
Variance	844.7438106	4014.868917
Observations	13	13
Pearson Correlation	0.956645755	
Hypothesized Mean Difference	0	
df	12	
t Stat	-3.76528055	
P(T<=t) one-tail	0.001347431	
t Critical one-tail	1.782287556	
P(T<=t) two-tail	0.002694863	
t Critical two-tail	2.17881283	

Figure 4.19: Paired t-test for Execution Time



Response Time		
t-Test: Paired Two Sample for Means		
	OSCAR	*NBIHA*
Mean	0.612153846	1.016384615
Variance	0.220730308	0.22060159
Observations	13	13
Pearson Correlation	0.99979379	
Hypothesized Mean Difference	0	
df	12	
t Stat	-152.7633492	
P(T<=t) one-tail	2.0792E-21	
t Critical one-tail	1.782287556	
P(T<=t) two-tail	4.15839E-21	
t Critical two-tail	2.17881283	

Figure 4.20: Paired t-test for Response Time



Energy Consumption		
t-Test: Paired Two Sample for Means		
	OSCAR	*NBIHA*
Mean	632.9279	684.0884615
Variance	88144.02	105183.5043
Observations	13	13
Pearson Correlation	0.998204	
Hypothesized Mean Dif	0	
df	12	
t Stat	-5.56619	
P(T<=t) one-tail	6.13E-05	
t Critical one-tail	1.782288	
P(T<=t) two-tail	0.000123	
t Critical two-tail	2.178813	

Figure 4.21: Paired t-test for Energy Consumption

sample means.

4.8 Summary

This chapter represented the resource scheduling model and the resource allocation based algorithm. A detail description about the proposed model provided through the flow chart. A Heap Based Optimization resource scheduling algorithm has been proposed to map tasks on the resources to ensure the resource intensive task scheduling. Further, the dynamic behaviour of tasks has analyzed and allocated resources to ensure high utilization. The next chapter discusses the verification and validation of the proposed model and resource scheduling algorithm.

CHAPTER 5

Conclusion and Future Direction

The main contribution of the thesis are concluded in this chapter. A detailed study is done in the area of fog computing along with its application, key issues, architecture and research challenges. Fog computing provides services to the end devices and reduce the burden of cloud. By deployment of its nodes near to network edge, fog computing has resolved the main issue of delay which is faced by cloud computing. There are number of other issues such as resource management and task scheduling at the fog layer which can needs to be handled with some efficient techniques. Various existing approaches have been studied and their literature review have been provided. To resolve the problem of resource management in fog computing environment , Tri-fold Task Clustering framework has been designed.

Further, the proposed framework has been designed to resolve the issue of resource scheduling and allocation in the fog-cloud environment. To evaluate and analyse the proposed framework, a zero hour policy has been developed. Further a technique for resource allocation and management(TRAM) has been developed. For the implementation of task clustering and resource allocation, TRAM method has been used. For the resource allocation, heap based optimisation (HBO) algorithm is used and assigned the resources to the tasks.The proposed approach has been analyzed through the iFogsim simulator.The experimental results have been compared and analyzed with the existing approaches. The obtained results represent that the proposed approaches outperforms the existing approaches.

This chapter concludes the research work done in this thesis by highlighting its main contributions. It also discussed about the resource management and task scheduling in fog computing. The Chapter gives the briefing about the research work conducted in this thesis. In the end the chapter provides future scope of work and research direction for further researchers.

5.1 Conclusion

Resource utilization and Scheduling is a crucial aspect of fog Computing, which has a significant impact on system performance. The objective of this research was to minimize the complexity of resource allocation and increase the resource utilization in a cloud-fog environment.The dynamic nature of incoming tasks in the fog layer increases the makespan time and introduces a delay in completing tasks. In the fog computing environment, changes in the resource requirements depend upon the user's need and can change at any time. This eventually increases the failure rate of the task computation. The proposed method overcomes the limitations by optimally performing scheduling and preemption of tasks.

For this, we have implemented Heap based optimization algorithm for task scheduling and preemption of tasks to attain proper resource utilization.The proposed algorithm not only stabilize the resource management but also improves the whole framework performance.

