An Effective Framework for Service Placement in Fog Environment

A

Thesis

Submitted to

For the award of

DOCTOR OF PHILOSOPHY (Ph.D)

in

(Computer Science & Engineering)

By

M. Sri Raghavendra

(11616630)

Supervised By

Dr. Priyanka Chawla

LOVELY FACULTY OF TECHNOLOGY AND SCIENCES

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

2023

Declaration of Authorship

I, M Sri Raghavendra, declare that this thesis titled, “An Effective Framework for Service Placement in Fog Environment” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself, I have made clear exactly what was done by others and what I have contributed myself.

Signed:
.....

Date:
.....

Certificate

It is to certify that M. Sri Raghavendra, Ph.D. Scholar, Department of Computer Science has carried out research work entitled "**An Effective Framework for Service Placement in Fog Computing Environment**", for the award of the degree of Doctor of Philosophy in Computer Science, Faculty of Technology and Sciences, Lovely Professional University, Punjab, India.

Further certify that:

I. The thesis embodies the original work of the candidate and is not copied from any other sources.

II. The candidate has worked under my supervision for the period required under statues.

III. The candidate has put in the required attendance in the department during the period of research.

IV. The candidate has fulfilled all the requirements of the UGC- 2018 regulations.

Dr. Priyanka Chawla
Associate Professor
Department of Computer Science and Engineering
National Institute of Technology
Warangal-506004, Telangana, India
(Former Professor, SCSE, LPU)

Abstract

The term "Internet of Things" originally referred to linking physical items to the internet, allowing them to see, hear, think, converse, and execute jobs such as sharing information and taking action. When it comes to cloud services, latency is also a major concern. It is more difficult to measure and less predictable. Latency requirements for IoT are high, and cloud services cannot meet these requirements in the best way possible.

Fog computing is a new suggested architecture that extends existing cloud computing by employing extra layers of intermediary computer servers to address these issues. Using fog servers, data may be routed from end-user devices to the cloud. Like the Cloud, they give processing, memory, and storage resources to devices but are more distributed. Our new computer paradigm brings a new set of issues, which we hope to address in this study. Fog computing has evolved into a flexible and promising platform for delivering elastic resources at the network's edge. It reduces transmission delay and bandwidth utilization while processing incoming requests from various (IoT) devices. In a heterogeneous resource-restricted distributed fog environment, the principal critical concerns that Fog Computing faces are service placement and energy usage.

The proposed Fog computing framework entitled Deadline-oriented Service Placement (DoSP) offers services in both the fog and the cloud nodes. By effective Leveraging of fog resources thereby keeping response times within a predetermined time frame was the goal of this study. In a fog environment, it leverages the Genetic Algorithm (GA) to identify where services should be placed. DoSP modelling influence the service placement techniques on service deadlines were assessed in this study by using iFogSim simulator. We found the proposed approach to reduce service execution delay by 10.19 percent for Edge Ward and 2.58 percent for Cloud Only.

This research has proposed a model for deploying deadline-sensitive applications (DEEDSP) with minimal energy usage. We demonstrate the importance of such a system. However, distributing applications to computing nodes becomes a formidable

challenge that necessitates the use of optimization approaches. We employ Metaheuristics, specifically the hyper-heuristic algorithm, which can optimize IoT application service placement while being deadline-aware and lowering energy consumption in Fog Computing. We used iFogSim, a Java-based open-source network simulator, to simulate the suggested model. Finally, after running the simulation, we performed a result analysis of the model to several output parameters. We discovered that it outperforms previous state-of-the-art models.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Dr. Priyanka Chowla, for her supervision, advice, and guidance from the very first day of this research as well as giving me extraordinary experiences throughout the work. I am truly very fortunate to have the opportunity to work with her. I found this guidance to be extremely valuable.

I am grateful to the friend and fellow researcher, particularly Y Narsimhulu for his constructive criticism and suggestions.

I would like to show my gratitude to the entire family of Lovely Professional University for providing me a suitable research atmosphere to carry out my work in proper time. I would like to thank the Division of Research and Development for all the support encouragement throughout the research work.

I am also very much grateful to my wife, Mrs. Radhika, my father, Mr. M Kondaiah, my mother, Mrs. P. Rama Lakshmi and my children for their moral support and care that they show me during the period of this work.

Last but not least, I thank God for sailing me through all the rough and tough times during this research work.

Sri Raghavendra Date.....

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Fog Computing Overview

The internet has transformed our study, work, and personal lives in the same way water, gas, electricity, and telecommunications have. People today place a great deal of trust and reliance on the internet to conduct business, communicate, and collaborate. Because of the intelligence supplied by smart products, machines are increasingly connected to the internet, resulting in intelligent systems such as smart homes, intelligent communities, smart utilities, and smart cities, to name a few. To this purpose, the future internet is combining people and things, resulting in an unparalleled information period dubbed "the Internet of Things (IoT) or Internet of Everything (IoE) age." Due to various constraints, such as resource constraints, data is typically collected from distant smart objects before being transferred to and processed in faraway cloud data centers.

However, the vast number of smart objects generates a data deluge that consumes a tremendous amount of energy and bandwidth, resulting in significant Internet traffic congestion, decreased Internet performance, and even application failure for many apps. Fog computing was inspired by cloud computing and provides edge processing for heterogeneous and ubiquitous items. Because the fog is still in its infancy, our goal in this thesis is to hasten its development while also making employment easier for all levels of fog actors in this ecosystem. A cloud computing model that provides internet services at the network's edge might be investigated. As a result, it assists in overcoming the frequently major issues of delays in the usage of IoT cloud systems.

Cloud computing provides pay-for-pay customers with hardware and software resources. It is a mix of cluster and grid computing where resources for high-level performance come together at a central point. The advanced form of distributed computing services is Fog Computing (Gao et al., 2017). It improves performance and uses computer processing and storage resources to handle user requests at the

network's edge. Cloud computing is not replaced. If cloud computing uses fog computing, this will reduce delays, easily measure and reduce processing costs (Xiao et al., 2018).

1.1.1 Fog Evolution

Cloud computing is enhanced with fog computing. Fog Computing (Wang et al., 2019) is a new technology for providing high-quality localized services. Cloud services are extended by installing Fog computing servers close to mobile users, including parks or commercial centers. It provides a "custom cloud" specified local software in a user location with a cloud that can process, store and transmit the network.Fog computing expands its facilities at dispersed areas around users compared to Amazon EC2, IBM SoftLayer, and other centrally managed cloud-based services and, thus, offers more focused, localized mobile services customized to deployment locations, such as streams of software, localized ads, and sensor networks (Li et al., 2018).

A central cloud service operates a Fog computing network. Consequently, Fog computing is a three-tier system of hierarchy, Mobile-Fog-Cloud. This model has three directions: Mobile device to Fog node, Mobile device to Cloud node, and Fog node to Cloud node (Ramirez et al., 2017).

The primary goal of fog computing is to distribute application's content over the fog, where most of the information has downloaded from cloud to lower layer computational nodes (Qi et al., 2019). Using Internet access, app users may rapidly access data due to fast transmission on a reliable network environment. If the user cannot find the data on lower Fog servers, then the d ata can be transferred from the cloud. To provide this Fog computing service,computational nodes must obtain the most recent content from the cloud node(Aljumah and Ahanger, 2018). In most cases, this kind of data delivery is accomplished through the internet. The Fog servers use a cellar or a satellite network or WAN network for communicating with the cloud node (Rafique et al., 2019).

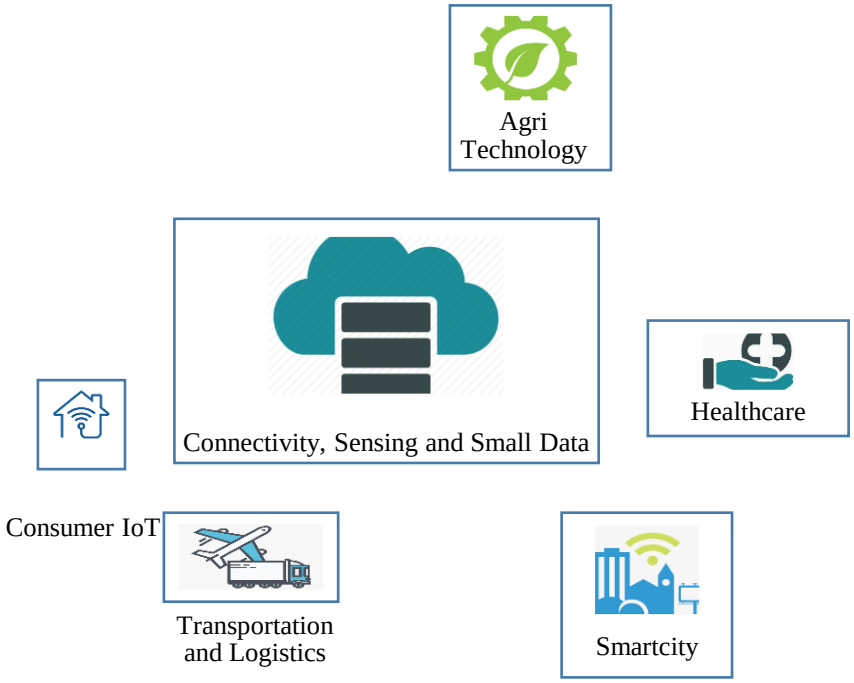


Figure 1.1: FOG Computing (Abdulkareem et al., 2019)

Current IT innovations of the state-of-the-art accelerate the growth of intelligent production significantly. More and more goods are connected to the internet in an intelligent manufacturing environment, allowing a vast amount of data to be accessed at all points of the product life cycle. Cloud-based intelligent manufacturing enables the processing of vast volumes of data. It allows large-scale manufacturing cooperation in various applications and solutions (Abdulkareem et al., 2019).

Significant development of the computing paradigms has been witnessed in the last decade. Without a doubt, cloud computing is the most well-known and well-established paradigm, resulting from the necessity to utilise "computer as a utility," allowing for the quick emergence of new Internet services. Cloud computing was a prominent research topic until an avalanche of smart gadgets and appliances, termed the Internet of Things (IoT), highlighted all of the limitations of such a centralized architecture (IT) (Padmavathi et al., 2017). The Internet of Things revolution has given rise to a growing interest in decentralized models (Oma et al., 2018).

To extend the cloud's power to network edges, edge computing tackled most of the new difficulties that cloud computers alone couldn't manage, such as bandwidth, latency, and connection. (DeDonno et al., 2019) Several implementation approaches have been proposed to apply the Edge Calculation Principles, including Mobile Cloud Computing (MCC) and Cloudlet Computing (CC).

Fog Computing was created by the crowd and it embodies the progression of Edge computing principles (Svorobej et al., 2019). Fog computing depicts a complete architecture that dispenses resources vertical and horizontal directions over the Cloud node-to-IoT devices. As a consequence, it's a leading service that interacts with the IoT and Cloud on a regular basis to assist and improve their engagement. (Kaur and Chana, 2016). As a result, Fog Computing was created by the crowd, and it embodies the progression of Edge computing principles (Svorobej et al., 2019). Fog computing attempts to depict a full architecture that distributes resources horizontally and vertically over the Cloud-to-Things continuum. As a result, it is a new actor who interacts with the cloud and IoT trivially to help and enhance their interaction (Saroa et al., 2018).

The primary computing paradigm in the last decade is cloud computing. For several years it will continue to be a key research topic. The rapid proliferation of IoT, however, has weakened its power. Indeed, cloud computing can hardly tackle several challenges related to IoT. As a result, the edge's interest increased due to its goal to address the IoT challenges by moving the computation processing at the lower fog node near to the user (Gia et al., 2015).In this shift, fog computing has taken the shape of a paradigm that entirely spans the gap between Cloud computers and IoT.In the smart urban environment, extreme data are generated, thanks to the global use of sensors, which enhance the need for an architecture that can meet the data maintenance requirements of devices (Barik et al., 2017). Cloud computing has a genre called "Fog Computing." The cloud services are close to edge devices in this genre. Data processing in the fog with the help of the network, conduct latency-sensitive and latency-tolerant workflow that require a lot of computation (Mahmud et al., 2018).

Increasingly linked Internet-of-Things (IoT) gadgets are creating a massive amount of data that cloud computing can't handle. The number of devices in 2015 amounted to 15.41billionfrom 2017 to 20.35, with a forecast of EUR 30.73 billion in 2020.

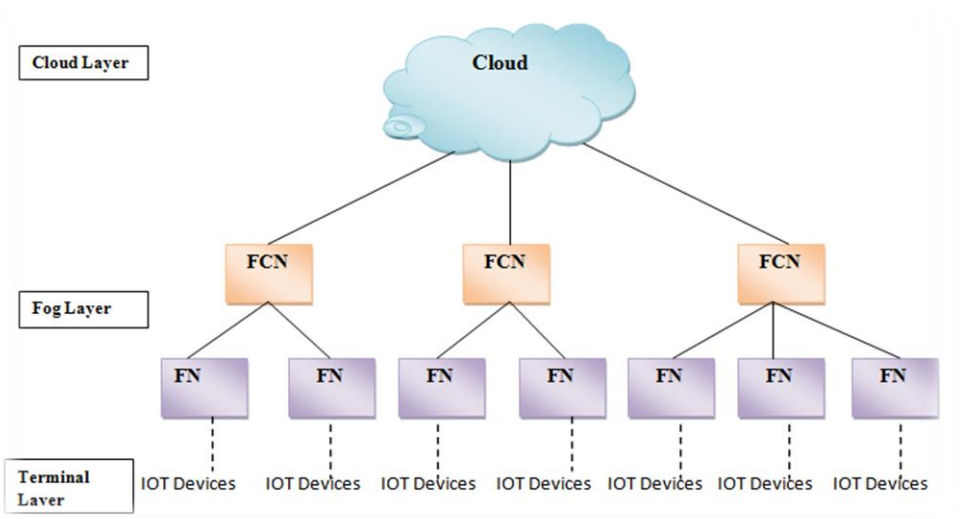


Figure 1.2: Evolution of Fog Computing (M. Sri Raghavendra et al.,2018)

1.1.1.1 Fog Architecture

Fog computing (Shi et al., 2015) is a modern computing model, expanding conventional cloud computing and network infrastructure. It offers network edge power for measurement, contact, control, storage, and operation. The open network varies from other conventional software programming models (Sun and Zhang, 2017). It's design and functionality and the difference between cloud and Edge Computing are outlined in figure 1.2.

Fog computing architecture (Shin et al., 2016) is the structure for implementing a functional IoT network with physical and logical network components, hardware, and software. Important decisions in architecture include the physical and geological location of fog nodes, the hierarchy of their structures, their number, size, topology, protocols, and data bandwidth capability, the connection of fog nodes, stuff and cloud, the design of individual fog Node hardware and software and how a full IoT network is structured and managed (Hu et al., 2017). A three-tiered Mobile-Fog-Cloud arrangement is common in fog computing (Naha et al., 2018). In the Fog layer, Fog

servers provide services to end users and synchronize data with the cloud. The Cloud provider provides content services to Fog servers that are geographically dispersed (Jin et al., 2018). Whenever a mobile user receives information, the Fog server and the mobile user exchange information. This Fog server sends the requested content to the mobile user if it is accessible (Aazam et al., 2018). Alternatively, this Fog server will need to contact its Cloud provider to place and access data. It's also important to note that Fog servers must constantly check their Cloud providers to determine whether they have the most recent information, and then update their storage by obtaining from the cloud over wired or wireless networks, such as cellular networks, if necessary.

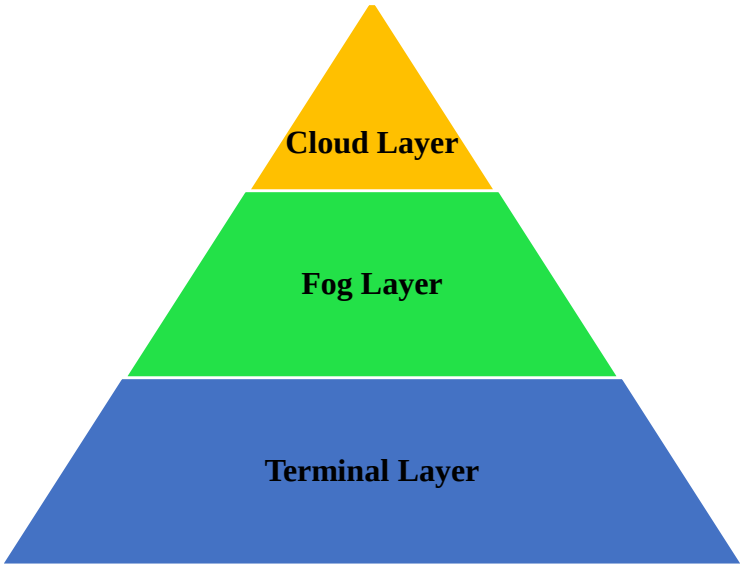


Figure 1.3: FOG Computing Architecture(M. Sri Raghavendra et al.,2018)

Hierarchical architecture has three layers: top, middle, and bottom.

❖ **Terminal Layer**

The layer closest to the user and the physical world is this one. IoT devices such as cameras, smartphones, smart vehicles, intelligent cards and printers are all included. We're only utilizing them as smart sensors in this case. Equipment like these may be found all around the world. Using physical object or event functional data, they are responsible for detecting and delivering data to the upper layer.

❖ **Fog Layer**

The network's edge is host to this layer. The Fog Computing network contains a large number of fog nodes, such as routers, gateways, switches, access points, base stations, and specimen fog servers, among other components. It's common to see these fog nodes spread out throughout a wide area, such as cafés; retail malls; transit stations; streets; as well as public spaces like parks. On a moving carriage, they might be stationary or movable. Fog nodes may be readily accessed by the terminals. It is able to calculate, transmit, and store temporary data. Real-time analysis and latency-sensitive applications can both benefit from the fog layer. By collaborating and interacting with cloud resources, fog nodes are able to deliver more efficient computation and storage than would be possible on their own.

❖ **Cloud Layer**

Many high-performance computers and storage devices make up the cloud computing layer. Smart housing, smart transportation, and smart factories are all available. The cloud core modules are controlled and planned in accordance with demand load requirements in order to maximize the usage of the cloud resources.

1.1.2 Features of Fog Computing

Fog computing uses end-user network edge devices for processing, connection, and storing of data. Like end-users, it has fairly similar service functionalities. (Rahman et al., 2018). Fog computing (Madakam and Bhagat, 2018) relies on this fundamental property, making it the most illusory benefit it has over other more traditional kinds of computing. Additional advantages and benefits of fog computing are presented in the following table:

Table 1.1: Fog Computing Characteristics

S. No.	Fog Computing Characteristics	
1.	Low Latency	Fog computing provides the best services to endpoints at the network's edge.
2.	Mobility	To enable mobility strategies like isolating host identification from location identity, LISP protocol fog devices can be used.

3.	Interactions in real time	To provide uninterrupted operation, fog computing requires significant interactions.
4.	Distribution by location	Through proxies and access points along roads and rails, the fog will play an active part in delivering high quality streaming to moving cars.
5.	Heterogeneous Devices	It is possible to use fog nodes in a wide range of situations. Computers, laptops, pagers, and smart phones are just some of the heterogeneous items or gadgets that make up these settings.
6.	Interoperability	The term "interoperability" refers to the way in which various IoT devices and hardware connect and function together when data is being transmitted by software via multiple networks.
7.	Save bandwidth	In the cloud, just a small portion of the useful information is carried over. The vast majority of the data does not necessitate the use of the internet.
8.	Data protection and security	It has the ability to encrypt and isolate data to keep it safe. Encryption, integrity checks, and isolation are all provided by fog nodes in order to keep private information safe. (Stojmenovic and Wen, 2014).
9.	Low energy consumption	A more environmentally friendly computing model is provided by fog computing. By using short-range communication, you may save money on your monthly energy bill. Reducing energy use and saving money are two of the benefits.

1.2 Background Technologies

- **Internet of Things (IoT)**

The trend has been to transfer computers, controls, and data stores in the cloud during the last decade. The primary data center, Backbone IP networks, and cellular core networks have been converted from storage, shops, and network management functions. Nevertheless, today's cloud computing faces growing difficulties in meeting evolving Internet of Things (IoT) new requirements (Chiang et al., 2016).

Many technological challenges face the Internet of Things (IoT), which will create an immense network of trillions or thousands of "things" that interact with each other. After the internet, the Internet of Things (IoT) is seen as a technology and an economic stream in the global information business (Chen et al., 2014). The IoT is an intelligent network connecting everything to the internet to exchange information and communicate by accepted protocols via information sensing devices. The aim is to define, locate, map, control, and manage matters intelligently (Souza et al., 2018). The network extends and enhances the connectivity from humans to people and things or things and stuff. It is an Internet-based network.

The term "Internet of Things" originally referred to linking physical items to the internet, allowing them to see, hear, think, converse, and execute jobs such as sharing information and taking action. A smart home allows the residents to open their garage automatically when they arrive home, cook meals, and control an in-house climate system, entertainment system, and other home appliances. Cisco promotes the Internet of Everything (IoE) (L. T. Yang, B. Di Martino et al, 2017), which is defined as the intelligent linking of people, processes, data, machines talking to machines; machines talking to humans; and humans talking to human's communications and interactions.

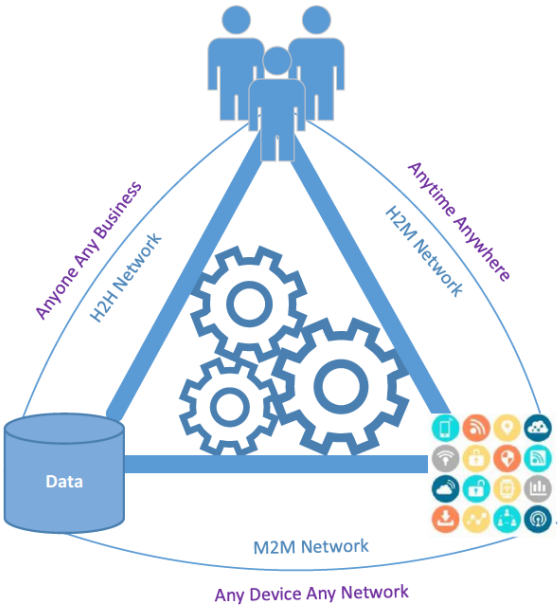


Figure 1.4: Internet of Things (L. T. Yang, B. Di Martino et al, 2017)

Anyone can collaborate via any device and any network in an IoT ecosystem, as shown in Figure 1.4. As the number of connected devices grows, the Internet of Things (IoT) is expected to take over as the dominant technology. However, it is uncertain what it will be like when hundreds of things surrounding each person are average. For predicting the extent of the IoT, numerous estimates have been produced. Cisco estimates that by 2020, there will be 50 billion linked things (J. Bradley, J. Barbier, et al, 2013). The Internet of Things is projected to improve life, employment, and economy throughout time.

Internet of Things (IoT) Elements

Semantics, identification, processing, storage, communication, networking, and application are crucial parts of the IoT ecosystem's functionality, as depicted in Figure 1.5. We take a quick look at each to see what the IoT is all about.

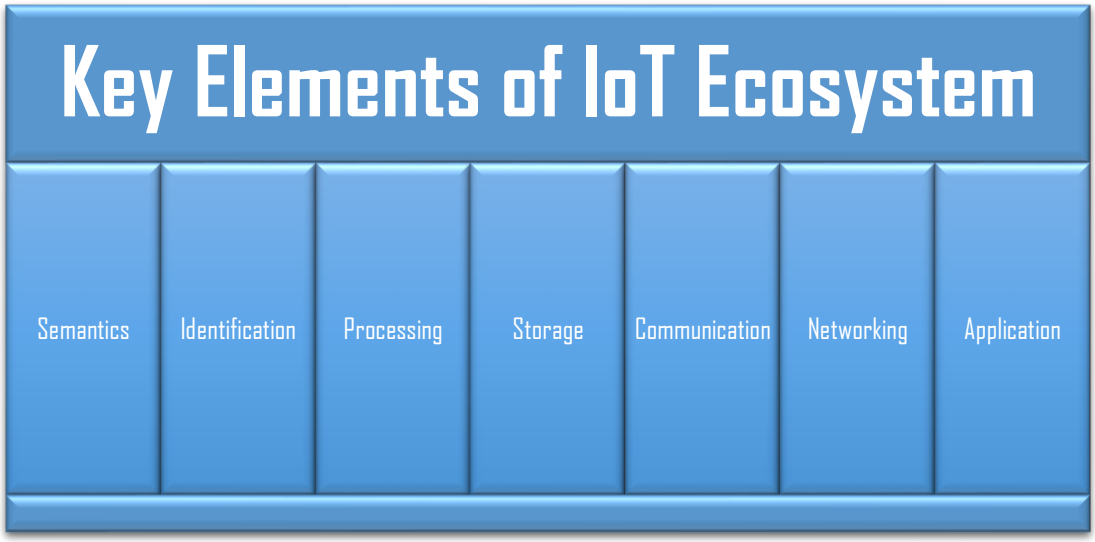


Figure 1.5: IoT Ecosystem Key Elements (N. Koshizuka and K. Sakamura, 2010)

Semantics

The ability of IoT nodes to intelligently extract knowledge from various things to execute functions such as resource discovery, utilization, information modeling, data collecting, and analysis is referred to as semantics.

Identification

From addressing, naming, networking, programming, and security, identification is critical for the IoT. Electronic product codes (EPC) and ubiquitous codes (uCode) are two methods of identifying that have been made available (N. Koshizuka and K. Sakamura, 2010). For example, a temperature SO called "T1" may have a different address depending on networking protocol stacks like IP, IPv6, 6LoWPAN (N. Kushalnagar, G. Montenegro et al., 2007). A domain name service (DNS) mechanism is necessary for this scenario to translate the name to its address for communication purposes. Furthermore, identity is a critical component in programming, communication, and security.

Processing

Each node is typically equipped with several processing units, such as controllers, processors, and virtual processors, which, when combined with the operating system and software applications, comprise the computing intelligence of the node. Various microprocessors and sensor-specific software have been created specifically for SOs to execute multiple applications.

Storage

Because storage is a core function and essential component of a computing system, it is critical for IoT applications. Some SOs feature only a few hundred bytes of RAM and a few tens of thousands of bytes of non-volatile RAM (NVRAM), while others were built with significantly greater storage space.

Communication

Communication between network nodes must be effective and efficient in order for upper-layer applications to work. Many low-power protocols have been developed to permit communication between heterogeneous items via lossy and noisy channels to regulate communication between heterogeneous objects.

Networking

As opposed to communication, which relates to the choice of media and protocols between two nodes, it is concerned with transmitting and receiving data for data sharing and advanced collaborations (J. Paek, O. Gnawali et al., 2017). In general, networking considers a variety of elements. First and foremost, how are the nodes physically and logically connected? To put it another way, how should network nodes

be organized for data channel formation and selection? Second, how to allocate addresses to networked nodes efficiently? What kind of address should they provide each other? Third, what network functions must be provided, and how should these functions be provided?

Application

Software applications, web applications, mobile applications, and other applications must be built to accomplish a group of coordinated operations, tasks, and activities to connect with people. In order to produce utilities, applications must be coupled with IoT nodes and their software systems. An application may modify numbers, text, graphics, or a combination of these components, depending on the functionality and purpose for which it was created.

IoT Architecture

These IoT economic predictions represent the IoT's potential for revolutionary growth shortly, although it is an evolutionary process. That is to say, we should turn traditional equipment and existing appliances into smart objects, which will protect existing hefty investments while also opening up numerous new commercial potentials. Moreover, disseminating ubiquitous goods and services across a large region presents a unique opportunity for telecommunication providers (A. Vulpe, G. Suciu et al., 2017) to rethink and reconfigure their network (Y. Gadallah, M. H. Ahmed et al., 2017) to enable H2M, H2H, and M2M traffic flows. Following that, we will talk about IoT architecture, which is the IoT's backbone.

Various IoT architectures have been developed to address the necessity of connecting widespread and heterogeneous SOs since the IoT gained traction. Several layered models have been investigated to provide flexibility, but none are agreed upon as a reference model. As shown in Figure 1.4, three of these models, middleware-based, SOA-based, and business-oriented, are the most popular in IoT polls. The following are some additional details. Tan et al. presented a middleware-based IoT architecture for H2H, H2M, and M2M communication, with middleware's objective to collaborate among corresponding applications (L. Tan and N. Wang, 2010).

In contrast, Spiess et al. proposed a service-oriented architecture (SOA)-based paradigm in (P. Spiess, S. Karnouskos et al., 2009), claiming that service composition is to conduct new services and orchestrate business operations. Regardless of the two models, Khan et al. looked at a business-oriented model based on present trends and predicted future uses (R. Khan, S. U. Khan et al., 2012). This approach, interestingly, includes a business layer for managing the complete IoT system services, activities, data analytics, and infrastructure.

The common components of IoT are found in various architectures.

- Objects Layer: In some articles, this layer is referred to as a perception layer, symbolizing SO's goal of gathering and processing data.
- Object Abstraction Layer: From the object layer to the top layer, data is safely and rapidly sent through this layer. It's a thin layer.
- Service Management Layer: This layer provides APIs for programmers to deploy services while processing data to make decisions.
- Application Layer: This layer interacts with users at all levels, each with its own set of business requirements.

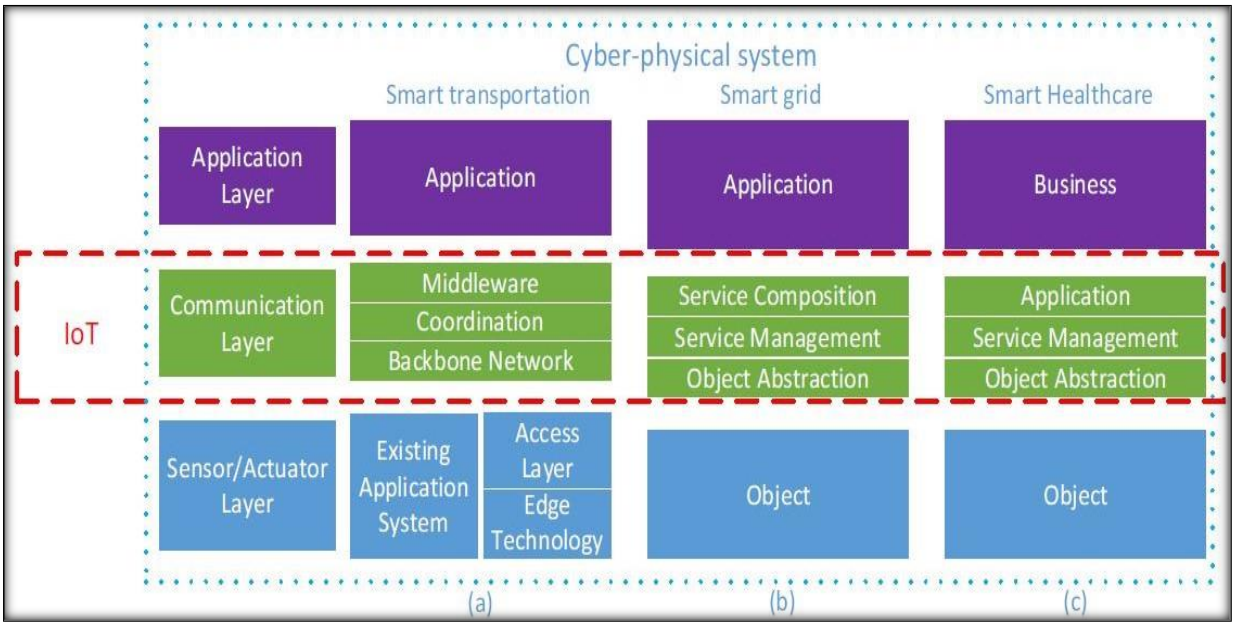


Figure 1.6: The layered IoT architecture model. (a) Middleware based. (b) SOA based. (c) Business-oriented (H. Yetgin, K. T. K. Cheung et al., 2017)

The architectures described in Figure 1.6 above were created to address the needs of a specific IoT context, such as wireless sensor networks (WSNs) (H. Yetgin, K. T. K. Cheung et al., 2017) (Z. Fei, B. Li, S. Yang et al., 2017)(I. Khoufi, P. Minet et al., 2017). When integrated with the appropriate underlying technologies for data delivery to an IoT platform, such architectures are supportable for many IoT applications. However, heavy tasks like service composition and management are difficult to complete because the bulk of these layers are designed to function on resource-constrained devices.

Similarly, heterogeneity is ignored by using a gateway device to interpret between different protocol stacks, which has been proved to be prone to serious concerns such as single point of failure (SPoF). More significantly, none of them adequately handle mobility, scalability, latency, security, and price features. A more generic framework to address such challenges is still required (J. Li, J. Jin, D. Yuan et al., 2018).

- **Cloud Computing**

For those who work in the computer industry, cloud computing is a crucial topic. It refers to the usage of computer processing power, storage, and servers worldwide over the internet. It provides a pool of resources that may be shared, such as data storage space, networks, computing power, and specific corporate and user applications. (Rithvik, K., Kaur, S., Sejwal et al, 2019). According to (A. Fox, R. Griffith, et al., 2009), cloud computing covers both web-based apps and the data centre infrastructure needed to support them. The cloud refers to the data center hardware and software. (SaaS), (IaaS) and (PAAS) are the three basic types of cloud computing. Such services prevent the need for businesses and individuals to invest in their massive servers or data center networking equipment. Furthermore, these devices do not require users to purchase or install software or programs.

When a provider makes their cloud open to the public, users can lease some services on a pay-as-you-go basis. Amazon web services, Azure, Google, IBM Cloud, Oracle are some of the most popular public cloud providers today (L. Dignan, 2018).

While being on the alternative, a private cloud is an internal data center available to serve a single firm. Unless otherwise stated, the cloud in this thesis refers to the public cloud.

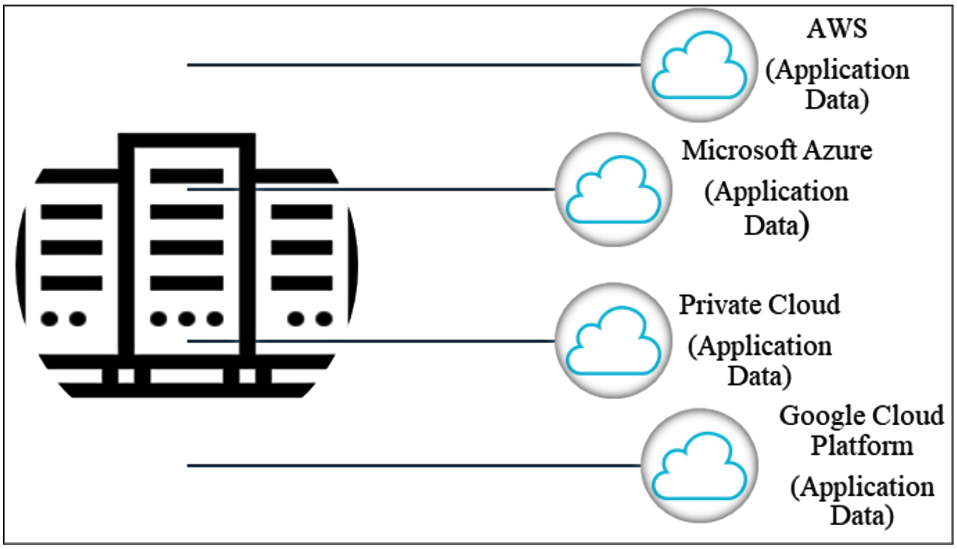


Figure 1.7: Cloud Service Providers (L. Dignan, 2018)

The best services offered by each company are paired with a multimedia product. It allows businesses to configure a business-specific network. Much less risk is posed with a multi-cloud architecture. If one host fails, an organization will continue to operate in a multi-cloud environment with other services over storing all the data at one site.

Table1.2: Comparison of Different Computing Technologies

Properties	**Cloud Computing**	**IoT**	**Fog Computing**
Delay	High delay than fog	High	Fog delivers minimal delay because it brings all services close to the edge.
Access Time	High: based on VM connection	High	High
Network Bandwidth	More bandwidth is required.	Low	Requires low bandwidth

Protection	Less Protection	Less protected	High Protection
Deployment	Centralized	Decentralized	Decentralized
Jitter	High	Very Low	Very Low
Computational Power	On-demand	Low	Ample
Storage Capacity	Ample	Limited	Limited

Architecture of Cloud Computing

It has two parts namely Frontend &Backend. We have showed the said architecture in Fig 1.8 below:

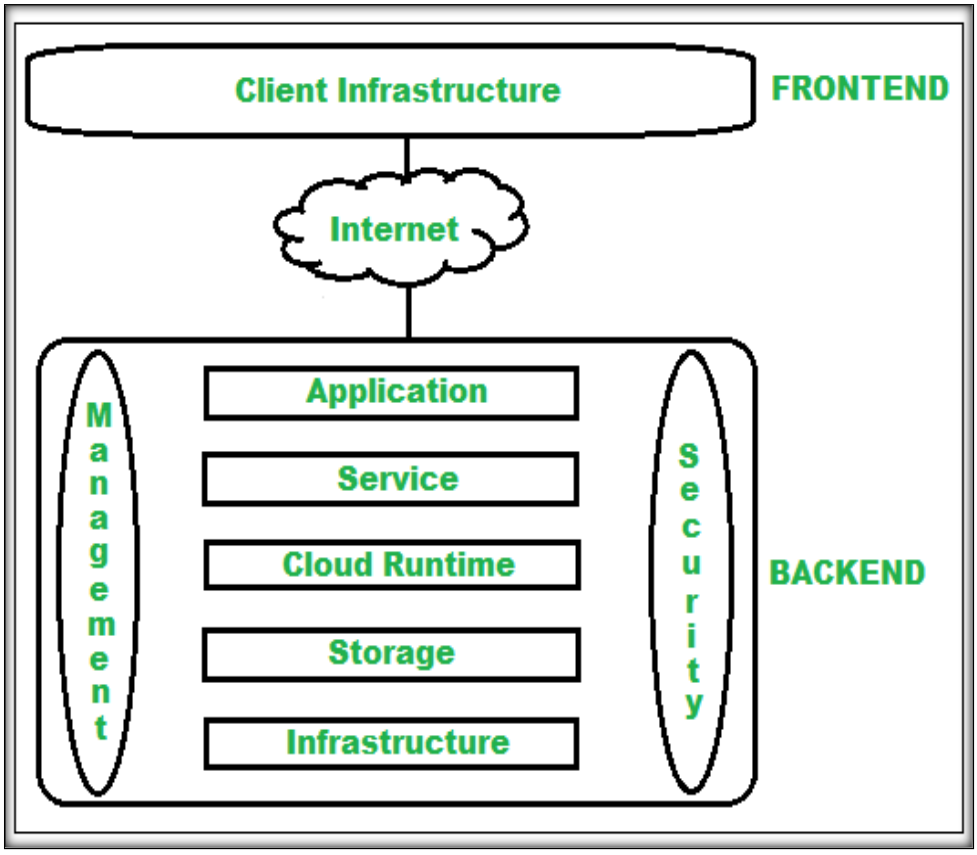


Fig 1.8: Cloud Architecture

SoA and EDA are merged in this hybrid system to provide a hybrid solution (Enterprise Data Architecture). This includes everything from the client

infrastructure to the application to the runtime to the storage to the administration of security and privacy.

1. Client Infrastructure:

This refers to the client-side of a cloud computing system. Cloud computing services and resources may be accessed through a single user interface and app. A web browser, for example, may be required if you plan on accessing the cloud platform. It refers to the front-end elements. You get everything you need to get started with cloud computing in this package.

2. Backend:

The service provider's cloud is referred to as the backend. In addition to containing the resources, it maintains them and implements security processes. Storage, virtual apps and virtual machines, and deployment methods are also included.

- ✓ **Application:** In the backend, an application is a type of software or a platform that a customer may access. That is, the service is tailored to the customer's needs on the backend.
- ✓ **Service:** SaaS, PaaS, and IaaS are the three primary backend cloud service categories referred to as "services." The user's access to services is likewise regulated by the system.
- ✓ **Cloud Runtime:** Cloud runtime refers to a virtual machine's processing and runtime framework in the back-end context.
- ✓ **Storage:** Backend storage and data management refer to delivering a flexible and scalable storage solution.
- ✓ **Infrastructure:** Infrastructure in the cloud refers to all of the servers, storage, networks, and virtualization software that make up the cloud.
- ✓ **Management:** A term used to describe the management of backend components including apps, infrastructure, and runtime framework is handling in the backend.
- ✓ **Security:** Installing various security solutions in the backend to ensure the security of cloud resources, systems, files, and infrastructure for end-users is referred to as backend security.

- ✓ **Internet:** An Internet connection acts as a mediator or bridge between the front and backend, allowing them to communicate and interact.

1.3Application Service Placement: Motivation and Challenges

Fog computing brings cloud services to the network's edge. The game-changing technique of placing IoT apps with optimal cloud and fog computing resources would have a tremendous influence on the deployed applications (Syed et al., 2016).It is vital that the edge of the network include storage and computing resources for IoT workflow applications that have low latency(Tseng et al., 2018).

The placement of distributed IoT applications in a fog infrastructure exhibits the following challenges:

1. Service placement is an important issue, requires special attention during the architectural design, and placing services (Ye et al., 2016) at non-optimal fog nodes, which reduces the performance.

2. The fog computation provides various levels of cloud service and economies of scale, as well as virtualization with the distribution of complex content and services and requests from multiple clients without breaking the Service Agreement. The process of finding and selecting the right computation nodes to place an application task be made more efficient while addressing non-functional constraints like network latency and QoS. We need a service placement mechanism that is flexible enough to meet varied goals. (Apat et al., 2018).

3. A fog infrastructure (Haouari et al., 2018) comprises heterogeneous devices and linkages with vastly different resource capacities. Furthermore, these devices are located at various network tiers and positions. Heterogeneous fog computing architectures that hybridize various edge node types can achieve greater scalability and lower costs than the centralized cloud architecture to support many Internets of Things (IoT) devices.

4. For proper functionality, IoT applications must adhere to a variety of limits and requirements. Consumable and non-consumable characteristics (such as network

latency) and various sorts of entities (such as components or communication channels) are subject to these restrictions, which result in a high degree of diversity.

5. Physical servers are projected to use about 45%, and networks use about 15% of the total capacity. As we design architecture for fog computing, fuel prices are rising, making the issue of carbon emissions and environmental effect more challenging as we consider the environmental impact. As there are multiple fog nodes, each with variable memory and storage capacity, the energy consumption of each node varies in the fog architecture. (Kitanov and Jinavski, 2017).

6. One advantage of Fog environments is low latency (Velasquez et al., 2018). It enables the deployment of different services that do not inherently work in the cloud with real-time and low latency constraints but also includes a new set of mechanisms ensuring the achievement of these low latency rates. The problem of fog computing is multi-level. One of the first challenges (Brogi et al., 2019) in the broader sense is shaping the orchestration component, which must be capable of deploying the cloud and managing tasks within the environment.

7. We must make placement selections quickly to respond to deployment requirements. When it comes to large-scale difficulties, however, solving the placement problem becomes increasingly challenging due to an increase in the number of computation devices and the number of applications.

The problems outlined above have prompted researchers to focus on developing a Fog computing model for minimizing response time and energy consumption while ensuring reliable operation in a diverse geographic context.

Benefits of Cloud Computing Architecture

Some of the benefits of this technology are as follows:

- ✓ It decreases the quantity of data processing necessary and streamlines the overall cloud computing system.
- ✓ It contributes to a high level of security.

- ✓ It starts to be modularized.
- ✓ As a result, disaster recovery tends to be improved.
- ✓ It allows people to get information quickly.
- ✓ It reduces the expense of running an IT department.

1.6 Thesis Contribution

This thesis develops a framework for placing deadline-sensitive applications with minimum energy consumption and shows the significance of such a system. However, the optimization task of allocating applications to computation nodes becomes a complex problem to solve that requires optimization methods. We leverage the use of Metaheuristics, a particularly hyper-heuristic algorithm capable of taking advantage of optimization of IoT application service placement with deadline-aware and minimizing energy consumption in Fog Computing. The simulation of the suggested model will be executed using iFogSim, an Open-Source Java-based network simulator. It works by stimulating the surrounding consisting of many IoT devices, and corresponding fog based we will show node Simulation results for justification.
Further, the testing of simulation results will be done for different output parameters. It will verify that the proposed policy is superior to previous state-of-the-art works. The following is a list of the thesis' key contributions.

