REPUBLIC OF TÜRKİYE
ALTINBAŞ UNIVERSITY
Institute of Graduate Studies
Electrical and Computer Engineering

ENHANCING IOT RESOURCE UTILIZATION IN FOG-CLOUD ENVIRONMENT FOR PROVIDING REAL TIME SERVICES

Naseem Adnan Hameedi ALSAMARAI

Doctor of Philosophy

Supervisor
Prof. Dr. Osman Nuri UÇAN

İstanbul, 2024

ENHANCING IOT RESOURCE UTILIZATION IN FOG-CLOUD ENVIRONMENT FOR PROVIDING REAL TIME SERVICES

Naseem Adnan Hameedi ALSAMARAI

Electrical and Computer Engineering

Doctor of Philosophy

ALTINBAŞ UNIVERSITY

2024

The thesis titled " Enhancing IoT Resource Utilization In Fog-Cloud Environment For Providing Real Time Services" prepared and presented by “Naseem Adnan Hameedi Alsamarai” and submitted on ……. has been accepted as a Doctor of Philosophy Dissertation in Electrical and Computer Engineering.

Prof. Dr. Osman Nuri Ucan
Supervisor

Thesis Defense Jury Members:

Academic Title First LAST NAME	Department of …, University	_____
Academic Title First LAST NAME	Department of …, University	_____
Academic Title First LAST NAME	Department of …, University	_____
Academic Title First LAST NAME	Department of …, University	_____
Academic Title First LAST NAME	Department of …, University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Naseem Adnan Hameedi ALSAMARAI

Signature

DEDICATION

First, I am greatly indebted in my work and success to our merciful " Allah " who gave me the ability to terminate this work and special peace to our messenger (Mohammed) Peace Be Upon Him.

I would like to express my gratitude to those who have made this thesis possible.

I offer my sincerest gratitude to my supervisor, Professor (Osman Nuri Uçan), who supported me throughout my thesis with his patience and guidance while allowing me the room to work in my own way.

I would like to express my sincere thanks my friends my colleges And to my teachers (Prof .Hasan Hüseyin BALIK, and Dr. Abdullah Abdu Ibrahim).

Last but not least, I would like to dedicate this thesis to my Parents, my Wife, my family and the people I love and who love me. Without their continuous support, I cannot complete the thesis alone.

I would also like to thank my Friends and anyone who has helped me with their thought and opinions in successfully completing the project.

ABSTRACT

ENHANCING IOT RESOURCE UTILIZATION IN FOG-CLOUD ENVIRONMENT FOR PROVIDING REAL TIME SERVICES

ALSAMARAI, Naseem Adnan Hameedi

Ph.D., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Prof. Dr. Osman Nuri UÇAN

Date: 06/2024

Pages: 86

When resources and computers are made available on demand over the internet, this is called fog-cloud computing. This makes it easy to offer people a variety of integrated computing services without being limited by local resources.

This means giving people more than just places to store and back up their data and ways to sync their own files. but it also has processing power and a simple software interface that lets the user control when it's connected to the network. This makes things easier by ignoring many details and internal processes.

When it comes to the cloud, scheduling algorithms are necessary to provide services that meet goals like high performance, low prices, minimal energy use, and so on. It is an NP-hard problem to come up with scheduling methods that meet more than one of these goals. We introduce some new heuristic scheduling algorithms in this thesis that allow for multi-objective optimization. We then compare their effectiveness with some well-known scheduling algorithms to study how well they work.

The first algorithm, the BDA, looks at the Make-span and the period it takes to complete jobs by the due date. Our algorithm takes into account a task's due date by giving the most weight for the job with the earlier due date and send it to the resource that can complete it in the shortest amount of time to achieve the shortest Makespan. Fog Max-Cloud Min is the name of the first part of our method. Ant Colony Optimization is the name of the second part. We looked at how well our suggested algorithm methods met deadlines and how long the system took to make. Compared to the tools we have now. The study results show that

our algorithm is better at getting things done with the lowest Makespan and the best deadline satisfaction.