The main contributions of this research work are described as follows:

- A Tri-fold task clustering framework is proposed for resource scheduling. The proposed framework collects all the tasks and create cluster based on the intensity of tasks.
- A Zero hour policy is designed to minimize the complexity of resource allocation in the fog-cloud environment. All the task consider and process in atomic manner. The dynamic nature of tasks are handled to design the resource grading process.The user's requirements were evaluated by the resource scheduler and allocate resources as per zero hour policy.
- A TRAM model is designed for resource allocation and management. TRAM model is used to create the LoR based on the availability of resources. Initially, the tasks are clustered based on the intensity by using the EM clustering model.The performance of proposed model is evaluated and concluded that the TRAM model is efficient in achieving the increased throughput and reduced response time in the completion of tasks.
- A Heap based optimization algorithm has been proposed for task scheduling and preemption of tasks to attain proper resource utilization to improve the overall system performance. The clustered tasks are scheduled based on the QoS and SLA constraints by implementing HBO resulting in reduced makespan time

– The allocation of resources is carried out based on the current and required resources for the effective computation of tasks using the C-DQN model. The preemption of tasks is performed to reduce the tasks' waiting time, thereby increasing the percentage of tasks satisfying the deadline and reducing the loss ratio.

– The proposed model is experimented using the iFogsim simulation tool and evaluated in terms of average response ratio, loss ratio, resource utilization, average makespan time, queuing waiting time, percentage of tasks satisfying the deadline and throughput.

– The comparative output analysis revealed that the proposed model has superior performance in terms of average loop delay, network consumption, execution time and energy consumption.

5.1.1 Main Contribution

The main contribution of this research work are summarised as follows:

– Resource utilization based task scheduling framework has been proposed that ensure the maximum utilization of available resources in the fog computing.

– A resource scheduling approach has been proposed to optimize the resources in the fog layer.

– Four main parameters have been considered to analyze the proposed resource utilization based resource scheduling framework and reinforcement Learning, i.e., Energy consumption, Response time, Network consumption and Execution time

– This thesis presents the development and implementation of the proposed Tri-fold task clustering framework and resource scheduling approaches.

– The proposed framework provides efficient resource scheduling and maximizes resource utilization in the fog layer, minimizing energy consumption.

5.2 Limitations of the Research

Due to the nature of fog devices, it will increase the complexity of the system in the network. The main limitations are mentioned as below.

- The main focus of the study is to minimize the latency and response time but the security of fog nodes are not considered at the fog layer.
- The proposed Zero Hour policy is designed to complete users' resources requests with all available resources at fog layer but there is no encryption algorithms which can lead to exposure of data to the hackers.
- The implementation is done on the iFogSimulator with number of tasks in the for of DAG and it is representing better resource utilization, whereas the results can vary in real-time applications and scenarios.
- The proposed framework can be deployed at any system with Java and Android mobile phones.

5.3 Future Work

This research has developed a lot of possibilities for future research. Fog computing methodology is developed to resolve the issues which are remain unhandled in the cloud computing methodology. All the possible future directions are mentioned as below:

- The approach ahead for sustainable directions will be focused on the use of Artificial Intelligence in the optimization approach.
- The resource security will be handled via implementing security policies and extends its usability beyond automotive industry.
- The application of artificial intelligence will help to handle the overloaded issue of fog nodes and bring the benefits to all enterprises. Moreover, artificial intelligence techniques can be implemented and embedded near the user's tasks on the fog layer.
- The prospective of fog computing in 5G enabled environments extends the benefits if AI at the edge level.
- In the case of extensive task, resource consumption in fog computing increases. Hence there is a need to evaluate the scalability and relability of resources in real-time fog computing applications, i.e., e-healthcare, smart traffic management system.

- Although fog computing is used in almost all areas now, fog is not very secure. A full security solution must be developed to determine all the security mechanisms required for all fog devices used in smart applications.

- The Fog system needs focus to decentralize the security model and best solution in the current time in Blockchain. However, Blockchain needs substantial research contributions to make it suitable for the fog system.

- The use of rule-based and module deployment algorithms increases the computational overhead at the proposed layer. As a result, optimization algorithms can be designed that take into account the other factors to make the Optical-Fog layer more computational.