1. A survey and taxonomy of the state-of-the-art Inservice placement in a fog environment have been designed to meet the specific needs of applications that objectively concentrate on latencies like augmented reality and IoT applications that generate large volumes of data unworkable for analyzed remote cloud data centers.
2. To design a practical framework for fog application placement, it can take the load in another fog node by appropriately positioning the device, inflating the heat that a specific fog node produces.

3. To design an efficient service placement algorithm concerning deadline awareness and energy efficiency. We have used the heuristic (Yates et al., 2019) algorithms to improve the scheduling problem's performance.
4. Extensive simulation-based performance evaluation of the proposed work is compared with the state-of-the-art algorithms across the multiple parameters along with the analysis and discussion of the deserved results.

1.7 Thesis Organization

Figure 1.9 shows the pictorial representation of the whole thesis. The overall schema of the chapters is formulated in the following manner:

Chapter 1 gives a concise introduction about the term Fog Computing, service placement and describes the various parameters used in the thesis work.

Chapter 2 depicts the literature review related to the fog computing environment, different application models, service execution models, and placement and orchestration models. A comparative analysis of the existing simulating tools is also discussed.

Chapter 3 The importance of a fog environment in the application module deployment is discussed. A multi tire fog architecture based on the suggested Deadline oriented Service Placement (DoSP) model is illustrated. In terms of response time and resource utilization, the DoSP was compared to the cloud-only and EdgeWard placements.

Chapter 4 provides the deadline-aware and energy-efficient dynamic service placement (DEEDSP) technique for IoT environment to minimize energy and response time. This proposed system is implemented by using simulation tool, i.e., iFogSim. The performance has been measured and compared based on various performance metrics.

Chapter 5 summarizes this thesis and proposes future research opportunities for

further study.

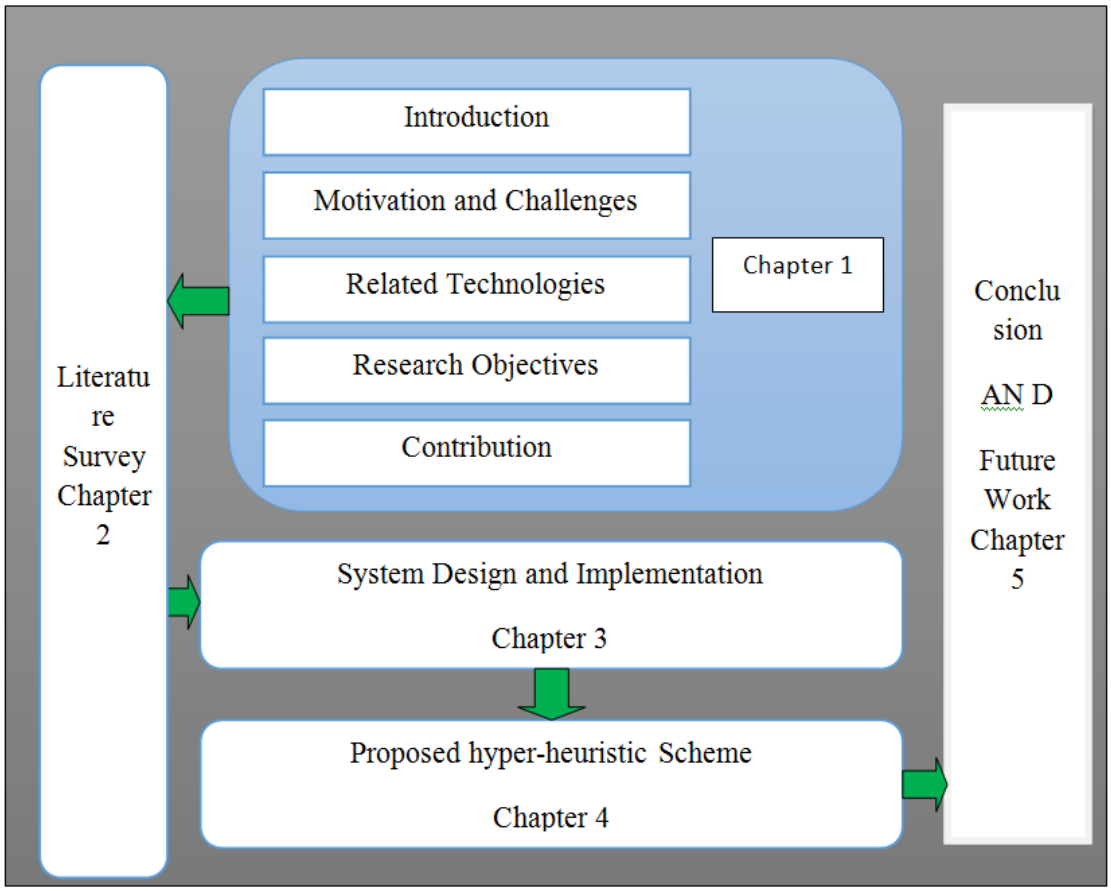


Figure 1.9: Thesis Organization

Chapter 2

Literature Survey

As a result of the IoT, virtualization and Cloud computing firms are now collaborating to deliver on-demand computer resources while also establishing ever-present computer systems in public, private, and business settings. F fog computing is one of the primary platforms for addressing IoT-related concerns to fulfill the demand for such computing tools. As the compute nodes in fog computing are heterogeneous and distributed, Fog computing services must cope with various issues (Apat et al., 2018).

Cloud computing and network edge services are combined in fog computing, a recent concept Fog computing is characterized by the use of network computers, temporary storage, and computation near to the end user. With one or more fog nodes, data can be calculated (Agarwal et al., 2016). As the demand for data processing grows, more fog nodes can be added. Routers, set-top boxes, gateways, and access points can all be used to integrate fog computing resources.

In this chapter, we will do in depth discuss of the Fog Computing Environment. This environment consists of various components such as Resource Provisioning, Application Placement & Application Scheduling. There are three types of fog application placement management: Fog Management, Cloud Management, and Fog Management in Both.

We will be focusing on the 2nd aspect, i.e.. Service Management in Fog has 2 categories of controls, including the Centralized & Distributed Control. The mechanism is further classified into Single Level, Two Level & More Level wherein the First part is further classified as Resource Utilization in Single cluster & in Neighbor cluster. The neighbor cluster is further categorized into 2 processing models named Batch Processing & Stream Processing.

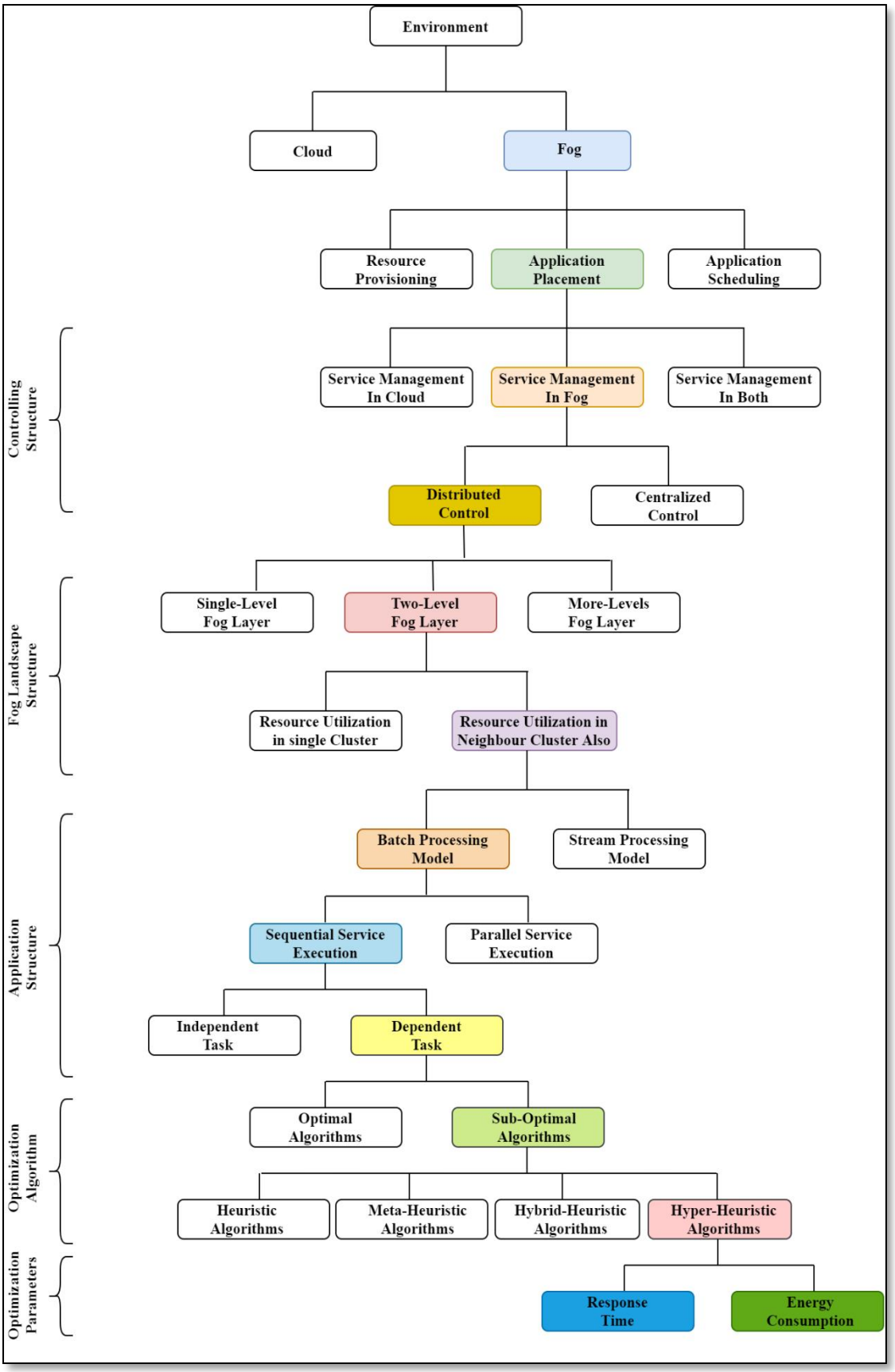


Fig 2.1: Literature Classification

2.1 Fog Computing Environment

Fog computing, also known as fogging, is a type of computer architecture that is focused on decentralization and communicates data-generating devices to the cloud. Here, we will discuss the 3 main environmental factors of Fog Computing: Resource Provisioning, Service Placement & Service Scheduling.

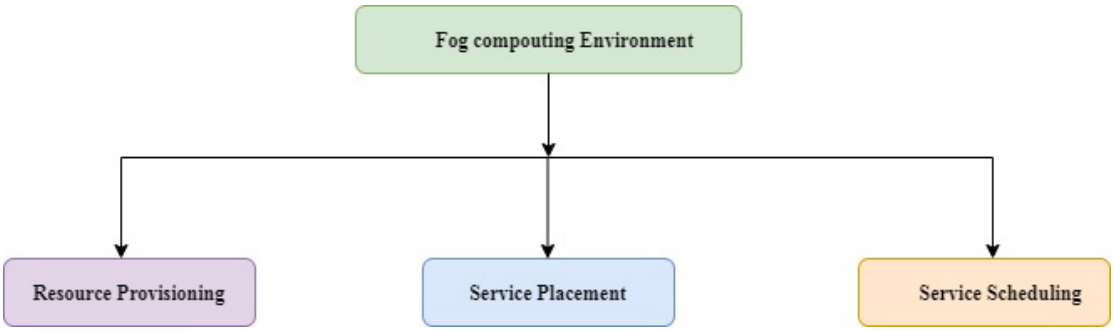


Fig 2.2: Computation Environment Classification

In recent years, the cloud computing, and fog computing have all received a great deal of interest from both business and academics. (De Donno et al., 2019) investigates the origins and evolution of important current computing paradigms as Cloud computing, Internet of Things, Edge computing, and Fog computing. Fog computing and cloud computing cannot help by themselves due to specific restrictions. Both technologies, however, are part of Smart City's intelligent IoT network.

The Internet of Things (IoT) played a critical role in creating intelligent cities in the twenty-first century. With the growth of IoT, data is increasing at a rapid rate. As the amount of data expands, so does the requirement for storage. Saving and retrieving data from the cloud takes a substantial amount of time. Fog computing is a strategy for minimizing the latency of data access to and from cloud resources. At the network's endpoint, Fog Processing provides storage, computation, and networking capabilities. Fog nodes also offer some basic device capabilities(Saroa et al., 2018). It also serves as a foundation for a city-wide waste management system. Fog computing could be a smart technique to handle rubbish collection in the city.

Fog computing brings cloud computing to end devices, making it easier to support time-sensitive, location-based, and large-scale applications. This research proposes a

fog computing ecosystem and develops and analyzes a real-world testbed for several usage cases.

(Hong et al., 2017) are looking at three different scenarios for how to use and optimize our fog computing platform. These are the possibilities for use: (i) content distribution to impaired networks, (ii) computer fog from crowdsourcing, and (iii) programmable internet analytics. Authors use and develop open-source projects to create our nebulizing platform.

(Shi, Y., Ding et al., 2015) This study analyses the characteristics in fog framework and the resources that it may be able to provide both within and outside the healthcare system. It increases the real-time quality of services, usability, security, and privacy of online applications by providing low latency and local knowledge. (Rahman G et al., 2018) summarize and define the Fog Computer Model's architecture, functionality, related paradigms, security problems etc.

(Qi et al., 2019) have designed the architecture which can be applied at the digital twin-story level, giving new manufacturing applications a clear perspective.

(Gao et al., 2017) introduced a hybrid data distribution system that integrates Fog computing's methods to software-defined networks and DTNs to explain the architecture and handle high content updating costs. The Cloud is the control unit for processing content update requests and arranging data flows.

Fog computing has surpassed significant cloud computing techniques and set new trends and heights for current networking worldwide. It poses a risk to data and service privacy and security. We may find a summary of current problems and challenges in fog computing here (Aljumah et al., 2018).

The state-of-the-art and similar works that fall under the same banner are investigated in this book (Stojmenovic et al., 2014). Protection and secrecy issues are also exposed in the new Fog computing concept. As an example, a man-in-the-middle threat is frequently discussed in the context of fog computing security, as is a remote code execution assault. We examine the attack's persistent characteristics by looking at its CPU and memory use on the Fog computer. In addition, fog computers and their

implementations in different real-world scenarios, such as the smart factory, smart traffic lights in motor networks, and existing software communication systems, are discussed in this article.

2.1.1 Resource Provisioning

In the different existing cloud computing platforms, efficient resource allocation is a difficulty. (Padmavathi et al., 2017) provides a fuzzy technique that dynamically assigns the required resources to maximize the usage of available resources. Unlike other algorithms, such as the FCFS priority-based algorithm, our monitoring agency tracks the needed resources and assigns them efficiently and according to availability. This grouping of incoming requests is done so that the maximum amount of resources is allotted. Their performance is assessed to achieve resource allocation efficiency.

(Skarlat et al., 2016) formulate the optimization problem by taking into consideration existing fog / IoT landscape resources. The goal of this topic is to make fog-based computing resources available for time-sensitive applications. In comparison to a reference strategy, the findings of the resource supply model reveal that delays are reduced by 39%, resulting in shorter journey times and machinery.

(Rakshith et al., 2018) offer a resource provisioning system that allocates resources and manages registered services in a complicated topological design. Fog computing, according to the findings, reduces total service time.

For storage system management, the majority of the data was saved on Cloud by the businessmen as it provides remote access to the Client on a usage-based basis. In that instance, we can rent the computer for their needs rather than purchasing any physical equipment from the user. In the cloud computing environment, resource provisioning is a difficult operation. The Cloud resource delivery framework for Big Data applications is discussed in (A. Sheshasaayee et al., 2017). Many users in the Cloud consider resource management to be a critical duty. There are numerous techniques of resource provision that are currently in use. For individuals with vast enterprises in need of resources, this recommended structure is just too valuable. It uses MapReduce to plan tasks, which reduces both runtime and response time.

The supporters of resources are all rational people who wish to get the most out of their resources. The owners of resources will not contribute unless there is an effective incentive system in place. Based on the previous issues have proposed architecture mostly on the neurons of the human brain is proposed by the properties of fog and cloud data centers. This system structure's activity is studied (Sun et al., 2017).

2.1.2 Service Placement

To maximize the decentralization in fog computing, a strong framework for positioning and reaction time and the utilization of resources in services has been designed. To support IoT service delivery, (Tran et al., 2019) suggested a new multi-tier fog computing architecture. The proposed approach optimizes the performance of IoT services by utilizing the network edges of virtual resources by improving the reaction time, energy, and cost savings. The proposed technique maximizes fog use while minimizing latency, energy consumption, network load, and operating expenses according to experimental data.

A conceptual fog computer framework is presented to improve the resource provisioning system model regarding resource cost, dependability, and service accessibility (Skarlat et al., 2017). Fault tolerance strategies can improve the architecture, allowing for more mobility in the Fog. It may be necessary to investigate parallel heuristic algorithms to identify a suitable alternative to the exact optimization strategy.

(Taneja et al., 2016) This paper describes an adaptively deployed resource-aware data analytics platform in a fog framework, that could reduce users' network expenses and reaction time by deploying the analytic platform to operate in the Cloud. (Kayal and Liebeherr,2019) A collaborative effort suggested a distributed service placement method for fog applications to maximize energy use and network resource usage. The proposed Neighbor Exchange Local technique, unlike cloud placement approaches, Neither central control nor global state data are required. The Markov approximation approach is used in the algorithm design to resolve the combinatorial optimization aim.

A profit-aware application placement policy is presented (R.Mahmud et al., 2020). Simultaneously, the program boosts its gross and net income by expanding to suitable instances without surpassing its time limits. This technique employs a refund scheme to lessen the impact of an SLA breach on users. The compensation scheme is based on machine production and helps both producers and consumers. We can use any ILP solver or best-fit heuristic approach to find the application-instance map in the proposed policy. To demonstrate the efficiency of the suggested technique, the heuristic approach is used, and its performance is compared to numerous existing application placement strategies in an iFogSim simulated environment. According to the testing results, the proposed technique enhances gross and net income, time to wait, and a QoS satisfaction level.

2.1.3 Service Scheduling

On a gateway-based edge computing service paradigm, (Tseng et al., 2018) proposed to reduce data transfer latency and network bandwidth to and from the Cloud. To alter the work program of the edge gateway, the lightweight virtualization technology Docker can be used to assign computing resources on-demand. The edge gateway can handle service needs in the local network. The suggested edge computing model eliminates the calculating charge associated with typical cloud service models and increases the efficiency of edge computing systems. This paradigm is also applicable for many innovative applications in the 5 G and beyond the cloud-based computing environment.

It is the authors' intention to investigate at a fog-enabled software-defined embedded platform wherein the job images are kept on storage servers and computations are executed either on an IoT device or on computational nodes. It is vital to have an efficient process allocation management technique with a short job completion time for enhance customer experience. This research looks into three issues throughout this study relating: 1) task scheduling 2) resource management and 3) how to manage I/O interruption queries across storage servers. They're considered at this as a non - linear optimization challenge. They propose a computationally

efficient technique based on this approach, which has been validated through simulated experiments (D. Zeng, L. Gu et al., 2016).

(Name et al., 2017) suggest a policy to ease the difficulty of user mobility by reducing the time necessary to react to an application using an effective algorithm that works with the Seamless Handover Scheme for Mobile IPV6. (Yin et al., 2018) develop a task scheduling method in the fog node to ensure timely task completion and an optimal number of concurrent activities. Finally, a mechanism for reallocating task delays based on container properties is proposed. The suggested task planning and reassignment technique has demonstrated that it can successfully reduce task delays and enhance task concurrency in fog nodes.

The resource planning problem has been investigated (Mtshali et al., 2019). In the fog computing environment, many cloud computation policies have been examined. Most of these scheduling algorithms are evaluated using multiple criteria, including energy consumption, implementation time and network utilization and the amount of data that is sent and received. The First Come First Serve, shortest job Scheduling algorithm, GP, and RR policies were adhered to in the investigation of the multi-objective optimization problem. Compared to the SJF, GP, and RR, the FCFS improves efficiency by 11 percent, 15 percent, and an average delay of 7.78 percent and 4.4 percent, respectively. FCFS is thus the optimal planning approach for task computing and job efficiency that uses the least energy.

(Xuan-Qui Pham et al., 2016) People have come up with a way to use both cloud nodes leased from cloud providers and fog nodes that are owned by cloud providers. A scheduling approach is proposed that ensures application performance while simultaneously lowering the obligatory cost of employing cloud resources.

2.2 Orchestration Model

Container scheduling, cluster management and possibly the provisioning of extra hosts all fall under the umbrella of orchestration. In this section of the Orchestration model, we will discuss the 3 categories: 1. Service Orchestration in Cloud, 2. Service Orchestration in Fog, 3. Service Orchestration in both cloud & fog.

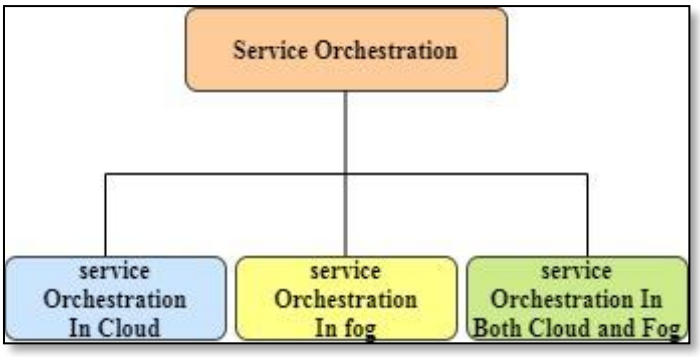


Fig 2.3: Service Orchestration Classification

2.2.1 Orchestration Model in Cloud

The topic of Cloud orchestration was investigated by (Bousselmi et al., 2014). SaaS services orchestration or IaaS services orchestration were the two options offered. There is a lack of attention paid to the administration of orchestrated software and services i.e., scheduling, installation, dependencies and the appropriate provisioning of Cloud resources.

Senna et al. (2014) introduced hybrid cloud architecture for orchestrating Hadoop programs. The Cross-Domain Hadoop Cluster can be automated; the supply of used files can be made possible. It can manage the application, and the results can be made available to the user through its components. The proposed model informs the user about repeated environmental behaviors. The approach aids in the identification of the best available resources, including the utilization of leased resources on demand. It also demonstrates how leased resources can be successfully managed, with VMs being deleted when no longer needed. The proposed architecture enables the creation of solutions for complicated contexts like hybrid clouds with Hadoop-as-a-Service.

Distributed service orchestration architecture (Sebrechts et al., 2016) is proposed to manage better the sophisticated orchestration logic required in such scenarios. To address the different components' adaptability, a novel service-engine-based technique is given. Cloud modeling languages, autonomous devices, image schedulers, and PaaS systems can all benefit from a hybrid integration strategy. The notion is included in Tengu, a distributed data experimentation platform, making it scalable and stable.

The authors created a new CAMP platform that enables the specification, deployment, and management of applications and components across diverse cloud environments. (Alexander et al., 2017). Policies written in declarative YAML are used in the proposed platform. Declarative policy directives, which specify actions we can do in response to policies, are also available with an eye on the extended platform's policy processing. Descriptive policy representations and fully automated orchestration of applications across different cloud providers are two ways CAMP extensions help platforms that are compatible with CAMP work better with other platforms.

(Alexander et al. 2017) proposed a new solution that included combining the upcoming TOSCA and CAMP standards and extending CAMP to orchestrate a multi-cloud application using declaration policies. There are other additions to the CAMP platform that bring the standards closer together for easier integration. The proposal proposes a cloud-based solution that allows cloud applications to extend and run across various clouds by supporting the modeling and deployment of cloud-based applications.

(Carnevale et al., 2018) suggested Osmotic Computing Paradigm-based orchestration architecture. Finally, start by looking at IoT applications that are installed in different places like a graph of Microelements, Osmotic architecture for multiple tenants capable of performing a workflow to drive MELS registration and transition to a computing application for the study process was described (MELS).
(Brabra et al., 2019) proposed a new model-driven orchestration approach that:

1. To characterize cloud resource objects, TOSCA is used.

2. Translates these items into native DevOps data using a model-driven translation approach.

3. Provides Connectors with a way to connect the Operations teams contents created to the underlying DevOps technologies.

They looked at the POC implementation and several applications to see how efficient it was and how well it transitioned. As indicated by the study, the use of TOSCA for

resource object representation and MDE methods for mapping between TOSCA and DevOps tools covers the existing complex implementation and inherent variability in these tools. They'd cut the amount of time spent on design and expert-led physical work in half.

2.2.2 Orchestration Model in Fog

For the various domains of infrastructure management, a hybrid strategy (Velasquez et al., 2017) is offered to manage the Fog by integrating orchestration and choreography. The architecture described here, framed by a SORT'S project, allows for a global view that permits broad optimization and automated low-level dynamic reactions. Furthermore, we can use different orchestrator instances to achieve different optimization goals based on the applications. The Fog paradigm (Velasquez et al., 2018) has been overhauled. Several Orchestrator architectures for the Fog service have been introduced and how the Fog issues are tackled. A comparative analysis of the various architectures is provided.

An approach for autonomously controlling and organizing fog computing networks was proposed by Dlamini et al. (2018), who used a finite state machine to achieve this goal. The scheme is based on prior work that linked the Open Fog architectural style towards the ETSI framework to address fog computing orchestration challenges, and also the special features with state machines providing information about the network event responses. This suggested methodology seems to be a key option to learning algorithms, that has been widely known in networking to resolve orchestration issues and provides sufficient connectivity intelligence without proceeding it through different phases required to establish a machine learning feature to support network selection, and it provides significant network abilities without proceeding through all different phases required to implement a machine learning process to support network selection.

We may find an overview of the INPUT in (Wamser et al., 2018). (Supporters of the Next Generation of Personal Cloud Services with In-Network Programmability). The Fog computing-architecture approach uses the above technologies in its Open Volcano implementation to give future network cloud personal Internet access. It

provides processing and storage capacity for end-user and edge cloud services and IaaS and PaaS templates with severe time and performance constraints. It provides a basic, programmable Ethernet domain that is based on SDN. The focus of this study is on multi-service orchestration and monitoring, which is an important component of any computer architecture.

2.2.3 Orchestration Model in Both

Academic works regarding the paradigm change to Fog, were assessed by (Osanaiye et al., 2017). By categorizing fog computing applications between real-time and just next to real time, taxonomy of diverse fog computing applications are offered. Despite the lower latencies of these services, the Cloud must be extended to the network's edge, resulting in fog computation.

Both computing technologies i.e., Cloud and fog are virtualized systems that provide resources for computation, networking, and storage. The concept of an adaptive pre-copy live vm migration within Xen discussed here is driven by the expectations for high-end users' availability. The suggested method predicts overall stop and copy points downtime while in an iterative pre-copy.

(Ren et al., 2019) provides a thorough examination of these new computing paradigms from the perspective of end-of-cloud orchestration. The designs and properties of several computer paradigms are first introduced and compared. Then, to address cutting-edge studies on uploading, caching, security, and privacy.

2.3 Placement Controller

An entity in Fog computing that oversees various application management actions is referred to as the placement controller. It also aids in measuring the time between requesting and installing the applications with in Fog computing environment, which significantly influences efficiency of cyber physical systems. Within Fog, there have been two kinds of placing control systems.

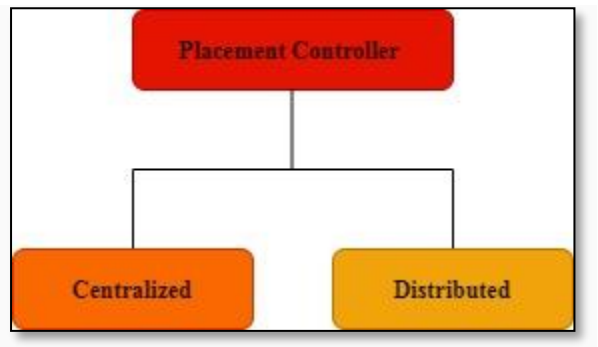


Fig 2.4: Placement Controller Classification

2.3.1 Centralized Controller

IoT applications require strong analytics approaches like Deep Learning in order to extract useful information. Existing IoT apps send data to data centers with a lot of processing power. It may, however, clog networks, overburden data centers, and expose security vulnerabilities. Connecting data centers to end users is the goal of our platform. We execute distribution of analytical algorithms across the devices without transferring any data to the data centers. To overcome the hurdles of implementing such a platform, we study the challenges and carefully adopt prominent opensource projects.

We next run extensive tests on the platform that has been implemented. Findings reveal the necessity of making decisions about (i) how to deploy an app across numerous devices, (ii) the overhead incurred by various factors in our system, and (iii) the advantages and limits of distributed analytics. (P.Tsai et al., 2017).

Tsai et al. present a TensorFlow and Kubernetes-based analytic fog computing architecture. A centralized server and fog devices make up this RaspberryPi-based architecture. Cloud servers, edge infrastructures, and fog devices are all detailed in their work. The application model is programmed in TensorFlow. Using Kubernetes, the fog landscape may be managed and resources found and deployed by the system's operator containers. Our approach to orchestrating differs from Kubernetes in that it is distributed rather than centralized. When compared to Kubernetes, fog nodes are smaller and easier to deploy on edge devices.

Dynamic distribution topologies for Internet of Things (IoT) applications may be developed using the DIANE framework. (M.Vogler et al., 2016). This framework keeps track of the deployment infrastructure, organizes it into categories based on available resources, and saves the data for subsequent study. The framework dynamically deploys topologies for IoT applications and enables monitoring by Using a rule-based mechanism. While DIANE provides a rule-based approach for resource provisioning, we pose a concrete optimization problem in our study. We also study and implement a fog landscape's hierarchical structure, focusing on creating communication between different fog landscape devices and offering application management.

2.3.2 Distributed Controller

In contrast to the centralized controller, the distributed placement controllers govern the Fog nodes from a local perspective on the Fog environment, rather than the centralized controller's perspective.

For applications in smart cities, the authors propose a multi-tiered fog computing architecture; this research proposes a multiple layer of fog environment supporting smart cities, as well as a large volume of growing data analytics service. This multi-layer fog is made up of application - specific fogs and reserved fogs using spontaneous or reserved computational resources, correspondingly. The proposed unique fog computing paradigm, which includes well-defined essential features, can help to overcome the challenges of specialized computing environments and delayed cloud services response. Researchers undertake analytics benchmark tests over fogs generated using Raspberry Pi equipment that used a distributed computational engine in order to measure overall processing capacity of varied analytical workloads and build efficient workload models.

Placement controls, and resource provisioning algorithms that are fully aware on Quality of service are meant to make advanced analytics services more accessible and useful. There are several factors to consider while designing a Quality of Service (QoS) strategy. This flexible framework simulator is built to evaluate proposed fog-based analytical services with QoS control mechanisms. Experiments indicate these

analytical services are effective across multi-tier fogs, and that the recommended QoS techniques work. When compared to a cloud-only model, fogs can greatly improve the efficiency of smart city analytical services to customers in terms of task blocking likelihood and service usefulness (J. He, J. Wei et al., 2018).

Many people are interested in fog computing because of the problems with cloud computing. There is a possibility that fog computing could be used to move computational power closer to the data that is being created by the devices on the edge of the network. This is because there are more and more devices at the edge of the network, which means that they need quick and local processing. This vision paper overviews current research activities, explains applications where fog computing is useful and suggests future difficulties that must be overcome to reach its full potential (M. S. de Brito et al., 2017).

2.4 Fog Architecture

Fog architecture makes use of end-device services (switches, routers, multiplexers, and so on) for computing, storage, and processing. It consists of 2 major components: Fog Layers & Clusters, which we will discuss in this section.

2.4.1 Fog Layers

2.4.1.1 Single

Heterogeneous fog computing systems, including multiple edge nodes to support many Internets of Things (IoT) devices, provide superior scalability and cheaper costs than cloud computing's centralized architecture. This paper presents an energy-efficient scheduling approach for IoT workflows running on heterogeneous fog environment. Our first step is to develop the integer programming model that reduces overall energy use. Rather than being used directly for computing, the ILP model's goal is to disclose essential components for lowering energy use in a distributed system. Based on model data, authors develop new energy minimization scheduling method that combines numerous strategies to achieve relatively close to the actual values. To create and test their concept, authors used simulations. The results of the

tests show that, it achieved near-optimal energy utilization while operating at a much rapidly (H. Y. Wu, C.R. Lee et al., 2018).

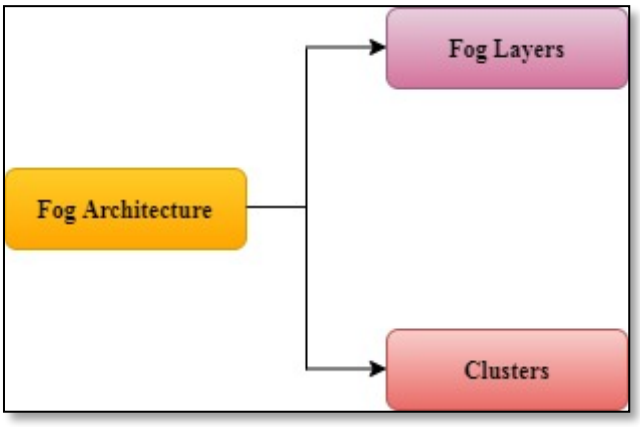


Fig 2.5: Elements of Fog Architecture (Sri Raghavendra et al.,2021)

In order to accommodate the growing number of Internet of Things smart devices, the researchers suggest a work scheduling mechanism in the fog layer that is mainly focused on priorities. It will enhance efficiency and cut down on costs. Priority-based job scheduling techniques and the appropriate architecture are clearly mentioned. According to performance evaluation, the proposed technique reduces overall reaction time. It significantly lowers total cost compared to existing job scheduling algorithms. The priority-based approach is applicable in many application domains. This study is essential to the growing fog computing technology (T. Choudhari, M. Moh et al., 2018).

Using a fog - cloud architecture, the researchers have been trying to deliver Internet - of - things types of services to their customers. On the basis of this architecture, a unique service placement approach has been developed that increases service distribution on the Fog cluster by leveraging context-aware information about the context, duration, and service standards to position services in the most appropriate locations. With many simulations used for smart grid applications, they experimented with testing the proposed approach. Compared to the traditional cloud computing model, the results prove that the proposed technique is helpful in decreasing latency, energy usage, and network stress (Minh, Q. T et al., 2017).

In order to promote the health and well-being of humanity, IoT based systems provide a realistic and well-organized approach. Technological solutions built upon wireless sensor network are becoming increasingly more accessible as the number of elderly citizens who needs health services on a daily basis rises. In such a framework, the energy consumption of a sensor node is critical. One possible solution to these problems involves the use of the gateway having fog computing technology. Fog computing services include distributed processing management, electrocardiography extraction and classification, graphical interface including access management, as well as pushed notations. Fog computing may reduce the burden on a cloud server, and a sensor node can save up to 50% of its power usage. The system also employs fog computing to assure that the obtained health information can be presented and evaluated in real - time basis, however if the gateway and cloud server are unavailable (Gia T. N., Jiang M et al., 2015).

2.4.1.2 Multiple

In order to reduce operating expenditures and system dependability, lowering energy consumption for Cloud services has been an essential priority. Anyone can get any service they want through the Cloud. This is called XaaS, and it's a way to get new real-time Cloud services. The authors examine power-aware virtual machine provisioning for real-time services in this research. These are the things they want: (i) The users request consider as a virtual server and (ii) use DVFS (Dynamic Voltage Frequency Scaling) techniques to deliver virtual machines in data centers. They offer many power-saving measures and test their effectiveness using simulation data. (S. Sharma and H. Saini et al., 2019).

The Internet of Things (IoT) has enabled computers to penetrate these most mundane aspects of everyday life, leading in a paradigm change to the way services are developed and deployed. A consistent method to applications deployment is critical for optimal network infrastructure use, especially when the volume of the impact grows rapidly. A typical Internet of Things application comprises multiple modules that operate in tandem with active interdependencies; these modules are typically hosted in the Cloud in worldwide data centers. According to the findings of

this study, a Module Mapping Algorithm is proposed that facilitates actual resource utilization in communications infrastructure by effectively installing Software Applications within Fog Infrastructure for Internet of Things-based services. Fog computing allows application modules to be placed closer to the source on Fog devices because processing is spread effectively anywhere in Cloud and Fog layers. The whole job's output may be used as a benchmark in fog computing, such as for evaluating Service quality and Service Performance Objectives for Applications. This method is general and may be used with a variety of standardized IoT programmers operating on various network topologies, irrespective of load (Taneja et al., 2017).

As a result of its emphasis on delivering cloud-based services efficiently and effectively towards the users in an efficient and timely way, the fog computing paradigm has attracted a great deal of attention in research circles. The majority of fog computing's physical equipment, known as fog devices, are geographically dispersed, resource restricted, and diverse. A system of fog nodes may be used to install large-scale programs that are broken down into smaller, more manageable units called "Application Modules." In this paper, authors describe a fog-based latency-aware Service Module management strategy that takes into account the changing service delivery latency and quantity of transmitted data to be analyzed per unit of time for different applications. In the fog environment, the policy aims to ensure application Quality of Service (QoS) while fulfilling client service timelines and improving resource efficiency. Using iFogSim, we simulate and evaluate our proposed strategy together in synthetic fog environment. The simulation findings show a significant performance improvement over other latency-aware approach (R. Mahmud et al., 2018).

The Internet of Things connects billions of devices (IoT). These gadgets produce a lot of data, which puts much strain on traditional networking infrastructures. Using edge computing, in conjunction with cloud computing, you may minimize network latency and energy usage by relocating computational and storage services that are resource heavy to edge devices. Meanwhile, software-defined networks (SDNs) might help sophisticated IoT-based applications enhance their quality of service (QoS).

Building an SDN-based computing platform, on the other hand, poses major challenges, high latency sensitive, more computation complexity, and high availability needs of new applications are not being met by present computing paradigms. As a result, this study presents a collaborative work offloading technique based on cloud-mobile edge computing (MEC) and service orchestration (CTOSO). In order to make distinct offloading mechanisms for jobs with varied resource needs and delay tolerance, the CTOSO system first assesses the cost of offloading tasks in terms of computation, communication, and latency.

Additionally, the CTOSO system has an ODaS (orchestrate data as services) approach that is based on SDN. MEC servers arrange the gathered metadata into high-quality services, decreasing the network burden generated by cloud resource uploads. On the one hand, data processing is performed to the greatest extent possible at the edge layer, allowing for load balancing and minimizing the risk of data leaks. Comparing the CTOSO approach toward the RDTO scheme and the MCJO scheme, we find that it decreases latency by 73.82 percent to 74.34 percent and energy consumption by 10.71 percent to 13.73 percent. (N. K. Giang, M. Blackstock et al., 2015).

2.4.2 Clusters

2.4.2.1 Single Cluster Resources

The objective of fog computing is to enable substantial Internet of Things applications (IoT). The demand for vital information to be transmitted across disparate IoT devices via event alerts, such as traffic conditions observed when undertaking traffic monitoring, is a critical feature of such systems. In the Internet of Things, the CEP paradigm is a strong tool for bridging the existing gap between IoT devices that monitor sensor data and IoT application users that get event alerts. However, deploying CEP in a highly dynamic IoT environment, such as one with mobile and heterogeneous devices, needs a CEP system with an extremely flexible architecture that can adapt to changing requirements and operating conditions. They demonstrate "how to boost adaptability in a fog environment by utilizing a technique called mechanism transitions" using a CEP use case. By describing and analyzing two

typical IoT use scenarios, we highlight the potential for mechanism transitions. They discover and investigate potential promising mechanism transitions as part of CEP. We experiment to determine the optimal operator placement and demonstrate how transitions adapt to competing performance objectives (Luthra et al., 2019).

In conjunction with the rise of the Internet of Things, fog computing has been promoted as a powerful complement to cloud computing for addressing both data and communications requirements of the IoT. The communication and cooperation between both the cloud and the fog has received a lot of attention recently (cloud). The goal of this paper is to examine service allocation in a fog computing system, in which a fog provider may efficiently execute clients' offloading applications by leveraging collaboration between its fog nodes and cloud nodes. In this cloud-fog environment, we first discuss the work scheduling problem. Then authors present a heuristic-based strategy with the primary objective of maintaining a balance across cloud resource makes span and computation cost. Their analytical results indicate that the proposed algorithm generates a greater trade-off value than the other methods (Xuan-Qui Pham and E.-N. Huh, 2016).

Network edge computing, communication, and storage are all made possible by the fog paradigm. Resource allocation is critical in this heterogeneous and distributed environment. As a result, scheduling will be difficult for increasing productivity and allocating resources to assignments appropriately. Use of a classifying data mining method to assist in computer task scheduling. The development of the I-Apriori method has been aided by a revolutionary classification mining approach based on the apriority algorithm.

There are also new task scheduling models and a new method for scheduling tasks in the fog, called TSFC (Task Scheduling in Fog Computing). The task set's least completion time for every task is coupled with I-Apriority algorithm's associate criterion. Furthermore, the job with the quickest completion time is picked to be executed somewhere at fog layer with the quickest completion time. Finally, we do practical simulations to see how well maybe the I-Apriority as well as TSFC strategies work. The TSFC technique surpasses other algorithms in terms of job

execution time and average wait period reduction, according to the findings of the experiments (Liu, L., Qi et al., 2018).

2.4.2.2 Clusters Co-operation

A Fog-of-IoT paradigm is predicted to emerge for the arrival of 5G access networks, combining internet of things and the Cloud computing. This computing supports distributed networking and computational resources distribution over the internet. The FC paradigm is well suited for high computation intensive and time-critical streaming applications in the power constrained wireless IoT environment. Consequently, these objectives have been established for this study:

- It provides a compelling analysis of the proposed Fog-of-IoT paradigm's major "killer" application areas.

- It demonstrates how to use Containers to build a virtualized networked computing architecture. The proposed design is implemented in the Middleware layer. It makes use of the Container Engines' inherent capabilities to enable dynamic real-time scaling of virtualized compute and network resources.

- Using a low-complexity penalty-aware bin packing-type algorithm, the researchers were able to actively manage the virtualized computing and networking resources. The proposed optimization implements the combined significant reduction of communication delay and energy while assuring hard optimum values on task processing and networking delays by adaptively trying to scale the computational power of both the virtual processors and transmission capacity of the parameterized TCP/IP connectivity.

- Energy efficiency against implementation complexity of the proposed resource manager is quantitatively tested under static and mobility of Fog scenarios, and compared to the related achievements are made to back up these claims (P. G. V. Naranjo et al., 2018).

The continued growth in popularity of automotive tools, smart and IoT devices, developing a reliable architecture for managing the huge volumes of data by the Vehicle network has become a top concern for the future smart city.

Cloud computing centralized data processing has an inherent weakness, and fog computing has been offered as a solution with lower fog computational nodes. Based on regional collaborative fog nodes, the authors have proposed a vehicle network for the smart city, taking into account variables such as latency and mobility as well as localization and scalability.

The Mobility control, data collecting from multiple sources, distributed compute and storage, and multi-path data transfer are some of the services discussed for IoV applications. In CFC-IoV, the authors show how to build a hierarchical system that manages resources inside and outside of the fog. This makes the LFS more energy efficient and packets less likely to be dropped (Zhang, Z. Zhang et al., 2017).

Internet of Things (IoT) services would be deployed using fog environment. The instability of resources makes it challenging to enable coordinated cooperation across compute, storage, and networking resources in the fog. As a result, we created an architecture and developed FogFrame.

This representational framework describes the required communication channels for launching and maintaining fog-based service execution. (V. B. C. Souza et al., 2018).

2.5 Workload Type

Workload refers to the total amount of information a system gets from its surrounding environment during a certain period of time. Understanding the workload's components is an important part of defining its nature and quantifying and probabilistic characterization of the workload factors in relation to event counts and service requests. Batch Processing and Stream Processing are the two main types of work that make up the workload.