In the second algorithm, We develop improvements to the performance and cost (PC) algorithm in order to give more weight Considering the significant expenses involved, reduce energy consumption, and reduce the duration of create a product. In this work, we describe an approach that is a combination of the PCA and GWO techniques. This algorithm is called the Performance and Cost-Gray Wolf Optimization (PC-GWO) algorithm. The results of the test indicate that the PC-GWO The algorithm decreases the average total energy usage by 12.17 percent, 11.5 percent, and 7.19 percent, as well as the Makespan by 16.72 percent, 16.38 percent, and 14.10 percent. When compared to the GWO algorithm, the PCA method, and the PSO algorithm, it also improves the best average resource consumption by 13.2 percent, 12.05%, and 10.9 percent respectively.

Keywords: Fog Computing, Cloud Computing, Task Scheduling, Energy Consumption, Makespan, Resource Utilization, Cost Processing, Internet of Things, Bandwidth and Deadline

ÖZET

GERÇEK ZAMANLI HİZMET SUNMAK İÇİN SİS-BULUT ORTAMINDA IOT KAYNAK KULLANIMININ ARTIRILMASI

ALSAMARAI, Naseem Adnan Hameedi

Doktora, Elektrik ve Bilgisayar Mühendisliği, Altınbaş Üniversitesi,

Danışman: Prof. Dr. Osman Nuri UÇAN

Tarih: 06/2024

Sayfalar: 86

Kaynaklar ve bilgisayarlar internet üzerinden talep üzerine kullanıma sunulduğunda buna sis bulut bilişim denir. Bu, yerel kaynaklarla sınırlı kalmadan insanlara çeşitli entegre bilgi işlem hizmetleri sunmayı kolaylaştırır.

Bu, insanlara verilerini depolayacakları ve yedekleyecekleri yerlerden ve kendi dosyalarını senkronize etme yollarından daha fazlasını vermek anlamına gelir. ancak aynı zamanda işlem gücüne ve kullanıcının ağa ne zaman bağlanacağını kontrol etmesine olanak tanıyan basit bir yazılım arayüzüne de sahiptir. Bu, birçok ayrıntıyı ve iç süreci göz ardı ederek işleri kolaylaştırır.

Bulut söz konusu olduğunda, yüksek performans, düşük fiyatlar, minimum enerji kullanımı vb. hedefleri karşılayan hizmetleri sağlamak için planlama algoritmaları gereklidir. Bu hedeflerden birden fazlasını karşılayan planlama yöntemleri bulmak NP-zor bir problemdir. Bu tezde, çok amaçlı optimizasyona izin veren bazı yeni buluşsal planlama algoritmalarını tanıtıyoruz. Daha sonra ne kadar iyi çalıştıklarını incelemek için bunların etkinliğini bazı iyi bilinen planlama algoritmalarıyla karşılaştırıyoruz.

İlk algoritma olan BDA, yapım süresine ve işlerin vade tarihine kadar tamamlanması için geçen süreye bakar. Algoritmamız, daha erken bitiş tarihi olan işe en fazla ağırlığı vererek görevin bitiş tarihini dikkate alır ve en kısa Makespan'a ulaşmak için onu en kısa sürede tamamlayabilecek kaynağa gönderir. Fog Max-Cloud Min yöntemimizin ilk kısmının adıdır. Karınca Kolonisi Optimizasyonu ikinci bölümün adıdır. Önerilen

algoritma yöntemlerimizin son teslim tarihlerini ne kadar karşıladığını ve sistemin bunu yapmasının ne kadar sürdüğünü inceledik. Şu anda sahip olduğumuz araçlarla karşılaştırıldığında. Çalışma sonuçları, algoritmamızın işleri en düşük Makespan ve en iyi son teslim tarihi memnuniyetiyle halletmede daha iyi olduğunu gösteriyor.

İkinci algoritmada, önemli harcamaları göz önünde bulundurarak, enerji tüketimini azaltmak ve ürün oluşturma süresini kısaltmak için performans ve maliyet (PC) algoritmasında iyileştirmeler geliştiriyoruz. Bu çalışmada PCA ve GWO tekniklerinin birleşiminden oluşan bir yaklaşımı tanımlıyoruz. Bu algoritmaya Performans ve Maliyet-Gri Kurt Optimizasyonu (PC-GWO) algoritması adı verilmektedir. Testin sonuçları, PC-GWO algoritmasının ortalama toplam enerji kullanımını yüzde 12,17, yüzde 11,5 ve yüzde 7,19 oranında azalttığını, Makespan'ın ise yüzde 16,72, yüzde 16,38 ve yüzde 14,10 oranında azalttığını gösteriyor. GWO algoritması, PCA yöntemi ve PSO algoritmasıyla karşılaştırıldığında, en iyi ortalama kaynak tüketimini de sırasıyla yüzde 13,2, %12,05 ve yüzde 10,9 oranında artırıyor.