References

1. Dempster, A. P., Laird, N. M. & Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* **39,** 1–22 (1977).
2. Iansiti, M. & Clark, K. B. Integration and dynamic capability: evidence from product development in automobiles and mainframe computers. *Industrial and corporate change* **3,** 557–605 (1994).
3. Bradley, P. S., Fayyad, U., Reina, C., *et al.* Scaling EM (expectation-maximization) clustering to large databases. *Microsoft Research,* 0–25 (1998).
4. Satyanarayanan, M. Pervasive computing: Vision and challenges. *IEEE Personal communications* **8,** 10–17 (2001).
5. Balan, R., Flinn, J., Satyanarayanan, M., Sinnamohideen, S. & Yang, H.-I. *The case for cyber foraging* in *Proceedings of the 10th workshop on ACM SIGOPS European workshop* (2002), 87–92.
6. Damiani, E., di Vimercati, D. C., Paraboschi, S., Samarati, P. & Violante, F. *A reputation-based approach for choosing reliable resources in peer-to-peer networks* in *Proceezdings of the 9th ACM conference on Computer and communications security* (2002), 207–216.
7. Ernemann, C., Hamscher, V., Schwiegelshohn, U., Yahyapour, R. & Streit, A. *On advantages of grid computing for parallel job scheduling* in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)* (2002), 39–39.
8. Krauter, K., Buyya, R. & Maheswaran, M. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience* **32,** 135–164 (2002).

9. Oprescu, A.-M. & Kielmann, T. *Bag-of-tasks scheduling under budget constraints* in *2010 IEEE second international conference on cloud computing technology and science* (2010), 351–359.

10. Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. & Buyya, R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* **41,** 23–50 (2011).

11. Çiftçioğlu, E. N. & Gürbüz, Ö. Scheduling for next generation WLANs: filling the gap between offered and observed data rates. *Wireless Communications and Mobile Computing* **11,** 654–666 (2011).

12. Han, H., Sheng, B., Tan, C. C., Li, Q. & Lu, S. A timing-based scheme for rogue AP detection. *IEEE Transactions on parallel and distributed Systems* **22,** 1912–1925 (2011).

13. Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. *Fog computing and its role in the internet of things* in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (2012), 13–16.

14. Bakıcı, T., Almirall, E. & Wareham, J. A smart city initiative: the case of Barcelona. *Journal of the knowledge economy* **4,** 135–148 (2013).

15. Gubbi, J., Buyya, R., Marusic, S. & Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems* **29,** 1645–1660 (2013).

16. Sharkh, M. A., Jammal, M., Shami, A. & Ouda, A. Resource allocation in a network-based cloud computing environment: design challenges. *IEEE Communications Magazine* **51,** 46–52 (2013).

17. Touati, F. & Tabish, R. U-healthcare system: State-of-the-art review and challenges. *Journal of medical systems* **37,** 1–20 (2013).

18. Aazam, M. & Huh, E.-N. *Fog computing and smart gateway based communication for cloud of things* in *2014 International Conference on Future Internet of Things and Cloud* (2014), 464–470.

19. Gia, T. N. *et al. Customizing 6LoWPAN networks towards Internet-of-Things based ubiquitous healthcare systems* in *2014 NORCHIP* (2014), 1–6.

20. Aazam, M. & Huh, E.-N. *Dynamic resource provisioning through fog micro datacenter* in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)* (2015), 105–110.

21. Aazam, M. & Huh, E.-N. *Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT* in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications* (2015), 687–694.

22. Cao, Y., Chen, S., Hou, P. & Brown, D. *FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation* in *2015 IEEE international conference on networking, architecture and storage (NAS)* (2015), 2–11.

23. Datta, S. K., Bonnet, C. & Haerri, J. *Fog computing architecture to enable consumer centric internet of things services* in *2015 International Symposium on Consumer Electronics (ISCE)* (2015), 1–2.

24. Gazis, V. *et al. Components of fog computing in an industrial internet of things context* in *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops)* (2015), 1–6.

25. Giang, N. K., Blackstock, M., Lea, R. & Leung, V. C. *Developing iot applications in the fog: A distributed dataflow approach* in *2015 5th International Conference on the Internet of Things (IOT)* (2015), 155–162.

26. Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M. & Kwak, K.-S. The internet of things for health care: a comprehensive survey. *IEEE access* **3,** 678–708 (2015).

27. Oueis, J., Strinati, E. C. & Barbarossa, S. *The fog balancing: Load distribution for small cell cloud computing* in *2015 IEEE 81st vehicular technology conference (VTC spring)* (2015), 1–6.

28. Saharan, K. & Kumar, A. Fog in comparison to cloud: A survey. *International Journal of Computer Applications* **122** (2015).

29. Sarkar, S., Chatterjee, S. & Misra, S. Assessment of the Suitability of Fog Computing in the Context of Internet of Things. *IEEE Transactions on Cloud Computing* **6,** 46–59 (2015).