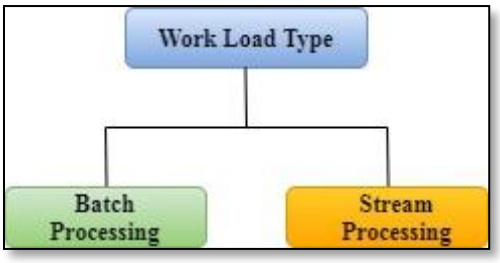


Fig 2.6: Workload Classification (Sri Raghavendra et al.,2021)

2.5.1 Batch Processing

Batch workload refers to an application's collection of non-interactive inputs. When the batch task is received, it has been constructed using data from many sources.

Fog computing lowers latency caused by distant clouds by allowing some application components to be deployed on fog nodes at the network's edge while others remain in the cloud. Virtual Network Functions (VNFs) can be used to build application components. Sequence, parallel, selection, and loops can all be used to indicate their execution sequences. They must map the application components into the infrastructure nodes using effective placement strategies. Current solutions do not account for fog node movement, which can occur in real-world systems. This study determines the projected end to end delay and applications execution cost for fog devices based on the random pointer mobility model developed in the previous research. After that, the problem is described as an ILP formulation that reduces a weighted aggregation feature of both the makespan and computation cost. A Tabu Search Component Placement (TSCP) approach is used to find sub-optimal component placements. The suggested technique is used to improve the computation time and application execution cost, according to the findings (C. Mouradian, S. Kianpisheh et al., 2019).

One of the most crucial components of an industrial automation system is the control loop. Fog computing, which brings distributed computing resources closer to linked devices, is a feasible industrial control solution for time-sensitive applications. However, because of the large volume of data exchanged across fog nodes, the communication burden is significant, minimizing the response time from fog computational nodes to actuators. In this study, we take into account of fog

environment. We use batched sparse (BATS) algorithms to reduce communication and compute demands in the distributed fog computing environment. According to numerical data, the BATS-based system can simultaneously minimize communication and processing demands and the overall reaction time from fog nodes to actuators (J. Yue, M. Xiao et al., 2018).

2.5.2 Stream Processing

Several sources generate this type of task regularly. As a result, the stream processing is preferred when developing real-time IoT systems. An IoT device's sensing frequency can vary the stream workload's specifications and processing requirements over time.

With enhancing the Cloud infrastructure concept to accommodate widely scattered fog nodes at network edges, fog computing alters the distributed computing environment. This scattered technique is ideal for using local fog computation nodes resources to run high data stream applications. Storm is an open-source cluster-based infrastructure with dynamic scalable, and fault-tolerant which are locally distributed. In order to run a distributed Quality of Service (QoS) aware scheduler and to enable self-adaptation, more components have been added to the Storm, which now has the ability to function in a geographically dispersed and continuously changing environment. Authors employ two different sets of DSP applications in this work to offer a comprehensive experimental assessment of the recommended approach: the first one is characterized by a basic topology with a wide variety of criteria, while the second comprises numerous applications. The results reveal that in terms of application performance and runtime flexibility, this proposed scheduler beats the centralized default scheduler. On the other hand, complex topologies with multiple operators may induce instability, limiting the availability of DSP applications. (V. Cardellini et al., 2015).

Fog computing is becoming more and more popular as a paradigm in academia and business. Fog computing has the ability to disrupt the industry with new services and user experiences. Fog computing, on the other hand, is still in its infancy. If you want it to be used as a realistic, low cost, feasible, and easy-to-use alternative to the cloud,

you really ought to build a strong foundation first. Fog computing can give a local network cloud-like services while cutting expenses. Fog computing allows for a novel resource-conserving technique for distributed video summarization, according to the authors. The Fog network's nodes are low-resource Raspberry Pi devices. Surveillance video is distributed among multiple nodes. For less bandwidth, a summary is made through the Fog network and sent to the cloud often. The suggested method is evaluated using a variety of realistic workloads recorded as surveillance recordings. Experiments show that the suggested framework has extremely little overhead and strong scalability compared to off-the-shelf expensive cloud alternatives, even when using a single-board computer with limited resources, indicating its utility for IoT-assisted smart cities. (Mansoor Nasir, Khan Muhammad et al., 2019).

One of today's most pressing issues is lowering electricity consumption and costs. New energy management systems are urgently required due to the massive growth in demand. As a consequence of the advent of new Information Communication Technologies (ICT), traditional electrical grids, residential areas, and devices often become Smart Grid nodes (SGs), Smart Meters (SMs), and Smart Equipment's (SEs). These smart equipment' data is exchanged on a regular basis. More complicated algorithms, faster retrievals, and higher storage capacity are all required to handle this data. A new fog-based energy management system is now being created with all of that in mind. Many data processing and permanent storage capacities are available with Cloud Computing (CC). It does, however, have limitations in terms of quick data retrieval, resulting in reaction times.

Alternatively, fog computing (FC) allows rapid information retrieving with less response delays, with main negative becoming the requirement for temporary storage. The proposed system architecture unifies the properties of both cloud and fog by combining their services. To regulate the load across multiple Virtual Machines in Fog servers, a novel load balancing approach called Modified Shortest Job First (MSJF) is introduced. The suggested algorithm's performance is evaluated using several performance factors such as Computational Time, Response Time, and Cost. The suggested system's performance is validated using simulation in the Cloud

Analyst tool. It is expected that the suggested approach would not outperform based on the findings (T. Nazar, N. Javaid et al., 2018).

2.6 Service Execution Model

Service Execution is responsible for the execution of individual jobs that make up applications. They are in charge of configuring the runtime environment in which jobs are executed. There are 2 categories of the Service Execution Model: Sequential & Parallel, and we will discuss these 2 categories in this section.

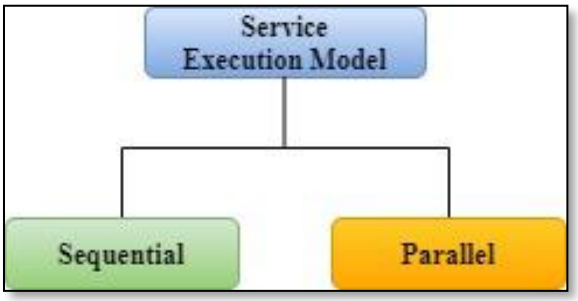


Fig 2.7: Types Of Service Execution

Stable infrastructures, vast volumes of motor vehicles, computers, and strong networks are typical components of the mobile cloud paradigm of car fog computing. The required service must be given fast as a portal for service delivery to correctly save the resources of the related nodes and increase network survival. However, the restricted capacity of the components compounds the situation. Four networks are assigned the bandwidth available to save operating time (Lin et al., 2018). According to the above serving methods, a utility model is designed and solved by a two-step procedure. The solutions on a Lagrangian algorithm for the first step are all sub-optimal. The second stage entails determining and assessing the best solution selection procedure. They have done a computational simulation for demonstrating the assignment results and the best utility model for maximizing survival run.

2.6.1 Sequential Service Execution

A unique fog-like service distribution approach is proposed (Yousefpour et al., 2019). The fundamental idea is to merge three components into a single goal function (application time, network congestion, and server usage). The best service

deployment strategy minimizes application response time, decreases network congestion, and keeps cloud-level server utilization minimum. It is suggested that optimal service deployment be used. This work has considered security concerns for device data transit, vulnerable executing device data in the process, and data privacy, segregated into executed components on other devices. Data protection is a current security concern that will be addressed in future research. It will show that fog computing is a cost-effective and safe solution for low-latency applications.

(Ghobaei-Arani et al., 2019) Fog management solutions in this critical field are described in conventional taxonomy terms using current methodologies and unresolved problems. Service placement, dynamic resource allocation, load balancing, and provisioning are the required fields in the taxonomy. To contrast the methods to resource management, key factors such as performance metrics, case studies, methodologies, assessment tools, and their advantages and downsides are used.

2.6.2 Parallel Service Execution

With several IoT devices, fog nodes, and datacenters all working together, the authors provide an energy and time savings weighted cost model. In addition, for concurrent IoT applications, a novel batch application placement approach that relies on the Memetic Algorithm. Given the different of IoT applications, they provide an ultra - light pre-scheduling solution to enhance the concurrent execution (M. Goudarzi, H. Wu et al., 2020).

(La et al., 2018) The researchers investigated if intelligence could be used as a fog computing facilitator. As shown in the first case study, human-driven data analytics can increase context knowledge and network adaptation in resource planning. Furthermore, when work offloading is assisted by a cluster of fog nodes, the second case study reduces energy consumption and latency for the EU device. Although fog computing intelligence is still in its infancy, the research shows that it has much potential for practical applications and requires greater attention.

By working together on task offloading and scheduling, (Guo et al., 2019), the RAN for a delay-effective computing offloading mechanism exploited parallel

communication and computation in Fog. Specifically, the average mission execution delay minimization problem has been formulated, and the RCO has proposed an efficient approach to its optimal solution. The RCO is shown to be scalable and effective by extensions of application and numerical evaluations.

2.7 Application Dependency

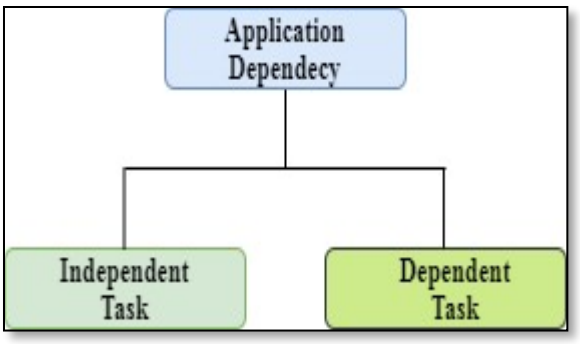


Fig 2.8: Application Module Type

2.7.1 Independent Task

A large number of edge nodes and a single cloud node are used to offload computing burdens from a large number of wireless devices (WD). Each task indicates whether it should be handled regionally at its edge nodes or offloaded to the cloud node when considering various real-time computation workloads at various WDs. They look into simple computational offloading strategies in order to keep MEC network service quality while lowering WD energy consumption. They also suggest a heterogeneous DDLO that beats

DDLO in terms of convergence. The DDLO techniques perform better than the LR-based methodology, according to extensive numerical study. It also takes less than a millisecond for the DDLO algorithm to make scheduling decisions, which is hundreds of times quicker than that of the LR-based approach (L. Huang, X. Feng et al., 2019).

The authors discuss the development of cloudlet-based mobile cloud computing (MCC) to handle resource constraints on mobile devices (energy consumption, processing and memory resources, etc.) as well as communication channel delays.

The architecture described by them is more effectively meets the compute and storage demands of mobile devices while also eliminating network delay. At the same time, the article detailed the tasks connected to energy conservation and communication channel delays by solving problems that need complicated computation and memory resources in cloudlets placed near the user (D. G. Roy, D. De et al., 2017).

The need to extending Cloud infrastructure towards the end user has prompted the development of innovative computer architectures such as Fog computing. When fog and cloud computing are mixed and used regularly, a new, extremely heterogeneous computing ecosystem emerges that benefit from both. Early research efforts such as the OpenFog Consortium's reference design or a new project, Fog-to-Cloud, emphasizes an administrative infrastructure for managing such a combination of resources (F2C). For this purpose, they investigate the possible merits of F2C in distributed environments that take into account the mobility and consumption scenarios of computing resources. They conduct comprehensive simulations to evaluate the issues like application responsiveness, communication bandwidth usage, energy consumption, and the probability of service interruption. The studies demonstrate that maybe a fog architecture gives considerable performance gains over a standalone Cloud (Ramirez, X. Masip-Bruin et al., 2017).

Mobile Edge Computing (MEC) has been proposed to substantially reduce latency due to the inability of Mobile Cloud Computing (MCC) to meet the needs of delay-sensitive applications. However, because edge servers have restricted capabilities that negate the latency gains during periods of high load, hierarchical edge cloud architecture has been investigated as a solution. However, depending on the cloudlet layer, such a model incurs variable computing costs. In this research, we optimize the transmission power and the assigned server computation of mobile devices in a multilayered MEC to minimize their energy consumption and computational cost while maintaining their latency threshold. We formally define the mixed-integer non-convex program and provide an efficient algorithm for solving and obtaining a high-quality solution based on the Successive Convex Approximation (SCA) approach. We investigate various scenarios using numerical data and demonstrate the effectiveness

of our approach in delivering an approximation solution that reduces total energy usage and computing cost (E. E. Haber, T. M. Nguyen et al., 2018).

2.7.2 Dependent Task

Similar studies employed Directed Acyclic Graphs to model their applications in the dependent category (DAG). An individual IoT application service is represented by each triangle. Each edge depicts the flow of data (or dependence) between two jobs.

When it comes to providing rich services to mobile users, emerging Internet of Things represents a new strategy to creating, storing, and analyzing enormous volumes of real-time data. A widely accepted concept of offloading mobile applications for execution to centrally managed and distributed data centers, including such cloud and fog servers, is being promoted to reduce disputes among mobile device' resource restrictions and user' requirements for reduced processing latency and longer battery life. However, despite the improved efficiency of Internet of Things in cloud-edge computing, randomly offloading mobile applications remains a significant barrier to enhancing execution time and overall energy consumption for portable devices due to complexity and variety of big data collected from mobile devices. It is proposed that COM be used to solve this problem, which is a computation offloading solution for Internet of Things enabled cloud edge computing. A system model is investigated in detail, including processing time and energy consumption for mobile devices. Dynamic scheduling is then demonstrated for data-constrained computing workloads. NSGA-III is also used to solve the multi-objective optimization issue of job offloading in cloud-edge computing. Finally, rigorous tests and comprehensive simulations are carried out to ensure that our proposed technique is effective (X. Xu, Q. Liu et al., 2019).

Offloading some code to the cloud is a promising method for increasing mobile app speed while also reducing device energy usage. Existing offloading solutions, on the other hand, are greatly affected by a long communication latency among both mobile devices and cloud servers. They suggest cutting-edge nodes near mobile devices as a solution to this problem, and they study how they improve code offloading. Edge-centric code offloading system Echo relies on a multiple

layer computing architecture that includes smart phones or tablets, edge computing and cloud computing—determining which operations should be offloaded where is a major challenge for Echo. In contrast to other offloading systems that allow mobile devices to make their own best possible offloading decision, Echo uses a centralized control towards the bottom computational nodes . Authors could perhaps fully leverage constrained hardware resources at the bottom computational nodes with this edge-centric design to perform services with guaranteed quality of service. We also have unique ways to make the system run faster, like slow object transfer and updating different parts of the same object at the same time. According to the finding of actual functioning implementation and process simulations, Echo significantly outperforms earlier code offloading systems on the basis of computation time and energy consumption (L. Lin, P. Li et al., 2018).

Code offloading is a promising technique to speed up mobile apps and minimize mobile device energy usage by transferring work to the cloud. Code offloading solutions that are now in use, suffered from a large amount of communication latency among mobile devices and cloud. To address this problem, we propose placing edge devices nearby mobile devices and investigating how they may help with code offloading. Authors proposed Echo, an Edge-centric code offloading system, proposed a three-layer computing environment that incorporates mobile devices, local edge, and also the cloud. The choice of which procedures to offload towards which computer platform is a critical issue that Echo must handle. When compared to conventional offloading methods, Echo provides a centralized deciding algorithm at the edge rather than allowing individual mobile devices to determine its own offloading decisions. An edge-centric architecture may make the most of the devices with limited available resources at the edges to deliver assured quality offloading services. Delayed communication and divergent object updating are two of the unique solutions we provide for boosting system performance. According to the findings of a smaller range deployment with simulations, Echo outperforms earlier code offloading systems on the basis of processing time and energy consumption (S. Bi, L. Huang et al., 2019).

2.8 Optimization Algorithms

Optimization algorithms examine numerous solutions iteratively until the best or most pleasing one is discovered. They are also known as optimization techniques. Optimization has been an integral aspect for computer-aided system design since the invention of computers. There are two sorts of algorithms: Optimal and Sub Optimal. Sub Optimal Algorithms are divided into four categories: Heuristic Algorithms, Meta-Heuristic Algorithms, Hybrid Heuristic Algorithms, and Hyper Heuristic Algorithms.

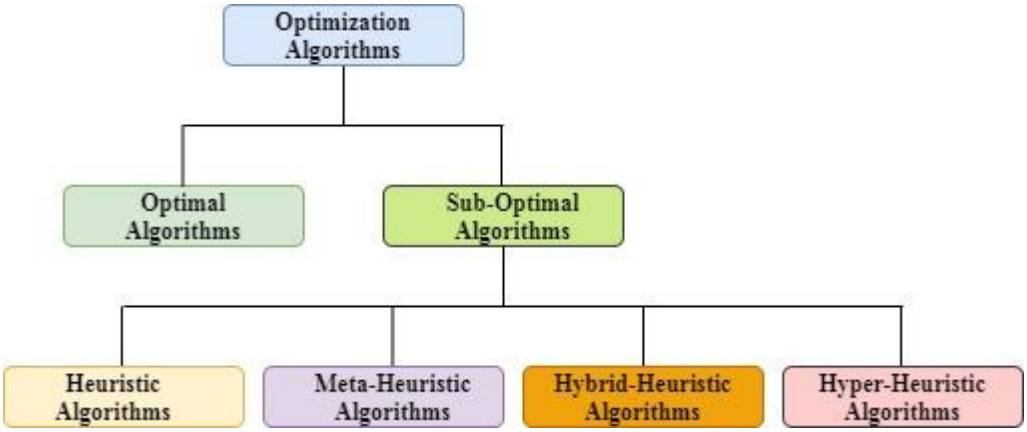


Fig 2.9: Classification of Optimization Algorithms

In this paper, we look at how to optimize QoS and save energy across cloud node, edge node, fog, and IoT models. (Qu et al., 2020). The authors assist instructors in better comprehending the principles and methods used in these models to increase QoS while conserving energy. Quality assurance, SLA violations, and resource management are all issues that the writers have discussed. When it comes to ensuring QoS and reducing SLA infringements, effective VM management is the most important factor to consider. The solution's suitable scheduling and VM integration can meet the needs of customers.

2.8.1 Optimal Algorithms

Fog Computation strategy extends the cloud services to offering extensive IoT-based application services. The Service Placement Problem concerns the placement of services and applications across the lower computational fog nodes and cloud node resources (SPP). In this fog environment, it is critical for response time and

energy consumption. However, problems such as changing service requirements, limited processing resources, and different latency and power consumption characteristics of fog domain devices make providing an efficient solution challenging. As a result, in this study, they suggest MinRE, an efficient SPP technique for fog-cloud systems. They recommend MinRes for important services to reduce reaction time and mining for ordinary services to save energy in the fog environment (Hassan, H. O., Azizi et al., 2020).

When used in conjunction with IoT devices and cloud computing, the fog computing paradigm can help to minimize job scheduling and load balancing. To deal with the huge volume of data received by various IoT devices, current research provides a multiple tier architecture with latency aware scheduling and load balancing with in fog computing environment. Tier-1 consists the IoT devices which is the lowest tier. The gateway routers divide the applications in the second tier into two classes using the Fuzzy Algorithm: Higher Priority and Lower Priority.

Fuzzifier takes into account a variety of criteria, including the task's length, time of arrival, minimum computational time, as well as completion time. The third tier receives a high-priority task. In the third level, Synthetic Patterns, a novel fog structure made up of nodes, were implemented. The fog nodes are grouped using the K-means++ clustering algorithm. Each fog node is responsible for a number of functions, such as monitoring, management, and communications.

The EDF task scheduling method has been suggested to schedule work anywhere in the fog nodes. The current fog node usage is estimated using Artificial Neural Networks. Whenever the IoT device does not find the required resources, the request is forwarded towards the cloud. Using iFogSim in a real-time Video Surveillance and Object Tracking application, then performance of the proposed scheme was evaluated on the basis of energy usage, response time, task scheduling frequency, and delay (A.A.Alsaffar, H.P.Pham et al., 2016).

The authors created a policy on improved learning with genetic algorithm which use a dynamic resource allocation technique (Talaat et al., 2020). LBOS continuously monitors network traffic, collects and distributes data for each server load to available servers via a dynamic resource allocation procedure. Because of this, it increases efficiency even during peak hours. As a result, LBOS provides easy and effective within fog environment for complex applications such as healthcare. The proposed fog framework is composed of multiple layers: First layer consists of the IoT devices, the second and the third layer maintains the fog, cloud node. The IoT layer monitors the patient symptoms.

The fog layer is taken into account while handling incoming requests, and the demand is sent to the appropriate server. The cloud layer is used to send and receive data to and from the fog layer. The patient information is supplied to the most appropriate server for management. This proposal's principal goal is to achieve a low latency. There are two key elements in the fog layer: one is Load Balancer Agent and the other is Resource Allocator.

The Load Balancer is a software that determines which Fog Server is capable of handling a given request. In the fog environment, the Resource Allocator algorithm is employed to attain a high Load balance. According to experimental data, the proposed approach improves cloud/fog device quality of service in terms of allocation costs and time reduction.

2.8.2. Sub Optimal Algorithms

This section will discuss the 4 categories of sub-optimal algorithms: Heuristic, Meta-Heuristic, Hybrid Heuristic & Hyper Heuristic algorithms.

2.8.2.1. Heuristic

The Min-conflict scheduling algorithm is provided as a load balancing scheduling algorithm (Kamal et al., 2018). To resolve a CSP, the algorithm uses a heuristic technique. The suggested Min-Conflict heuristic approach solves the limitation problem by assigning VMs to processing requests with the fewest possible conflicts. Optimal resources are offered for applications where modest tasks are required, and

machine resources are raised. This work yields three separate performance matrices: cost, RT, and processing time.

In the subject of fog computing, the WOA algorithm is presented to handle the problem of job scheduling (Jayasena et al., 2019). Energy usage and execution costs are factored into the fog computation paradigm. The augmented wave method is then suggested as a solution to this scheduling issue. Experimental studies are being undertaken to compare the proposed model to the PSO, RR, SJF, and RR in terms of effectiveness. In addition to these applications, iFogSim has been used to create and simulate a smart health framework.

Fog computing, which brings data storage to the network's edge, makes automated placement even more crucial for deploying distributed applications. An application's resources criteria must be met when a deployment is made in a heterogeneous fog infrastructure. The Internet of Things to the geographical locations of actual objects/things further complicates placement selections. Framework, objective function, and procedure are shown in this research to deal with the issue of deploying IoT applications that are spread out in the fog, as well as how to do it.

Through the use of a backtrack search method with supporting heuristics, the proposed mechanism is capable of dealing with large-scale challenges and making efficient placement decisions that meet the goal of reducing the reaction time of deployed apps. The proposed technique is tested using simulations of various configurations of algorithms and heuristics on various infrastructures as well as workloads (Xia, Y., Etchevers et al., 2018).

2.8.2.2. Meta-Heuristic

Data volumes have increased dramatically as a result of the rapid rise of IoT applications. IoT nodes' resources are insufficient to manage such massive workloads. They could move some of the burden to the cloud to address this issue, but this would reduce the quality of services provided to end consumers. Latency for end-users can be reduced by putting computation in fog devices that are strategically placed at the network's edge.

The difficulty of optimizing fog device energy consumption to cloud device energy consumption is critical. However, the propagation latency between fog and cloud nodes, which is highly variable owing to cloud dispersion with a wide variety of workloads, is largely responsible for delivering the appropriate quality of the services for processing the required workloads. No strategy has yet been successful in handling the problem of job distribution while also limiting the energy consumption and delay caused by fog devices or clouds. This study proposes a processing model for the fog processing problem, defining a trade-off between energy use and latency. A method known as NSGAII is used to solve this multi-objective problem. In a fog-cloud scenario, the numerical results demonstrate that applying the suggested technique for job allocation can reduce both energy usage and latency (M.Abbasi et al., 2020).

A great deal of attention has been given to the use of meta-heuristics in cloud and grid systems recently (Kalra, M., & Singh, 2015). Based on these three prominent meta-heuristic approaches: Ant Colony Optimization, Genetic Algorithm, and Particle Swarm Optimization, as well as two classical strategies: League Championship Algorithm (LCA) and BAT algorithm, a comprehensive study and analysis of various Cloud and grid optimization algorithms is provided. In order to improve the speed and quality of convergence of meta-heuristic algorithms, the majority of research is focused on increasing the speed and quality of meta-heuristic algorithms. They addressed these concerns by adjusting the transformation operator, prepping the input population, or using hybridizing methods.

The use of meta-heuristic approaches in cloud and grid environment planning is examined (Shishira et al., 2016). The majority of research is devoted to increasing the speed and quality of solution convergence for meta-heuristic approaches, that are really slow than deterministic algorithms and give less optimal answers. Adjusting the transformation operator, preparing the input population, and hybridizing methods addressed these issues. Furthermore, different programming algorithms focused on diverse optimization criteria. The approach utilized for meta-heuristic improvement, optimization criteria, task nature, and the context in which the algorithm is implemented are all compared in comparative algorithm analysis based on each meta-

heuristic technique. Today's data centers are huge consumers of electricity and a substantial contributor to global warming pollution. Hence new investigative work has been performed to plan the usage of energy. The major goal is to reduce data center energy usage without sacrificing performance or violating SLA requirements.

Butt et al. (2019) proposed three layers of cloud infrastructure: Fog, consumers, and cloud infrastructure itself. The Cloud and Fog both provide virtual machines (VMs) that can easily operate the user device. The following is the proposed meta-heuristic algorithm: The fog and cloud VM applications set are balanced using a genetic algorithm and Particle Swarm Optimization. The proposed approach is compared with the current Particle swarm optimization to assess efficiency. Nearest Data Center, Reaction Time, as well as Reconfigure Load is used to optimizing Response and Processing Time. These regulations also dictate which data centers are used to process requests.

2.8.2.3. Hybrid

Edge computing can improve Internet of Things and Cyber-Physical System applications through providing computation resources closer to the user. Computationally heavy modules of applications, including a thread, and a task, can be offloaded towards edge nodes because these edge devices have underused processing capabilities. The near-optimal approach of scheduling off loadable modules in an application is established using a Particle Swarm optimization and Genetic Algorithm strategy in this work to substantially improve the application completion time and device energy consumption. Particle swarm optimization is used to optimize offloading without breaching the deadline constraint of an application using an Adaptive Genetic Approach technique based on a novel inertial weight equation (Ezhilarasie, R., Reddy et al., 2019).

A multi-objective hybrid fruitfully optimization technique (Lawanyashri et al., 2017) is suggested to improve convergence rates and optimization precision based on simulated ringing. It is hoped that the suggested technique would help cloud computing users make better use of resources while consuming less power and money.

Use of WorkflowSim allowed the development and implementation of an algorithm for scheduling workflow tasks in cloud settings using the GA-PSO algorithm (Manasrah and Ali, 2018). PSO, GA, and MTCT were among the algorithms that performed comparably to the recommended strategy. With the suggested technique, it is hoped to ensure that the available virtual machines (VMs) are distributed equitably in cloud computing environments, while also taking into consideration the order that workflow activities are completed in order to save processing time and costs.

(Rafique et al., 2019) suggested the NBIHA, a modified particle swarm optimization method, as a novel biologically inspired hybrid algorithm (MPSO). To Program tasks across fog devices, the proposed approach uses MPSO. To control fog device resource levels, it employs a mix of MPSO and MCSO. In the suggested system, resources are divided and regulated based on the requests received. The primary objective is to minimize average reaction time and maximize resource usage by appropriately coordinating operations and regulating available fog resources. iFogSim is used to mimic the performance. The proposed technique (NBIHA) shows promising outcomes of energy use, completion time, and average reaction time when compared to state-of-the-art scheduling algorithms.

2.8.2.4. Hyper-Heuristic

Putting the evolutionary algorithm into practice, Hyper-heuristics is a time-consuming operation that initially intimidates researchers and practitioners who would rather concentrate on the challenging application area. Hyper-Heuristics, in most circumstances, intelligently find the best heuristic or algorithm for a given situation. A hyper-heuristic can, of course, be (often) meta-heuristic and act on meta-heuristic. A meta-heuristic that works at a lower level (Meta-heuristic) operates at a somewhat greater hyper-heuristic activity than the traditional use of meta-heuristics to optimization problems. EvoHyp, a Java toolbox for constructing hyper-heuristics evolutionary algorithms, was introduced by (N. Pillay et al., 2017). EvoHyp includes libraries for hyper heuristic genetic algorithms (GenAlg), hyper heuristic genetic programming (GenProg), a Genalg distributed version (DistrGenAlg), and a GeneProg distributed version (DistrGenProg) as well as a Genalg distributed version

(DistrGenAlg) (DistrGenProg). They demonstrate and explain how to use libraries. The ultimate goal is to provide an instrument set that a non-expert can execute a hyper-heuristic evolutionary algorithm.

A data mining rule is used to introduce an HH-based security-aware technique (Rahbari et al., 2017). The proposed algorithm is 61.72 percent SA, 70.28 percent ACO, and 62.81 percent PSO in terms of average power usage. The proposed algorithm Furthermore, the SA algorithm costs 53.92 percent compared to 54.28 percent for the PSO method, resulting in a 53.84 percent raise for the ACO algorithm. It shrinks simulation time and energy consumption to the size of a heuristic method while increasing decision-making authority for allocating resources based on workflow type and user limitations.

To reduce a well-known series of tests, a hyper-heuristic detection method is applied (Yates et al., 2019). A database of heuristic selections is created using the series of low-level heuristic selections and objective function values that arise. The sequences in the database are separated into subsequences. The logarithmic return mathematical principle is used to distinguish between "efficient" subsequences that tend to lower the goal value and "disruptive" subsequences that tend to raise it.

A multi-objective evolutionary paradigm is proposed to create a selection Hyper-heuristic for the 2D bin packing problem (J. C. Gomez et al., 2017). A multi-target developmental learning procedure with specialized genetic operators is used to construct a collection of hyper-heuristic variable-length laws. Hyper-heuristics provide a solution to a single problem instance by detecting the situation and deciding which heuristic individual to employ at the decisive moment.

To construct Pareto approx. Hyper-heuristics, the proposed system integrates three well-studied multi-objective evolution algorithms: the non-dominated genetic sorting algorithm 1, the heavy evolutionary algorithm 2 of Pareto, and the generalized differential evolution algorithm 3.

Table 2.1: Optimization Algorithms

S.No	References	Optimization Algorithm	Control Parameters	Contribution
1	(Taneja M., Davy et al., 2017)	ModuleMapping Algorithm - Fog-Cloud Placement, lower bound Algorithm - Algorithm used for Search. Comparing the Network Node and the Application service	Resource Utilization. Latency, Network usage, Energy consumption	For Internet of Things based applications, a Module Mapping Algorithm is used to optimally use resources in its network architecture by efficiently distributing Applications in Fog infrastructure.
2	(Minh, Q. T., Nguyen et al., 2017)	Application module Placement	Latency, energy usage, and network burden are all being reduced in this effort.	To enable IoT service offerings, a multi layer fog architecture is being created.Based on this architecture, a novel service placement approach has been given that maximises service decentralisation on the Fog environment by using context-aware information about the location, duration, and service quality.
3	(Wang, D., Liu et al., 2019)	fog controller node selection procedure based on the Gini coefficient.	Computation resource allocation, Latency, Quality Loss	A Gini coefficient-based approach is suggested to get a sub-optimal strategy. The computing resource allocation problem is addressed using an improved resource optimization technique focused on a genetic algorithm.
4	(Attiya et al., 2020)	Using the Simulated Annealing Technique with Improved Harris Hawks Optimization.	Data Center, Host, Virtual Machines, Performance Improvement Rate	It offers a cloud-based version of the Harris hawks optimization method based on simulated annealing. In the proposed approach, simulated annealing is employed here as local search algorithm to improve the stabilisation rate and quality of the results produced by the regular HHO algorithm.
5	Qu, Z., Wang et al., 2020	QoS optimization	Ensuring QoS, Reducing SLA Violations	Researchers examined QoS improvement and energy efficiency in the settings of cloud, fog and IoT computing, as well as edge computing and IoT devices. They gave a description of the major difficulties and looked at the solutions suggested by past research.
6	Talaat et al., 2020	Solution with LB enhancement (LBOS)	Network Traffic monitoring, time, the privacy of data, and accuracy	An LB enhancement technique in Machine Learning as well as a genetic algorithm is suggested in this paper.

7	(Kamal et al., 2018)	Load balancing scheduling algorithm	low latency, high-security	The Min-conflicts scheduling technique is provided in this study as a load balancing scheduling algorithm. To solve a Constraint Satisfaction Problem, the method uses a heuristic technique (CSP).
8	(Jayasena and Thisarasinghe, 2019)	Whale Optimization Algorithm	energy consumption and execution cost	Compared to the RR, SJF, and PSO algorithms, the proposed algorithm reduced average energy usage by 4.47 percent and cost by 62.07 percent on average.
9	Kalra and Singh, 2015	Metaheuristic based techniques	Priority constraints, dependencies constraints, deadline constraints, and budget constraints are all types of constraints.	The authors have provided a full review and comparative analysis of different scheduling methods for cloud and grid settings in this paper, based on the standard metaheuristic approaches.
10	(Shishira et al., 2016)	League Championship Algorithm (LCA)		The authors offer a detailed review of cloud optimization algorithms based on three common metaheuristic techniques: ACO, PSO, and an unique technique: Algorithm for League Championships (LCA).
11	Butt et al. (2019)	This paper proposes the Genetic Algorithm with Binary Particle Swarm Optimization.	Both the response time and the processing time are important considerations.	Authors suggest a three-tiered architecture with cloud, and fog
12	Hassan, H. O., Azizi et al., 2020	MinRes for critical services	Latency, Power Consumption	For SPP in fog-cloud systems, they recommend MinRE, an efficient technique termed SPP-MinRE. Affordability for IoT and energy efficiency for Fog Service Providers. They propose MinRes for critical services, which tries to reduce reaction time, and mining for regular services, which reduces fog environment energy usage
13	A.A. Alsaffar, H.P.Pham et al., 2016	K Means ++ Clustering	reaction time, scheduling time, rate of load balancing, delay, and energy usage are all factors to consider.	An innovative multiple layer architecture with delay-aware schedule and load balancing towards the fog environment is presented.
14	Jayasena and Thisarasinghe, 2019	Wave Algorithm	Energy usage and execution costs	For this problem, the improved wave algorithm is suggested. Experiments are being undertaken to compare the WOA to the PSO, RR, SJF, and RR in terms of effectiveness. iFogSim has also been used to design and simulate a smart health framework, in addition to these applications.

15	Xia, Y., Etchevers et al., 2018	backtrack search algorithm	Execution Time, Scalability	The suggested technique uses a backtrack search algorithm and associated heuristics to efficiently determine placement selections that reduce deployed app reaction time.
16	M.Abbasi, E. Mohammadi Pasand et al., 2020	NSGAII algorithm	Energy usage and delay	The NSGAII algorithm is used to solve this multi-objective model of the issue. This strategy can minimise energy use and delay in a fog-cloud setting. Furthermore, by assigning 25% of IoT tasks to fog devices, energy consumption and latency are decreased.
17	Lawanyashri et al., 2017	Simulated Annealing Approach	Maximizes resource efficiency by lowering energy usage and costs	A multi-objective fruitfly optimization technique is suggested to improve convergence rates and optimization precision based on simulated ringing. The suggested technique optimises resource usage while lowering energy consumption and expenditures.

2.9. Optimization Metrics

2.9.1. Deadline aware

This placement parameter is intended to shorten the amount of time it takes to provide application services while still satisfying the deadline requirements. The computation, propagation, and deploying times are all taken into account while computing this statistic.

IoT applications, which operate in a dispersed heterogeneous environment, can provide a variety of services to their users. Quality of service standards are breached in such applications when computer tasks are outsourced to the public Cloud.It is essential for IoT applications to have a full framework that allows for service placement for the fog computing environment. It's challenging to orchestrate time-critical IoT applications in a fog environment.

The authors propose a DoSP algorithm, a unique four - layered fog computing architecture that distributes application modules in both cloud and fog nodes, as a solution to this challenge. This research demonstrated how to take use of fog resources when keeping a set response time. It uses the Genetic Algorithm to determine where services should be placed in a fog environment. It was necessary to

use the iFogSim simulator in order to simulate DoSP and evaluate the impact of service placement strategy on services deadlines. According to research, the proposed method reduces service execution delay by 10.19% for EdgeWard and 2.58% for Cloud Only (Meeniga Sriraghavendra et al., 2021).

Offloading a job to a fog node or a cloud server can be done in a timely manner, depending on the work's resource requirements. The goal of this study is to develop a DPTO approach enabling scheduling as well as executing Edge device tasks on suitable computing units. Each task is allocated to the proper multilayer feedback queue and prioritised according to its deadline.

This technique eliminates the problem with low priority jobs starvation by minimising the waiting period for delay-sensitive tasks in the queue. In addition, the DPTO technique chooses the optimum processing node for each job depending upon available resources and Connected devices transmission time. While fulfilling deadlines, this method reduces job offloading time by half. Finally, thorough simulation findings based on a variety of performance characteristics indicate that the proposed approach surpasses current baseline techniques (Adhikari, M., Mukherjee et al., 2019).

The IoT era's many connected gadgets provide a substantial communication and computational resource management challenge. Furthermore, the rivalry for limited resources among such devices will inevitably increase the delay experienced by users. The authors propose a priority service-providing approach to decrease the latency faced by delay-sensitive services. Priority groups are created to use a matching theoretical approach, which divides incoming work into time limit and delay-insensitive groupings.

The computing and communication node utilised to assess the queue delays experienced from each class. To deliver a good quality of experience concerning overall delay time for all jobs, a priority - based system is designed and managed using a heuristic algorithm. The dynamic priority system, which enhances their own category when the delay exceeds a certain threshold, is beneficial to users seeking

non-computing jobs at the communication node. The total delay encountered at both computing and communication nodes is compared using high priority, non-prioritized, as well as dynamic scheduling systems. The data show that having effective priority service could significantly minimise the amount of time consumers are delayed (A. Alnoman et al., 2018).

The matching game is also used to explain the challenge of distributing client tasks to cloudlets. During this game, client who surpass a latency threshold employ the idea of demands can enqueue the demand in some other cloudlet and put the task data towards the first cloudlet available. A matching approach that is based on delayed matching is used to solve this game. Simulation findings demonstrate that the proposed technique offers consistent service and minimum latencies, when compared to reactive baseline systems (Mohammed S et al., 2018).

A whole new computer paradigm has arisen in response to the rising popularity of IoT applications, known as fog computing. To make fog node deployment easier, IoT applications are separated down into modules. To achieve a common goal, these modules communicate with one another. The application's overall performance may be harmed if these modules are added without a plan in place. Furthermore, the fog nodes' capacity is insufficient in relation to the module requirements, causing a placement problem. The author’s aim is to lower the total latency of the application by deploying modules on fog nodes (Amira Rayane Beamer et al., 2018).

Fog computing, in order to support the Internet of Things, aims to expand the Cloud's services by providing processing, storing, and network connectivity to the edge nodes. Clustering, dispersion, and dynamic deployment capabilities from IoT devices to Cloud are complex tasks due to the aforementioned diversity, tree structure, and exceptionally large scaled architecture. Multi-task IoT applications could be deployed on Fog infrastructure, according with QoS. The operational characteristics such as latency, bandwidth of infrastructure, application module interactions, and organizational policies are addressed in its model. Algorithms that detecting configurations with Fog infrastructure were discussed. The proposed work is analysed with FogTorch (Brogi A et al., 2017).

Regional fog layers' dynamic and distributed development is particularly challenging to achieve due to unpredictable arriving and departing of adjacent fog nodes. To maintain low communication and processing, a fog node must be able to dependably detect a group of neighbouring nodes and effectively offload computing chores to all of these fog nodes and the Cloud. A combined fog-cloud infrastructure is used to examine the topic of fog network creation and task allocation.

Their goal is to decrease computation and communication time and ensures the fog nodes to create the effective fog connectivity and optimise job allocation when the arrival processes of neighbouring fog nodes are uncertain. The findings also show how the proposed architecture can appropriately transmit computing jobs between both the network environment and a faraway cloud server in a variety of network topologies (G. Lee, W. Saad et al., 2019).

2.9.2 Cost

There are a variety of financial expenses involved using fog computing, including infrastructure setup, operating costs, and instance leasing charges, among others. The use of cost as a placement metric relates to the reduction of costs during an application placement process in Fog. This statistic is taken into account in the following research papers.

Thanks to the Internet of Things, wearables and smart home/city applications have gotten a fresh lease on life. Instead of depending exclusively on the limited storage and computational capacity of small devices, they must remap the landscape of these applications that deal with vast volumes of data stored in the Cloud to accommodate their needs. It's becoming increasingly difficult for traditional cloud computing infrastructure to keep up with the growing number of IoT devices and their need for real - time data, latency sensitive applications.

Although fog computing appears to be a promising solution because it provides end-users at the network's edge with elastic resources and services. The emergence of innovative social software like crowd sensing has heightened the demand of scalability, budget effective infrastructure which can support distributed analytic with

optimize resource allocation and decreasing reaction time, among other functions. Based on recent breakthroughs, we are encouraged to propose MIST as a fog computing-based approach to assist crowd sensing activities within scope of IoT.

For cost-effective constrained resource provision, we further concentrate on database industry associations, job scheduling, as well as virtual machine deployment to MIST. Written as a mixed-integer nonlinear programme (MINLP), the problem is linearized to become a mixed-integer linear programme (MINLP) and then solved (MILP). Real-world aspects like Tehran region, Iran's headquarters, are used to perform a complete evaluation of MIST. According to the findings, the MIST fog-based technique outperforms traditional cloud computing as that of the numerous applications demanding real-time service rises. (H. R. Arkian et al., 2017).

In response to the fast growth of the IoT and smart manufacturing, enormous numbers of intelligent devices are connecting to cloud servers, producing tremendous congestion in the network and network latency concerns. Fog computing is a new computing paradigm that can bring computer resources available to end users and tackle problems that standard cloud-only solutions can't. Work scheduling in fog computing is still in development owing to Quality-of-Service restrictions such as costs and time, as well as the complexity of many provided facilities including edge devices, cloud, and fog nodes.

This paper provides the cost-effective scheduler of multiple workflows under time limitations to address this issue. We'll establish the execution time and cost models for fog computing workflows. After that, a novel multi-workflow scheduling approach is constructed utilising Particle Swarm Optimization. Workflow execution costs may be estimated by using a fitness function. As an example, the heart monitoring app is offered. Extensive testing has proven that, when compared to alternative options, our suggested methodology may greatly lower the cost of completing many procedures within given timelines (Ruimiao Ding, Xuejun Li et al., 2019).

2.9.3. Energy

At some point, offloading all of your work to a data centre in another country gets very pricey and inefficient because there are a lot of Internet of things devices and a lot of data sent from them. It's also challenging to strike a balance in between energy usage for application requests by IoT devices and the deadline constraints. Near-user fog computing provides speedier service while using fewer resources than cloud computing from afar. Fog does not appear to be a replacement for Cloud; rather, they seem to be complementing, and their partnership is worth investigating.

Using a basic fog architecture, the authors demonstrate how to fully harness the benefits of both fog and cloud computing. Assigning the resources in such an energy and time-efficient manner is then the first step in lowering energy and time costs. The problem is then solved with ETCORA, which minimizes application request energy usage and completion time. Before concluding, extensive simulations are performed to ensure that the proposed technique significantly outperformed the next two methods in terms of energy use and time required to fulfil requests (Sun, H., Yu et al., 2019).

The Internet of Things is really a network that links a variety of devices, including sensors and computers. Sensor data is transferred to network servers and analysed on cloud servers, according to the cloud computing concept. As a result of the high volume of sensor traffic, networks are congested and servers are overburdened. To decrease time delay as well as network activity while improving performance of the system, fog computing models distribute data and processes among cloud servers and fog nodes. While we can reduce cloud server traffic, the total amount of power used by fog nodes to analyse sensor data grows.