Anahtar Kelimeler: Sis Bilişimi, Bulut Bilişim, Görev Planlama, Enerji Tüketimi, Makespan, Kaynak Kullanımı, Maliyet İşleme, Nesnelerin İnterneti, Bant Genişliği ve Son Teslim Tarihi

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

ACO : Ant Colony Optimization

AWS : Amazon Web Services

BDA : Bandwidth Deadline Algorithm

CPU : Central Processing Unit

DATS : Dispersion-stable Task Scheduling

DRL : Deep Reinforcement Learning

EDF : Earliest Deadline First

FC : Fog Computing

FCFS : First Come, First Services

FEC : Fog/Edge Computing

FEC : Forward Error Correction

FNs : Fog Nodes

GA : Genetic Algorithm

GAE : Google App Engine

GWO : Gray Wolf Optimizer

IaaS : Infrastructure as a Service

IIoT : Industrial Internet of Things

ILP : Integer Linear Programming

ILP : Integer Linear Programming

IoT : Internet of Things

IT : Information Technology

M2M : Machine-to-Machine

MASM : Multiple Algorithm Service Model

MEETS : Maximal Energy-Efficient Task Scheduling

MILP : Mixed-Integer Linear Programming

NIST : National Institute of Standards and Technology

NSGA : Non-Dominated Sorting Genetic Algorithm

PaaS : Platform as a Service

PC-GWO : Performance and Cost-Gray Wolf Optimization Algorithm

PC : Personal Computer

PCA : Performance and Cost Algorithm

PDA : Personal Digital Assistant

PDF : Portable Document File

PGA : Priority-aware Genetic Algorithm

PSO : Particle Swarm Optimization

RM : Resource Monitoring

SaaS : Software as a Service

SCALE : Security, Cognition, Agility, Latency, and Efficiency

SDKs : Software Development Kits

SDN : Software-Defined Networking

TD : Task Director

TEA : Tide Ebb Algorithm

TS : Task Scheduler

UAV : Unmaned Arial Vevhele

1. INTRODUCTION

1.1 RESEARCH OVERVIEW

When resources and computers are made available on demand over the internet, this is called fog-cloud computing. This makes it easy to offer people a variety of integrated computing services without being limited by local resources [1]. This means giving people more than just places to store and Make copies of their data and methods as a precautionary measure to sync their own files. but it also has processing power and a simple software interface that lets the user control when it's connected to the network [2]. This makes things easier by ignoring many details and internal processes.When it comes to the cloud, scheduling algorithms are necessary to provide services that meet goals like high performance, low prices, minimal energy use, and so on [3]. It is an NP-hard problem to come up with scheduling methods that meet more than one of these goals. We introduce some new heuristic scheduling algorithms in this thesis that allow for multi-objective optimization [4]. We then compare their effectiveness with some well-known scheduling algorithms to study how well they work.

The initial method, referred to as the BDA, prioritizes the Makespan and work completion times in relation to their respective due dates. The system gives priority to projects with earlier deadlines and assigns them to resources that can complete them in the shortest amount of time, In order to attain the desired outcome, shortest possible Makespan. The first component of our approach is referred to as Fog Max-Cloud Min, while the second component is known as Ant Colony Optimization. We assessed the efficacy of our algorithm in fulfilling deadlines and the overall system processing time in comparison to preexisting technologies. The study results indicate that our algorithm performs very well in obtaining the shortest Makespan and highest level of deadline satisfaction.