30. Tang, B. *et al.* in *Proceedings of the ASE BigData & SocialInformatics 2015* 1–6 (2015).

31. Yi, S., Qin, Z. & Li, Q. *Security and privacy issues of fog computing: A survey* in *International conference on wireless algorithms, systems, and applications* (2015), 685–695.

32. Agarwal, S., Yadav, S. & Yadav, A. K. An efficient architecture and algorithm for resource provisioning in fog computing. *International Journal of Information Engineering and Electronic Business* **8,** 48 (2016).

33. Chiang, M. & Zhang, T. Fog and IoT: An overview of research opportunities. *IEEE Internet of Things Journal* **3,** 854–864 (2016).

34. Dastjerdi, A. V., Gupta, H., Calheiros, R. N., Ghosh, S. K. & Buyya, R. in *Internet of things* 61–75 (Elsevier, 2016).

35. Lee, W., Nam, K., Roh, H.-G. & Kim, S.-H. *A gateway based fog computing architecture for wireless sensors and actuator networks* in *2016 18th International Conference on Advanced Communication Technology (ICACT)* (2016), 210–213.

36. Pham, X.-Q. & Huh, E.-N. *Towards task scheduling in a cloud-fog computing system* in *2016 18th Asia-Pacific network operations and management symposium (APNOMS)* (2016), 1–4.

37. Skarlat, O., Schulte, S., Borkowski, M. & Leitner, P. *Resource provisioning for IoT services in the fog* in *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)* (2016), 32–39.

38. Sun, X. & Ansari, N. EdgeIoT: Mobile edge computing for the Internet of Things. *IEEE Communications Magazine* **54,** 22–29 (2016).

39. Yangui, S. *et al. A platform as-a-service for hybrid cloud/fog environments* in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)* (2016), 1–7.

40. Zeng, D., Gu, L., Guo, S., Cheng, Z. & Yu, S. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Transactions on Computers* **65,** 3702–3712 (2016).

41. Abbas, N., Zhang, Y., Taherkordi, A. & Skeie, T. Mobile edge computing: A survey. *IEEE Internet of Things Journal* **5,** 450–465 (2017).

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

43. Brogi, A. & Forti, S. QoS-aware deployment of IoT applications through the fog. *IEEE Internet of Things Journal* **4,** 1185–1192 (2017).

44. Byers, C. C. Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled iot networks. *IEEE Communications Magazine* **55,** 14–20 (2017).

45. Chang, C., Srirama, S. N. & Buyya, R. Indie fog: An efficient fog-computing infrastructure for the internet of things. *Computer* **50,** 92–98 (2017).

46. Dubey, H. *et al.* in *Handbook of Large-Scale Distributed Computing in Smart Healthcare* 281–321 (Springer, 2017).

47. Etemad, M., Aazam, M. & St-Hilaire, M. *Using devs for modeling and simulating a fog computing environment* in *2017 International Conference on Computing, Networking and Communications (ICNC)* (2017), 849–854.

48. Gia, T. N. *et al. Low-cost fog-assisted health-care IoT system with energy-efficient sensor nodes* in *2017 13th international wireless communications and mobile computing conference (IWCMC)* (2017), 1765–1770.

49. Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K. & Buyya, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience* **47,** 1275–1296 (2017).

50. Hao, Z., Novak, E., Yi, S. & Li, Q. Challenges and software architecture for fog computing. *IEEE Internet Computing* **21,** 44–53 (2017).

51. Hu, P., Dhelim, S., Ning, H. & Qiu, T. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of network and computer applications* **98,** 27–42 (2017).

52. Kabirzadeh, S., Rahbari, D. & Nickray, M. *A hyper heuristic algorithm for scheduling of fog networks* in *2017 21st Conference of Open Innovations Association (FRUCT)* (2017), 148–155.

53. Kocakulak, M. & Butun, I. *An overview of Wireless Sensor Networks towards internet of things* in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)* (2017), 1–6.

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

55. Lopes, M. M., Higashino, W. A., Capretz, M. A. & Bittencourt, L. F. *Myifogsim: A simulator for virtual machine migration in fog computing* in *Companion Proceedings of the10th International Conference on Utility and Cloud Computing* (2017), 47–52.

56. Mayer, R., Graser, L., Gupta, H., Saurez, E. & Ramachandran, U. *Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures* in *2017 IEEE Fog World Congress (FWC)* (2017), 1–6.

57. Mouradian, C. *et al.* A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE communications surveys & tutorials* **20,** 416–464 (2017).