To minimise node total electric energy consumption, the authors develop the tree-based fog computing (TBFC) infrastructure for distributing activities and data across cloud and fog servers in the IoT. The amount of electricity consumption of devices with in the TBFC concept seems to be relatively low than the cloud environment, according to the evaluation (Oma, R., Nakamura et al., 2018).

Cloud, on the other hand, has substantial challenges in meeting the computational and sophisticated network needs of the inevitable 5G communication network, which include low delay, high flexibility, and scalability.

Smart phones are currently the primary method by which many customers connect to high-speed Internet. The most efficient method to accomplish this is to move computation, storage towards the Cloud. On the other side, a new paradigm defined as Fog Computing or simply Fog, has emerged to overcome these restrictions. Inside a 5G network, fog can help boost spectrum and energy efficiency while also allowing direct wireless communication between devices. It can also help with the emerging concept towards network function virtualization. They investigate on energy consumption of cloud and fog computing, and also the enhanced communication within 5G mobile wireless networks. (Kitanov, S. Janevski et al, 2017).

Researchers in this study examine a method for offloading computation inside a fog computing system in order to save energy. We think that users must choose whether or not to shift work toward a local fog device based on energy use and delay restrictions. The energy consumption and execution delay of the offloading process are investigated in depth using queuing theory. Two queuing models reflect both processing only at mobile device and fog node. The energy-efficient optimization issue is based on a theoretical study and is meant to decrease energy usage while taking execution delay limitations into account. An alternative multiplier-based distributed solution is provided to tackle the described problem. Extensive simulation tests are carried out to demonstrate the efficacy and improved performance of the proposed plan over other existing proposals (Chang, Z., Zhou et al., 2017).

An effective task offloading approach was proposed by the authors for the purpose of constructing effective framework inside a Fog computing environment for the processing of time - intensive and resource-consuming applications. The Firefly algorithm is used in the proposed offloading approach to choose the best computing node based on two Performance constraints: energy usage and process time. The major aims of this method are to reduce both computational time and overall energy consumption of IoT applications with the shortest feasible delay. The Firefly

approach's control parameters are carefully studied. Through comparisons, we show that the suggested technique outperforms existing methods on a variety of performance parameters, such as computation time, energy efficiency, Emissions of carbon dioxide, and heat emissions (Mainak Adhikari, Hemant Gianey, 2019).

In the IoT environment, computational offloading is a challenging task. This offloading improves processing of data produced by various Devices, enhances the task processing, and extends battery life. In this research, authors describe a safe compute offloading approach for the Fog-Cloud-IoT environment. Machine learning techniques are used in a fog-IoT scenario to ensure efficient and safe offloading. At the smart gateway, we encrypt data using a Neuro-Fuzzy Model. The IoT device then selects a suitable Fog node where it may deploy its task via Particle Swarm Optimization via the smart gateway. If a fog node is unable to manage the load, after classification, the data is transmitted to the Cloud.

The historical data is stored in a cloud permanently. On the other side, the data which should be analysed if offloaded using a dynamic offloading technique. The availability of the fog node is determined by two parameters in PSO: which are Available Processing Capacity and Remaining Node Energy. The Cloud node is chosen by Reinforcement Learning. The suggested solution for smart city applications is evaluated by using the NS-3 simulator. The proposed framework is compared with the previous solutions. The proposed strategy appears to minimise latency, according on simulation data (Adam A. Alli and Muhammad Mahbub Alam. 2019).

In this highly distributed, energy-hungry environment, the quality of deployed services must be ensured, taking into account the heterogeneity of capabilities and protocols and the mobility of users and objects. SDNs and fog computing have been introduced to the deployment infrastructure, tweaked to give the required functionality. The authors want to see how IoT services work in a Fog architecture. They present an infrastructure and IoT applications model and a placement strategy that includes system energy consumption and application latency violations mitigation using a Discrete Particles Swarm Optimization method (DPSO). For the simulations, they employed the iFogSim simulator (T. Djemai, P. Stolfe et al., 2019).

Fog computing has gotten much interest to address the location awareness and real-time reaction needs of many applications. On the other hand, fog devices have restricted power supply, processing resources, and communication resources compared to cloud computing, posing design issues in meeting real-time reaction requirements. Energy-efficient offloading decision mechanisms and also an offloading scheduler are shown in this paper.

They show how to balance the response time and energy consumption of several fog devices that run multiple applications by effectively managing their communication and computation resources. Unloading responsibilities are divided into various subtasks, each with its own timeframe from start to end. A run-time dispatcher with the use of an end-to-end delay-based scheduling aspect is also provided to fulfil the response time needs of such applications. This framework saves a lot of energy, according to the assessment findings, and real-world platform research backs it up (Y. Jiang, Y. Chen et al., 2019).

2.9.4 Deadline and Energy

A cloud-fog computing system's power consumption and latency are examined to find a solution to this problem. They begin by quantifying the problem of employment allocation. The primary issue is then categorized into three problem instances with equivalent components in the system, each of which may be solved using an approximate solution strategy independently. Finally, we show that fog computing may considerably increase overall system performance by sacrificing minimum processing resources of the cloud, to retain bandwidth of the network and minimize transmission latency, based on comprehensive simulations and numerical data (M. Huang, W. Liu et al., 2018).

The most pressing challenge confronting the modern ICT business is energy efficiency. The ever-increasing ICT technologies and services have dramatically increased energy needs, emphasizing the importance of raising awareness to encourage the development of energy-saving techniques. However, the support of the underlying ICT platform is critical for the successful and effective implementation of such systems. reducing the energy consumption in Cloud environment. It introduces

the Green Cloud Scheduling Model (GCSM), where it uses a scheduler unit to assign and organize deadline-constrained activities confined to just energy-aware nodes, taking use of task and resource heterogeneity.

To avoid performance degradation and achieve specified QoS, GCSM algorithmically determines energy-aware job assignment decisions. Setting up a Cloud environment allows for evaluating and comparing the suggested model with two additional methodologies. According to the findings, GCSM saves 71% on energy and has an excellent performance in meeting deadlines (Kaur, T., & Chana et al., 2016).

2.10. Simulation Tools

2.10.1. iFogSim

Fog bed, an integration platform and toolkit for rapid prototyping of fog components in virtualized environments, was introduced by (Coutinho et al., 2018). With a desktop approach, Fogbed uses fog nodes as a software container in various network setups. The concept satisfies the requirements for low-cost deployment, versatility, and real-world technological compatibility. Unlike current approaches, the proposed framework allows testing of fog component standard interfaces with third-party systems**.**

FogNetSim++ (Qayyum et al., 2018), a novel simulator that allows users to simulate an extensive fog network with many configurable possibilities, has been proposed in this paper. It enables researchers to integrate custom mobility models and algorithms into fog node planning and management. They have used a traffic management system to test the simulator's scalability and efficacy concerning CPU and memory usage.

(Gupta et al., 2016) presented the IoT-Fog Simulator, iFogSim, to model and assess the influence of resource management approaches on latency, grid congestion, energy use, and cost. In two case studies, we show how to model the IoT environment and compare resource management strategy. In addition, the simulation toolkit's

scalability is measured on the basis of Memory usage and computation time in a range of scenarios.

(Lopes et al., 2017) addressed how fog-building resources are allocated based on user mobility and developed MyiFogSim. This iFogSim module allows virtual computers to migrate between clouds. In addition, a migration strategy will be presented and tested using MyiFogSim to see how it affects the quality of the service framework. The results show that the policy will support lower latency without a migration policy compared to a scenario.

To investigate personalized and dynamic techniques for creating and executing applications, (Brogi et al., 2019) presented a fog computing simulator. Complex network theory influences relationships between applications, network connections, and infrastructure characteristics, allowing topological behavior in dynamic and flexible strategies, including device module positioning, workload locations, path-routing, and service schedule. The most commonly reported, iFogSim, is presented with a comparative analysis of simulator test efficiency and convergence. To highlight the YAFS functionalities, we write three scripts that we know cannot be used with current fog simulators: dynamic allocation of new device modules, dynamic network node failures, and user mobility alongside topology.

2.10.2 CloudSim

In Cloud computing models, application services are very complicated to get, design, set up, and install. It's difficult to evaluate cloud approach, application workload configurations, as well as resource management modeling techniques in different requirements and user configurations with needs. CloudSim (Calheiros et al., 2011) is an expanded simulation toolkit that allows for the simulation and modeling of cloud computing environment including application - aware scenarios. In order to better understand the tactics and behavior of Cloud system components, which including physical servers, virtual servers, and allocating resources, CloudSim was developed as a set of tools and utilities.

It has standardized application supply approaches that may be scaled up rapidly and effectively. It now offers cloud computing environment modeling and simulation in a single, interconnected cloud environment. In inter-networked cloud computing environments, it also exposes customizable APIs for policy implementation and the provision of VM allocation strategies. Several academics use CloudSim to study efficient resource distribution as well as power aware server and storage management systems.

CloudSim is used by HP Labs with in United States, as well as other universities, in their research. In a study that investigated application-level provision of services in this hybrid cloud environment, CloudSim's utility is illustrated. The federated cloud computing approach significantly improves QoS application needs under variable demand patterns for resources and services, as demonstrated in this case study. Simulation is one of the most frequent ways of evaluation in scientific workflow studies.

(Chen and Deelman, 2012) launched WorkflowSim, which extends the existing workflow management layer to incorporate a current CloudSim simulator. According to the authors, failures in the simulation of scientific procedures and system overheads may result in large inaccuracies during the projected runtime. WorkflowSim has chosen two intriguing research topics. It provides a specialized and effective assessment tool to demonstrate its continued commitment to increasing research activity.

(Zhou et al., 2013) present FTCloudSim, which extends CloudSim's core features. The FTCloudSim framework has been expanded to help researchers introduce new ways for improving cloud service efficiency. FTCloudSim will also examine the behavior of the suggested new mechanisms. Four upgraded dependability techniques are used to demonstrate the possibilities of FTCloudSim. A tool for simulating and modeling containerized cloud computational infrastructures, Container CloudSim (Piraghaj et al., 2016), is described in detail below. As a CloudSim expansion, containerized cloud simulation architecture is proposed and implemented. In order to demonstrate how their container planning and supply rules may be integrated and

contrasted in order to reduce energy consumption and SLA satisfaction, a number of use scenarios are offered. Because there will be more containers in a data centre compared to virtual machines, this developed system is particularly scalable because it allows several containers to be simulated. Table 2.2 shows a comparison of different simulation tools.

Table 2.2: Comparison of Different Simulation Tools

Simulators/ Characteristics	**Language**	**Community Support**	**Application-Level Modelling**	**Scalability**	**Mobility**	**Fault Injection**
iFogSim	Java,XML	Low	Stream Processing	Yes	No	No
CloudSim	Java	Moderate	Service Chain	Yes	No	No
Fogbed	Java	Moderate	Fog Network	No	No	No
FogNetSim++	C++	Moderate	Fog Network	Yes	Yes	No
YAFS	Python	Moderate	Modular	Yes	Yes	Yes

The Table 2.3. summarizes various existing frameworks in fog environment with respect to their application properties, architectural properties, and placement properties. The frameworks listed in the table are as follows:

1. Application properties
    a. Workload Type
    b. Execution Type
    c. Number of applications and number of modules in each application
    d. Service Modules
2. Architectural properties and
    a. Number of Fog node layers
    b. Cluster cooperation
3. Placement properties.
    a. Priority
    b. Strategy
    c. Approach
    d. Optimization Parameters

Table 2.3: Summary of Literature

Work	Application Properties				Architectural Properties		Placement Properties			
	Workload Type (Batch/ Streaam)	Execution Type (Sequential /Parallel)	Apps No. & Modules No.	Service Module (Dependent/Independent)	No. of Fog Layers	Cluster Co-operation	Prioritized	Place-ment Strategy (Static/Dynamic	Approach	Optimization Parameters (Deadline/Energy)
M. Taneja et al., 2016	**Stream**	**S**	**S & M**	**D**	**M**	**No**	**-**	**S**	**H**	**-**
X.-Q. Pham and E.-N. Huh, 2016	**Batch**	**S**	**S & M**	**D**	**S**	**No**	**-**	**S**	**H**	**-**
H.Gupta et al., 2016	**Stream**	**S**	**S & M**	**D**	**S**	**No**	**-**	**D**	**H**	**-**
X.-Q. Pham et al, 2016	**Batch**	**P**	**S& M**	**D**	**S**	**No**	**Task**	**D**	**H**	**-**
R. Mahmud et al., 2018	**Batch**	**S**	**M & M**	**D**	**M**	**No**	**App**	**D**	**H**	**Deadline**
T. Huang et al, 2017	**Batch**	**S**	**S & M**	**D**	**S**	**No**	**-**	**D**	**H**	**Deadline**
Skarlat et al., 2017	**Stream**	**P**	**M & M**	**D**	**M**	**Yes**	**App**	**D**	**H**	**Deadline**
T. Choudhari, et al., 2018	**Stream**	**P**	**M & S**	**I**	**S**	**No**	**App**	**D**	**H**	**Deadline**
R. Mahmud et al., 2019	**Batch**	**S**	**M & M**	**I**	**M**	**No**	**App**	**D**	**FL**	**Response time**

Meeniga Sriraghavendra et al., 2021	**Batch**	**S**	**M & M**	**I**	**M**	**Yes**	**App**	**D**	**H**	**Deadline**
R. Mahmud et al., 2020	**Batch**	**-**	**M & S**	**I**	**S**	**No**	**-**	**D**	**LP**	**Deadline**
P. G. V. Naranjo et al., 2018	**Batch**	**S**	**S & M**	**I**	**S**	**Yes**	**-**	**D**	**H**	**Energy**
W.Ramirez et al., 2017	**Stream**	**S**	**M & S**	**I**	**M**	**No**	**-**	**S**	**H**	**Energy**
H. Y. Wu. et al., 2018	**Stream**	**P**	**M & S**	**I**	**S**	**No**	**-**	**D**	**LP**	**Energy**
K.H. Kim et al., 2007	**Batch**	**-**	**M & S**	**I**	**S**	**No**	**App**	**S**	**H**	**Deadline & Energy**
R. Deng et al., 2015	**Batch**	**-**	**M & S**	**I**	**S**	**No**	**-**	**D**	**LP**	**Deadline & Energy**
S. Sharma et al., 2019	**Stream**	**S**	**M & S**	**I**	**M**	**No**	**App**	**S**	**FL**	**Deadline & Energy**
M. Huang et al., 2018	**Batch**	**S**	**M & S**	**I**	**S**	**No**	**App**	**D**	**H**	**Deadline & Energy**
M. Goudarzi et al., 2021	**Batch**	**P**	**M & M**	**D**	**M**	**Yes**	**-**	**D**	**H**	**Cost & Energy**
DEEDS P	**Batch**	**S**	**M & M**	**D**	**M**	**Yes**	**App**	**D**	**HH**	**Deadline & Energy**

Column-wise abbreviations:

S – Sequential, P- Parallel

M-Multiple, S- Single

I-Independent, D-Dependent

S- Single, M-Multiple

S-Static, D-Dynamic

H-Heuristic, FL-Fuzzy Logic, LP-Linear Programming, HH-Hyper heuristics.

2.11. Formulas for Existing Parameters such as Cost and Energy:

2.11.1 Cost

There are various cost models that are utilized in fog service placement, and the particular formulas employed may differ based on the approach and assumptions used. Fog service placement commonly involves several cost models and their associated formulas, including:

Energy cost: This cost model refers to the energy utilized by fog nodes to execute a service or process data. The energy cost can be determined by multiplying the power consumption rate, execution time, and the energy price per unit. The energy cost formula is given as:

Energy Cost = Power Consumption Rate x Execution Time x Energy Price per unit

Processing cost: This cost model refers to the computational cost of executing a service or processing data, which can be estimated in terms of CPU cycles, memory usage, or other resources. The processing cost can be estimated by multiplying the resource usage and the unit price of the resource. The processing cost formula is:

Processing Cost = Resource Usage x Unit Price of Resource

Bandwidth cost: This cost model refers to the cost of transmitting data between fog nodes or between fog and cloud nodes. The bandwidth cost can be determined by multiplying the data size and the unit price of bandwidth. The formula for bandwidth cost is:

Bandwidth Cost = Data Size x Unit Price of Bandwidth

Delay cost: This cost model refers to the penalty for not meeting the service deadline or response time requirement. The delay cost can be estimated as a function of the waiting time or response time and the penalty cost per unit time. The formula for delay cost is:

Delay Cost = Waiting Time x Penalty Cost per unit time

It should be noted that the above formulas are just examples, and other cost models and associated formulas may be used based on the particular fog service placement problem and requirements.

2.11.2 Energy

There are several formulas available for calculating energy consumption in fog service placement, depending on the specific factors considered. Some of the commonly used formulas include:

Energy Consumption of a Device: This formula estimates the energy consumed by a device based on its power consumption and the duration of usage.

Energy Consumption = Power × Time

Energy Consumption of a Communication Link: This formula calculates the energy consumed by a communication link based on its data rate, distance, and the transmission time.

Energy Consumption = Data Rate × Distance^2 × Transmission Time

Total Energy Consumption of a Fog Service Placement: This formula calculates the total energy consumed in a fog service placement scenario, considering the energy consumption of all devices and communication links involved.

Total Energy Consumption = ∑(Energy Consumption of Devices) + ∑(Energy Consumption of Communication Links)

It is important to note that these formulas are just examples and may not be applicable to all fog service placement scenarios. Other factors such as data compression, data aggregation, and load balancing can also affect energy consumption and may require additional formulas or models.

2.12. Research Questions

The research challenges of the globally dispersed fog environment in terms of energy and response time are addressed in this paper. Based on the motivation stated above, the following research investigations are proposed:

- **How to identify latency-sensitive applications?**

  Determining the applications with the highest priority is essential for the fog computing environment. Moreover, identifying the applications and providing the response time below the deadline is crucial.

- **How to map the application modules?**

  Determine which category of application module it is decided according to the category of application module that affects the application deadline and energy consumption.

- **How to choose the appropriate computation node for application deployment?**

  Finding a suitable node in the hardware heterogeneity of the fog environment is very challenging. Different parameters should be considered for the computation node selection.

- **What are the main parameters for effective service placement in a fog environment?**

  Determine the parameters which provide effective service placement is important for the fog computing environment. Moreover, identify the parameters which affect application deployment in a fog environment.

- **How to design the application placement algorithm in a Fog environment?**

  Considering the parameters for application placement, evaluate the performance, and identify the most significant impact on application deadline and energy consumption.

- **What are the most effective strategies & simulation tools for the service placement?**

  Finding the effective algorithm for application modules placement, and suitable simulator tool for evaluating the proposed work.

2.13. Objectives

In order to fulfill the aim of the research, the following are the objectives identified.

1. To analyze and assess the existing service placement methodologies in a fog computing environment.

2. To design and develop a practical framework for service placement in a fog computing environment.

- ❖ To design and implement an algorithm for deadline-aware service placement.
- ❖ To design and implement algorithm for deadline aware and energy efficiency service placement.

3. To evaluate the proposed architecture.

Chapter 3

Deadline oriented Service Placement (DoSP)

3.1 Introduction

Since it provides subscription-based access to remote computing resources, cloud computing has helped both individuals and corporations. Cloud computing employs servers located in a remote location that gives computer power on demand. Servers have high latency, bandwidth, and energy consumption due to their remote position. In light of these limitations, cloud computing cannot be suitable for Internet of Things (IoT) applications that require low latency, such as smart farming and traffic monitoring. Other IoT applications that are not suitable for cloud computing include home automation, smart buildings, as well as smart health monitoring units and intelligent supermarkets. A common research topic that has gotten a lot of interest is offering services together in short length of time. The fog computing technology solves the problem of moving computation closer to the data-generating devices. Fog computing is a type of cloud computing that relies significantly on bandwidth usage, heterogeneous processing, workload distribution, and mobility.

Resource provisioning has been found to minimize latency, improve compliance with Service Level Agreements (SLAs), and give additional benefits. However, simply providing services is insufficient to achieve the intended results; service placement strategies are also required. A complete technique that addresses service placement while also increasing Quality of Service is therefore required in light of both the heterogeneous nodes and dynamic fog-cloud computing environment.

This study will make the following major contributions, which are listed in alphabetical order: The proposed work provides a comprehensive approach in Quality of Service-aware service placement in a fog environment, allowing for optimum sequential IoT application service placement.

- It is presented a Deadline-oriented Service Placement (DoSP) algorithm, that evaluates a service placement response time of in different levels and

determines where workflow-based Internet of Things applications should be located in different layers of the fog framework.

- A fog simulation is carried out using the iFogSim toolbox, and indeed the efficiency of the service distribution approach is evaluated. Based on the distinct levels, many IoT application modules/services are placed on different tiers of the fog architecture.

3.2 System Model

There are three layers of nodes in the fog computing environment, as indicated in Figure 3.1. Sensors and actuators in IoT devices are linked to fog nodes with processing, storage, as well as network connectivity at a lower level. The controlling role is performed by a fog node in the fog environment. This node is referred to as the fog controller node.

The following are the functions of the fog controller node:

1. Accepting the user's incoming request.
2. fog controller node regulates the availability of resources across cloud and fog nodes, including certain processing capacity, storage, and memory size.
3. It finding out where the ideal place to put an application is.

When running the modules for resource utilisation, this however identifies which application modules should be installed on which computational node. This fog controller node forwards the module placement to the cloud node, fog nodes, and the neighbour controller node.

- The application receiver component is in charge of receiving the user's application request. For each application that makes it to the controller node, parameters and information such as the number of modules, workloads, and data input size were established. In the fog computing architecture, the fog controller node (FCN) determines the right placement of modules.

- The resource collector is in charge of gathering and storing information about all processing nodes' current system statuses in the resource database. As resources are added or removed from the computer system, state information

is frequently updated. It assures that the fog controller node's final application placement conforms to the most recent updates on computing node resource use, improving precision.

- With information about all computing nodes' processing speed and communication delay, the application placer examines the application, calculates the needed location, then allocates the tasks of each application to the computing node with the most processing power and communication delay. The computer nodes examined in the proposed framework include fog, fog controller, neighbour controller, and Cloud nodes. The sensors' frequency is considered to be the same for convenience, as well as fog nodes at same system level are presumed to be the same kind.

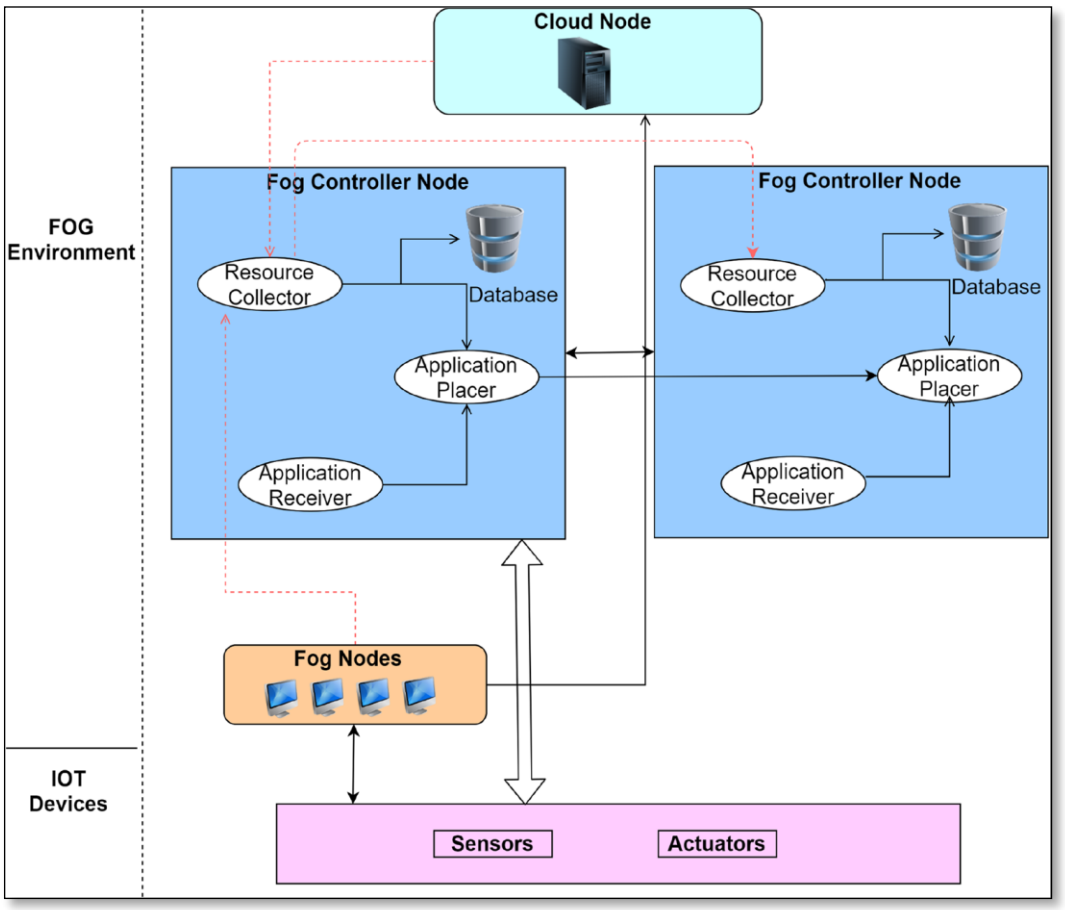


Figure 3.1: System model

3.3 Proposed Methodology

The suggested approach seeks to establish an effective framework for the installation of IoT-based applications in a fog environment that is aware of the quality of service (QoS). It is designed to function in a fog computing environment. Mostly in context of building Internet of Things applications in an array of sectors, which including healthcare and the military, we can see the relevance of this paradigm, as well as the need for fog computing, moreover to cloud computing, in order to manage these applications.

3.3.1. Overview of the proposed work

Fig. 3.2 shows a system for service placement that is aware of the quality of service (QoS), which guarantees the service placement, was effectively optimized in order to increase performance. This method can aid service placement in a fog environment while also allowing for effective decision-making based on a given IoT application's QoS needs.

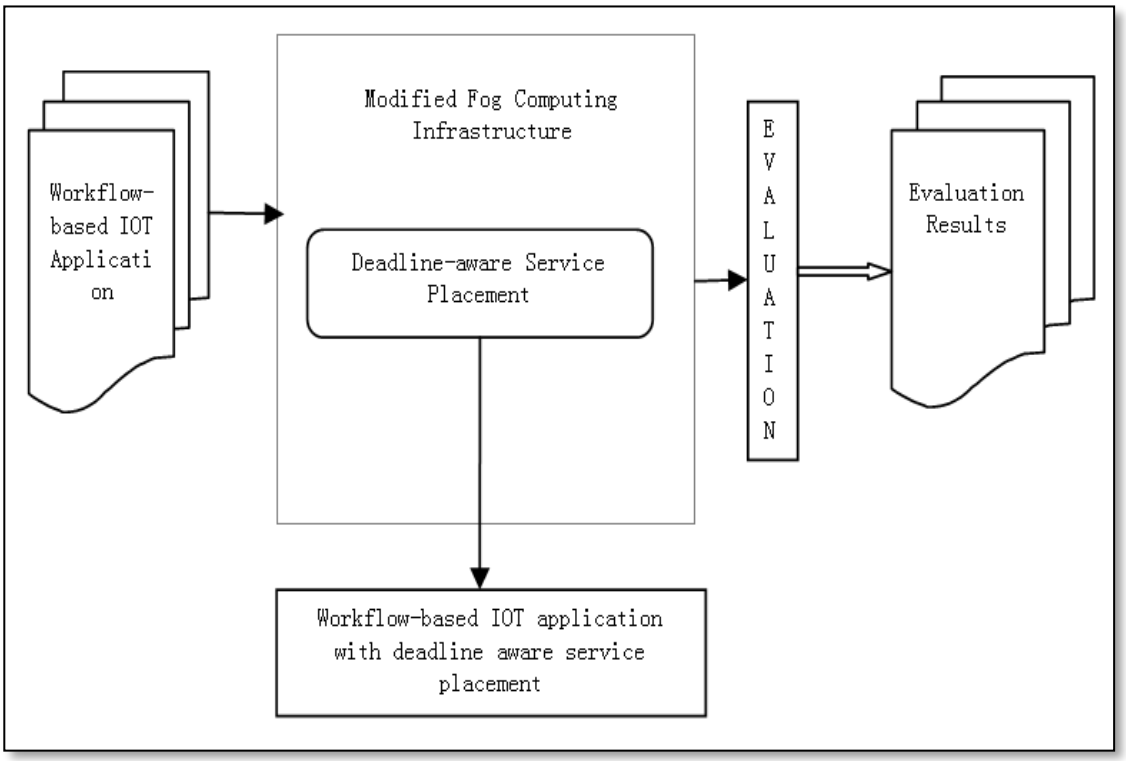


Figure 3.2: Overview of the framework

IoT workflow-based application is used as the source of information. The suggested system is then in charge of analyzing its needs and running an algorithm for effective service placement that considers service deadlines. The service placement meets or exceeds timelines. For the sake of simplicity, no service layer agreements are addressed in the proposed work.

3.3.2. Functional Details of the Proposed Work

Detailed architectural design of deadline-aware service placement is depicted in Figure 3.3, which also includes functional aspects. Upon selection with one or even more workflow-based Internet of Things applications, the proposed architecture is subjected to a number of checks to verify that the service placement fits the requirements of the application. Modules from the IoT application are used in this application. For each module/service, the number of modules to be installed, their sequencing, and the deadline associated with each module are among the details obtained. Following that, each module's requirements are assessed. Each module has its own pool of resources, including storage, computing power, and bandwidth, which makes it difficult to maintain a high level of productivity.

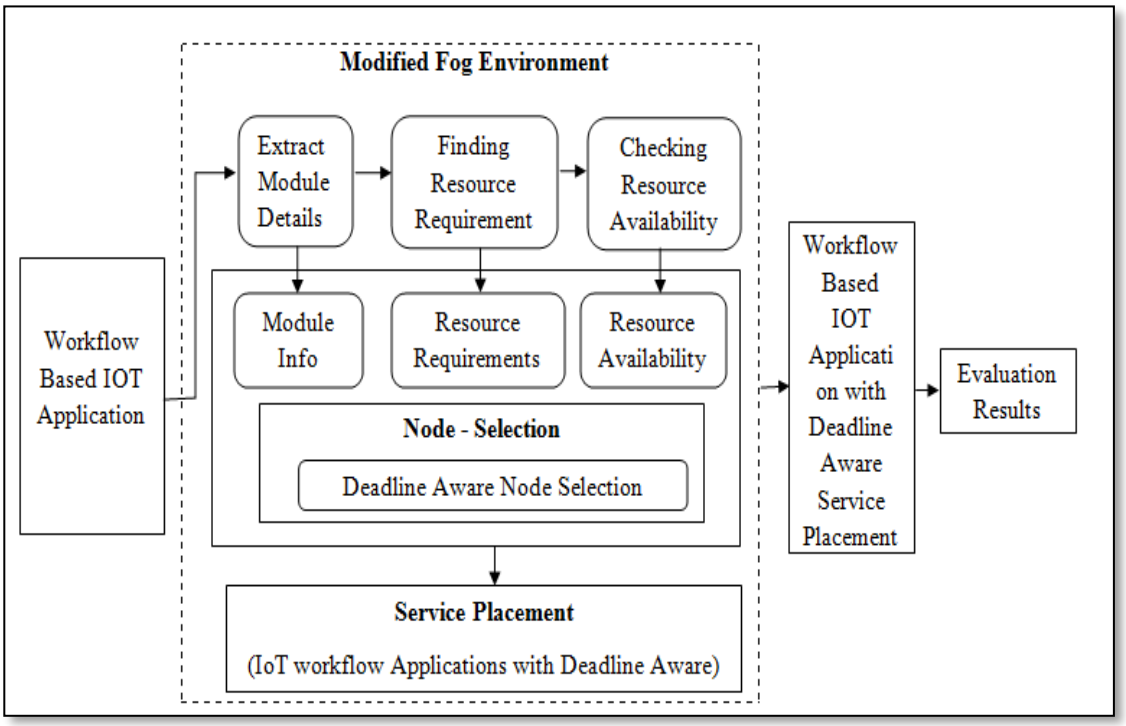


Figure 3.3: Functionalities of the proposed architecture

It is also possible to fulfill this need for a specific service using FN, FCN, NFCN, or CN networks. After that, the availability of resources is investigated in order to gather information across all computational nodes with associated resource availability. The information about resource availability is then used to select a node for a particular service or module. Finally, the benefits are deployed in a particular layer. After node selection, the selected nodes are picked based on the most recent study of resource availability and the service deadline.

3.4. Application Model

3.4.1 Application Data flow

Before a program is required to be run in the fog environment, it should be divided into modules. The groupings of interconnected modules aid the concept of a distributed application. Every module in the program is required and performs at least some of the application's functions. Application edges are the connections that exist between two application tasks; if two applications have an edge between them, it indicates that they are both reliant on the other. The distributed data flow model (DDF) represents application dependency and data flow in the directed network that can be unidirectional or sequential.

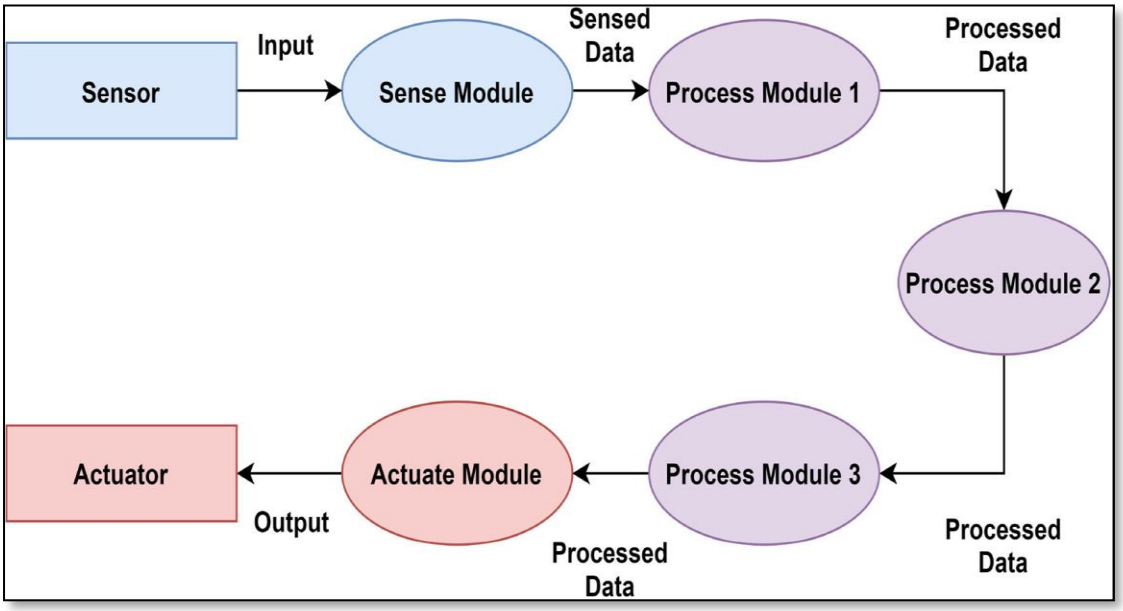


Figure 3.4: Flow of Data in Application

Figure 3.4 depicts the five key modules (tasks) that perform processing in the IoT applications discussed in this work: the sensor module, the process module 1, the process module 2, the process module 3, and the actuation module (as depicted in the figure). Modules are a set of interrelated activities that make up an application. They are carried out one by one in a sequential manner. A sensor is shown in Figure 1 that transmits information from the Sensing module towards the application process modules. Authorization, data normalization, as well as multi-sensor data aggregation are all handled by this module.

The data coming from the Sense module is handled by process modules. The application comes with a number of features. As a result, each processing module is capable of carrying out at least one data function. The actuate module determines and controls the actions of the associated actuator. In the iFogSim simulator, the app components are mimicked using the AppModule class. We used the AppEdge class in the iFogSim simulator to build data dependencies across modules, as seen in Figure 3.4.

3.4.2 Application Process flow

An example of a service placement request is illustrated in Figure 3.5, which is received by a fog controller node from the user. It then determines which resources are required for the specified IoT application. It also necessitates the setting of a deadline for the service. The resources on fog nodes are assessed with the deadline in mind. Then, determine which fog nodes are capable of meeting service deadlines.

It is then computed and validated when the deadline and deployment time will be met or exceeded. The service is distributed in the fog node (or) fog controller node (or) Cloud if the service is low. If the time is excessively long, it is forwarded to the NFCN, which evaluates resource availability and QoS expectations. The application modules are posted to NFCN once they have been approved. Otherwise, they're sent to the Public Cloud, which hosts the service.

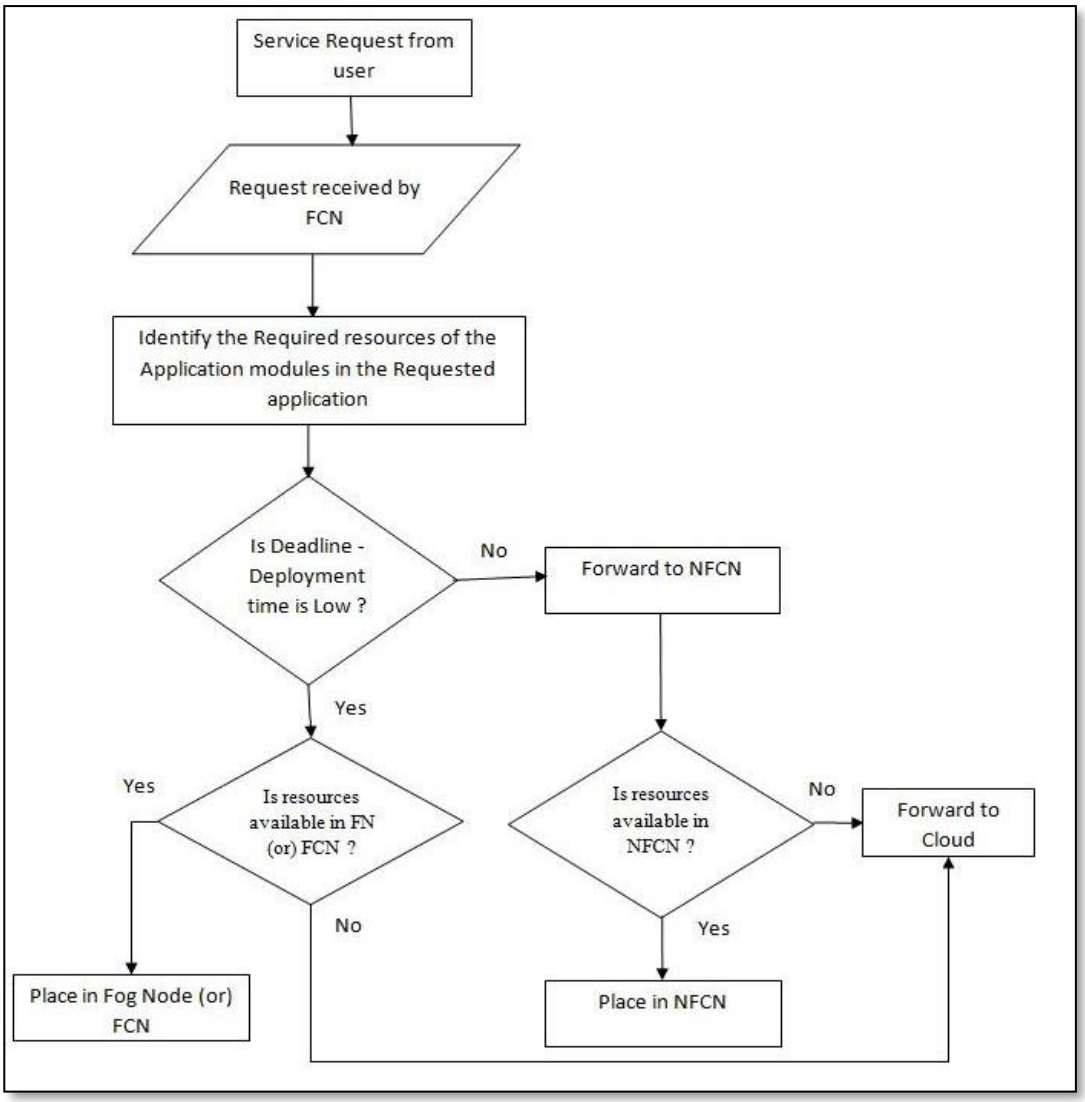


Figure 3.5: Flowchart of Application Module Placement in Fog area

3.5. Model for Performance metrics (formulas & Constraints)

3.5.1. Application Selection Prioritization

During this phase, the requests for service placement received from various applicants are prioritized. This is accomplished by ranking applications based on the time elapsed during respective deadline and its deployment time, which would be defined as follows:

$$\text{prioritization} = \text{Deadline}_{App_i} - \text{Deptime}_{App_i} \quad (1)$$

Where $Deadline_{Appi}$ represents application deadline, and $Deptime_{Appi}$ represents waiting time for deployment. $Deptime_{Appi}$ is associated with the time waited by the application prior to its assignment to suitable computing resources. It is always better to initially assign the applications with high waiting times based on the deadline to the fog resources. As the deadline cannot be violated, it needs to be made so. Allocating the shortest computation applications first to fog resources may lead to starvation. Hence it is not desirable.

3.5.2 Placement Constraints

In order to deploy services of an application, it is essential to have certain constraints that can help achieve optimal performance.

Constraint 1: In addition to storage, this constraint is related to RAM and CPU (important computer resources). These resources are to be considered as the fog resources allocation should not exceed these available resources.

$$\sum_{App_i}^{App} \sum_{AppMod_i}^{App_i} Res_{AppMod_i} . Place^{fn}_{AppMod_i} \leq Avail^{Res^{fn}}, \forall fn \qquad (2)$$

Where fn={FN, FCN, NFCN}denotes fog node in a fog cluster, Res={CPU, MEM, STORAGE}denotes required resources, Place denotes service placement, and Avail denotes available resources.

Constraint 2: This condition is related to the expected response time of the applications. No application should violate its deadline, therefore:

$$Resptime_{Appi} \leq Deadline_{Appi}, \forall_{Appi} \qquad (3)$$

Where $Resptime_{Appi}$ represents application response time and $Deadline_{Appi}$ represents application deadline.

Estimating application response time

In a deadline-aware approach, it is important to estimate application response time. Therefore, the response time can be estimated by:

$$Resptime_{Appi} = TotDeptime_{Appi} + Exectime_{Appi} \quad (4)$$

$$Exectime_{Appi} = Makespan_{Appi} + Comm_{Appi} \quad (5)$$

$TotDeptime_{Appi}$ denotes deployment time, $Exectime_{Appi}$ denotes execution time, $Makespan_{Appi}$ denotes makespan time, and $Comm_{Appi}$ denotes communication time.

$Deptime_{Appi}$ takes into account the amount of time that has passed since we placed each service appropriately. Depending on the runtime scenario, the placement is done in cloud or fog. The $TotDeptime_{Appi}$ represents both the elapsed and expected deployment times. It is created when the service AppModi Appi is propagated to the nearest colony. The AppModi Appi execution time indicates the time required to execute all services of a specific application and the necessary amount of communication between services. As a result, the overall execution time AppModi Appi is calculated as the sum of makespan and the communication time CommAppi. As shown in the architecture, delays in the link indicate communication time in service placement in various places.

3.6. Deadline-oriented Service Placement Approach

3.6.1. Proposed Policy

The Deadline-oriented Service Placement (DoSP) method is supposed to assess the fog landscape for this given sequential applications with deadline requirement and make judgments regarding applications service placement in the fog area. The suggested approach is depicted in Figure 3.3 with Node selection, respectively. It is based on the notion of Genetic Algorithms (GA).

3.6.2. Service Placement Algorithm

Each operation within the GA will operate in accordance with this concept. The length of the vector is proportional to the number of services which will be employed in the algorithm. The vector is the layout blueprint that determines where the

application modules will be placed. The I - th module is inserted in node 2 whether the vector[i] = 2. As a result, the I - th module is assigned to the value of the processing node. The whole vector (the placement strategy) is referred to as the chromosome.