The second algorithm aims to enhance the performance and cost (PC) algorithm by giving more importance to high-profit costs, reducing energy consumption, and shortening product creation time. The Performance and Cost-Gray Wolf Optimization (PC-GWO) method is a novel strategy that integrates the PCA and GWO approaches. The test findings indicate that the PC-GWO algorithm achieves a reduction in mean total energy consumption of 12.17%,

11.5%, and 7.19%, as well as a drop in the Makespan by 16.72%, 16.38%, and 14.10%. In comparison to the GWO algorithm, the PCA method, and the PSO algorithm, it also enhances Mean resource usage by 13.2%, 12.05%, and 10.9% correspondingly, Scheduling can encompass various policies, like as [5]:

i. Enhance the productivity of the tasks.
ii. Enhance the rate of data processing.
iii. Reduce the amount of time it takes for data to travel between two points.
iv. Reduce expenses.
v. Prevent the system from sustaining any harm.
vi. Attain optimal distribution of workload.

In this study, we introduce a novel scheduling algorithm paradigm to lower delay, make better use of resources, and cut down on task waiting time. This will ensure that IoT activities run as planned. This model of an algorithm takes into account both the task's limit and its scope.

1.2 THE PROBLEM DEFINITION

The majority of conventional algorithms used for fog-cloud scheduling send the work to any cloud or fog resource as soon as it arrives without considering its bandwidth or deadline. This leads to some problems, such as unbalanced resource utilization and high latency time in execution tasks, in case most IoT applications need real-time services.

1.3 THE TARGET OF DISSERTATION

The purpose of this dissertation is to create an algorithm that can manage several essential job requirements, such as time, system requirements (CPU and memory usage), energy usage, and temperature. Scheduling pieces a crucial role in enhancing the performance of fog-cloud computing systems. Therefore, a hybrid algorithm has been developed that combines multiple task requirements to find a solution that meets more than one requirement simultaneously.

We came up with the Bandwidth Deadline Algorithm (BDA) to solve the problem of how to schedule tasks in the Fog-Cloud setting. It is a heuristic algorithm since it changes based

on the jobs and resources it is given. When planning the process, it also takes into account the due dates of each job and how important it is. It gives it to the right resource to see if that can help it find the best answer to an optimization problem.

1.4 ORGANIZATION OF DISSERTATION

This dissertation's remaining sections are arranged as follows:

Chapter 2: This text provides a general outline of IoT (Internet of Things), fog computing, and cloud computing. It explains the architecture and deployment models for these technologies, as well as their respective services. The article analyzes the technical, qualitative, and cost-effective characteristics of these technologies and identifies areas for future development. Additionally, it summarizes previous research in fog-cloud computing scheduling and proposes a new algorithm to address issues with earlier methods. Finally, the text concludes with a comparison of various scheduling algorithms based on their methods, parameters, and resulting environments.

Chapter 3: This chapter outlines the structure of our proposed algorithms, which consist of two sub-algorithms each. The goal of these algorithms is to address one or two characteristics of a given task To identify the nearest optimal solution for the system. Specifically, we aim to reduce latency time and energy consumption while also preventing damage to the system by lowering its temperature.

Chapter 4: This chapter provides a concise overview of the evaluation, simulation model, and outcome. These aspects are elucidated by comparing our methods with the following baselines:

The Bandwidth Deadline Algorithm was compared to the First Come, First Serve (FCFS), Earliest Deadline First (EDF), and Max-Min algorithms.

The second approach is the Performance and Cost-Gray Wolf Optimization algorithm (PC-GWO), which is compared to the Performance and Cost approach (PCA), the Gray Wolf Optimizer, and Particle Swarm Optimization (PSO).

Chapter 5: Presents the conclusion and outlines the prospective future work.

2. OVERVIEW OF IOT, FOG AND CLOUD COMPUTING AND SCHEDULING ALGORITHMS IN FOG-CLOUD ENVIRONMENT

2.1 INTRODUCTION

Chapter 2 begins with its infancy., the standard definitions of IoT, Fog, and Cloud computing, then comes system model architecture cloud and fog evaluation, service type of fog and Cloud, the characteristics of fog and cloud computing,

2.2 INTERNET OF THINGS (IOT) DEFINITIONS:

i. Oracle Defines the IoT[1]: In the context of the Internet of Things (IoT), "things" refers to physical entities that are outfitted with software, hardware, and other advanced technologies. These objects have the ability to connect to and communicate with various other systems and devices that are connected to the internet. This category includes a diverse range of devices, spanning from simple household goods to advanced industrial machines.

ii. Amazon Web Services (AWS)Define the IoT [2]: " The Internet of Things (IoT) The term "Internet of Things" means an internet of interconnected devices and the hardware and software that enables communication between these devices and the cloud. It also pertains to the interconnections that gadgets establish with each other. Thanks to the advancement of inexpensive computer chips and the implementation of high-speed internet connections, we currently own an immense number of internet-connected devices. This suggests that everyday objects like toothbrushes, vacuum cleaners, autos, and machinery can utilize sensors to collect data and respond intelligently to individuals".