58. Munir, A., Kansakar, P. & Khan, S. U. IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. *IEEE Consumer Electronics Magazine* **6,** 74–82 (2017).

59. Ni, L., Zhang, J., Jiang, C., Yan, C. & Yu, K. Resource allocation strategy in fog computing based on priced timed petri nets. *IEEE Internet of Things Journal* **4,** 1216–1228 (2017).

60. Östberg, P.-O. *et al. Reliable capacity provisioning for distributed cloud/edge/fog computing applications* in *2017 European conference on networks and communications (EuCNC)* (2017), 1–6.

61. Pooranian, Z., Shojafar, M., Naranjo, P. G. V., Chiaraviglio, L. & Conti, M. *A novel distributed fog-based networked architecture to preserve energy in fog data centers* in *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)* (2017), 604–609.

62. Rahbari, D. & Nickray, M. *Scheduling of fog networks with optimized knapsack by symbiotic organisms search* in *2017 21st Conference of Open Innovations Association (FRUCT)* (2017), 278–283.

63. Taneja, M. & Davy, A. *Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm* in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)* (2017), 1222–1228.

64. Tang, B. *et al.* Incorporating intelligence in fog computing for big data analysis in smart cities. *IEEE Transactions on Industrial informatics* **13,** 2140–2150 (2017).

65. Velasquez, K. *et al. Service orchestration in fog environments* in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)* (2017), 329–336.

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

67. Yigitoglu, E., Mohamed, M., Liu, L. & Ludwig, H. *Foggy: A framework for continuous automated iot application deployment in fog computing* in *2017 IEEE International Conference on AI & Mobile Services (AIMS)* (2017), 38–45.

68. Zhang, H., Zhang, Y., Gu, Y., Niyato, D. & Han, Z. A hierarchical game framework for resource management in fog computing. *IEEE Communications Magazine* **55,** 52–57 (2017).

69. Albahri, O. S. *et al.* Systematic review of real-time remote health monitoring system in triage and priority-based sensor technology: Taxonomy, open challenges, motivation and recommendations. *Journal of medical systems* **42,** 1–27 (2018).

70. Benamer, A. R., Teyeb, H. & Hadj-Alouane, N. B. *Latency-aware placement heuristic in fog computing environment* in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (2018), 241–257.

71. Farahani, B. *et al.* Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare. *Future Generation Computer Systems* **78,** 659–676 (2018).

72. Ghobaei-Arani, M., Jabbehdari, S. & Pourmina, M. A. An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. *Future Generation Computer Systems* **78,** 191–210 (2018).

73. Kamal, M. B. *et al. Heuristic min-conflicts optimizing technique for load balancing on fog computing* in *International Conference on Intelligent Networking and Collaborative Systems* (2018), 207–219.

74. Lee, D. & Lee, H. IoT service classification and clustering for integration of IoT service platforms. *The Journal of Supercomputing* **74,** 6859–6875 (2018).

75. Liu, J. *et al.* Secure intelligent traffic light control using fog computing. *Future Generation Computer Systems* **78,** 817–824 (2018).

76. Mahmud, R., Koch, F. L. & Buyya, R. *Cloud-fog interoperability in IoT-enabled healthcare solutions* in *Proceedings of the 19th international conference on distributed computing and networking* (2018), 1–10.

77. Mahmud, R., Kotagiri, R. & Buyya, R. in *Internet of everything* 103–130 (Springer, 2018).

78. Puthal, D. *et al.* Secure and sustainable load balancing of edge data centers in fog computing. *IEEE Communications Magazine* **56,** 60–65 (2018).

79. Qayyum, T., Malik, A. W., Khattak, M. A. K., Khalid, O. & Khan, S. U. FogNetSim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access* **6,** 63570–63583 (2018).

80. Rahmani, A. M. *et al.* Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems* **78,** 641–658 (2018).

81. Saroa, M. K. & Aron, R. *Fog computing and its role in development of smart applications* in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)* (2018), 1120–1127.

82. Simmhan, Y., Ravindra, P., Chaturvedi, S., Hegde, M. & Ballamajalu, R. Towards a data-driven IoT software architecture for smart city utilities. *Software: Practice and Experience* **48,** 1390–1416 (2018).

83. Sun, Y., Lin, F. & Xu, H. Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II. *Wireless Personal Communications* **102,** 1369–1385 (2018).

84. Tu, S. *et al.* Security in fog computing: A novel technique to tackle an impersonation attack. *IEEE Access* **6,** 74993–75001 (2018).