In the previous paragraph, it was stated that it would take the shape of an integer vector. Fog cell IDs, Cloud, nearest neighbour, and fog orchestration control node are all possible integers. These IDs are just as follows: Zero as Cloud, One as FCN, Two as NFCN, as well as Three for FN. The parameter "number of Placement Locations" is assigned as the number of potential integers. The first generation must be created after the variables "Chromosome Length" and "Number of Placement Locations" have been assigned. It is necessary to predetermine the population size of the GA, which is one of the sections of the GA. Through this quantity, we'll be able to create a series of chromosomes for the first generation. There have been various distinctive ways to fill a vector using numbers with a chromosome. One method is to add numbers to the vector at random from a list of potential integers.

This might be the stage during which a new chromosome be formed. The status check should be satisfied by all of the produced chromosomes. This status check is a procedure for determining if a plan has been effectively put. When the needed MIPS is far less than the available MIPS, a module could be successfully inserted first. Second, only FN contains the sensing and actuation components. Moreover, the process modules are really not permitted in the FN.

In order to determine how excellent an individual chromosome is, its fitness must first be determined. Then it is added to the population. According on how near an application has come to the deadline, its fitness may be measured. Whenever the application is filed close to the deadline, then high penalty will be imposed; else, a low penalty would be applied. The fitness value must be minimized in this case which is possible by using the minimum fitness chromosome placement strategy. The variable "fittest Chromosome" is assigned to the population's fittest chromosome. We return that chromosome if it meets all of the restrictions, and we are done. If that is not the case, we will have to go through the rest of the process. First, we determine

the number of crossovers that we will perform. It is also possible to use "crossover" as an additional operator with in genetic algorithm.

This number of generations, which would be entirely user-controlled, is another aspect of the genetic algorithm. We utilized the GA operator "selection" in each generation to find the best (fittest) chromosome. In this case, the most suited chromosome is the one with the highest fitness among the ones that have been chosen. As a result, we're not focusing at the most fit chromosome in the population. This algorithm's selection part can be implemented in a variety of ways. Using the Tournament option is one option. Tournament selection is a two-step selection method.

The first stage is to group all of the population's solutions into a "circle," with each solution occupying a different area of that circle depending on its fitness. The greater the significance of the chromosome, the higher the fitness will be. The second stage is to choose the best suited option from the list of possible options. This choice was made in order to obtain both parents' approval. The crossover can be carried out in a variety of ways.

The uniform crossover is one option. Genes (chromosome portions) are uniformly picked between parents to generate a kid chromosome in this crossover. The fixed mixing ratio inside the uniform crossover is another genetic algorithm parameter that tells us how many genes originate from each parent. With crossover, we are making two-child chromosomes. Those two chromosomes are opposed. If the first one utilizes the first gene from the first parent, the second one will use the second parent's first gene, and so on. We have employed uniform crossover in the proposed algorithm. The legitimate chromosome that passes the status check is the resultant chromosome. After crossover, a mutation can occur. Another genetic algorithm operator is mutation. It is a rarely utilized technology, yet it is quite helpful to create a diversified population. We choose 'm' random places and stuff each one with a new node value that passes the state check during the mutation procedure.

After the mutation, we calculate the fitness of the new chromosomes and keep them (chromosomes) in a temporary and constant population list. In that temporary list, we also included the whole present population. After that, we practice elitism by crossing all chromosomes and assigning them to the temporary population. Elitism is stated as a percentage of 20%, implying that we will pass down 20% of the best chromosomes from the current population to the next generation. The remaining members of the temporary population will be chosen by sorting them (chromosomes) by fitness value and picking the best 80%. It is the case when the user-defined algorithm comes to a standstill.

Algorithm 1: deadline-oriented Service Placement (DoSP)

Inputs: No. of applications, No. Application modules, CPU, Memory Storage, make-span time, deployment time, population.

Output: Efficient and deadline-aware service placement

// build the first generation

Chromosome Length = total No. of applications * No. Application modules;

Number of Placement Locations = FN or CN or NFCN or FCN;

for *i* = 1 to *population Size* **do**

new Chromosome = create Chromosome (*Chromosome Length*, *Number of Placement Locations*);

if (state-check (Chromosome))

calculate fitness of new Chromosome;

add new Chromosome to *population*;

i++

end for

```
        fittest Chromosome = get Fittest Chromosome(population);
 if fitness of fittest Chromosome > 0 then
                return fittest Chromosome;
        add the current population to the temporary population;
 number Of Chromosomes Used for Crossover = generation Size * defined Crossover Percentage;
        number Of Crossovers = number Of Chromosomes Used for Crossover / 2;
// two chromosomes are involved in each crossover
        for j = 1 to number Of Crossovers do
                parent Chromosomes = select Chromosomes(population); // selection
                children Chromosomes = mate Chromosomes (parent Chromosomes); //
crossover
               state_check (children Chromosomes)
          state_check (mutate Chromosomes (children Chromosomes)); //mutation and state
check
                calculate the fitness of the new chromosomes;
                add the new chromosomes to temporary population;
        end for
        add elite chromosomes from temporary population to the new population;
        add remaining best chromosomes from temporary population to the new population;
end for
```

3.6.3. Time Complexity

The time complexity increases as the number of applications and nodes grow; this may be seen using genetic algorithms. The proposed algorithm's asymptotical time consumption or complexity is O (N log N + IPS2), where n denotes the number of applications for sorting, I the number of iterations, P the population size, and S the number of sub solutions.

3.7. Performance Evaluation

3.7.1. Experimental Methodology

We employed fog, fog controller, neighbour controller, and cloud nodes as compute nodes for our benchmarking. Meeting application deadlines while decreasing reaction time and boosting energy efficiency are the performance goals for the proposed work. Figure 3.6 depicts the overall experimental framework. We present a genetic algorithm for an analytics algorithm application placement. We assessed the suggested work by conducting benchmarking analyses with various existing methods. Observations on applications modules location are made through using experimental setup indicated in the preceding section.

In comparison to current approaches such as EdgeWard and Cloud Only, the recommended deadline-aware technique for application module placement is presented. Because EdgeWard is a simple greedy optimization heuristic, it was chosen. Based on resource availability, EdgeWard organises application modules in a bottom-to-top order. In all circumstances, Cloud Only offloads all modules to the Cloud. Response time is a key observation at certain deadlines. A range of distributed data flow-based sequential IoT applications are studied for the empirical study. Simulations using iFogSim have provided the critical insights across application areas such as motion, sound, video, humidity, and temperature.

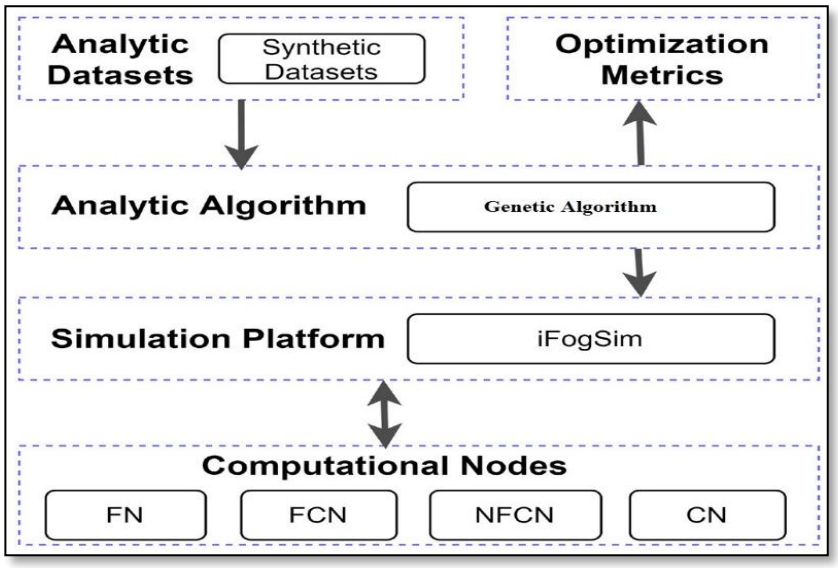


Fig 3.6: Experimental framework

3.7.2. Simulation Setup

To assess the effectiveness of resource management strategies, simulation studies are commonly utilised in fog computing. We employ iFogSim, a simulator toolkit that has been used in a number of research papers as a viable foundation for fog computing research. Experiments must be conducted in a safe atmosphere. Table 3.1 summarises the setting employed in this research endeavour for the empirical investigation.

Table 3.1: Simulation Setup & its Configuration

Processor	Intel core i5-2430 CPU, 2.40 GHz
Memory	8 GB
Simulator	iFogSim
Operating system	Windows 7 Professional
Topology model	Hierarchy

iFogSim with JDK 1.8 was used to conduct the simulation investigation. A visual model of the fog computing infrastructure is available through our simulation framework's API.

It may be used to simulate the fog, fog controller, cloud nodes, gateways, sensors, as well as actuators, among several others. 1 Fog Orchestration node, 10 Fog Nodes governed through by a Fog Controller Node, 1 Neighbor Controller Node, and a Cloud make up the network. The processing capabilities of the node may be configured to such potential measured in millions of instructions per second. Drag-and-drop functionality in the iFogSim allows users to design and comprehend with intuitively.

Figure 3.7 depicts the fog infrastructure that has been built for the planned simulation investigation. Different nodes can be accommodated in each fog cell. After the modelling is finished, the nodes are properly configured. In the simulation study, each node is modelled like a real-world unit with similar features. As a result, it's vital that they're set up correctly. The fog node and fog controller nodes are configured for various settings in Table 3.2. Among the criteria are the computational capacity, memory, and storage details.

Other parameters are required for simulation, as stated in Table 3.3. The communication link delays are taken into account when configuring these parameters (in seconds). The outcomes of the simulation study will be easier to notice and measure as a result of this. It is critical to keep track of numerous parameters when simulating the service placement scenario because they are employed in real-world fog computing networks.

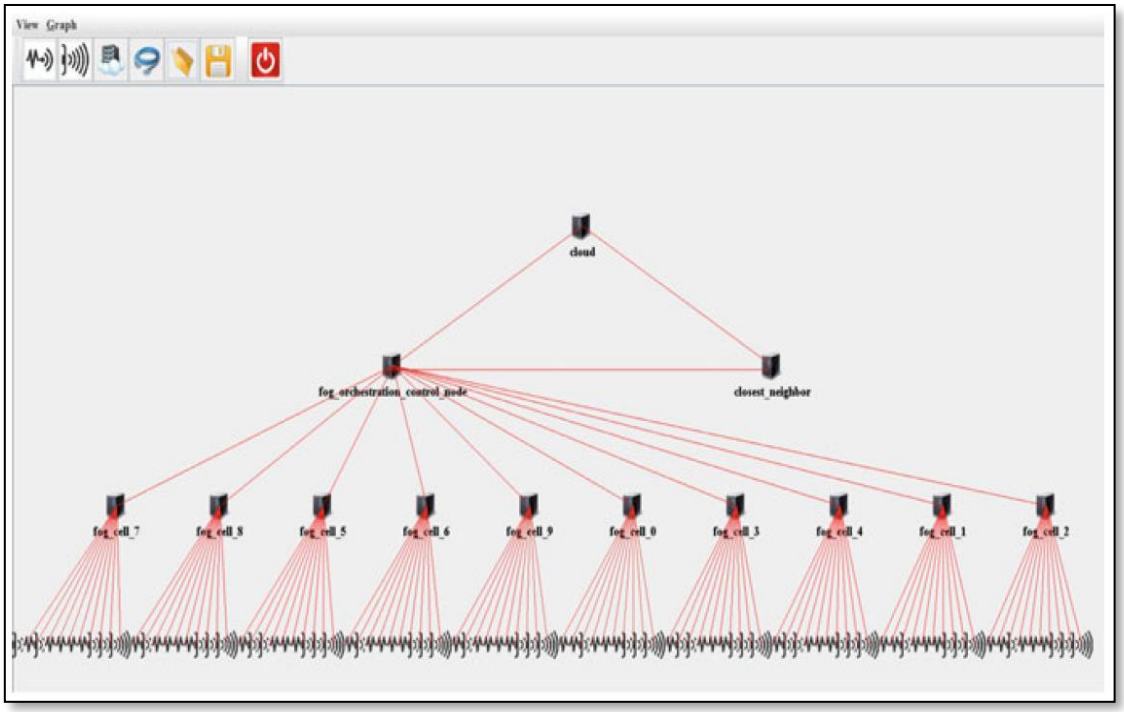


Fig 3.7: Fog infrastructure modeled using iFogSim for the simulation study

Table 3.2: Characteristics of the fog controller node and fog node

Parameter	Fog controller node	Fog node
Processing rate (MIPS)	1000	250
Memory (MB)	512	256
Storage (GB)	8	4

Table 3.3: Different parameters and their corresponding communication link delay in seconds

Parameter	Communication link delay (sec)
Fog controller node-cloud	9
Fog controller node-neighbor controller node	0.5
Fog controller node-fog node	0.3

We set up the node communication link latency as 0.3 and delay as 0.5 for the fog controller. Seconds. The fog controller node–cloud communication link is set to a 9-second delay. After such setups have been completed, we must take care of service modules as well. They require various resources such as CPU, memory, storage, and time. Table 3.6 shows the multiple arrangements. These specifics are provided to conduct a thorough simulation analysis. The actuation module, for example, is configured with a 50 CPU (MIPS), 20 MB main memory, 10 MB storage, and a 0.50 makespan time in seconds.

Table 3.4: Different parameters and their corresponding communication link delay in seconds

Node Type	**No. of Nodes**
Fog	10
Fog Controller	1
Neighbor Controller	1
Cloud	1

Table 3.5: Required resources for application modules

Service module	CPU (MIPS)	Memory (MB)	Storage (MB)	Makespan (sec)
Sensing module	50	30	10	0.90
Data aggregation module	200	10	30	0.10
Data analysis module	200	20	30	0.10
Decision making module	100	30	30	0.25
Actuate module	50	20	10	0.50

Table 3.7 shows the parameters, deadlines, and deployment configurations for various types of applications. The application deadline and deployment time for motion, video, sound, temperature, and humidity applications are determined and provided—tables 3.5,3.6,3.7 picture that the assumed values were based on the average of the preceding experimental run.

Table 3.6: Deadline and deployment time of each application

Application type	Application deadline	Deployment time
Motion	120	60
Video	300	0
Sound	300	60
Temp	360	60
Humidity	240	0

3.7.3. Analysis and Results

As illustrated in Fig. 3.7, the deadline for Motion Application is 120 seconds, whereas the deadline for Video Application is 300 seconds. The horizontal axis shows several techniques for placing services, while the vertical axis shows reaction time in seconds. There was a deadline breach in the instance of a motion application, according to the results. Both the EdgeWard and Cloud Only approaches are 184.45 and 31.85 seconds late, respectively. The EdgeWard, Cloud Only, and DoSP methods might be useful in ensuring that the video application's length restriction isn't exceeded.

The findings are provided for sound and temperature applications, as shown in Fig. 3.8. The reaction time for IoT applications with deadlines of 300 and 360 seconds is measured. The findings found that in the case of sound application, there were deadline violations. The EdgeWard approach could not deliver optimal service placement since it was 4.45 seconds late. In the case of a temp application, the three ways are used to place the service without infringing the deadline. EdgeWard, on the other hand, took longer to respond, and the DoSP technique took the shortest.

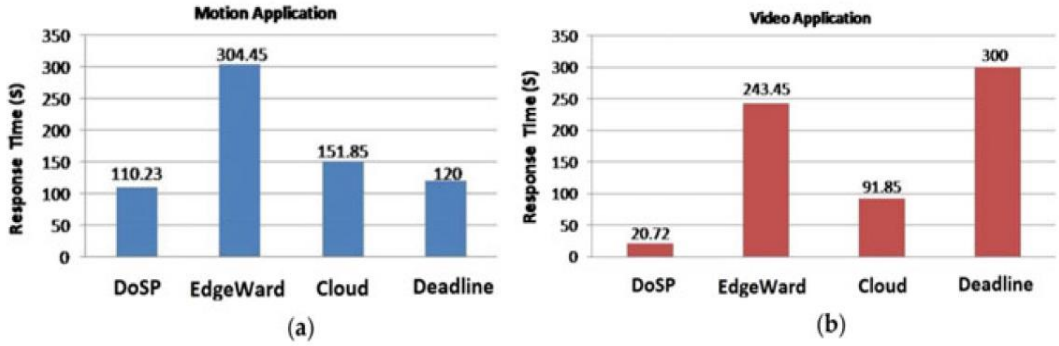


Fig 3.7: Performance comparison with a) motion, and b) video applications

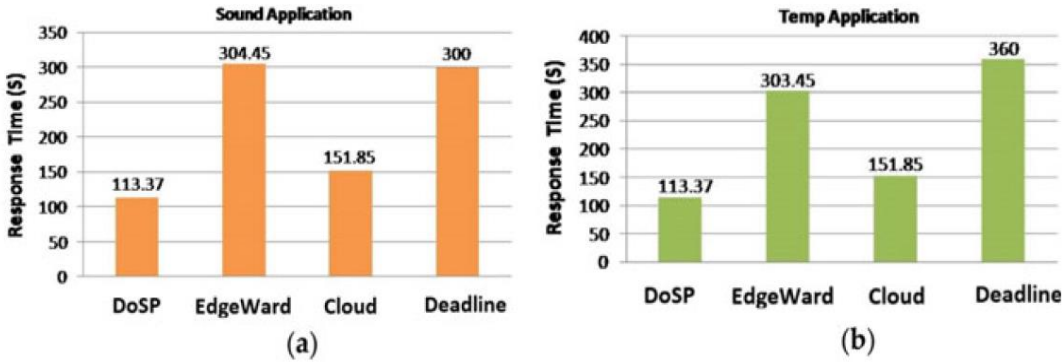


Fig 3.8: Performance comparison with a) sound and b) temp applications

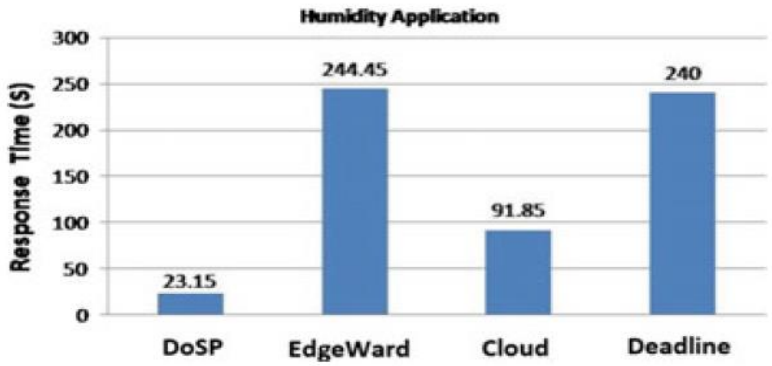


Fig 3.9: Performance comparison with humidity application

The performance of service placement approaches revealed varying reaction times, as shown in Fig. 3.9. The proposed DoSP method has the potential to meet the deadline. It is the same with the Cloud-Only strategy. The deadline is missed by 4.45 seconds when using the EdgeWard technique. This level of performance is unacceptable since SLAs may be related to deadlines in cloud and fog computing scenarios.

3.7.3.1 Response Time Against Deadline

In order to evaluate the suggested DoSP method, response time, also referred as latency, is an important parameter to measure. This response time with a specified deadline is an important empirical observation. As shown in Fig. 3.10, these findings were obtained in terms of reaction time versus various deadlines. The results of all of the applications are displayed with a 1 second communication distance between both the fog master control station and the Cloud. On the bottom axis, deadlines run from 50 to 500 seconds, with each increment of 50 seconds.

The vertical axis depicts the response time. Humidity and Video applications began when the elapsed time was 50 seconds, whereas other applications started at 100 seconds. Responding time values may be grouped into three categories, according to the findings: low, medium, and high. When services are deployed in a fog master controller node, response time is lowered, implying greater performance. Whenever services were being hosted in the Cloud, its response time is stated as medium, which is faster than the fog controller node's response time. When services are installed in an adjacent fog controller node, the response time is increased due to its deployment delay and extra deployment time needed to position the service

modules. As a result of these observations, the observed patterns in reaction time may be explained. The data indicated that the deadline has an impact on response time. The explanation is so that the suggested DoSP algorithm tries to promote service placement which may not exceed the stated timeframe. The results showed that the recommended approach performs deadline-aware, with no violations across the board. It should suggest that DoSP has made appropriate service placement selections in light of the timeframe and available resources.

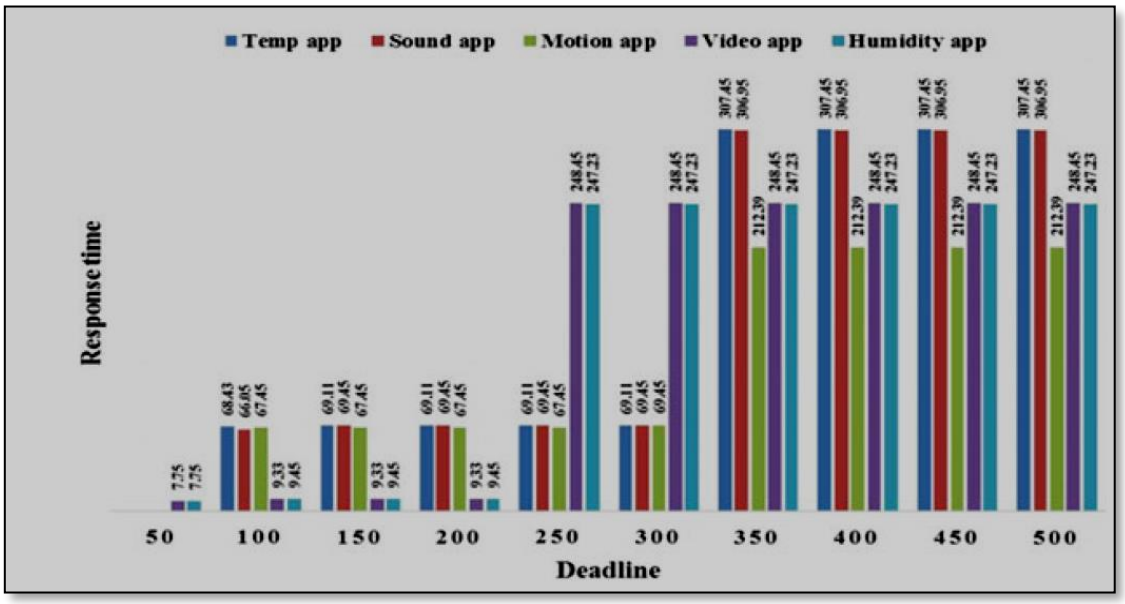


Fig 3.10: Performance of the DoSP method concerning all applications

3.7.3.2 Resource Utilization Analysis

Using the suggested DoSP approach, service placement selections are made based on an analysis of resource availability. The EdgeWard technique places both sensing and actuating services in multiple fog nodes all across the fog layer. Other functions that are relocated to the controller node and propagated to the next fog master controller node include data collecting, analysis, and decision making. In the DoSP, things are a little different. The sensor and actuation units are housed in the fog nodes. Processing-related modules, on the other hand, are located either in the fog master controller node or cloud node.

As according resource analysis, the fog node is responsible for 40% of all service deployments, which is considerably faster than the rest of the system (see Fig. 3.11).

The service placement of the fog controller node is 12 percent in the closest neighbour node and 6 percent with in fog controller node. Cloud deployments account for the remaining 42% of service deployments. As previously noted, there is a distinction in EdgeWard conditions. It splits the services between fog nodes, nearest neighbour nodes, and fog orchestration control nodes, with 40 percent going to fog nodes, 32 percent going to closest neighbour nodes, and 28 percent going to fog orchestration control nodes.

The Cloud Only technique is used to store everything on the cloud. As a result, the fog nodes and fog master controller nodes are not available. It has the drawback that perhaps the Cloud Only services placement may not meet the requirements of specialized sequential IoT workflow applications that demand quick response times. The Cloud significantly slower than all of the other levels in the fog framework.

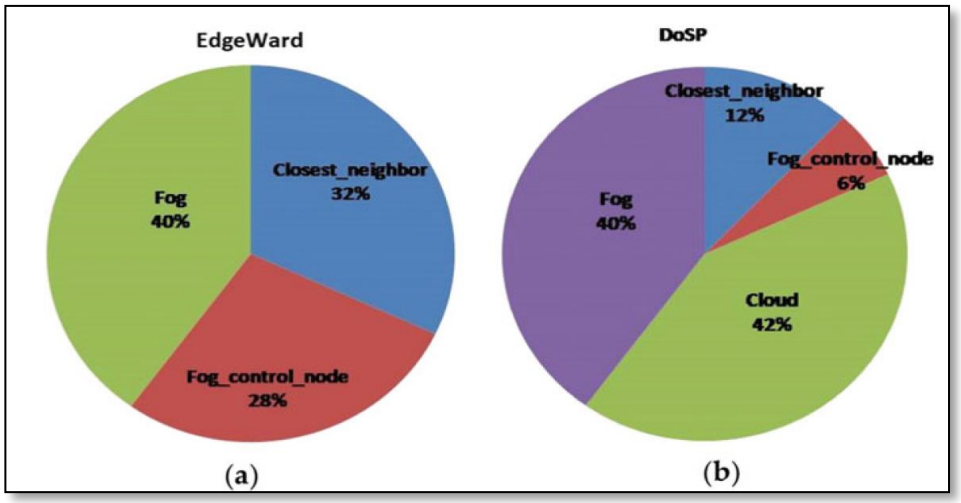


Fig 3.11: Resource utilization for a) Edge Ward method and b) DoSP

3.8. Summary

The proposed Deadline-Aware Dynamic Service Placement (DoSP) method is designed for dealing with a large number of sequential workflow applications that follow the sense-process-actuate paradigm and contain the same set of modules for each application. The algorithm uses the DDF model to deploy the applications and every service must be hosted across any one of the available processing nodes.

The dynamic nature of the DoSP algorithm allows it to adapt to changes in the network environment, such as fluctuations in the availability of processing nodes and

changes in service deadlines. As a result, the algorithm can optimize service placement to ensure efficient resource utilization and meet tight SLA requirements.

This approach is particularly interesting because it has the potential to accelerate the introduction of Internet of Things (IoT) applications in fog computing. The DoSP algorithm is tested using the iFogSim simulator, and the results indicate that it outperforms previous approaches such as EdgeWard and Cloud Only.

DoSP's placement strategy is unique, allowing it to enhance performance and reduce reaction time in the face of tight Service Level Agreement (SLA) requirements. Although the study only used five workflow-based IoT apps, the assumed values are calculated by taking the average of the previous experimental run.

Chapter 4

Deadline and Energy Efficient Dynamic Service Placement (DEEDSP)

4.1. Introduction

In the next five years, the number of Internet of Things (IoT) objects is predicted to exceed fifty billion. Wearables, autonomous vehicles, drones, robotics, augmented reality (AR), and virtual reality (VR) devices are just a few examples of items that use a lot of power and compute. The seamless integration of all "things" connected in the network poses an unprecedented difficulty. Due to the tremendous surge of data traffic generated by billions of IoT devices, the traditional cloud computing paradigm has proven untenable under time-critical needs. Fog computing is the idea of adding a new networking layer, storage, processing, and resources at the network's edge or end-users. Fog is needed to improve Quality of Service (QoS) characteristics like lowering latency, increasing efficiency, and better overall service to end-users. Sensing and actuation modules, for example, are IoT services that connect data with IoT or edge devices in an IoT application. Real-time applications such as intelligent lighting systems, automobile networks, smart grid, pipeline monitoring, wired trains, wind farms, petroleum and gas sector applications, and industrial loop control benefit greatly from these capabilities. A distributed computing paradigm is gradually replacing the current central computer paradigm, in which IoT system control, data, and intelligence are exclusively available on the Cloud. On the other hand, meeting such criteria requires careful selection of hosting systems and appropriate application assignments.

Application placement decision-making is an NP-hard problem. When compared to cloud computing, the placement of modules in a fog computing environment becomes more difficult. The reasons for the increased complexity are as follows:

- Devised diversity (in terms of device configuration, hardware, and software, such as operating systems),

- The location of IoT devices must be considered in IoT application placement, and there are a large number of these devices, adding to the complexity of placement; and,
- Specific application constraints, such as computation and latency requirements, must be met.

Cloud servers consume more electricity due to their vast scale and expanding service needs. The fog computing system's energy saving is also vital and must be taken into account. On the other hand, ensuring QoS, such as application response time within the stipulated deadline, is equally crucial. In the fog computing system, we investigate the trade-off between power usage and response time. Based on the most relevant and novel approaches published in IoT, cloud computing, and fog computing, we proposed a framework for a deadline-aware and energy-efficient service placement strategy. By making effective decisions dynamically, the suggested method brings originality to the fog-computing environment. Various heuristic techniques are used in the proposed hyper-heuristic-based algorithm. This technique combines the strengths of heuristic algorithms like simulated annealing (SA), genetic algorithms (GA), and particle swarm optimization (PSO) into a single approach.

The aim of the proposed method is to find optimal locations for services, which may include fog nodes, fog controller nodes, neighbor controller nodes, and cloud nodes, while ensuring that the response time of the application does not exceed the deadline and the energy consumption is minimized. The article highlights several key contributions in this regard.

- For IoT devices, we proposed a hyper-heuristic based on above mentioned technique that prioritizes different user applications and places application modules in a fog environment.
- DEEDSP makes dynamic judgments about mapping application services in a fog computing environment to meet application deadlines and save energy.
- We evaluated DEEDSP's efficiency in this fog environment using the iFogSim simulator, which revealed a significant reduction in energy utilization and service response time when compared to conventional techniques.

4.2. Proposed Methodology

Figure 4 depicts the proposed system. 1. The framework described in Figure 4 allows for deadline-aware service placement. 2. The placement of services is carefully considered in terms of energy efficiency. An IoT application's Quality of Service criteria can be taken into consideration when deciding where to locate services in a fog environment. The input is a sequential workflow IoT applications. Following that, the proposed system must analyse its needs and utilise an algorithm to develop an appropriate service location that takes into account service deadlines and energy efficiency.

The services of workflow-based IoT applications are placed in Fog, fog controller, neighbor fog controller, or cloud nodes according to the deadlines and energy consumption associated with the services—the service location results in energy efficiency and satisfies deadlines.

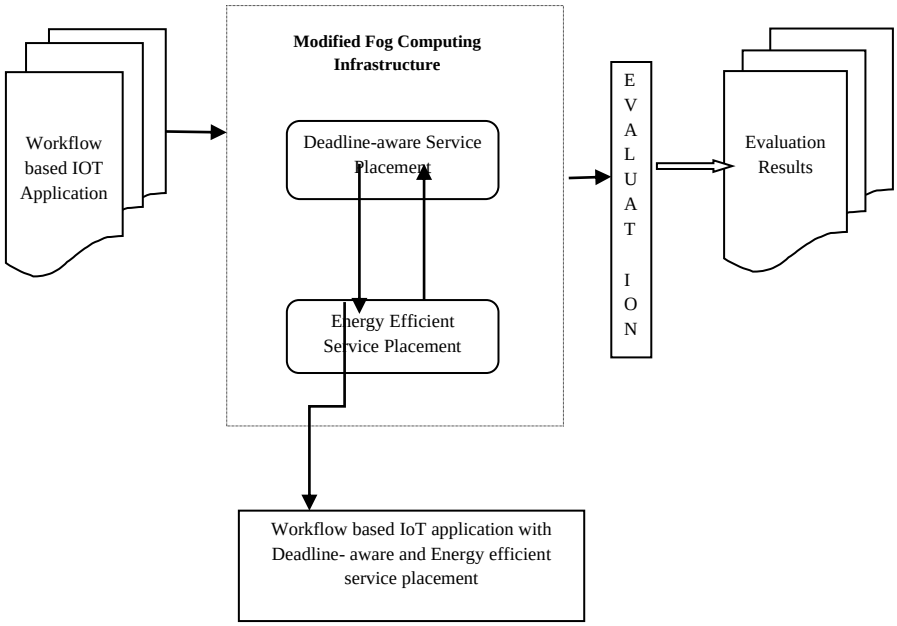


Fig 4.1: Proposed Methodology

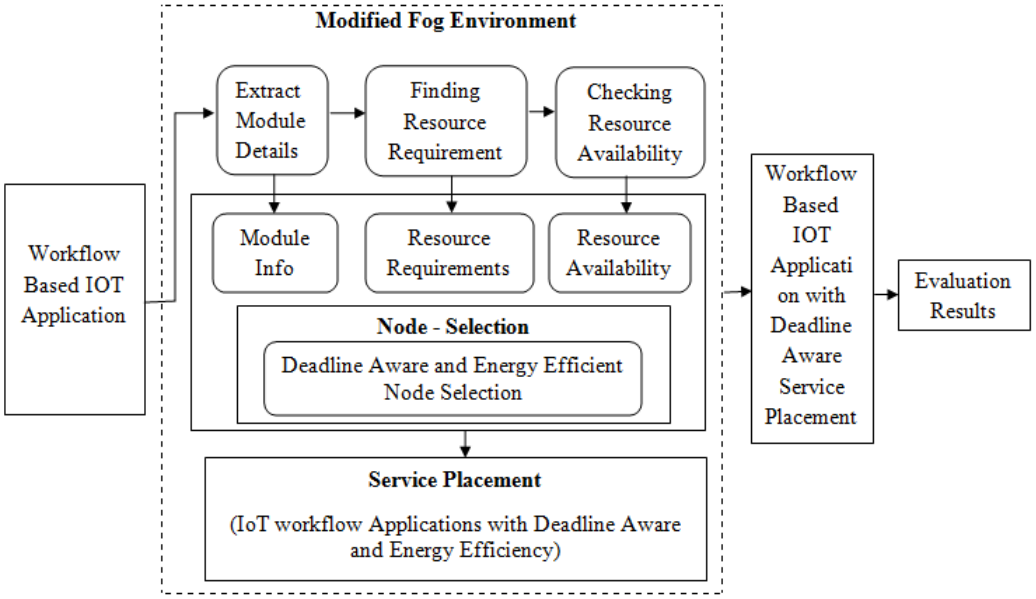


Fig 4.2: Modified Fog Environment

4.3. Application Model

Prior to the programming, software modularity is being required to be performed with in distributed environment, it is necessary to break it down into smaller modules. The concept of a distributed application is aided by the groups of interconnected modules. Every module in the program is necessary that performs at least part of the functions of the application. Such interconnections among application modules are known as application edges; if two application modules have an edge between them; it signifies that both are dependent on one other. The distributed data flow model is a directed network that may be whether sequential and unidirectional, and it depicts application relationships and data flow.

4.3.1. Application Sequence Diagram

Figure 4.3 depicts application modules deployed to VMs in fog nodes, cloud nodes, fog brokers 1 and 2, and fog brokers 1 and 2. After processing, the application modules are transferred back to fog broker 1 to be combined and supplied to the user.

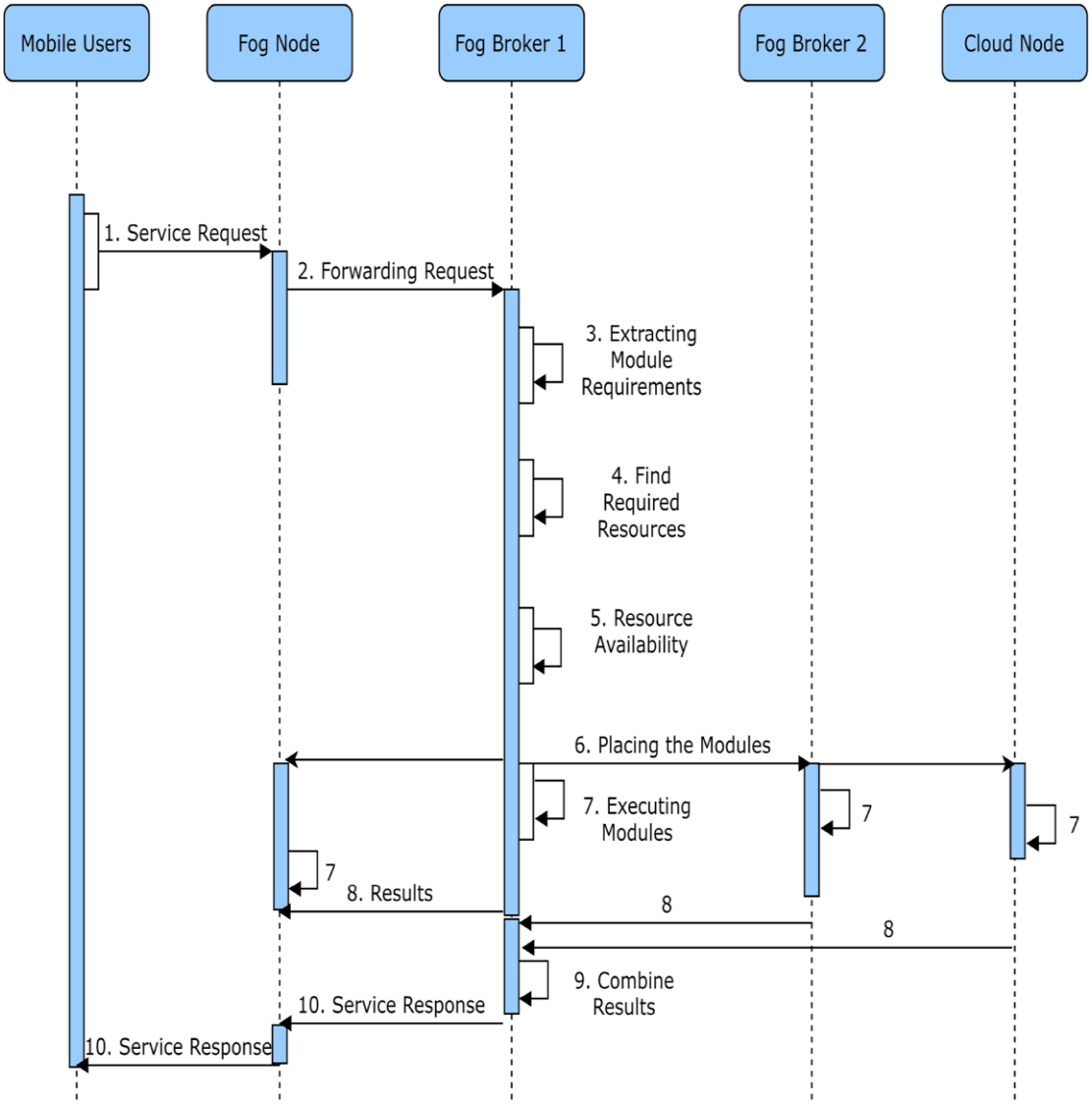


Fig 4.3: Application sequence diagram

Step 1: The mobile user requests for the application, this request is handled by the fog node which it is connected.

Step 2: The fog node forwards the request to the fog broker 1.

Step 3: Fog broker 1 extracts and sequences the modules of the requested application.

Step 4: Fog broker 1 finds the required resources of the modules for execution.

Step 5: Fog broker 1 finds the resource availability information from itself, fog node, fog broker 2, and cloud node.

Step 6: Fog broker 1 runs the placement algorithm to find the optimal module placement.

Step 7: The module assign nodes are responsible for processing the assigned module.

Step 8: The assigned nodes send the results of the modules back to the Fog broker 1.

Step 9: Fog broker 1 combines the results which are send by the module assigned nodes.

Step 10: The response send to the mobile user through the fog node which it is connected.

4.3.2 Application Process flow

Figure 4.4 depicts how applications are scheduled in the fog controller node's application queue to achieve the goals of minimizing response time and maximizing energy efficiency for all applications. To implement the application buffering policy, we can use the application queue. As shown in Figure 4.1, we suggest a parallel virtual queuing paradigm at the fog controller node that buffers arriving application modules of the same type into a separate virtual queue. In the fog environment system, the framework supports a variety of IoT applications.

4.4. Model for Performance metrics (formulas & Constraints)

4.4.1. Application response time:

The optimization problem is designed to reduce reaction time while meeting constraints such as deadlines and energy consumption.

$Res_{\text{time}}(n)$, is the response time of an nth application. The application response time is calculated as follows:

$$Res_{\text{time}}(n) = MKspan_{\text{time}}(n) + Total_{\text{time}}(n). \quad (1)$$

Where:

- The application makes span time, MKspantime(n), is made up of the time it takes to execute all application modules, Exectime(n), and the time it takes to communicate with the controller node's application modules, Commtime(n).
- The time elapsed until we properly located each service on the computational framework resource sources or computing nodes is factored into the total deployment time of an application TotalDept(n).

$$MKspan_{\text{time}}(n) = Exec_{\text{time}}(n) + Comm_{\text{Dept}}(n). \quad (2)$$

$$Exec_{\text{time}}(n) = \frac{Module\ Capacity_n}{Node\ Capacity_n}. \quad (3)$$

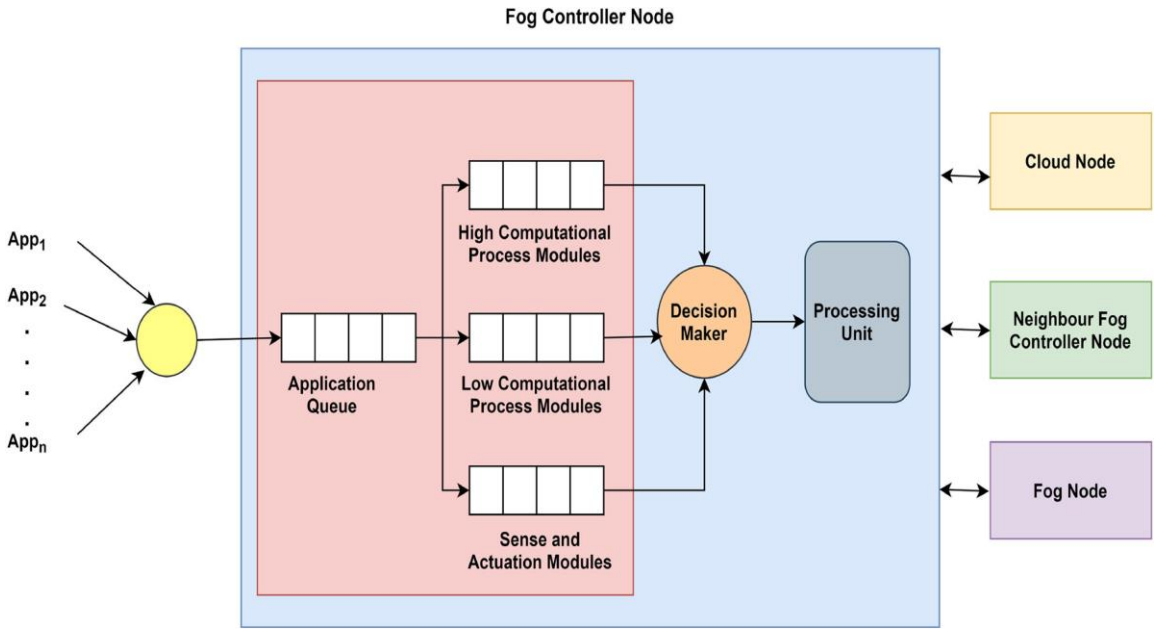


Fig 4.4: Queuing the Application Modules

*Exec*time (n) is the module capacity divided by the node capacity. Node capacity is "the CPU power of the computational node where the application service is deployed," whereas module capacity is "the required CPU power for the module." The communication time is the sum of the undeniable communication delays between the nodes in which each module in an application is located.

The distances between the fog node (FN), the neighbor control node (NFCN), the controller node (FCN), and the cloud node (CN) are represented by the variables distfog, distnbr, distctrl, and distcloud, respectively. The fog controller node serves as a manager, assigning modules to the appropriate nodes. The controller node receives the results from the nodes and sends them back to it. As a result, communication between the nodes takes place twice.

$$MKspan_{\text{time}}(n) = \sum_{i=0}^{m} \{Exec_{\text{time}}(n_i) + dist_{\text{fog}} * x_{\text{fog}} + 2 * dist_{\text{nbr}} * x_{\text{nbr}} + dist_{\text{ctrl}} * x_{\text{ctrl}} + 2 * dist_{\text{cloud}} * x_{\text{cloud}}\}. \quad (4)$$

In Equation (4), 'm' is the number of modules in each application. Total deployment time of the application $\text{Total}_{\text{Dept}}(n)$ Consists of $\text{Dept}_{\text{time}}(n)$ and the estimated extra time when an application module n_i is sent to the adjacent controller node.