2.3 WHY IS THE INTERNET OF THINGS (IOT) SO IMPORTANT?

The Internet of Things (IoT) has become increasingly important in the twenty-first century. Embedded devices enable the Internet connectivity of common objects such as cars, baby monitoring, appliances for the home, and thermostats. This allows for seamless communication between people, processes, and entities. Efficient computing, cloud computing, extensive data processing, data analysis, and mobile technologies enable the

communication and data collecting of physical things with minimal human involvement. In this networked environment, the digital and physical realms collaborate as digital systems possess the power to record, observe, and alter the interactions among connected items [3], [4].

2.4 WHAT IS INDUSTRIAL IOT?

The Industrial Internet of Things (IIoT) is a concept that applies Internet of Things (IoT) technologies specifically in industrial environments to manage and supervise cloud-based sensors and equipment. The PDF of the Titan use case provides a clear demonstration of this point. The term "machine-to-machine" (M2M) communication has historically been used by the industry to refer to wireless automation and control. However, the implementation of cloud technology, along with advancements in analytics and machine learning, has enabled firms to achieve higher degrees of automation. This not only enhances profitability but also facilitates the implementation of new business models. The phrase IIoT, short for "Industrial Internet of Things," is also referred to as "Industry 4.0," denoting the fourth stage of the Industrial Revolution. Below are few instances of typical applications of the Industrial Internet of Things [5], [6].

i. Smart Industrial.
ii. Connected assets and preventive and predictive maintenance.
iii. Smart power grids.
iv. Smart cities.
v. Connected logistics.
vi. Smart digital supply chains.

2.5 FOG COMPUTING DEFINITIONS

i. The definition of the National Institute of Standards and Technology (NIST)[7]: “Similar to a computer systems analyst, fog computing is designed as a hierarchical structure that provides broad access to a shared and scalable share of computing resources. This approach enables the smooth integration of apps and services that are

distributed that demand minimal delay. The system consists of fog nodes, which can be physical or virtual, positioned within smart end-devices and centralized services in the cloud".

ii. Cisco defines Fog Computing[3]: "Fog computing extends the functionalities of cloud computing and its services to the network's periphery, offering a broader spectrum of capabilities. Like cloud computing, fog computing delivers data, computing, storage, and application services to users. It has three main characteristics: proximity to end-users, wide geographical distribution, and support for mobility. These services are located either at the network edges or on devices such as set-top boxes or routers".

2.6 FOG COMPUTING DEPLOYMENT

2.6.1 Private Fog Node

Not more than one person from the same company can use a fog node, such as people from different business units. That person or group could be the company itself, a third-party service provider, or a mix of the two. The fog hub could be on or off the business's land [8].

2.6.2 Community Fog Node

There is a certain clientele of firms that share common difficulties, such as purpose, security requirements, rules, and compliance expectations. A fog node is specifically intended for this clientele so that it can share these issues. An individual group, many groups within the same organization, an external third party, or a combination of these organizations may be responsible for the Taking full responsibility for the business's ownership, leadership, and operation of the facility. Alternatively, the responsibility may be shared by all of these entities. The location could be either within or outside of the premises as the case may be [8].

2.6.3 Public fog Node

A fog node can be used by anyone, and it can be owned, controlled, and run by a number of groups, including businesses, universities, and governments. The provider's building is where this fog node is located [8].

2.6.4 Hybrid Fog Node

There may be Multiple distinct private entities, community, or public fog nodes that make up a complex fog node. Each one keeps its own identity while being linked using standard or custom technology. This makes it possible for data and programs to move between these nodes. For example, fog bursts can be used to spread the work among them [8][9].

2.7 FOG COMPUTING SERVICES

2.7.1 Computing Services

Because devices in the perception layer don't have a lot of computing power, remote processing methods have been created. The Fog layer is used for processing, both because sensor nodes don't have a lot of processing power and to make computing location-specific and save energy. Processing that is done in the cloud can be moved to the Fog layer to make it more focused and quick[8][9].