85. Wadhwa, H. & Aron, R. *Fog computing with the integration of Internet of Things: architecture, applications and future directions* in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)* (2018), 987–994.

86. Xu, X. *et al. A heuristic virtual machine scheduling method for load balancing in fog-cloud computing* in *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)* (2018), 83–88.

87. Yang, Y. *et al.* MEETS: Maximal energy efficient task scheduling in homogeneous fog networks. *IEEE Internet of Things Journal* **5,** 4076–4087 (2018).

88. Zhang, G. *et al.* DOTS: Delay-optimal task scheduling among voluntary nodes in fog networks. *IEEE Internet of Things Journal* **6,** 3533–3544 (2018).

89. Adhikari, M., Mukherjee, M. & Srirama, S. N. DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multi-level feedback queueing. *IEEE Internet of Things Journal* (2019).

90. Alturki, B., Reiff-Marganiec, S., Perera, C. & De, S. Exploring the effectiveness of service decomposition in fog computing architecture for the Internet of Things. *IEEE Transactions on Sustainable Computing* (2019).

91. Chang, C., Srirama, S. N. & Buyya, R. Internet of things (IoT) and new computing paradigms. *Fog and Edge Computing: Principles and Paradigms,* 1–23 (2019).

92. Donassolo, B., Fajjari, I., Legrand, A. & Mertikopoulos, P. *Fog based framework for IoT service provisioning* in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (2019), 1–6.

93. Gope, P., Lee, J., Hsu, R.-H. & Quek, T. Q. Anonymous communications for secure device-to-device-aided fog computing: architecture, challenges, and solutions. *IEEE Consumer Electronics Magazine* **8,** 10–16 (2019).

94. He, Z., Zhang, Y., Tak, B. & Peng, L. Green fog planning for optimal internet-of-thing task scheduling. *IEEE Access* **8,** 1224–1234 (2019).

95. Krishnan, P. & Aravindhar, D. J. Self-adaptive PSO memetic algorithm for multi objective workflow scheduling in hybrid cloud. *Int. Arab J. Inf. Technol.* **16,** 928–935 (2019).

96. Luo, J. *et al.* Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. *Future Generation Computer Systems* **97,** 50–60 (2019).

97. Maiti, P., Shukla, J., Sahoo, B. & Turuk, A. K. in *Emerging Technologies in Data Mining and Information Security* 13–21 (Springer, 2019).

98. Mukherjee, M., Guo, M., Lloret, J., Iqbal, R. & Zhang, Q. Deadline-aware fair scheduling for offloaded tasks in fog computing with inter-fog dependency. *IEEE Communications Letters* **24,** 307–311 (2019).

99. Naranjo, P. G. V., Pooranian, Z., Shojafar, M., Conti, M. & Buyya, R. FOCAN: A Fog-supported smart city network architecture for management of applications in the Internet of Everything environments. *Journal of parallel and distributed computing* **132,** 274–283 (2019).

100. Nguyen, B. M., Thi Thanh Binh, H., Do Son, B., *et al.* Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment. *Applied Sciences* **9,** 1730 (2019).

101. Ning, Z., Huang, J. & Wang, X. Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications* **26,** 87–93 (2019).

102. Pang, S., Li, W., He, H., Shan, Z. & Wang, X. An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing. *IEEE Access* **7,** 146379–146389 (2019).

103. Rafique, H. *et al.* A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing. *IEEE Access* **7,** 115760–115773 (2019).

104. Rahbari, D. & Nickray, M. Low-latency and energy-efficient scheduling in fog-based IoT applications. *Turkish Journal of Electrical Engineering & Computer Sciences* **27,** 1406–1427 (2019).

105. Santos, J., Wauters, T., Volckaert, B. & De Turck, F. Resource provisioning in Fog computing: From theory to practice. *Sensors* **19,** 2238 (2019).

106. Tang, C. *et al. Fog-enabled smart campus: Architecture and challenges* in *International Conference on Security and Privacy in New Computing Environments* (2019), 605–614.

107. Tortonesi, M. *et al.* Taming the IoT data deluge: An innovative information-centric service model for fog computing applications. *Future Generation Computer Systems* **93,** 888–902 (2019).

108. Wang, W. *et al. Data scheduling and resource optimization for fog computing architecture in industrial IoT* in *International Conference on Distributed Computing and Internet Technology* (2019), 141–149.