Here n_i represents i^{th} module in n^{th} application. The x_{fog}, x_{nbr}, x_{ctrl}, and x_{cloud} are binary decision variables. The value of these variables will be 1 if the n_i is placed in the appropriate node, else the value is 0. The additional deployment time comprises of delay for propagation $P_{Delay}(n)$ and expected deployment time $\text{Expt}_{\text{Dept}}(n)$ in neighbor controller node.

We depend on the variable F(n) to cover the additional deployment time. If the neighboring controller node has at least one application module of n^{th} application propagated then F(n) = 1 otherwise F(n) = 0. If F(n) == 1, we add $P_{Delay}(n)$ and $\text{Expt}_{\text{Dept}}(n)$ to Dept(n). Else we add nothing to Dept(n). We formalize the total deployment time of an application $\text{Total}_{\text{Dept}}(n)$ as follows:

$$Total_{\text{Dept}}(n) = Dept(n) + \begin{cases} P_{\text{Delay}}(n) + Expt_{\text{Dept}}(n) & : F(n) = 1 \\ 0 & : F(n) = 0 \end{cases}. \quad (5)$$

4.4.2. Energy Consumption

Only the energy consumption of compute nodes is taken into account; communication and cooling energy are omitted. Our goal is to reduce overall energy consumption,

including energy consumed when working and consumed while idle. When the computation node is inactive, the static energy is depleted. In contrast, dynamic energy pertains to the host's resource utilization. Util depicts the consumption point of the host node. The nominal power implied by dissipating the entire power unit is E_{max}. The number of modules in each application is denoted by the letter m.

$$E_{\text{node}} = E_{\text{static}} + E_{\text{dynamic}}, \tag{6}$$

$$E_{\text{dynamic}} = (E_{\max} - E_{\text{static}}) * Util, \tag{7}$$

$$E_{\text{total}} = \sum_{i=0}^{m} E_{\text{FN}} + E_{\text{FCN}} + E_{\text{NFCN}} + E_{\text{CN}}. \tag{8}$$

4.4.3. Placement Constraints

Let placefog(ni), placectrl(ni), placenbr(ni), placecloud(ni) ∈ {0, 1} be binary factors that specifies whether the module ni is deployed on a fog node (placefog(ni) = 1), or a fog controller node (placectrl(ni) = 1), or a neighbor controller Nodeplacenbr(ni) = 1), or on the cloud node (placecloud(ni) = 1). Since themodule is deployed only once, in the constraint1 is held:

$$constraint_1 : \{place^{\text{fog}}(n_i) + place^{\text{ctrl}}(n_i) + place^{\text{nbr}}(n_i) + place^{\text{cloud}}(n_i)\} \quad \forall n_i \in n, n \in N. \tag{11}$$

The $constraint_1$ value will be 1, because the module will be placed in at least one of the available nodes.

$$constraint_2 : \{Res_{\text{time}}(n) < D(n), \forall n \in N\}. \tag{12}$$

The module placement approach must ensure that Res_{time} (n), defined in $constraint_2$, does not jeopardize the application's deadline, D(n).

$Constraint_3$ in this article is that the application's requested resources, *Req*res*ni*, must be sufficient to the accessible resources, $Avail_{res}$(r, nodes), such as processing power (CPU), storage space (STR), and memory capacity (MEM). It can be stated as follows:

$$constraint_3 : \left\{ \sum_{n}^{N} \sum_{n_i}^{n} Req_{\text{res}} n_i <= Avail_{\text{res}}(r, \text{nodes}) \right\} \quad (13)$$

$r \in \{\text{CPU, MEM, STR}\}, \text{nodes} \in \{\text{fog, ctrl, nbr, cloud}\}.$

The sensor (x_i) and actuation (y_i) modules of the application that we should only deploy at lower fog nodes are represented by constraint4 in this article. The following is how it's defined:

$$constraint_4 : \{place^{\text{fog}}(n_{x_i}) \ \&\& \ place^{\text{fog}}(n_{y_i})\} = 1. \quad (14)$$

The above '$constraints_{1,2,4}$' are applicable independently for each application.

4.5. Our Proposed Approach

4.5.1. Proposed Policy

We used existing methods to process the modules in the fog layer in reality. According to our technique, delay-tolerant applications are placed in the cloud and neighbor nodes, whereas delay-sensitive apps are placed in low computational fog nodes. When comparing the cloud with Fog, the cloud has more computing capacity and has a higher communication delay between the cloud and the device layer.

On the other hand, Fog has limited computing resources and has a shorter communication delay with the device layer. Furthermore, our system considers application deadlines and energy consumption during the placing process. The data used to execute the suggested method is artificial.

$$Objective\_function = \min \left\{ \sum_{n=1}^{N} \{Res_{\text{time}}(n) + E_{\text{cons}}(n)\} \right\}, \quad (9)$$

Where, n is the index of modules in each application and N is the number of applications. The aim of the proposed algorithm is to minimize the objective function presented as above.

4.5.2. Service Placement Algorithm

The placement strategy is divided into 3 phases:

Phase 1: Selection of an application

Phase 2: Selection of module

Phase 3: Selection of the node

These phases are explained in the following subsections.

4.5.2.1. Phase 1: Selection of an application

Algorithm 1. Deadline-aware and Energy-Efficient Applications Placement

```
input : Z{}, H{}, APP[N][M]
Initialize Threshold W.
 Calculate Application Priorities, AppsortedbasedonA_pri[].
 Construct MinHeap(App). ;
 for (i=0;i<N;i++) do
    EnQueue(SA_Queue,App[i][0],App[i][M-1]);
     for (j=1;i<M-1;j++) do
        if (App[i][j] > W) then
            EnQueue(HC_Queue,App[i][j]);
        else
            EnQueue(LC_Queue,App[i][j]);
call Module_Placement(Z,H);
```

Step 1: Applications are prioritized based on their deadlines and deployment times. The application with the shortest deadline-to-deployment time is given precedence. If many ready jobs have the same priority, a FCFS application is chosen. Priorities for the application A_{pri} is a month that is given by:

$$A_{\text{pri}} = D(n) - Dept(n). \tag{10}$$

4.5.2.2. Phase 2: Selection of module

The modules of the application are retrieved upon an application selection. After that, we will look at how the application modules should be classified. It is accomplished in the following way:

Step 2: In the sense and actuation modules queue, selected applications of sense and actuation modules are placed.

Step 3: Place the application's process modules in the high or low computational process modules queues, depending on their CPU value.

The steps mentioned above in 2 and 3 are described in Algorithm 1. Using the Govern function (Hi; Imp Flag; Div Flag), as defined in Algorithm 2 and Algorithm 3, the selected heuristic algorithm H_i will evolve the solution Z for the provided number of iterations.

4.5.2.3. Phase 3: Selection of the node

Step 4: The ModulePlacement function is called by Algorithm 1 to position the modules in the specified nodes. ModulePlacement() selects nodes that satisfy the application module placement restrictions listed below.

Let *place*fog(*ni*), *place*ctrl(*ni*), *place*nbr(*ni*), *placecloud*(*ni*) ∈ {0, 1} be binary factors that specifies whether the module *ni* is deployed on a fog node (*place*fog(*ni*) = 1), or a fog controller node (*place*ctrl(*ni*) = 1), or a neighbor controller *Nodeplace*nbr(*ni*) = 1), or on the cloud node (*place*cloud(*ni*) = 1). Since themodule is deployed only once, in the *constraint*1 is held:

$$constraint_1 : \{place^{fog}(n_i) + place^{ctrl}(n_i) + place^{nbr}(n_i) + place^{cloud}(n_i)\} \quad (11)$$
$$\forall n_i \in n, n \in N.$$

The *constraint*1 value will be 1, because the module will be placed in at least one of the available nodes. The module placement strategy must be assured that *Res*time(*n*) as specified in the *constraint*2, application does not compromise its deadline, *D*(*n*).

$$constraint_2 : \{Res_{time}(n) < D(n), \forall n \in N\}. \quad (12)$$

The *constraint*3 is that, available resources, *Avail*res(*r, nodes*), such as processing power (CPU), storage space (STR), memory capacity (MEM) must be sufficient to required resources, *Req*res*ni*, of the application in the deployment node. This can be defined as:

$$constraint_3 : \left\{ \sum_{n}^{N} \sum_{n_i}^{n} Req_{\text{res}} n_i <= Avail_{\text{res}}(r, \text{nodes}) \right\} \quad (13)$$

$r \in \{\text{CPU, MEM, STR}\}, \text{nodes} \in \{\text{fog, ctrl, nbr, cloud}\}.$

The *constraint*4 represents the sensing (*xi*) and actuation (*yi*) modules of the application that should be placed in lower fog nodes only. It is defined as below:

$$constraint_4 : \{place^{\text{fog}}(n_{x_i}) \ \&\& \ place^{\text{fog}}(n_{y_i})\} = 1. \quad (14)$$

The above $constraints_{1,2,4}$ is applicable independently for each application.

Algorithm 2. Module Placement

input : $Z\{\}, H\{\}, Max\_iterations, Not\_improve$
Select a Heuristic Algorithm H_i from set H.
while ($Max_{iterations}$ *reached or* $Not_{improve}$ *is reduced to zero*) **do**
  Using H_i, improve the population of solutions Z ;
  $Imp\_Flag = Improvement\_Detection()$;
  $Div\_Flag = Diversity\_Detection()$;
  Flag = Govern(H_i, Imp_Flag, Div_Flag) ;
  if ($Flag == TRUE$) **then**
    Select next H_i *from set* H;
    $Z = Fine\_Tune(Z)$;

Step 5: The 2nd algorithm, referred to as module placement, is depicted below. The population of the solution and the candidate pool are the algorithm's first two input parameters. The application modules' placement plans in the computational nodes are used to populate the solution. The module placement algorithm selects the heuristic algorithm H_i from the candidate pool H in a sequential manner. Three hyper-heuristic methods are included in the set H: genetic algorithm, particle swarm optimization, and simulated annealing (H = GA, PSO, SA).

The 2nd algorithm, referred to as module placement, is depicted below. The population of the solution and the candidate pool are the algorithm's first two input parameters.

The application modules' placement plans in the computational nodes are used to populate the solution. The module placement algorithm selects the heuristic algorithm Hi from the candidate pool H in a sequential manner. Three hyper-heuristic methods are included in the set H: genetic algorithm, particle swarm optimization, and simulated annealing (H = GA, PSO, SA). As indicated in Algorithm 2, the selected heuristic algorithm H_i will be rerun until the maximum number of iterations is achieved, or there is no improvement.

Step 6: When the Govern() function returns True, the Module Placement algorithm will choose the next algorithm H_i from H at random. Then, as indicated in Algorithm 3, solution Z is acquired to refine the solution.

Algorithm 3. Govern

input : H, *Imp_Flag*, *Div_Flag*
S = Single Solution Based Heuristic Algorithms
P = Population Based Heuristic Algorithms
if *((H ∈ S && Imp_Flag == TRUE) || (H ∈ P && Imp_Flag, Div_Flag == TRUE))* **then**
  return FALSE ;
else
  return TRUE ;

Step 7:

Algorithm 4. Improvement_Detection

BSDEE = Best so far Deadline and Energy Efficient
if *(BSDEE is not Improved)* **then**
  return FALSE
else
  return TRUE

In Algorithm 4, while selecting the next heuristic algorithm H_i from the candidate pool H a sequential selection method is employed. The Algorithm *Module Placement* suggests when to change to the next heuristic algorithm from the list H_i.

If after a defined number of iterations, the selected *Hi*'s $Not_{improve}$, cannot better the fitness value (which is obtained by BSDEE value) then we pick a new heuristic algorithm by returning a *False* value to the calling function.

The *Improvement Detection* function will suggest choosing a new heuristic algorithm in three situations: when iterations exceed the $Max_{iterations}$, the flag $Not_{improve}$ has been met, and If the condition of termination is satisfied.

Step 8:

Algorithm 5. Diversity_Detection

```
CS = Current Solution
if (CS>Threshold ) then
  return FALSE
else
  return TRUE
```

The Diversity Detection () method in Algorithm 5 detects when to switch the heuristic algorithm from the list H. The threshold value is determined by the diversity of the original solution D(Z). (i.e., the initial solution's fitness values). The diversity of the present solution D is used to calculate the fitness function's average (Z). If the threshold value is smaller than the current solution's diversity D(Z), Module Placement () will choose the next new Heuristic algorithm.

Step 9:

Algorithm 6. Fine_Tune

input : Z
Send the solutions obtained by H_i to the next chosen H_i as input

The fine-tuned answers produced by Hi from the Fine Tune() method are passed to the newly selected hyper-heuristic algorithm in Algorithm 6. It indicates that the heuristic algorithm Hi's candidate solutions can balance the search's intensification and variety. The Fine Tune() algorithm, to be more specific, is responsible for changing the population's solutions.

4.5.3. Time Complexity

The complexity of the running time is proportional to the size of the application list, which is N. To arrange this list of applications, the scheduler requires O(NlogN) time complexity, and the space complexity is expected to be O. (N). Each heuristic algorithm in list H has a time complexity of $O(SP^2)$, where S denotes the number of sub solutions in each problem solution, and P represents the population size. So, if all heuristic algorithms are confined to SP^2 and r is the number of iterations required, the hyper-heuristic algorithm's time complexity is $O(rSP^2)$. The total computing complexity of the proposed algorithm is $O (NlogN + rSP^2)$.

4.6. Performance Evaluation

4.6.1. Experimental framework

For system benchmarking, we must address three critical aspects of representation: computational platform, method, and data analysis on a collection of data. We used a simulation tool and an analytics technique to evaluate the system in the suggested work. iFogSim, FogNetSim++, MyiFogSim, and YAFS are some of the simulation programs available, each with its unique set of benefits. We used the iFogSim simulator, which is the most used fog computing simulator, in this study. Furthermore, in a hierarchical design, the iFogSim simulator is ideal for emulating IoT, fog, and cloud processing nodes.

It's an excellent choice for assessing various essential aspects of IoT applications, such as reaction time and energy consumption. We used fog nodes, fog controller nodes, neighbor controller nodes, and cloud nodes as computing nodes for benchmarking. The proposed work's performance metrics are meeting application deadlines while lowering reaction time and increasing energy efficiency. Figure 4.5 depicts the overall experimental framework.

We present a hyper-heuristic method for application placement as an analytics algorithm. We assessed the suggested work by conducting benchmarking trials with various existing methods.

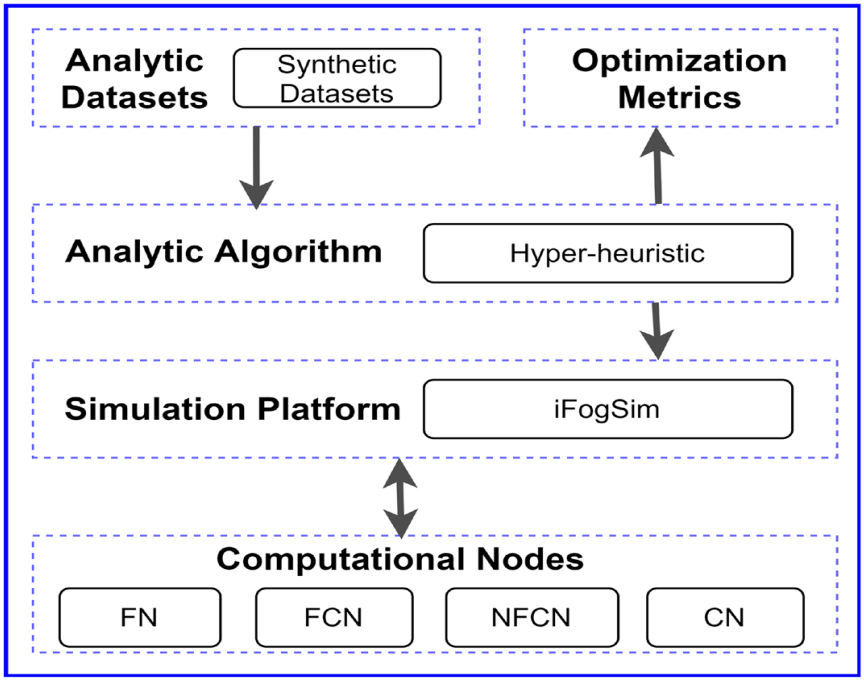


Fig 4.5: Experimental framework

4.6.2. Simulation Setup

The experimental context for our model's evaluation is shown in Table 4.1. The iFogSim simulator uses Java with JDK 1.8 to develop fog infrastructure. In our simulations, the processing power of processors is expressed by MIPS (million instructions per second).

4.6.2.1. Network topologies and resource description:

The suggested approach presents module placement ideas for various scenarios in cloud Network Connectivity from the controller node and applications. The following is the topology of the network:

- The lowest level in the hierarchy has sensors and actuators.
- The next level contains numerous fog nodes (FN) managed by a fog orchestration controller node (FCN).
- The next level includes one neighbor cluster fog controller node (NFCN).
- The top-level contains a cloud node (CN).

Table 4.2 depicts a setup framework that includes a single cluster with ten fog nodes controlled by the FCN and connected to the NFCN and the CN node.

4.6.2.2. Module configuration and resource demands

Motion, video, sound, temperature, and humidity are just some of the applications we've analyzed. Table 4.3 lists the application deadlines and deployment times. Each application is made up of five parts. Sensing, data collection, data processing, decision making, and actuation are among them. The processing power (CPU), storage space (STR), and memory capacity (MEM) of these modules are listed in Table 4.4.

The next part, which explains several application scenarios, provides concrete examples for these concepts. To guarantee that the simulations are diverse, different parameters have been modified. Table 4.5 shows the parameters that affect the results and their values in the GS, DoSP, GA, PSO, SA, and suggested DEEDSP algorithms. The GS29 and DoSP31 algorithms are both GA-based. As a result, the GS, DoSP, and GA algorithms have the same population size, crossover, mutation, and elitism rate.

TABLE 4.1: Simulation setup

System	**Intel Core i5-2430 CPU,2.40GHz**
Memory	8 GB
Simulator	iFogSim
Operating system	Windows 7 professional
Topology model	Hierarchy

TABLE 4.2: Characteristics of the computation node

					Placement	
Node type	**Number of nodes**	**Delays (s)**	**CPU (MIPS)**	**RAM (GB)**	**Power max**	**Power idle**
Fog	10	0.2	100	0.5	88.57	80.23
Fog controller	1	0	1000	2	109.33	85.47
Neighbor controller	1	0.5	1000	2	109.33	85.47
Cloud	1	5	10000	4	170.62	110.82

TABLE 4.3: Applications configuration

Applications	Deadlines	Deployment time
Motion	120	60
Video	300	0
Sound	300	60
Temp	360	60
Humidity	240	0

TABLE 4.4: Required resources of application modules

Service module	Scenario 1 CPU (MIPS)	Scenario 2 CPU (MIPS)	MEM (MB)	STR (MB)
Sensing module	50	50	30	10
Data aggregation module	200	300	10	30
Data analysis module	200	300	20	30
Decision making module	100	300	30	30
Actuate module	50	50	20	10

TABLE 4.5 Parameters of GS, DoSP, and DEEDSP algorithms for IoT applications placement

Algorithm	Parameter values
GA	Population size = 50, crossover rate = 0.5, mutation rate = 0.02, elitism = 0.2, generations = 10
PSO	Swarm size = 50, acceleration rate = 2
SA	Starting temperature = 10, cooling rate = 0.05
DEEDSP	Max iteration of low-level algorithm = 50, nonimproved iteration threshold = 5

4.6.3. Analysis and Results

This evaluation aims to see how successful framework module placement plans are in comparison to various current techniques. Cloud only placement, collaborative task offloading scheme with service orchestration (CTOSO), edge-ward placement (EWP), genetic scenarios (GS), deadlines-oriented service placement (DoSP), genetic algorithms, particle swarm optimization, and simulated annealing are among the methods used.

4.6.3.1 Deadline satisfaction

4.6.3.1.1 Scenario 1

This section examines the outcomes of time frame-aware application placement and energy consumption reduction as functions with varying application complexities (i.e., latency-sensitive, latency tolerable applications with various computational capacities). They were thoroughly examined using existing state-of-the-art methodologies and the suggested approach using the iFogSim simulator. The outcomes are split into two groups as mentioned below:

Scenario 1: It has applications that do not require a lot of processing power.

Scenario 2: It has apps that require a lot of processing power. By comparing them to FN computations, the computations are distinguished.

We will discuss these scenarios in detail.

Each application's response time, Res_{time} is determined by the Equation (5). The execution time ($Exec_{\text{time}}$) and the total time of deployment ($Total_{\text{Dept}}$)is computed with (1) and (2) with the respective data given in Tables 3–5. After computations, the corresponding deadline $D(n)$ of the application in Table 4 shall be compared to Res_{time} with various cloud distances. The results for the first scenario are shown in Figures 4.6–4.8. The cloud-only policy exceeds the application deadline for motion, video, sound, temp, and humidity applications as shown in Figures 4.6,4.7, and 4.8. It exceeds the deadline with an average cloud distance of 25, but it misses the deadline with a cloud distance of 10 because the deadline is short.

At a cloud distance of 50, the CTOSO policy surpasses the deadline in video and temp applications, as demonstrated in Figures 4.6B and 7B, respectively. Because it stores the application in FN, this policy does not exceed the deadline for short deadline applications, even at large cloud distances. Applications with an extended deadline will be given preference for placement in CN, even if submitted after the deadline. Two latency-tolerable applications (video and temperature) exceed the deadline at a cloud distance of 50.

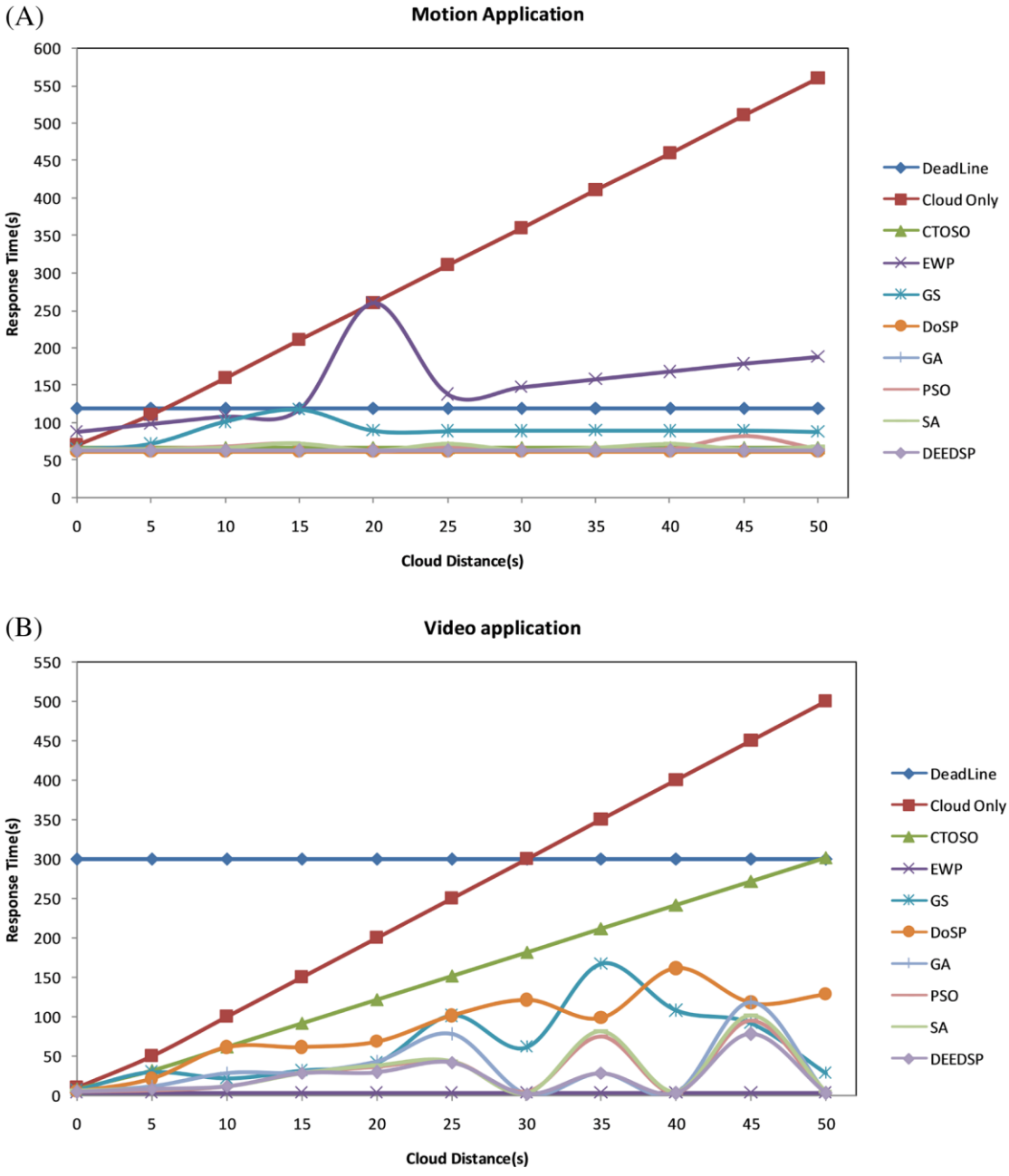


Fig 4.6: Performance comparison of motion and video applications

At a distance of 20, the EWP policy exceeds the move application deadline. Because the previous programs may have used up all of the resources, FN and FCN, the subsequent applications will only have NFCN and CN to work. When the distance between TAU and CN is large, there is a possibility of missing the deadline. In any circumstance, the GS, DoSP, GA, PSO, SA, and DEEDSP policies do not exceed the deadlines.

The application deadline is exceeded by the techniques cloud Only, CTOSO, and EWP. For latency tolerance apps at high cloud distances and latency-sensitive applications with significant compute, CTOSO beats the deadline.

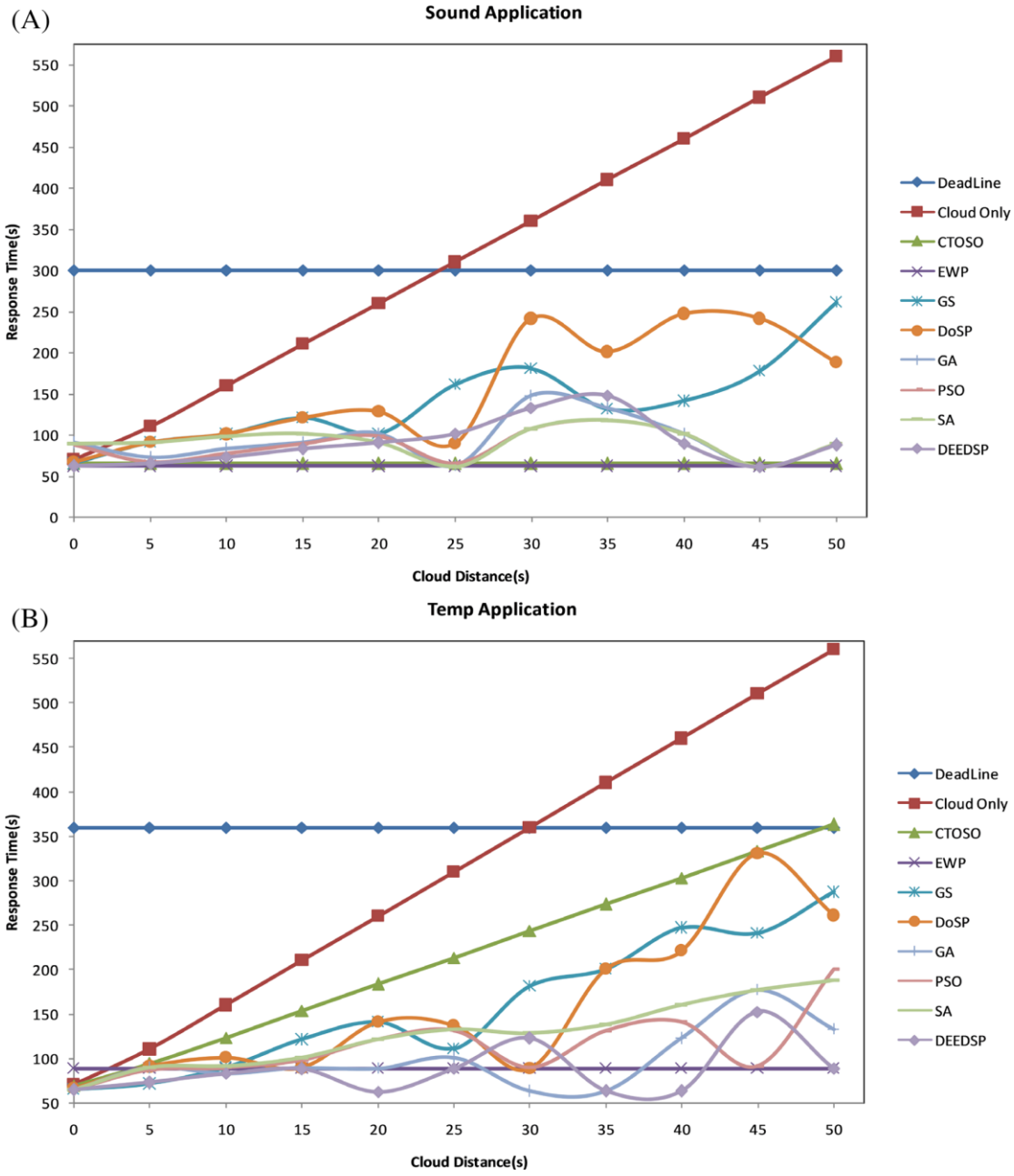


Fig 4.7: Performance comparison of sound and temp applications

For latency-sensitive applications with low and high compute, EWPpolicy exceeds the deadline. DoSP has a faster response time for tight deadlines, but DEEDSP has a faster response time for applications with long cloud distances.

In comparison to GS and DoSP, GA, PSO, SA, and DEEDSP policies yielded a low "combined response time." EWP has proven to be the best for applications like video and temp with long deadlines, but the suggested technique exceeds EWP at a point, 50. DoSP provides the fastest reaction times for latency-sensitive applications in both low and heavy computational activities.

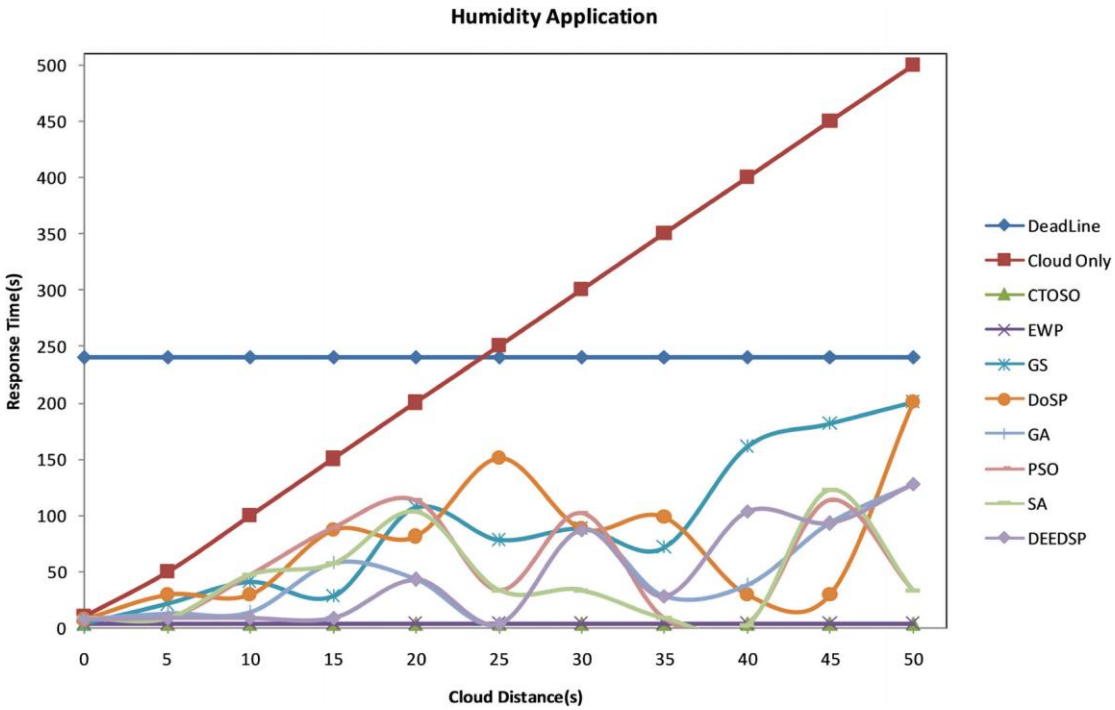


Fig 4.8: Performance comparison of humidity application

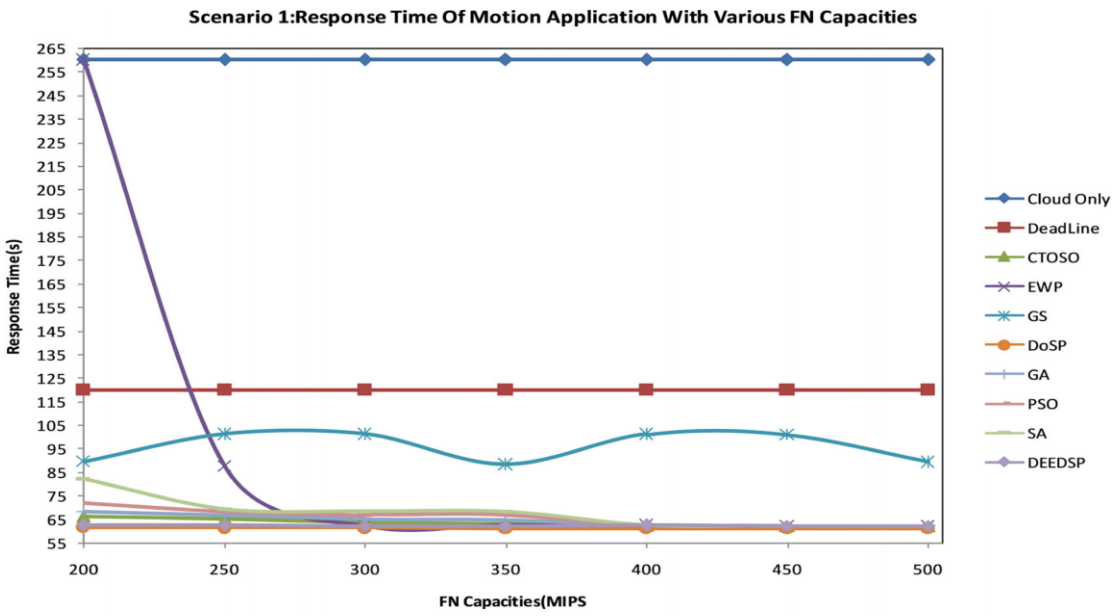


Fig 4.9: Respone time Of Motion Applicaion With Various FN Capacities

In Figure 4.8, we can see that the GS and DoSP policies have pushed the application near its deadline. Figure 4.9 shows the motion application reaction times for TAU = 25, cloud distance = 20, and different FN MIPS. The cloud-only and EWP policies, in this case, overshoot the deadline by 200 MIPS. All other policies had higher "combined response times" than the proposed strategy.

4.6.3.1.2. Scenario 2

In Figures 4.10–4.13, the results for the second scenario are shown. Except for the Motion application, all of the comparisons policies do not exceed the deadline in this scenario at a cloud distance of 10. Only CTOSO and EWP policies exceed the deadline in this motion application cloud. CTOSO misses the deadline because the high computational modules cannot be placed in FN. Because the suggested DEEDSP algorithm places this application in FCN, the results show that DEEDSP performs well as DoSP.

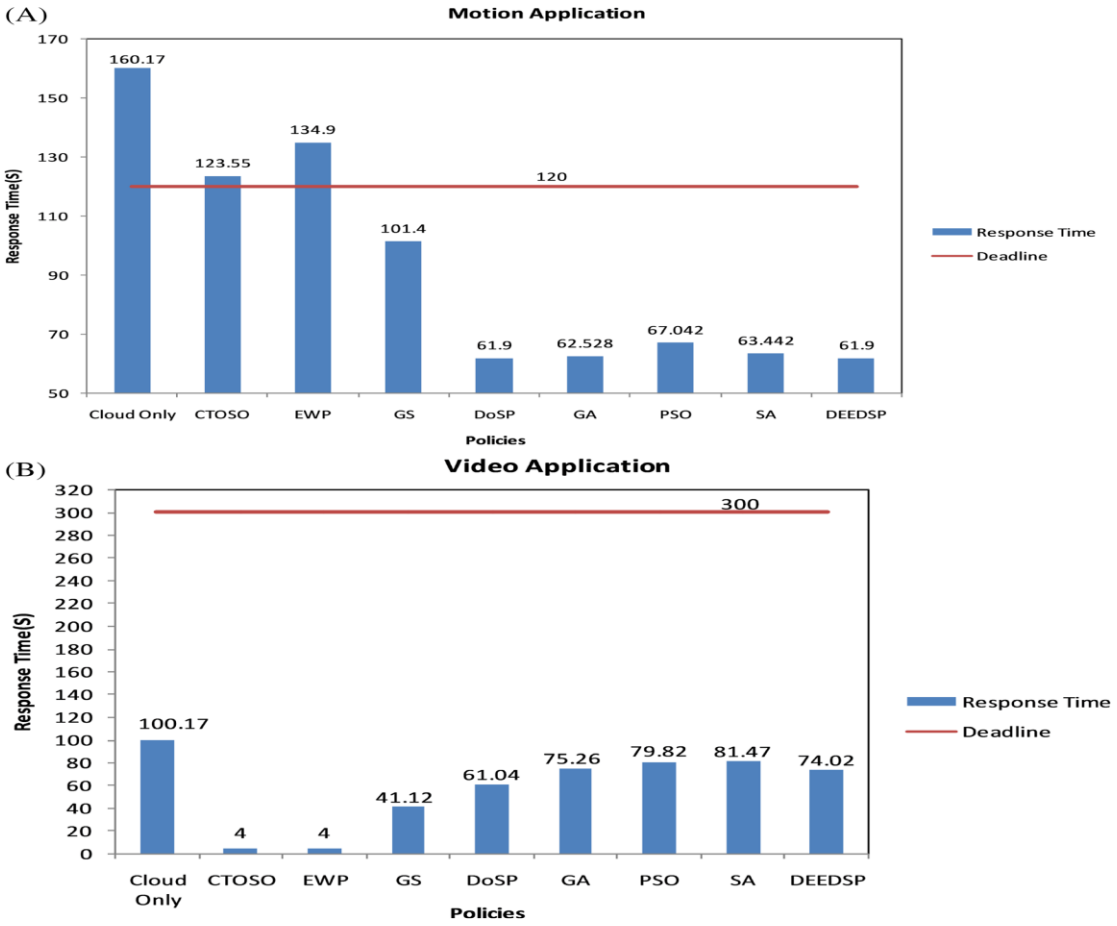


Fig 4.10: Performance Comparison Of Motion and Video Application(H.Gupta et al., 2016, Skarlat et al., 2017, Meeniga Sriraghavendra et al., 2021, M. Huang et al., 2018,Holland JH et al.,1975,Kennedy J,1995,Kirk,1983)

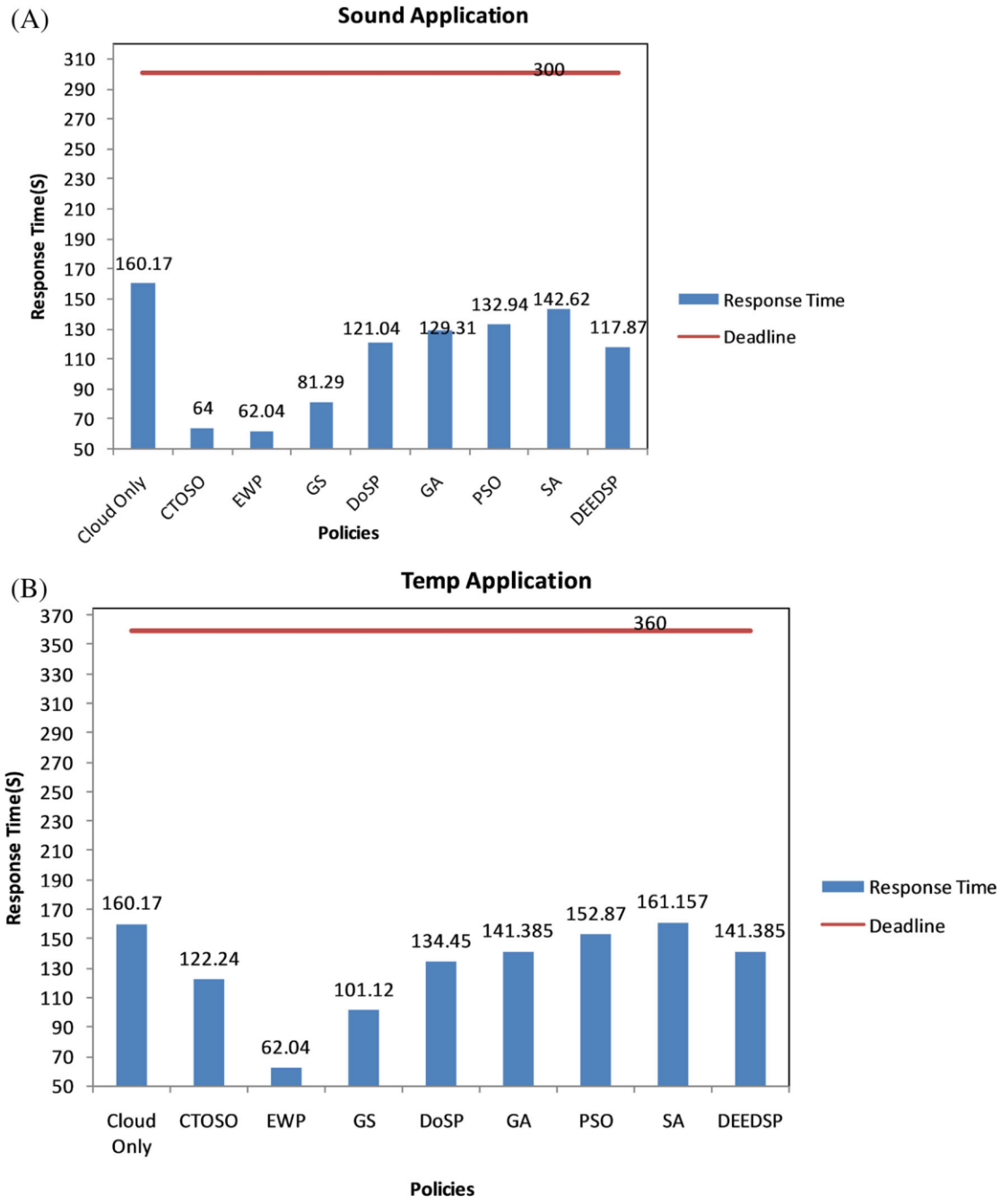


Fig 4.11: Performance Comparison Of Sound and Temp Application(H.Gupta et al., 2016, Skarlat et al., 2017, Meeniga Sriraghavendra et al., 2021, M. Huang et al., 2018,Holland JH et al.,1975,Kennedy J,1995,Kirk,1983)

Figure 4.13 depicts the motion application reaction times for TAU = 25, cloud distance = 20, and different FN MIPS. In this case, the cloud-only, EWP, and CTOSO policies all surpass the deadline until the MIPS exceeds 250. Until MIPS 250, the DEEDSP scheme and DoSP performed equally well; however, DoSP outperforms the proposed DEEDSP algorithm marginally for larger MIPS values.

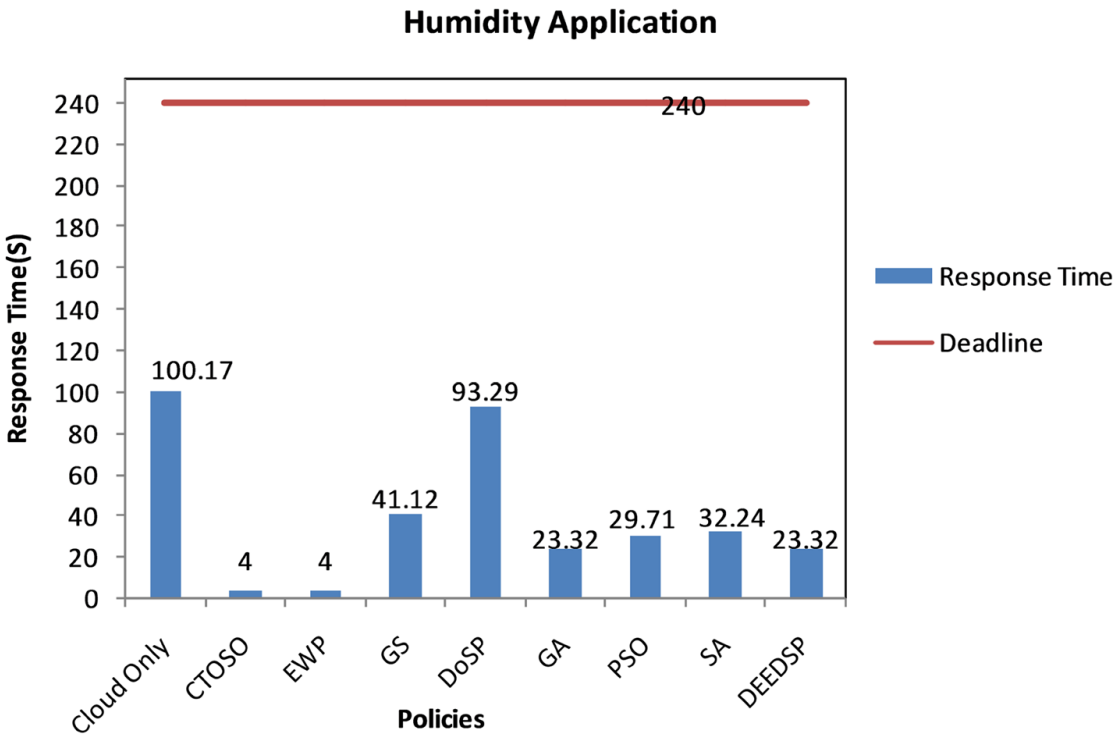


Fig 4.12: Performance comparison of humidity application

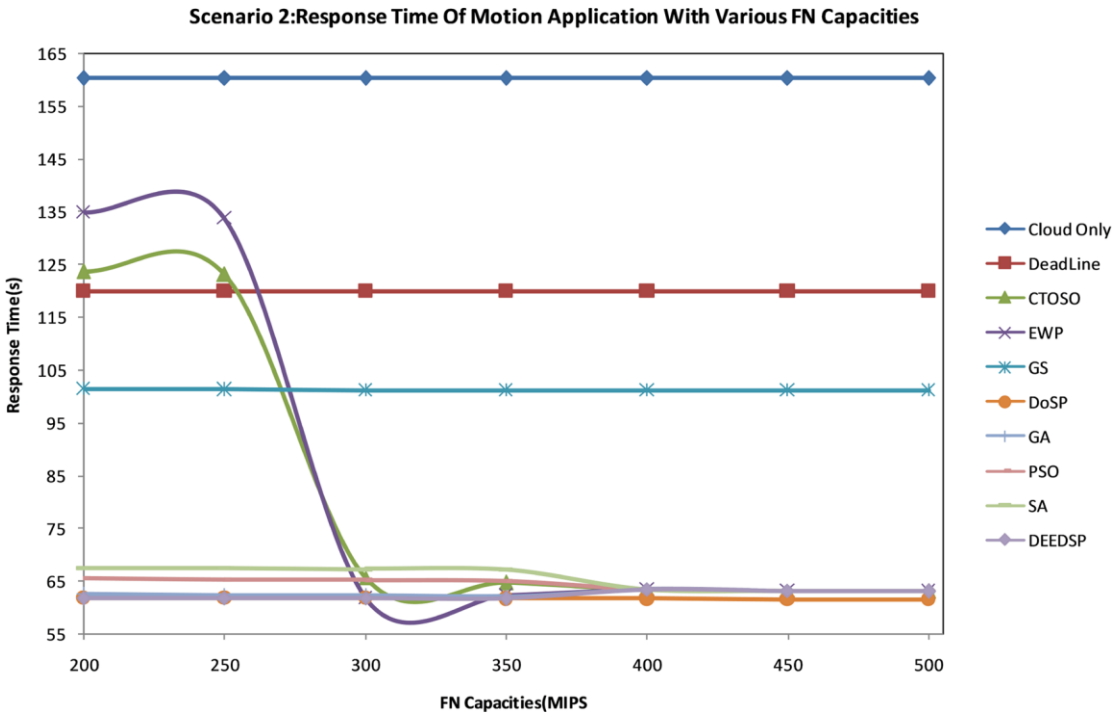


Fig 4.13: Response time of motion application with various FN capacities

4.6.3.2. Resource utilization

Figure 4.14 shows that the cloud utilization for Cloud Only, CTOSO, EWP, GS, DoSP, GA, PSO, SA, and DEEDSP policies are 100 percent, 24 percent, 0 percent, 36 percent, 32 percent, 4 percent, 8%, 8%, and 0%, respectively. Cloud Only, CTOSO, EWP, GS, DoSP, GA, PSO, SA, and DEEDSP policies have an FN utilization of 0%, 76 percent, 76 percent, 40 percent, 40 percent, 68 percent, 68 percent, and 68 percent, respectively. Finally, the NFCN utilization for cloud-only, CTOSO, EWP, GS, DoSP, GA, PSO, SA, and DEEDSP policies is 0%, 0%, 12%, 4%, 12%, 12%, 8%, 8%, and 16%, respectively. Only an application's sensing and actuation modules are allowed to run on fog nodes in both GS and DoSP rules.

Looking at all of the policies, we can see that the EWP and DEEDSP policies haven't put any application modules in the cloud node. Except in the Cloud Only policy, the GS policy has added more modules to the cloud node. GS, DoSP, and Cloud Only policies deployed more or nearly the same number of modules in FN nodes than CTOSO, EWP, GA, PSO, SA, and DEEDSP policies.

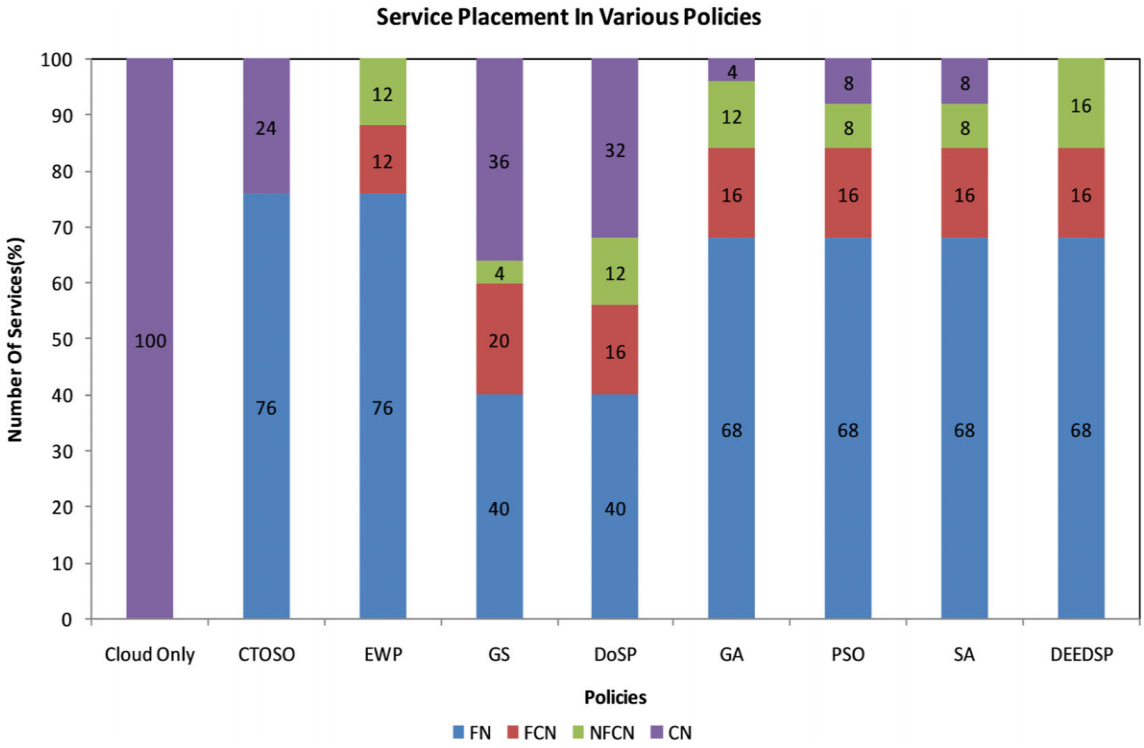


Fig 4.14: Service placement in various policies

Figure 4.15 depicts the share of each resource type as the cloud distance changes. The module placement in the cloud node steadily decreases as the cloud distance increases. To prevent applications from exceeding their deadlines, they are placed in a fog node, which increases fog node consumption. In a deadline-conscious setting, when the TAU value and cloud distances increase, the modules are more likely to be placed in FCN rather than NFCN. Because of the application deadline, the module placement in the NFCN steadily decreases when the TAU value rises, as shown in Figure 4.16.

Cloud node, FCN node, or FN node module location increases. When the cloud distance and TAU parameters are increased, the modules prefer to execute in the FN or FCN node. Applications submitted after the deadline should be submitted to either NFCN or CN.

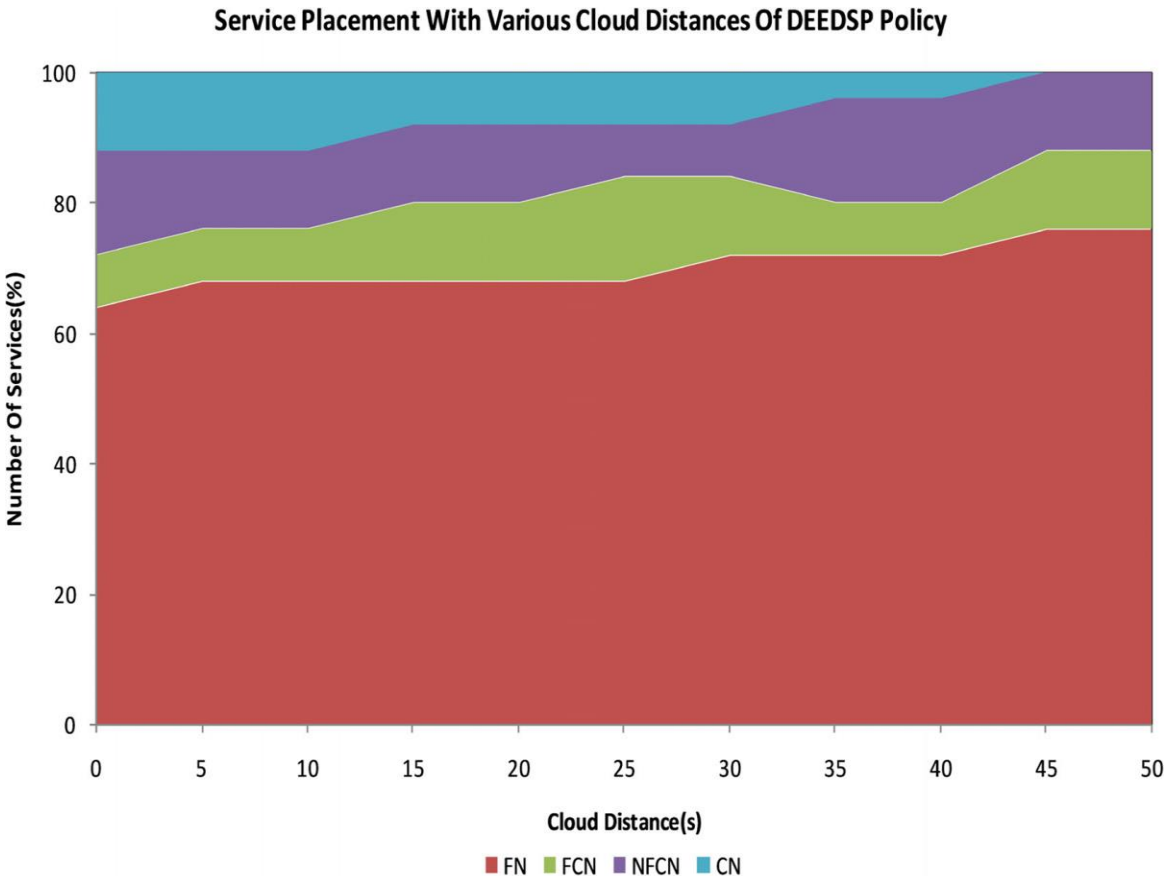


Fig 4.15: Service placement with various cloud distances of DEEDSP policy

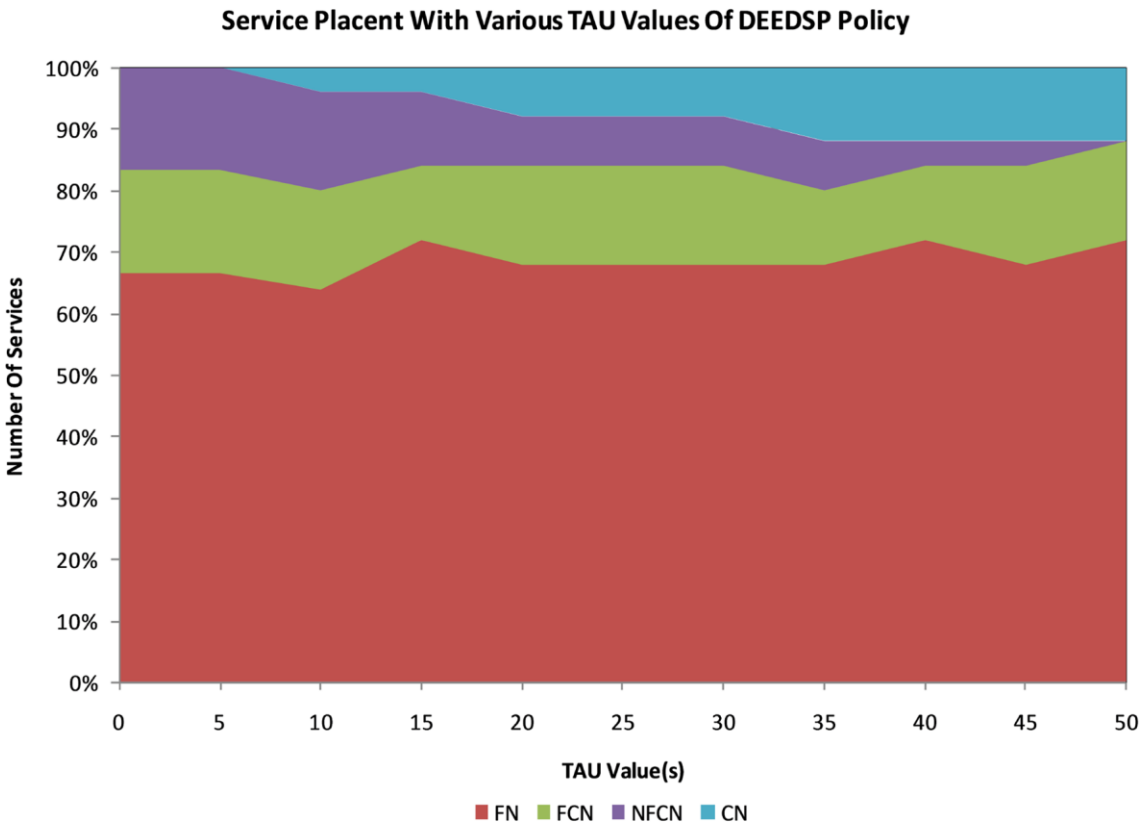


Fig 4.16: Service placement with various TAU values of DEEDSP policy

4.6.3.2.1. Scenario 1

Figure 4.17 depicts FN consumption as a function of FN MIPS levels. It is believed that the applications will have a minimal computing cost and a short deadline. The Cloud Only policy does not place the application modules in the FN node, as can be seen. In the FN node, GA, PSO, SA, CTOSO, and EWP policies place an equal amount of modules.

For all FN node MIPS, the GS and DoSP policies use 40% of the fog node resources. These two policies prevent the application's process modules from being placed. FN node use is lower in the GA, PSO, SA, and DEEDSP schemes till 350 MIPS, compared to CTOSO, EWP, GA, PSO, and SA policies. CTOSO, EWP, and DEEDSP policies position all the modules after 400 MIPS in FN node.

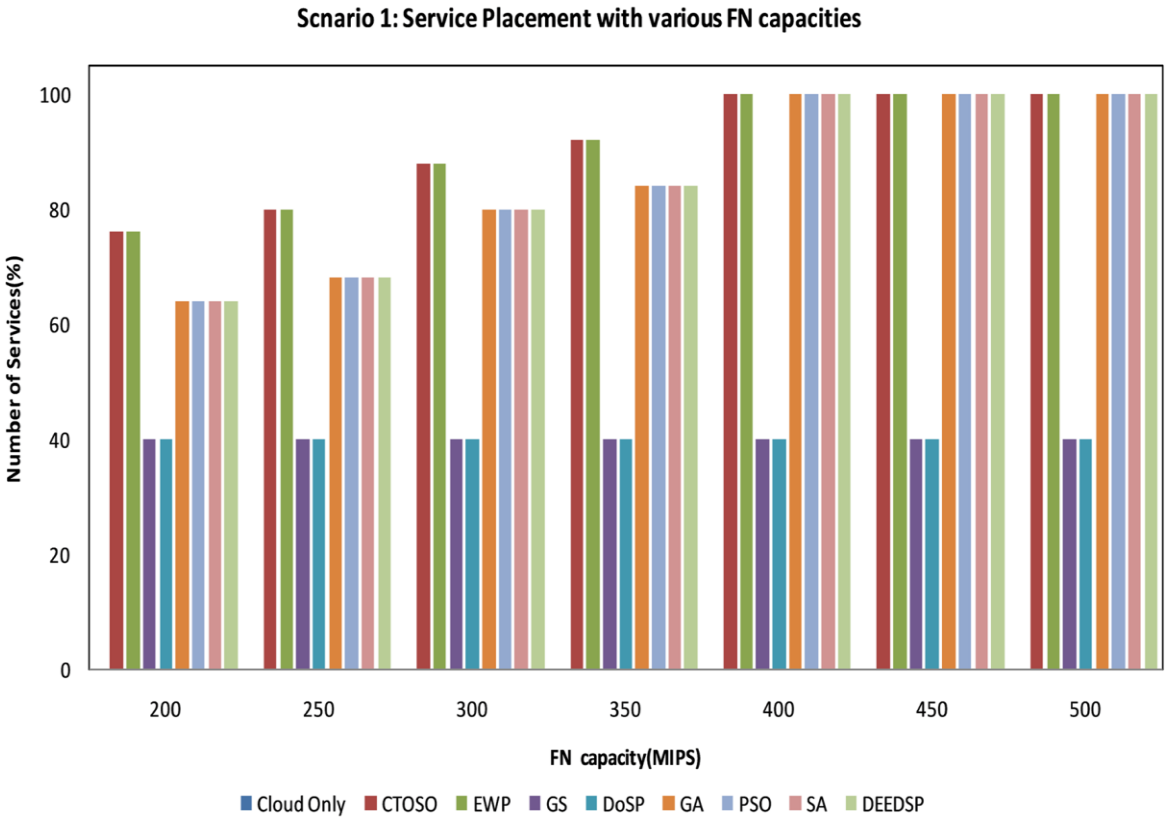


Fig 4.17: FN consumption

4.6.3.2.2. Scenario 2

Figure 18 depicts FN usage to various FN MIPS values. It is believed that the applications will have a high computing cost and a short deadline. In the case of the cloud-only policy, the application modules are not placed in the FN node. CTOSO and EWP policies place the same number of modules in the FN node as in the prior scenario. Only the sensing and actuation modules of the apps are allowed to run in FN under the GS and DoSP regulations, which use 40% of the fog node resources.

The application process modules are not placed in FN by these two policies. FN node utilization is lower in GA, PSO, SA, and DEEDSP policies than CTOSO, EWP, GA, PSO, SA, and DEEDSP policies. In the same policy, sensor and actuation modules occupy FN nodes, preventing the FN nodes from running other modules. CTOSO and EWP policies place all modules on the FN node after 400 MIPS.

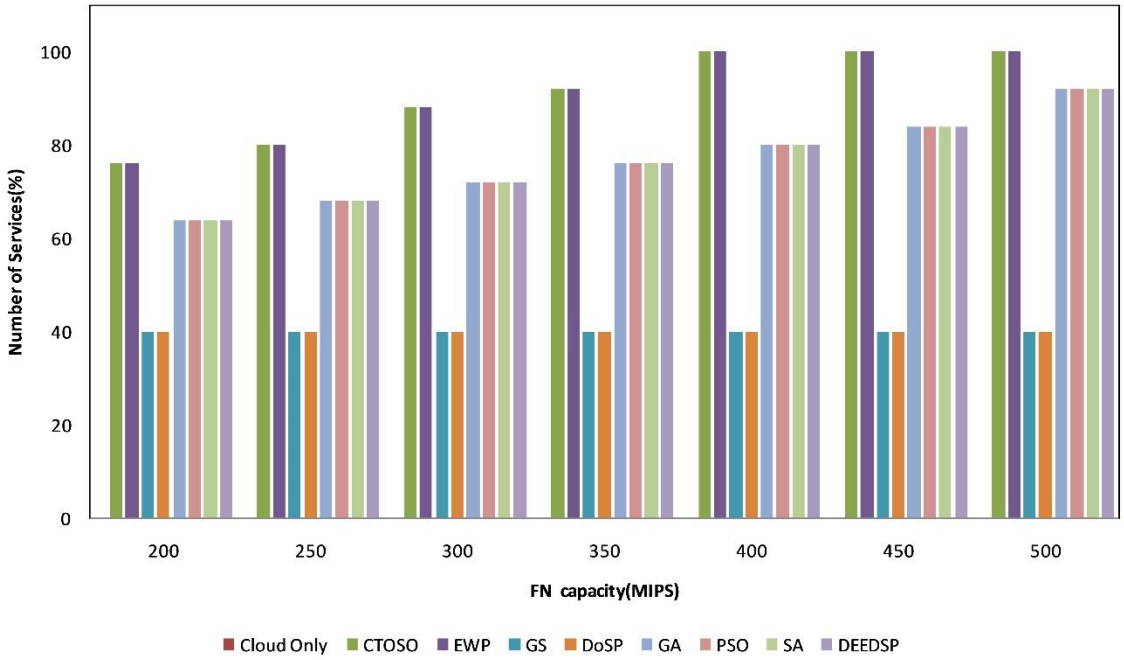


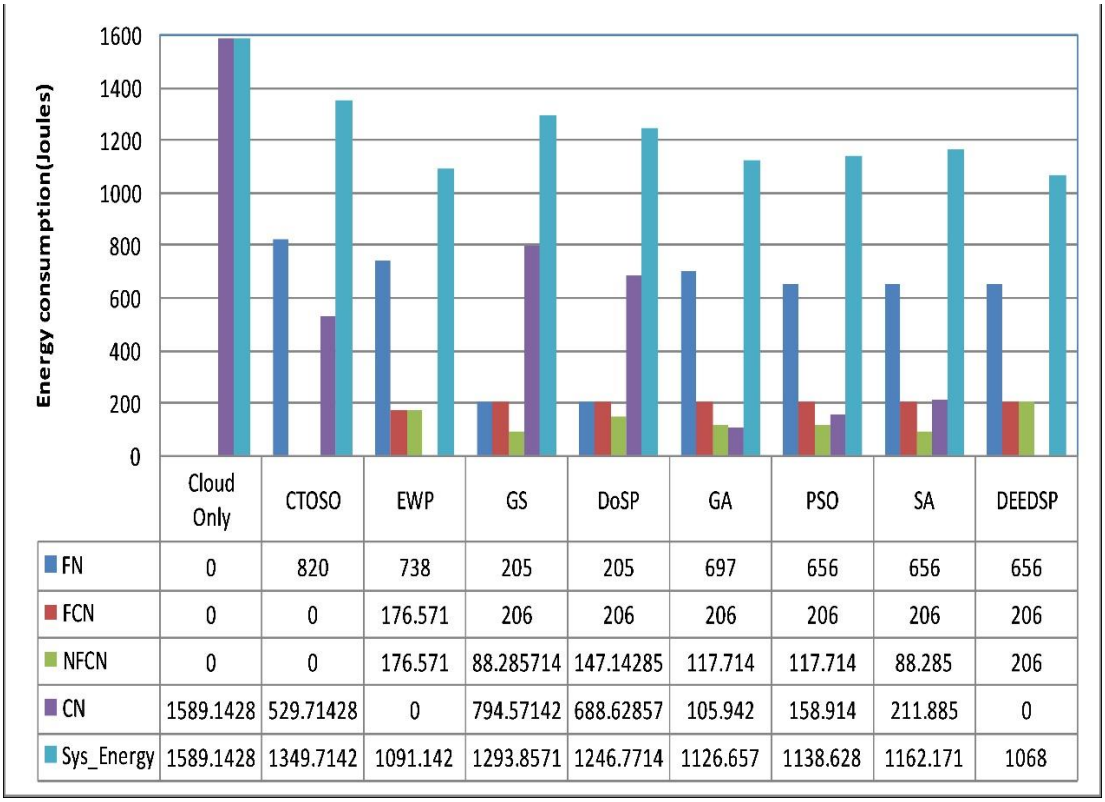
Fig 4.18: FN utilization with various capacities

4.6.3.3 Energy utilization

Figure 4.19 illustrates this. All other options consume more energy than the cloud-only policy. Because it stores all modules in the cloud, the cloud consumes more energy to run the modules. Except for EWP and DEEDSP, the policies GA, PSO, and SA use less energy than the others.

The EWP and DEEDSP policies are useless or have the same cloud (CN) energy as the other policies. The DEEDSP policy uses less energy than the other policies combined. More modules are placed in FN as a result of the CTOSO policy. Because it installs more modules in FN, CTOSO requires more FN energy. The fog node is given priority in the CTOSO policy, while the remaining modules are placed in the cloud (CN). The majority of EWP and DEEDSP policies do not place modules in the cloud. Because of the high FN energies, EWP requires more energy than DEEDSP.

Except for EWP, all policies that consider FCN for placement spend the same amount of FCN energy. DEEDSP has a higher NFCN energy than any other policies that take NFCN into account. In terms of using the NFCN node without surpassing the application deadline, this strategy outperforms different rules.

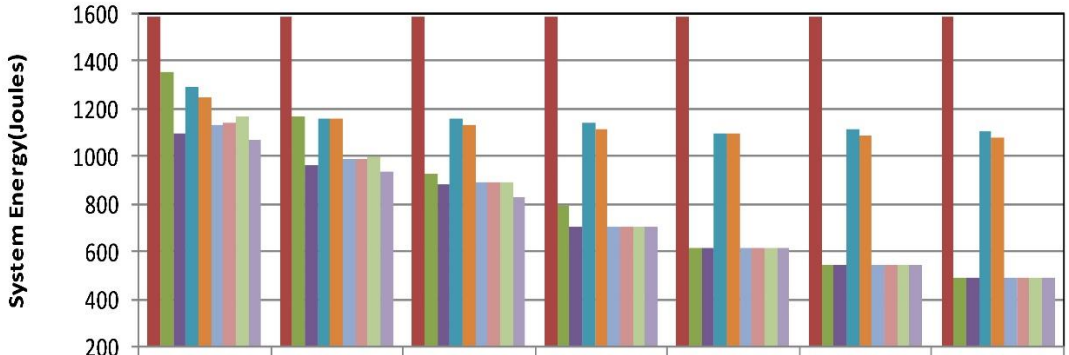


	Cloud Only	CTOSO	EWP	GS	DoSP	GA	PSO	SA	DEEDSP
FN	0	820	738	205	205	697	656	656	656
FCN	0	0	176.571	206	206	206	206	206	206
NFCN	0	0	176.571	88.285714	147.14285	117.714	117.714	88.285	206
CN	1589.1428	529.71428	0	794.57142	688.62857	105.942	158.914	211.885	0
Sys_Energy	1589.1428	1349.7142	1091.142	1293.8571	1246.7714	1126.657	1138.628	1162.171	1068

Fig 4.19: System energy consumption of various policies

4.6.3.3.1. Scenario 1

The energy usage for various fog node MIPS is shown in Figure 4.20. In the event of a cloud-only strategy, the cloud energy usage remains constant for all FN MIPS values. DEEDSP uses less energy than any other plans up to 350 FN MIPS. Because it deploys more modules in the cloud node, the CTOSO consumes more energy than the DEEDSP policy, up to 250 FN MIPS. Because FN has enough room to support all of the modules alone, the CTOSO, EWP, GA, PSO, SA, and DEEDSP policies require the same amount of energy from 400 FN MIPS.



	200	250	300	350	400	450	500
Cloud Only	1589.142	1589.142	1589.142	1589.142	1589.142	1589.142	1589.142
CTOSO	1349.714	1165.542	922.552	789.5556	615	546.66	492
EWP	1091.142	957.028	883.847	702.857	615	546.66	492
GS	1293.857	1158.685	1154.895	1135.371	1097.185	1109.33	1100.22
DoSP	1246.771	1158.685	1131.352	1111.828	1097.185	1085.79	1076.68
GA	1126.657	984.21	892.34	702.857	615	546.66	492
PSO	1138.628	992.173	892.34	702.857	615	546.66	492
SA	1162.171	997.326	892.34	702.857	615	546.66	492
DEEDSP	1068	935.142	824.19	702.857	615	546.66	492

Fig 4.20: Energy consumption with various FN capacities

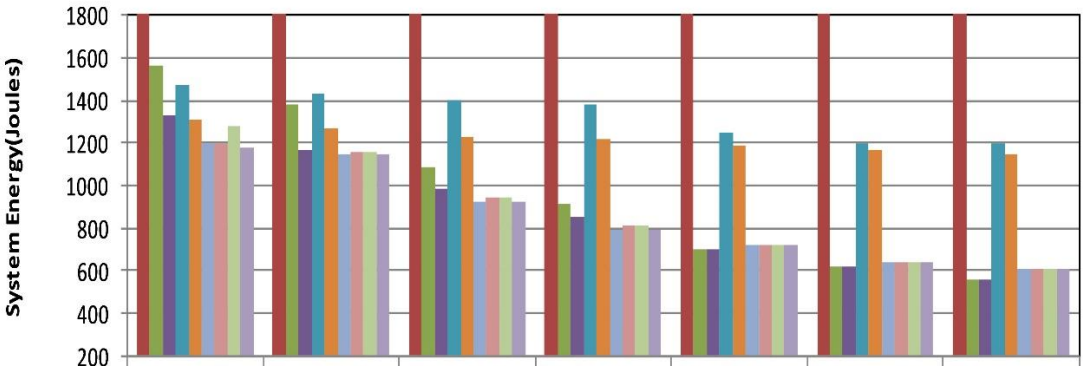
The rise in the FN MIPS value has no impact on the GS and DoSP policies. Only the sensor and actuation modules are placed in FN by these two regulations. Even as the FN MIPS value rises, it uses the FN resources equally. For various FN capabilities, the suggested DEEDSP spends less or equal energy than other policies.

4.6.3.3.2. Scenario 2

Because all modules are located in CN, the cloud-only policy, as illustrated in Figure 4.21, consumes a constant amount of energy for various FNMIPS. DEEDSP spends less or equal energy than all other policies until the fog node reaches 350 FN MIPS. DEEDSP policy has arranged for all high-performance computing modules to be housed in FN. CTOSO and EWP policies have been demonstrated to be the best in system energy consumption at 400 FN MIPS. The FN MIPS value does not affect the GS and DoSP policies; these two policies solely place sensing and actuation modules in FN.

Even as the FN MIPS value rises, it continues to use FN resources. Starting at 400 MIPS, GA, PSO, SA, and DEEDSP required the same amount of FN energy. The energy consumption of all policies with different cloud distances is shown in Figure 4.22. The change in cloud distance does not affect the energy consumption of the cloud alone, CTOSO, or EWP regulations. The energy consumption of the GS, DoSP, GA, PSO, SA, and DEEDSP policies decreases as the cloud distance increases.

When the distance between the cloud and the modules is extended, these three policies prevent the modules from being placed in the cloud. By reducing the number of modules in the cloud, the application will not miss its deadline. At high cloud distances, the proposed technique uses less energy than all existing policies. Except for EWP and DEEDSP policies, GA, PSO, and SA policies use less energy than other policies.



	200	250	300	350	400	450	500
Cloud Only	1801.028	1801.028	1801.028	1801.028	1801.028	1801.028	1801.028
CTOSO	1561.6	1377.428	1083.161	914.742	697	619.555	557.6
EWP	1326.971	1168.914	984.799	855.657	697	619.555	557.6
GS	1470.428	1429.428	1402.095	1382.571	1247.598	1197.015	1197.015
DoSP	1311.847	1266.292	1229.768	1217.515	1188.768	1166.933	1141.911
GA	1193.74	1151.87	927.276	789.555	718.5	642.888	609.511
PSO	1202.35	1157.23	942.64	812.42	718.5	642.888	609.511
SA	1273.54	1157.23	942.64	812.42	718.5	642.888	609.511
DEEDSP	1182.19	1147.028	927.276	789.555	718.5	642.888	609.511

Fig 4.21: Overall energy consumption with various FN capacities

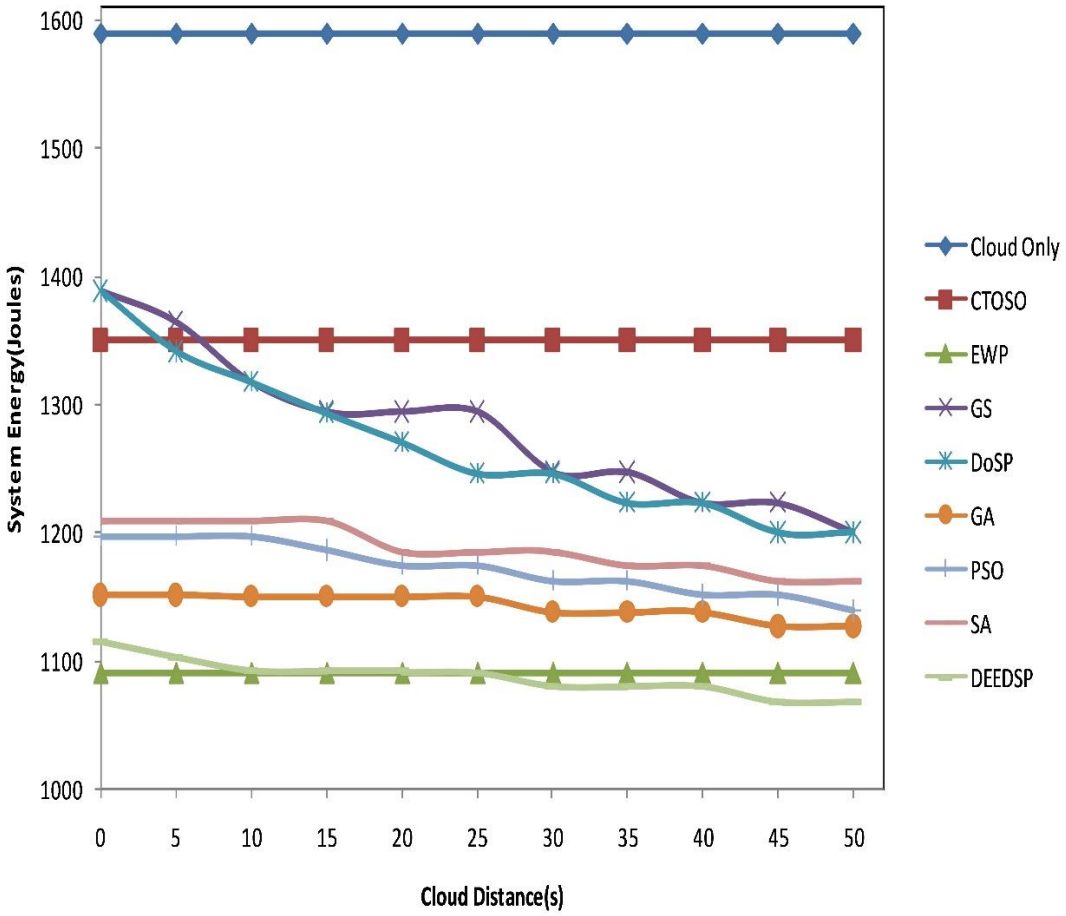


Fig 4.22: Overall energy consumption with various cloud distances

Figure 4.23 depicts the energy usage of all policies based on different TAU values. GA and DEEDSP consume less energy than EWP policy between TAU values of 5 and 20. TAU levels do not affect the energy usage of cloud-only, CTOSO, and EWP policies. All modules are placed in cloud nodes in the cloud-only policy. Thus, they are not affected by the TAU value, keeping the energy consumption constant. The scenario under the CTOSO policy is the same as in the cloud alone. Even after increasing the TAU value, the EWP policy arranges the modules in a hierarchical sequence in the neighbour controller node. As a result, greater TAU values do not affect energy usage.

The system's total energy consumption is increased in the GS, DoSP, GA, PSO, SA, and DEEDSP policies as the TAU values increase. The GS and DoSP policies strive to arrange the modules in FCN and CN when the TAU values rise. More modules in the controller node and cloud node lead to a rise in energy consumption. When the TAU value rises, the DEEDSP policy consumes less energy than the GS, DoSP, GA, PSO, and SA policies. To avoid surpassing the deadline, the GS, DoSP, GA, PSO, SA, and DEEDSP policies minimize module insertion in neighbour fog controller nodes when the TAU value is increased.

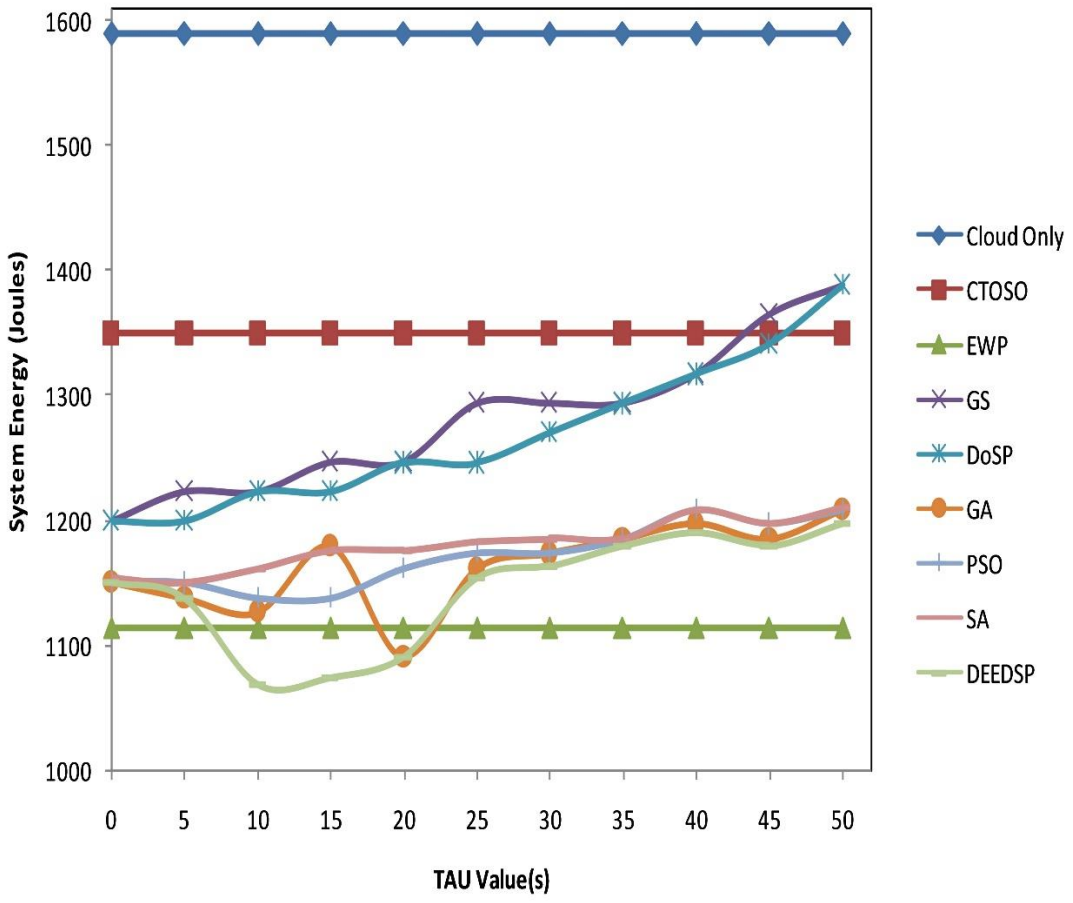


Fig 4.**23:** Overall energy consumption with various TAU values

4.6.3.4. Trade-off between energy and execution time

When considering the response time and energy utilisation parameters which are dependent on cloud distance D, Figure 4.24 illustrates how well the DEEDSP policy impacts one another. Due to the increasing amount of services assigned to CN, energy consumption is highest when D = 0, although makespan time is lowered to 20.64 seconds.

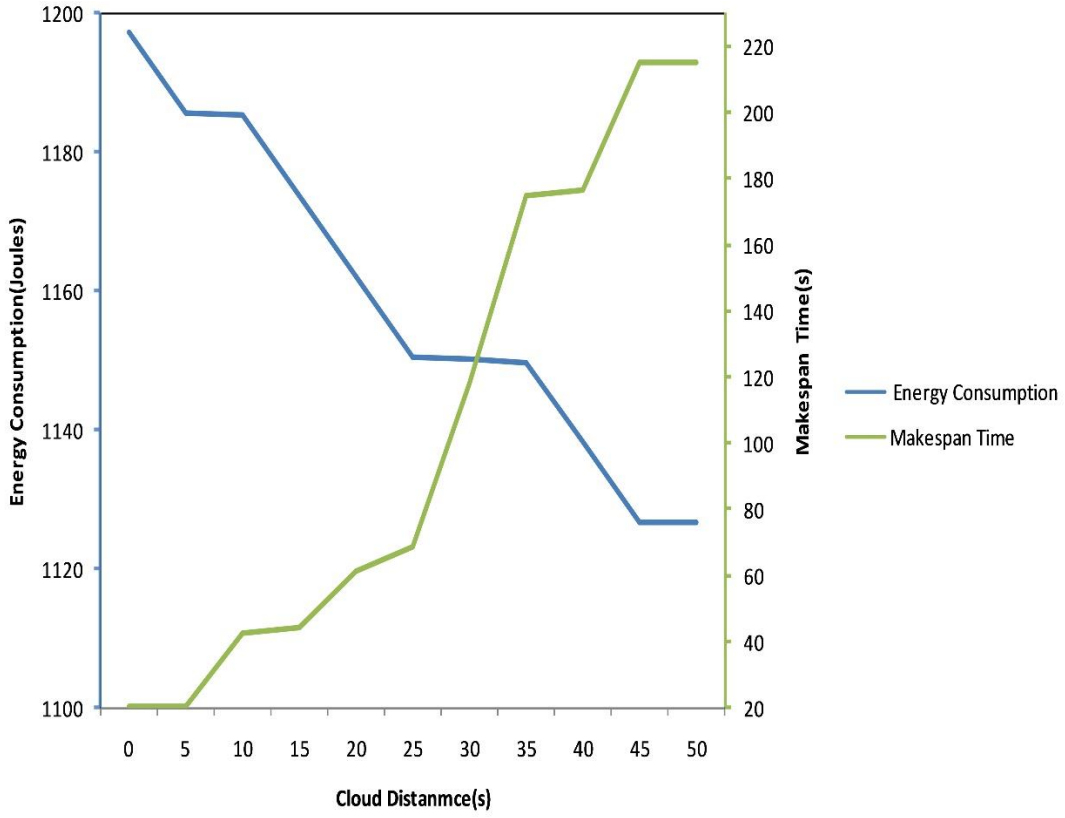


Fig 4.24: Trade-off between energy consumption and makespan time of DEEDSP policy

At a cloud distance of D = 15, consumption of energy as well as makespan time improved by 23.543 joules and 23.74 seconds, respectively. When D approached 50, overall energy consumption decreases but the makespan duration increases. The DEEDSP approach minimizes energy usage when increases the service placement in FN, leading in kind of a prolonged makespan time. The experiment demonstrated that our planned approach might be adaptable by modifying the cloud distance D.