109. Xu, J., Hao, Z., Zhang, R. & Sun, X. A method based on the combination of laxity and ant colony system for cloud-fog task scheduling. *IEEE Access* **7,** 116218–116226 (2019).

110. Yousefpour, A. *et al.* All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture* (2019).

111. Zhou, Z. *et al.* Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. *IEEE Transactions on Vehicular Technology* **68,** 3113–3125 (2019).

112. Abbasi, M., Yaghoobikia, M., Rafiee, M., Jolfaei, A. & Khosravi, M. R. Efficient resource management and workload allocation in fog–cloud computing paradigm in IoT using learning classifier systems. *Computer Communications* **153,** 217–228 (2020).

113. Abdel-Basset, M. *et al.* Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications. *IEEE Transactions on Industrial Informatics* (2020).

114. Abdelmoneem, R. M., Benslimane, A. & Shaaban, E. Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures. *Computer Networks* **179,** 107348 (2020).

115. Abualigah, L. & Diabat, A. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Cluster Computing,* 1–19 (2020).

116. Ali, I. M. *et al.* An Automated Task Scheduling Model using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems. *IEEE Transactions on Cloud Computing* (2020).

117. Alli, A. A. & Alam, M. M. The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications. *Internet of Things* **9,** 100177 (2020).

118. Altulyan, M., Yao, L., Kanhere, S. S., Wang, X. & Huang, C. A unified framework for data integrity protection in people-centric smart cities. *Multimedia Tools and Applications* **79,** 4989–5002 (2020).

119. Arisdakessian, S., Wahab, O. A., Mourad, A., Otrok, H. & Kara, N. FoGMatch: an intelligent multi-criteria IoT-Fog scheduling approach using game theory. *IEEE/ACM Transactions on Networking* **28,** 1779–1789 (2020).

120. Bonadio, A., Chiti, F., Fantacci, R. & Vespri, V. An integrated framework for blockchain inspired fog communications and computing in internet of vehicles. *Journal of Ambient Intelligence and Humanized Computing* **11,** 755–762 (2020).

121. Chen, L., Guo, K., Fan, G., Wang, C. & Song, S. Resource constrained profit optimization method for task scheduling in edge cloud. *IEEE Access* **8,** 118638–118652 (2020).

122. Chen, X. *et al.* A woa-based optimization approach for task scheduling in cloud computing systems. *IEEE Systems Journal* **14,** 3117–3128 (2020).

123. Goudarzi, M., Wu, H., Palaniswami, M. S. & Buyya, R. An application placement technique for concurrent IoT applications in edge and fog computing environments. *IEEE Transactions on Mobile Computing* (2020).

124. Hosseinioun, P., Kheirabadi, M., Tabbakh, S. R. K. & Ghaemi, R. A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. *Journal of Parallel and Distributed Computing* **143,** 88–96 (2020).

125. Hussein, M. K. & Mousa, M. H. Efficient task offloading for iot-based applications in fog computing using ant colony optimization. *IEEE Access* **8,** 37191–37201 (2020).

126. Jie, Y., Guo, C., Choo, K.-K. R., Liu, C. Z. & Li, M. Game-Theoretic Resource Allocation for Fog-based Industrial Internet of Things Environment. *IEEE Internet of Things Journal* (2020).

127. Lin, S.-Y. *et al.* Fog Computing Based Hybrid Deep Learning Framework in effective inspection system for smart manufacturing. *Computer Communications* **160,** 636–642 (2020).

128. Lohi, S. A. & Tiwari, N. *A high performance machine learning algorithm TspINA; scheduling multifariousness destined tasks by better efficiency* in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)* (2020), 603–607.

129. Madeo, D., Mazumdar, S., Mocenni, C. & Zingone, R. Evolutionary game for task mapping in resource constrained heterogeneous environments. *Future Generation Computer Systems* **108,** 762–776 (2020).

130. Naha, R. K., Garg, S., Chan, A. & Battula, S. K. Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. *Future Generation Computer Systems* **104,** 131–141 (2020).

131. Patel, E. & Kushwaha, D. S. Clustering cloud workloads: k-means vs gaussian mixture model. *Procedia Computer Science* **171,** 158–167 (2020).

132. Rahman, H. F., Chakrabortty, R. K. & Ryan, M. J. Memetic algorithm for solving resource constrained project scheduling problems. *Automation in Construction* **111,** 103052 (2020).