It meets the needs of users looking for high-performance execution or low energy consumption. At a cloud distance of 30 kilometres, the system achieves balance in terms of energy efficiency and makespan time. It is the best spot for balancing energy consumption and makespan time.

4.7. Summary

When used in sequential IoT workflow applications, overall efficiency of the multi-tier fog infrastructure is investigated. There are several fog nodes containing distinct system resources inside this framework. Regarding better service location in a fog environment, the deadline-aware and energy-efficient service allocation method is explained. For validate this service placement method, extensive benchmark tests in a fog environment were conducted to evaluate application response time and system energy usage. When trying to compare this with other state-of-the-art algorithms, the proposed deadline-aware and energy-efficient service placement method is assessed. As per the simulated results, the suggested approach functioned effectively in a multi-layer fog environment.

Chapter 5

Conclusion & Future Scope

5.1. Overview

The findings and outcomes of the proposed model are presented in this chapter. We have presented the multi-tiered fog computing paradigm application service in the previous chapters wherein we have proposed the 2 Fog placement models named as Deadline-Aware Dynamic Service Placement (DoSP) & (DEEDSP). According to the Literature Survey, no work has been done to provide real-time analytics that dynamically identifies analytical activities pushed to the cloud or edge with the least latency and throughput.

Most of the network security-related research in fog computing has concentrated on multi-node authentication and authorization mechanisms. Still, little work has been done to provide fault tolerance in the fog environment for individual nodes, networks, service platforms, and applications. One of the challenging problems is the minimization of the energy consumption by the nodes in fog environment. There is a need to work to be done on programming models and architectures with the consideration of dynamic configuration. Further, due to multiple cloud tenant & users, distributed resources very minimal work has been done on security aspects in the fog computing environment.

With the Internet of Things (IoT) looming huge around the globe, it's crucial to remember that such applications can encompass thousands of connected devices. In other words, an IoT application may be thought of as a collection of services. The IoT application is coupled with sensing devices that create data 24 hours a day, seven days a week. IoT applications require cloud assistance because their sensor devices generate data continuously, and this data has big data characteristics such as volume (a large amount of data), velocity (data is streamed), and variety (data types such as structured, unstructured and semi-structured).

In this context, it is acknowledged that deploying an IoT application in the public cloud causes the application's response time and throughput to suffer. It is the primary issue that has been identified. Hence, fog computing is a good fit to execute workflow-based IoT applications. Furthermore, the issue is the positioning of services that allow for such an application. In terms of storage, processing, and other services, the services do not have uniform needs.

As a result, one size does not fit all, and different services require different treatment. When a particular service is deployed in FN, FCN, or CN, it is unavoidable to imagine or foresee prospective improvements in the broader application. It is a difficult problem that we must handle. As a result, the goal of this study is to develop fault-tolerant QoS aware service placement.

The thesis focuses on developing a framework for placing deadline-sensitive applications with minimum energy consumption and shows the significance of such a system. However, the optimization task of allocating applications to computation nodes becomes a complex problem to solve that requires optimization methods. We leverage the use of Meta heuristics, a particularly hyper-heuristic algorithm capable of taking advantage of optimization of IoT application service placement with deadline-aware and minimizing energy consumption in Fog Computing.

The simulation of the suggested model will be executed using iFogSim. It works with stimulating the surrounding consisting of many IoT devices, and corresponding fog based we will show node Simulation results for justification.

Further, the testing of simulation results will be done for different output parameters. It will verify that the proposed policy is an improvement to previous works. The following is a list of the thesis' key contributions.

- ✓ A survey and taxonomy of the state-of-the-art in-service placement in a fog environment have been designed to meet the specific needs of applications that objectively concentrate on latencies like augmented reality and IoT applications which tend to generate huge amounts of data un-workable for analyzed remote cloud data centers.

- ✓ To design a practical framework for application placement in a fog environment. The optimal solution obtains the load in another fog node by appropriately positioning the device, raising the heat that a specific fog node produces.

- ✓ To design an efficient service placement algorithm concerning deadline awareness and energy efficiency. We have used the heuristic (Yates et al., 2019) algorithms to improve the scheduling problem's performance.

To compare the Extensive simulation-based performance evaluation of the proposed work with the existing algorithms across various multiple parameters along-with the analysis and discussion of the deserved results.

5.2. Conclusion

The proliferation of IoT devices has resulted in a tremendous volume of data with hardware heterogeneity, geographical dispersion, and limited bandwidth. Existing cloud models are ill-suited to handle the latency-sensitive applications in such bandwidth constrained networks. The primary objective of this research study is to propose a fog computing model that can mitigate the above-mentioned challenges by distributing local processing near the point of data production.

The proposed fog computing model comprises three layers - device, fog, and cloud. The fog layer consists of nodes with exploitable computational resources, which can be leveraged to accelerate IoT services. The research focuses on two primary parameters - response time and energy minimization.

To achieve the aforementioned objectives, two fog placement models named Deadline-Aware Dynamic service placement (DoSP) and Dynamic Energy-Efficient Deadline-Sensitive Placement (DEEDSP) are proposed. The DoSP algorithm leverages the Genetic algorithm for deadline-aware application placement in the fog environment. The algorithm is dynamic and adapts to changes in the network environment to optimize service placement. The DEEDSP algorithm improves service placement in terms of deadline-awareness and energy efficiency. The algorithm is based on a hyper-heuristic algorithm that balances the energy-delay tradeoffs to

dynamically place application modules while minimizing the system's energy consumption.

Simulation results show that both the DoSP and DEEDSP algorithms outperform other approaches such as Edge Ward and Cloud Only. The proposed algorithms have the potential to accelerate IoT services in fog computing by balancing the energy-delay trade-offs and providing a dynamic and adaptable service placement strategy. Further research can be conducted to test the proposed algorithms on larger datasets and real-world scenarios.

5.3. Future Scope

The future scope of this research includes the following areas of development to improve the existing approach:

Real-time information and energy efficiency: The proposed method can be extended to optimize service location based on real-time data and energy efficiency. Advanced optimization techniques such as machine learning can be employed to achieve better results.

Container virtualization and device mobility: In a fog computing environment, container virtualization and device mobility can be considered to improve the efficiency and flexibility of the system.

Security: The DEEDSP approach can be further enhanced by integrating blockchain or quantum computing concepts to improve the security and processing capability of the system.

IoT applications: DEEDSP can be applied in various IoT applications, such as Industry 4.0, healthcare, and agriculture, to optimize service placement and improve system performance.

Serverless edge computing: The concept of serverless edge computing can be incorporated to scale applications effectively and optimize service placement.

In conclusion, the proposed DEEDSP approach can be further enhanced and expanded to address the latest trends and challenges in the fog computing domain. The future research would require integrating advanced optimization techniques, security mechanisms, and new technologies to improve the performance and efficiency of the system.

Bibliography

A. Vulpe, G. Suciu, S. Halunga, and O. Fratu, “Innovative Platform for Resource Allocation in 5G M2M Systems”, in Proceedings of International Conference on Future Access Enablers of Ubiquitous and Intelligent Infrastructures (FABULOUS ’17), Bucharest, Romania, October 2017, pp. 16–24.

A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, “Above the Clouds: A Berkeley View of Cloud Computing”, in Technical Report No. UCB/EECS-2009-28-13, February 2009, pp. 1–23.

Aazam, M., Zeadally, S., & Harras, K. A. (2018). Fog Computing Architecture, Evaluation, and Future Research Directions. IEEE Communications Magazine, 56(5), pp. 46–52.

Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2018). Mobile Edge Computing: A Survey. IEEE Internet of Things Journal, 5(1), pp. 450–465.

Abdulkareem, K. H., Mohammed, M. A., Gunasekaran, S. S., Al-Mhiqani, M. N., Mutlag, A. A., Mostafa, S. A., Ibrahim, D. A. (2019). A Review of Fog Computing and Machine Learning: Concepts, Applications, Challenges, and Open Issues. IEEE Access, 7, 153123- 153140.

A. Alnoman and A. Anpalagan. 2018. A Dynamic Priority Service Provision Scheme for Delay-Sensitive Applications in Fog Computing. In 2018 29th Biennial Symposium on Communications (BSC). IEEE, 1–5.

A. Sheshasaayee and R. Megala, "A study on resource provisioning approaches in autonomic cloud computing," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and

Cloud) (I-SMAC), 2017, pp. 141-144, doi: 10.1109/I-SMAC.2017.8058325.

Adam A. Alli and Muhammad Mahbub Alam. 2019. SecOFF-FCIoT: Machine learning based secure offloading in Fog-Cloud of things for smart city applications. Internet of Things 7 (2019), 100070.

Adhikari, M., Mukherjee, M., & Srirama, S. N. (2019). DPTO: A Deadline and Priority-aware Task Offloading in Fog Computing Framework Leveraging Multi-Level Feedback Queueing. IEEE Internet of Things Journal, 1-9.

Agarwal, S., Yadav, S., & Yadav, A. K. (2016). An efficient architecture and algorithm for resource provisioning in fog computing. International Journal of Information Engineering and Electronic Business, 8(1), 48.

Amira Rayane Benamer, Hana Teyeb, and Nejib Ben Hadj-Alouane. 2018. Latency- Aware Placement Heuristic in Fog Computing Environment. In On the Move to Meaningful Internet Systems. OTM 2018 Conferences, Hervé Panetto, Christophe Debruyne, Henderik A. Proper, Claudio Agostino Ardagna, Dumitru Roman, and Robert Meersman (Eds.). Springer International Publishing, Cham, 241–257.

Alexander, K., Lee, C., & Chai, S. (2017). Declarative policy support for cloud application orchestration. 2017 19th International Conference on Advanced Communication Technology (ICACT).

Alexander, K., Lee, C., Kim, E., & Helal, S. (2017). Enabling End-to-End Orchestration of Multi-Cloud Applications. IEEE Access, 5, 18862–18875.

A.A.Alsaffar, H.P.Pham, C.-S.Hong, E.-N.Huh, and M.Aazam, “An architecture of IoT service delegation and resource allocation based

on collaboration between fog and cloud computing," Mobile Information Systems, vol. 2016.

Aljumah, A., & Ahanger, T. A. (2018). Fog computing and security issues: A review. 2018 7th International Conference on Computers Communications and Control (ICCCC).

Apat, H. K., Sahoo, B., & Maiti, P. (2018). Service Placement in Fog Computing Environment. 2018 International Conference on Information Technology (ICIT).

Ardagna, D. (2015). Cloud and Multi-cloud Computing: Current Challenges and Future Applications. 2015 IEEE/ACM 7th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems.

Attiya, I., Abd Elaziz, M., & Xiong, S. (2020). Job Scheduling in Cloud Computing Using a Modified Harris Hawks Optimization and Simulated Annealing Algorithm. Computational Intelligence and Neuroscience, 2020, 1–17.

Barik, R., Dubey, H., Sasane, S., Misra, C., Constant, N., & Mankodiya, K. (2017). Fog2Fog: Augmenting Scalability in Fog Computing for Health GIS Systems. 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE).

Bousselmi, K., Brahmi, Z., & Gammoudi, M. M. (2014). Cloud Services Orchestration: A Comparative Study of Existing Approaches. 2014 28th International Conference on Advanced Information Networking and Applications Workshops, 410-416.

Brabra, H., Mtibaa, A., Gaaloul, W., Benatallah, B., & Gargouri, F. (2019). Model- Driven Orchestration for Cloud Resources. 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), 422-429.

Brogi A., Forti S., Guerrero C., & Lera I. (2019). How to Place Your Apps in the Fog
State of the Art and Open Challenges. Software Practice and Experience, 1-22.

Butt, A. A., Khan, S., Ashfaq, T., Javaid, S., Sattar, N. A., & Javaid, N. (2019). A Cloud and Fog based Architecture for Energy Management of Smart City by using Meta-heuristic Techniques. 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC).

C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi and R. H. Glitho, "Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems with Mobile Fog Nodes," in IEEE Journal on SelectedAreas in Communications, vol. 37, no. 5, pp. 1130-1143, May 2019, doi: 10.1109/JSAC.2019.2906790.

Calheiros, R. N., Ranjan, R., Beloglazov A., DeRose, C. A. F., & Buyy, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource. Software Practice and Experience, 41, 23-50.

Cardellini, V., Mencagli, G., Talia, D., & Torquati, M. (2019). New Landscapes of the Data Stream Processing in the era of Fog Computing. Future Generation Computer Systems, 99, 646-650.

Carnevale, L., Celesti, A., Galletta, A., Dustdar, S., & Villari, M. (2018). From the Cloud to Edge and IoT: a Smart Orchestration Architecture for Enabling Osmotic Computing. 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), 419-424.

Chang, Z., Zhou, Z., Ristaniemi, T., & Niu, Z. (2017). Energy Efficient Optimization for Computation Offloading in Fog Computing System. GLOBECOM 2017 - 2017 IEEE Global Communications

Conference.

Chen, W., & Deelman, E. (2012). WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. 2012 IEEE 8th International Conference on E-Science.

Chen, S., Xu, H., Liu, D., Hu, B., & Wang, H. (2014). A Vision of IoT: Applications, Challenges, and Opportunities with China Perspective. IEEE Internet of Things Journal, 1(4), 349–359.

Chiang, M., & Zhang, T. (2016). Fog and IoT: An Overview of Research Opportunities. IEEE Internet of Things Journal, 3(6), 854–864.

Coutinho, A., Greve, F., Prazeres, C., & Cardoso, J. (2018). Fogbed: A Rapid-Prototyping Emulation Environment for Fog Computing. 2018 IEEE International Conference on Communications (ICC).

D. G. Roy, D. De, A. Mukherjee, and R. Buyya, "Applicationaware cloudlet selection for computation offloading in multi-cloudlet environment," The Journal of Supercomputing, pp. 1–19,2016.

De Donno, M., Tange, K., & Dragoni, N. (2019). Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. *IEEE Access*, 7, 150936- 150948.

Dlamini, S., Mwangama, J., Ventura, N., & Magedanz, T. (2018). Design of an Autonomous Management and Orchestration for Fog Computing. *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC).*

E. E. Haber, T. M. Nguyen, D. Ebrahimi and C. Assi, "Computational Cost and Energy Efficient Task Offloading in Hierarchical Edge-Clouds," 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2018, pp. 1-6, doi: 10.1109/PIMRC.2018.8580724.

Ezhilarasie R, Reddy MS, Umamakeswari A. A new hybrid adaptive GA-PSO computation offloading algorithm for IoT and CPS context application. Journal of Intelligent & Fuzzy Systems. 2019 Jan 1;36(5):4105-13.

Gao, L., Luan, T. H., Yu, S., Zhou, W., & Liu, B. (2017). FogRoute: DTN-based Data Dissemination Model in Fog Computing. *IEEE Internet of Things Journal*, 4(1), 225- 235.

Ghobaei-Arani, M., Souri, A., & Rahmanian, A. A. (2019). Resource Management Approaches in Fog Computing: a Comprehensive Review. Journal of Grid Computing, 18, 1-42.

Gia T. N., Jiang M., Rahmani A. M., Westerlund T., Mankodiya K., Liljeberg P., & Tenhunen H. (2015). Fog Computing in Body Sensor Networks: An Energy Efficient Approach. IEEE International Body Sensor Networks Conference (BSN'15).

Gupta, H., Dastjerdi, A. V., Ghosh, S. K., & Buyya, R. (2016). iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments, Software Practice and Experience, 47, 1275-1296.

Guo, K., Sheng, M., Quek, T. Q. S., & Qiu, Z. (2019). Task Offloading and Scheduling in Fog RAN: A Parallel Communication and Computation Perspective. IEEE Wireless Communications Letters, 9(2).

G. Lee, W. Saad, and M. Bennis. 2019. An Online Optimization Framework for Distributed Fog Network Formation with Minimal Latency. IEEE Transactions on Wireless Communications 18, 4 (April 2019),2244–2258.

Hassan, S. F., & Fareed, R. (2018). Video streaming processing using fog computing. 2018 International Conference on Advanced Science

and Engineering (ICOASE), 140-144.

Hong, H.-J. (2017). From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices. 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).

Haouari, F., Faraj, R., & AlJa'am, J. M. (2018). Fog Computing Potentials, Applications, and Challenges. 2018 International Conference on Computer and Applications (ICCA).

H. Y. Wu, C.R.Lee, "Energy Efficient Scheduling for Heterogeneous Fog Computing Architectures," 42nd IEEE International Conference on Computer Software & Applications,2018.

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

Holland JH. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Cambridge, MA: University of Michigan Press; 1975.

H. R. Arkian, A. Diyanat, and A. Pourkhalili. 2017. MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. Journal of Network and Computer Applications 82 (2017), 152 – 165.

Huang M, Liu W, Wang T, Liu A, Zhang S. A cloud–MEC collaborative task offloading scheme with service orchestration. IEEE Internet Things J. 2020;7(7):5792-5805. https://doi.org/10.1109/JIOT.2019.2952767

H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo, "A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks", IEEE Communications Surveys & Tutorials, vol. 19,

no. 2, pp. 828–854, January 2017.

I. Khoufi, P. Minet, A. Laouiti, and S. Mahfoudh, “Survey of Deployment Algorithms in Wireless Sensor Networks: Coverage and Connectivity Issues and Challenges”, International Journal of Autonomous and Adaptive Communications Systems, vol. 10, no. 4, pp. 341–390, December 2017.

J. C. Gomez and H. Terashima-Mar´ın, “Evolutionary hyper-heuristics for tackling bi- objective 2d bin packing problems,” Genetic Programming and Evolvable Machines, pp. 1–31,2017.

J. Yue, M. Xiao and Z. Pang, "Distributed Fog Computing Based on Batched Sparse Codes for Industrial Control," in IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4683-4691, Oct. 2018, doi: 10.1109/TII.2018.2857203.

Jayasena, K. P. N., & Thisarasinghe, B. S. (2019). Optimized task scheduling on fog computing environment using Meta heuristic algorithms. 2019 IEEE International Conference on Smart Cloud (SmartCloud), 53-58.

J. Bradley, J. Barbier, and D. Handler, “Embracing the Internet of Everything to Capture Your Share of 14.4 Trillion Dollars”, Cisco White Paper, February 2013.

J. Paek, O. Gnawali, M. A. Vieira, and S. Hao, “Embedded IoT Systems: Network, Platform, and Software”, Mobile Information Systems, vol. 2017, pp. 1–2, 2017.

J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang. 2018. Multitier Fog Computing With Large-Scale IoT Data Analytics for Smart Cities. IEEE Internet of Things Journal 5, 2 (April 2018), 677–686.

J. Li, J. Jin, D. Yuan, and H. Zhang, “Virtual Fog: A Virtualization Enabled Fog Computing Framework for Internet of Things”, IEEE Internet of Things Journal, vol. 5, no. 1, pp. 121–131, February 2018.

Jin, Q., Lin, R., Zou, H., & Yang, F. (2018). A Distributed Fog Computing Architecture Supporting Multiple Migrating Mode. 2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom).

Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud computing. Egyptian Informatics Journal, 16(3), 275–295.

Kamal, M. B., Javaid, N., Naqvi, S. A. A., Butt, H., Saif, T., & Kamal, M. D. (2018). Heuristic Min-conflicts Optimizing Technique for Load Balancing on Fog Computing. Lecture Notes on Data Engineering and Communications Technologies, 207–219.

Kanna, G. P., & Vasudevan, V. (2017). A New Approach in Multi Cloud Environment to Improve Data Security. 2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS).

Kaur, T., & Chana, I. (2016). Energy aware scheduling of deadline-constrained tasks in cloud computing. Cluster Computing, 19(2), 679–698.

Kayal, P., & Liebeherr, J. (2019). Autonomic Service Placement in Fog Computing. 2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM).

Kitanov, S., & Janevski, T. (2017). Energy efficiency of Fog Computing and Networking services in 5G networks. IEEE EUROCON 2017 -17th International Conference on Smart Technologies.

Kennedy J, Eberhart RC. Particle swarm optimization. Paper presented at: Proceedings of the IEEE International Conference on Neural Networks, Perth,WA, Australia; 1995:1942-1948;IEEE.

Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing.

Science.1983;220(4598):671-680.

L. Lin, P. Li, X. Liao, H. Jin, and Y. Zhang, “Echo: An edge-centric code offloading system with quality of service guarantee,” IEEE Access, vol. 7, pp. 5905– 5917, 2018.

L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, “Multi-server multi-user multi- task computation offloading for mobile edge computing networks,” Sensors, vol. 19, no. 6, p. 1446, 2019.

La, Q. D., Ngo, M. V., Dinh, T. Q., Quek, T. Q. S., & Shin, H. (2018). Enabling intelligence in fog computing to achieve energy and latency reduction. Digital Communications and Networks, 5(1), 3-9.

Lawanyashri, M., Balusamy, B., & Subha, S. (2017). Energy-aware hybrid fruitfly optimization for load balancing in cloud environments for EHR applications. Informatics in Medicine Unlocked, 8, 42–50.

Lin, F., Zhou, Y., Pau, G., & Collotta, M. (2018). Optimization-Oriented Resource Allocation Management for Vehicular Fog Computing. IEEE Access, 1-11.

Li J., Li X., Yuan J., Zhang R., & Fang B. (2018). Fog Computing-Assisted Trustworthy Forwarding Scheme in Mobile Internet of Things. *IEEE Internet of Things Journal*, 6(2), 2778-2796.

Li, J., 2019. Fog computing framework for the Internet of Things (Doctoral dissertation, Swinburne University of Technology).

L. T. Yang, B. Di Martino, and Q. Zhang, “Internet of Everything”, Mobile Information Systems, vol. 2017, no. 8035421, pp. 1–3, July 2017.

Liu, L., Qi, D., Zhou, N., & Wu, Y. (2018). A Task Scheduling Algorithm Based on Classification Mining in Fog Computing Environment. Wireless Communications and Mobile Computing, 2018, 1–11.

Lopes, M. M., Higashino W. A., Capretz, M. A. M., & Bittencourt, L. F.

(2017). MyiFogSim: A Simulator for Virtual Machine Migration in Fog Computing. UCC Companion,47-52.

Luthra, M., Koldehofe, B., & Steinmetz, R. (2019). Transitions for Increased Flexibility in Fog Computing: A Case Study on Complex Event Processing. Informatik Spektrum.

L. Tan and N. Wang, “Future Internet: The Internet of Things”, in Proceedings of Third International Conference on Advanced Computer Theory and Engineering (ICACTE ’10), Chengdu China, August 2010, pp. 1–376.

L. Dignan, Top Cloud Providers 2018: How AWS, Microsoft, Google Cloud Platform, IBM Cloud, Oracle, Alibaba Stack up, accessed on 13 September 2018. [Online]. Available: https://www.zdnet.com/article/cloud - providers - ranking - 2018 - how - aws - microsoft - google -cloud-platform-ibm-cloud- oracle-alibaba-stack/.

Mach, P., & Becvar, Z. (2017). Mobile Edge Computing: A Survey on Architecture and Computation Offloading. IEEE Communications Surveys & Tutorials, 19(3), 1628–1656.

Mainak Adhikari and Hemant Gianey. 2019. Energy efficient offloading strategy in fog-cloud environment for IoT applications. Internet of Things 6 (2019), 100053.

Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials*, 19(4), 2322–2358.

L. Vogler, J. M. Schleicher, C. Inzinger, and S. Dustdar, “A Scalable ¨ Framework for Provisioning Large-scale IoT Deployments,” ACM Transactions on Internet Technology, vol. 16, no. 2,2016.

M.Abbasi, E. Mohammadi Pasand, & M.R. Khosravi, "Workload Allocation

in IoT- Fog-Cloud Architecture Using a Multi-Objective Genetic Algorithm," J Grid Computing 18, 43–56 (2020). https://doi.org/10.1007/s10723-020-09507-1.

Madakam, S., & Bhagat, P. (2018). Fog Computing in the IoT Environment: Principles, Features, and Models. Fog Computing, 23–43.

M.S.de Brito et al., "A service orchestration architecture for Fog-enabled infrastructures," 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), 2017, pp. 127-132, doi: 10.1109/FMEC.2017.7946419.

Meeniga Sriraghavendra, Priyanka Chawla, Huaming Wu, Sukphal Singh Gill and Rajkumar Buyya, DoSP:A Deadline-Aware Dynamic Service Placement Algorithm for Workflow oriented IoT Applications in Fog-Cloud Computing Environments,in book title "Energy Coservation Solutions for Fog-Edge Computing Paradigms" to be published by Springer book series "Lecture Notes on Data Engineering and Communications Technologies".

M. Sri raghavendra and P. Chawla, "A Survey on QOS and Fault Tolerance based Service Scheduling Techniques in Fog Computing Environment," ICRITO, Amity university, Noida, India,August 29-31, 2018.

M. K. T. a. N. D. De Donno, "Foundations and evolution of modern computing paradigms: Cloud, IoT,edge, and fog," IEEE Access, vol.7,pp.150936-150948,2019.

Mohammed S. Elbamby, Mehdi Bennis,Walid Saad, Matti Latva-aho, and Choong Seon Hong. 2018. Proactive edge computing in fog networks with latency and reliability guarantees. EURASIP Journal on Wireless Communications and Networking 2018, 1 (20 Aug 2018), 209.

Koshizuka and K. Sakamura, “Ubiquitous ID: Standards for Ubiquitous Computing and the Internet of Things”, IEEE Pervasive Computing, vol. 9, no. 4, pp. 98–101, October2010.

N. Kushalnagar, G. Montenegro, and C. Schumacher, “IPv6 over Low power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals”, in,2007.

P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. M. S. De Souza, andV. Trifa, “SOA-based Integration of the Internet of Things in Enterprise Services”, in Proceedings of IEEE International Conference on Future Internet: The Internet of Things Web Services (ICWS ’09), Los Angeles, CA USA, July 2009, pp.968–975.

M. Huang, W. Liu, T. Wang, A. Liu and S. Zhang, "A Cloud–MEC Collaborative Task Offloading Scheme With Service Orchestration," in IEEE Internet of Things Journal, vol. 7, no. 7, pp. 5792-5805, July 2020, doi: 10.1109/JIOT.2019.2952767.

M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments, IEEE Transactions on Mobile Computing (TMC),2020.

Manasrah, A. M., & Ba Ali, H. (2018). Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing. Wireless Communications and Mobile Computing, 2018, 1–16.

Mansoor Nasir, Khan Muhammad, Jaime Lloret, Arun Kumar Sangaiah, and Muhammad Sajjad. 2019. Fog computing enabled cost-effective distributed summarization of surveillance videos for smart cities. J. Parallel and Distrib. Comput.126 (2019), 161 –170.

Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A Survey on

Mobile Edge Computing: The Communication Perspective. IEEE Communications Surveys & Tutorials, 19(4), 2322–2358.

Minh, Q. T., Nguyen, D. T., Van Le, A., Nguyen, H. D., & Truong, A. (2017). Toward service placement on Fog computing landscape. 2017 4th NAFOSTED Conference on Information and Computer Science,291-296.

Mtshali, M., Kobo, H., Dlamini, S., Adigun, M., & Mudali, P. (2019). Multi-Objective Optimization Approach for Task Scheduling in Fog Computing. 2019 International Conference on Advances in Big Data, Computing andData Communication Systems (icABCD).

M. Pillay and D. Beckedahl, "EvoHyp - a Java toolkit for evolutionary algorithm hyper-heuristics," 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 2706-2713, doi:10.1109/CEC.2017.7969636.

Naha R. K., Garg S., Jayaraman P. R., Gao L., Xiang Y., & Ranjan R. (2018). Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions," *IEEE Access*, 6, 47980- 48009.

N. K. Giang, M. Blackstock, R. Lea and V. C. M. Leung, "Developing IoT applications in the Fog: A Distributed Dataflow approach," 2015 5th International Conference on the Internet of Things (IOT), 2015, pp. 155- 162, doi: 10.1109/IOT.2015.7356560.

Name, H. A. M., Oladipo, F. O., & Ariwa, E. (2017). User mobility and resource scheduling and management in fog computing to support IoT devices. 2017 Seventh International Conference on Innovative Computing Technology (INTECH).

Oma, R., Nakamura, S., Duolikun, D., Enokido, T., & Takizawa, M. (2018). An energy-efficient model for fog computing in the Internet of Things (IoT). Internet of Things, 1(2), 14–26.

Osanaiye, O., Chen, S., Yan, Z., Lu, R., Choo, K. R., & Dlodlo, M. (2017). From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework. IEEE Access, 5, 8284-8300.

P. Tsai, H. Hong, A. Cheng and C. Hsu, "Distributed analytics in fog computing platforms using tensorflow and kubernetes," 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS),2017,pp.145-150,doi: .1109/APNOMS.2017.8094194.

Padmavathi, S., Soniha, P. K., Soundarya, N., & Srimathi, S. (2017). Dynamic resource provisioning and monitoring for cloud computing. 2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing(INCOS).

Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N., & Buyya, R. (2016).ContainerCloudSim: An Environment for Modeling and Simulation of Containers in Cloud Data Centers. Software Practice and Experience, 1- 17.

P. G. V. Naranjo, E. Baccarelli, M. Scarpiniti, "Design and energy efficient resource management of virtualized networked fog architectures for the real-time support of iot applications," in The Journal of Supercomputing 74 (6), 2018, 2470–2507.

Qayyum, T., Malik, A. W., Khan, M. A., Khalid, O., & Khan, S. U. (2018). FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment. IEEE Access, 6, 1-10.

Qi, Q., & Tao, F. (2019). A Smart Manufacturing Service System Based on Edge Computing, Fog Computing and Cloud Computing, IEEE Access, 7, 86769-86777.

Qu, Z., Wang, Y., Sun, L., Peng, D., & Li, Z. (2020). Study QoS Optimization and Energy Saving Techniques in Cloud, Fog, Edge, and IoT. Complexity, 2020, 1-16.

Rafique, H., Shah, M. A., Islam, S. U., Maqsood, T., Khan, S., & Maple, C. (2019). A Novel Bio-Inspired Hybrid Algorithm (NBIHA) for Efficient Resource Management in Fog Computing, IEEE Access, 7, 115760- 115773.

Rahbari, D., Kabirzadeh, S., & Nickray, M. (2017). A security aware scheduling in fog computing by hyper heuristic algorithm. 2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS),87-92.

Rahman G., & C. C. (2018). Fog Computing, Applications, Security, and Challenges, Review," International Journal of Engineering & Technology, 7(3), 1615- 1621.

Rakshith, G., Rahul, M. V., Sanjay, G. S., Natesha, B. V., & Ram Mohana Reddy, G. (2018). Resource Provisioning Framework for IoT Applications in Fog Computing Environment. 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems(ANTS).

Ramirez, W., Masip-Bruin, X., Marin-Tordera, E., Souza, V. B. C., Jukan, A., Ren, G.-J., & Gonzalez de Dios, O. (2017). Evaluating the benefits of combined and continuous Fog-to-Cloud architectures. Computer Communications, 113, 43–52.

Ruimiao Ding, Xuejun Li, Xiao Liu, and Jia Xu. 2019. A Cost-Effective Time- Constrained Multi-workflow Scheduling Strategy in Fog Computing. In Service-Oriented Computing – ICSOC 2018 Workshops, Xiao Liu, Michael Mrissa, Liang

Ren, J., Zhang, D., He, S., Zhang, Y., & Li, T. (2019). A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms: Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet. ACM Computing Surveys, 52(6).

Rithvik, K., Kaur, S., Sejwal, S., Narwal, P. and Jain, P., 2019. Cloud computing data security using encryption algorithms. IIOAB Journal, 10(2), pp.75-82.

R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future Internet: the Internet of Things Architecture, Possible Applications and Key Challenges”, in Proceedings of Tenth International Conference on Frontiers of Information Technology (FIT ’12), Islamabad, India, December 2012, pp. 257–260.

R. Mahmud R., Ramamohanarao K., & Buyya R. (2018). Latency-Aware Application Module Management for Fog Computing Environments. ACM Transactions on Internet Technology, 19(1),1-21.

R.Mahmud, R., Srirama, S. N., Ramamohanarao, K., & Buyya, R. (2020). Profit- aware application placement for integrated Fog-Cloud computing environments. Journal of Parallel and Distributed Computing.

R Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, “Quality of Experience (QoE)-aware placement of applications in fog computing environments,” Journal of Parallel and Distributed Computing, vol. 132, pp. 190–203, 2019.

R. Bi, L. Huang, and Y.-J. A. Zhang, “Joint optimization of service caching placement and computation offloading in mobile edge computing system,” arXiv preprint arXiv:1906.00711,2019.

Saroa, M. K., & Aron, R. (2018). Fog Computing and Its Role in Development of Smart Applications. 2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications.

Sebrechts, M., Vanhove, T., Van Seghbroeck, G., Wauters, T., Volckaert, B., & De Turck, F. (2016). Distributed Service Orchestration: Eventually Consistent Cloud Operation and Integration. 2016 IEEE International Conference on Mobile Services (MS), 156-159.

Senna, C. R., Russi, L. G. C., & Madeira, E. R. M. (2014). An Architecture for Orchestrating Hadoop Applications in Hybrid Cloud. 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 544-545.

Sharma, S and H. Saini, "A novel four-tier architecture for delay aware scheduling and load balancing in fog environment", Sustainable Computing: Informatics and Systems, vol. 24, pp. 100355, Dec.2019.

Shi, Y., Ding, G., Wang, H., Roman, H. E., & Lu, S. (2015). The fog computing service for healthcare. 2015 2nd International Symposium on FutureInformation and Communication Technologies for UbiquitousHealthCare(Ubi-HealthTech).

Shishira, S. R., Kandasamy, A., & Chandrasekaran, K. (2016). Survey on metaheuristic optimization techniques in cloud computing. 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 1434-1440.

Shin S., Seo S., Eom S., Jung J., & Lee K. H. (2016). A pub/sub-based fog computing architecture for internet-of-vehicles. *Proceedings - 8th IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2016*, 90-93.

Skarlat, O., Schulte, S., Borkowski, M., & Leitner, P. (2016). Resource Provisioning for IoT Services in the Fog. 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), 32-39.

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

Skarlat O,Nardelli M, Schulte S, et al. Optimized IoT service placement in the fog. SOCA. 2017;11:427-443. https://doi.org/10.1007/s11761-017-0219-8

Sriraghavendra M, Chawla P,WuH,Gill SS, Buyya R. DoSP: a deadline-aware dynamic service placement algorithm forworkflow-oriented IoT applications in fog-cloud computing environments,in book title "energy coservation solutions for fog-edge computing paradigms. Lecture Notes on Data Engineering and Communications Technologies;2021.

Sri Raghavendra, M, Chawla, P, Singh Gill, S. DEEDSP: Deadline-aware and energy-efficient dynamic service placement in integrated Internet of Things and fog computing environments. Trans Emerging Tel Tech. 2021; e4368. doi:10.1002/ett.4368.

Stojmenovic, I., & Wen, S. (2014). The Fog Computing Paradigm: Scenarios and Security Issues. Proceedings of the 2014 Federated Conference on Computer Science and Information Systems.

Sun, Y., & Zhang, N. (2017). A resource-sharing model based on a repeated game in fog computing. Saudi Journal of Biological Sciences, 24(3), 687–694.

Sun, H., Yu, H., Fan, G., & Chen, L. (2019). Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture. Peer-to- Peer Networking and Applications.

Souza, V. B., Masip-Bruin, X., Marín-Tordera, E., Sànchez-López, S., Garcia, J., Ren,G.J., Juan Ferrer, A. (2018). Towards a proper service placement in combined Fog-to-Cloud (F2C) architectures. Future

Generation Computer Systems, 87,1–15.

Stavrinides, G. L., & Karatza, H. D. (2019). An energy-efficient, QoS-aware and cost- effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. *Future Generation Computer Systems*, 96, 216–226.

Svorobej, S., Takako Endo, P., Bendechache, M., Filelis-Papadopoulos, C., Giannoutakis, K., Gravvanis, G., Lynn, T. (2019). Simulating Fog and Edge Computing Scenarios: An Overview and Research Challenges. *Future Internet*, 11(3), 1-15.

Syed M. H., Fernandez E. B., & Ilyas M. (2016). A Pattern for Fog Computing. *The 10th Travelling Conference*.

Djemai, P. Stolf, T. Monteil, and J. Pierson. 2019. A Discrete Particle Swarm Optimization Approach for Energy-Efficient IoT Services Placement Over Fog Infrastructures. In 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC).32–40.

T. Choudhari, M. Moh, and T.-S.Moh, "Prioritized task scheduling in fog computing," in Proc. ACMSE, Richmond, Kentucky, Mar. 2018, pp. 401– 405.

T. Nazar, N. Javaid, M. Waheed, A. Fatima, H. Bano, and N. Ahmed. 2019. Modified Shortest Job First for Load Balancing in Cloud-Fog Computing. InAdvances on Broadband and Wireless Computing, Communication and Applications, Leonard Barolli, Fang-Yie Leu, Tomoya Enokido, and Hsing- Chung Chen (Eds.). Springer International Publishing, Cham, 63–76.

Talaat, F. M., Saraya, M. S., Saleh, A. I., Ali, H. A., & Ali, S. H. (2020). A load balancing and optimization strategy (LBOS) using

reinforcement learning in fog computing environment. Journal of Ambient Intelligence and HumanizedComputing.

Taneja, M., & Davy, A. (2016). Resource Aware Placement of Data Analytics Platform in Fog Computing. Procedia Computer Science, 97, 153–156.

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

Tseng, C.-L., & Lin, F. J. (2018). Extending scalability of IoT/M2M platforms with Fog computing. 2018 IEEE 4th World Forum on Internet of Things (WF- IoT).

Tran, M.-Q., Nguyen, D. T., Le, V. A., Nguyen, D. H., & Pham, T. V. (2019). Task Placement on Fog Computing Made Efficient for IoT Application Provision. Wireless Communications and Mobile Computing, 2019,1–17.

V. Cardellini, V. Grassi, F. L. Presti and M. Nardelli, "On QoS-aware scheduling of data stream applications over fog computing infrastructures," 2015 IEEE Symposium on Computers and Communication (ISCC), 2015, pp. 271-276, doi: 10.1109/ISCC.2015.7405527.

Velasquez, K., Abreu, D. P., Goncalves, D., Bittencourt, L., Curado, M., Monteiro, E., & Madeira, E. (2017). Service Orchestration in Fog Environments. 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud).

Velasquez, K., Abreu, D. P., Assis, M. R. M., Senna, C., Aranha, D. F., Bittencourt,L. F. Madeira, E. (2018). Fog orchestration for the Internet of Everything:state-of-the-art and research challenges.

Journal of Internet Services and Applications, 9(1).

V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor, “Handling service allocation in combined fog-cloud scenarios,” in 2016 IEEE international conference on communications (ICC), pp. 1–5, IEEE,2016.

Wamser, F., Lombardo, C., Vassilakis, C., Dinh-Xuan, L., Lago, P., Bruschi, R., & Tran-Gia, P. (2018). Orchestration and Monitoring in Fog Computing for Personal Edge Cloud Service Support. 2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN).

Wang, D., Liu, Z., Wang, X., & Lan, Y. (2019). Mobility-aware Task Offloading and Migration Schemes in Fog Computing Networks. IEEE Access, 7.

Xia Y, Etchevers X, Letondeur L, Coupaye T, Desprez F. Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed iot applications in the fog. InProceedings of the 33rd Annual ACM Symposium on Applied Computing 2018 Apr 9 (pp. 751-760).

Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, L. Qi, A Computation Offloading Method over Big Data for IoT-Enabled Cloud-Edge Computing, Future Generation Computer Systems 95 (2019)522-533.

Xiao, Y., & Krunz, M. (2018). Distributed Optimization for Energy-efficient Fog Computing in the Tactile Internet. IEEE Journal on Selected Areas in Communications, 36(11), 2390-2400.

Xuan-Qui Pham and Eui-Nam Huh, "Towards task scheduling in a cloud-fog computing system," 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2016, pp. 1-4, doi:

10.1109/APNOMS.2016.7737240.

Yates, W. B., & Keedwell, E. C. (2019). An analysis of heuristic subsequences for offline hyper-heuristic learning. Journal of Heuristics, 25, 399-430.

Jiang, Y. Chen, S. Yang, and C. Wu. 2019. Energy-Efficient Task Offloading for Time-Sensitive Applications in Fog Computing. IEEE Systems Journal 13, 3 (Sep. 2019), 2930–2941.

Ye, D., Wu, M., Tang, S., & Yu, R. (2016). Scalable Fog Computing with Service Offloading in Bus Networks. 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud).

Yin, L., Luo, J., & Luo, H. (2018). Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacture. IEEE Transactions on Industrial Informatics, 14(1).

Gadallah, M. H. Ahmed, and E. Elalamy, "Dynamic LTE Resource Reservation for critical M2M Deployments", Pervasive and Mobile Computing, vol. 40, pp. 541–555, January2017.

Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A. Jue,J.P. (2019). All One Needs to Know about Fog Computing and Related Edge Computing Paradigms. Journal of Systems Architecture, 98, 289-330.

Zhang, F., Hwang, K., Khan, S. U., & Malluhi, Q. M. (2016). Skyline Discovery and Composition of Multi-Cloud Mashup Services. IEEE Transactions on Services Computing, 9(1), 72–83.

Zhou, A., Sun, Q., Zou, H., Yang, F., & Wang, S. (2013). FTCloudSim: A Simulation Tool for Cloud Service Reliability Enhancement Mechanisms. ACM/IFIP/USENIX Middleware 2013, 1-2.

Fei, B. Li, S. Yang, C. Xing, H. Chen, and L. Hanzo, "A Survey of Multi-objective Optimization in Wireless Sensor Networks: Metrics,

Algorithms, and Open Problems", IEEE Communications Surveys & Tutorials, vol. 19, no. 1, pp. 550–586, September2017.

AUTHOR'S PUBLICATIONS

International Conference Publications

- M. Sri Raghavendra and P. Chawla, "A Review on container-based lightweight virtualization for fog computing," ICRITO, Amity university, Noida, India, August 29-31, 2018.
- M. Sri Raghavendra and P. Chawla, "A Survey on QOS and Fault Tolerance based Service Scheduling Techniques in Fog Computing Environment," ICRITO, Amity university, Noida, India, August 29-31, 2018.
- M. Sri Raghavendra, P. Chawla and Ajay Rana, "A Survey of Optimization Algorithms For Fog Computing Service Placement," ICRITO, Amity university, Noida, India, June 04-05, 2020.
- M. Sri Raghavendra, P. Chawla and Y. Narasimhulu, "A Probability Based Joint-Clustering Algorithm for Application Placement in Fog-to-Cloud Computing," ICRITO, Amity university, Noida, India, 2021.

Book chapter

- Sri Raghavendra, M., Chawla, P., Wu, H., Gill, S. S., & Buyya, R. (2022). DoSP: A Deadline-Aware Dynamic Service Placement Algorithm for Workflow-Oriented IoT Applications in Fog-Cloud Computing Environments. In Energy Conservation Solutions for Fog-Edge Computing Paradigms (pp. 21-47). Springer, Singapore.

UGC Journal

- M. Sri Raghavendra and P. Jain, "VIRTUAL MACHINE VS CONTAINER: AN APPLICATION PERFORMANCE REVIEW," vol. 6, pp. 29–40, 2017.

International Journal Publications

- Sri Raghavendra, M, Chawla, P, Singh Gill, S. DEEDSP: Deadline-aware and energy-efficient dynamic service placement in integrated Internet of Things and fog computing environments. Trans Emerging Tel Tech. 2021; e4368. doi:10.1002/ett.4368.