133. Rathee, G., Sandhu, R., Saini, H., Sivaram, M. & Dhasarathan, V. A trust computed framework for IoT devices and fog computing environment. *Wireless Networks* **26,** 2339–2351 (2020).

134. Rezaee, A. & Adabi, S. *Jobs (DAG workflow) and tasks dataset with near 50k job instances and 1.3 Millions of tasks.* Sept. 2020.

135. Shadroo, S., Rahmani, A. M. & Rezaee, A. The two-phase scheduling based on deep learning in the Internet of Things. *Computer Networks,* 107684 (2020).

136. Shetty, C. & Sarojadevi, H. *Framework for Task scheduling in Cloud using Machine Learning Techniques* in *2020 Fourth International Conference on Inventive Systems and Control (ICISC)* (2020), 727–731.

137. Tuli, S., Ilager, S., Ramamohanarao, K. & Buyya, R. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. *IEEE Transactions on Mobile Computing* (2020).

138. Tuli, S. *et al.* Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments. *Future Generation Computer Systems* **104,** 187–200 (2020).

139. Tychalas, D. & Karatza, H. A scheduling algorithm for a fog computing system with bag-of-tasks jobs: Simulation and performance evaluation. *Simulation Modelling Practice and Theory* **98,** 101982 (2020).

140. Vijayalakshmi, R., Vasudevan, V., Kadry, S. & Lakshmana Kumar, R. Optimization of makespan and resource utilization in the fog computing environment through task scheduling algorithm. *International Journal of Wavelets, Multiresolution and Information Processing* **18,** 1941025 (2020).

141. Wang, S., Zhao, T. & Pang, S. Task scheduling algorithm based on improved firework algorithm in fog computing. *IEEE Access* **8,** 32385–32394 (2020).

142. Wang, X., Gu, H. & Yue, Y. The optimization of virtual resource allocation in cloud computing based on RBPSO. *Concurrency and Computation: Practice and Experience* **32,** e5113 (2020).

143. Zhou, Z., Liao, H., Wang, X., Mumtaz, S. & Rodriguez, J. When vehicular fog computing meets autonomous driving: Computational resource management and task offloading. *IEEE Network* **34,** 70–76 (2020).

144. Lai, C.-F., Weng, H.-Y., Chou, H.-Y. & Huang, Y.-M. A Novel NAT-based Approach for Resource Load Balancing in Fog Computing Architecture. *Journal of Internet Technology* **22,** 513–520 (2021).

145. Shadroo, S., Rahmani, A. M. & Rezaee, A. The two-phase scheduling based on deep learning in the Internet of Things. *Computer Networks* **185,** 107684 (2021).

146. Wadhwa, H. & Aron, R. *A Clustering-Based Optimization of Resource Utilization in Fog Computing* in *Proceedings of International Conference on Advanced Computing Applications* (2022), 343–353.

147. Wadhwa, H. & Aron, R. Resource Utilization for IoT Oriented Framework Using Zero Hour Policy. *Wireless Personal Communications* **122,** 2285–2308 (2022).

148. Wadhwa, H. & Aron, R. TRAM: Technique for resource allocation and management in fog computing environment. *The Journal of Supercomputing* **78,** 667–690 (2022).

LIST OF PUBLICATIONS

- **Journal Publication**

  Wadhwa, Heena, and Rajni Aron. "TRAM: Technique for resource allocation and management in fog computing environment." The Journal of Supercomputing 78.1 (2022): 667-690. (Impact Factor-2.6)

  Wadhwa, H., Aron, R. Resource Utilization for IoT Oriented Framework Using Zero Hour Policy. Wireless Personal Communication 122, 2285–2308 (2022). (Impact Factor-1.6)

- **International Conference**

  Wadhwa, H., Aron, R. (2018, December). Fog computing with the integration of Internet of Things: architecture, applications and future directions. In 2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom) (pp. 987-994). IEEE.

  Wadhwa, Heena, and Rajni Aron. "A Clustering-Based Optimization of Resource Utilization in Fog Computing." In Proceedings of International Conference on Advanced Computing Applications, pp. 343-353. Springer, Singapore, 2022.

  Wadhwa, H., Aron, R. (2020). Energy Based Resource Provisioning For IoT Application in Fog Computing. Journal of Natural Remedies, 21(3), S1.
  Fog computing framework for multi constraints based resource utilization in 2nd International Conference on Computational Methods in Science Technology [Presented]

- **Communicated**

  Optimized Task Scheduling And Preemption For Distributed Resource Management In Fog Assisted IoT Environment [Communicated]