Depending on the job, the way that layers share the load can be set up in different ways. As an example, a system that looks for patterns in data could divide the job so that patterns in specific areas are found at the Fog layer. Generalized patterns, on the other hand, can only be found in the Cloud. Not only does the Fog layer handle data, it can also handle events in real time and make the system more reliable. Through abstraction, agent-based management, and virtual machines, there are a number of middleware that can be used to control physical devices.

2.7.2 Storage Services

A lot of information is made by the sensor nodes, which are many of them. Most of the time, the perception layer doesn't have enough storage room to hold all the data that is created

every day. Not all the data needs to be sent straight to the cloud, especially if some of it is useless or already there. The data should be briefly stored in the middle fog layer so that it can be filtered, analyzed, and compressed for faster transmission or local learning. In situations where connection isn't strong, this helps make the system more reliable by making sure that client nodes always behave correctly. In their review of fog computing for IoT, Sarkar et al. talk about these aspects of fog computing [8][9].

2.7.3 Communication Services

Wireless nodes are the primary means of communication in the IoT, and the protocols used are specifically designed to minimize power consumption, enable narrow-band transmission, or provide a wider range of coverage. These design considerations are necessary owing to the limited resources available at the perception layer. There is a wide array of alternative protocols available in the market.

The Fog layer is strategically positioned to coordinate various protocols and integrate their communication with the Cloud layer. This facilitates the administration of subnetworks consisting of sensors and actuators, enhances system security, facilitates the transmission of messages between devices, and improves system reliability. In addition, the Fog layer can facilitate interoperability across multiple protocols by interpreting the format of the representation. Additionally, it has the capability to render non-IP-based devices detectable and reachable via the Internet [8][9].

2.8 BENEFITS OF FOG COMPUTING

Fog Computing (FC) offers five key advantages, namely security, cognition, agility, latency, and efficiency, which can be demonstrated using the acronym SCALE.

2.8.1 Security

Fog computing enhances the security of Internet of Things (IoT) devices, guaranteeing their safety and dependability during transactions. Managing wireless sensors in outdoor areas often involves the need to remotely update their source code to address security vulnerabilities. Nevertheless, a remote central backend server may encounter difficulties in promptly executing updates due to environmental factors such as fluctuating signal strength,

outages, and restricted bandwidth. This, in turn, heightens the vulnerability to cyber-attacks. Alternatively, with the implementation of a Fog Computing (FC) architecture, the backend has the capability to utilize different FC nodes to promptly update the software and ascertain the optimal routing path for the wireless sensors throughout the network [10].

2.8.2 Cognition

Essentially, FC assists clients in comprehending their goals in order to autonomously determine the timing and location for utilizing computing, storage, and control features. This is accomplished by a range of systems that facilitate self-adjustment, self-arrangement, self-repair, and self-communication. By imparting this consciousness, IoT devices undergo a transformation from being inactive to becoming active intelligent devices that has the ability to function and react to client requirements independently, without depending on decisions made by a remote Cloud [10].

2.8.3 Agility

Implementing Forward Error Correction (FEC) in Internet of Things (IoT) systems improves their adaptability. Unlike the current economic model for Cloud services, where big corporations have the sole responsibility of creating, establishing, and overseeing the basic infrastructure, FEC offers a different approach. It enables individuals and small businesses to provide FEC services using standard open software interfaces or open Software Development Kits (SDKs). By incorporating standards such as ETSI's MEC and embracing the Indie Fog business model, significant advancements can be made in the development of expansive IoT systems [10].

2.8.4 Latency

The dominant agreement regarding FC is that it enables quick answers for applications that demand extremely low latency. It is especially important for many commonly used applications and industrial automation systems to continuously gather and analyze sensory data in the form of a data stream in order to detect any events and respond quickly. By utilizing FC, these systems can expedite time-sensitive activities. Moreover, FC's

softwarization capability allows the remote central server to fully alter the functionality of physical devices using software abstraction, providing a highly adaptable platform for quickly reconfiguring IoT devices [10].

2.8.5 Efficiency

Applying Forward Error Correction (FEC) can enhance the effectiveness of Cellular Internet of Things (CIoT) by boosting performance and lowering costs. For example, in healthcare or eldercare systems, FEC can distribute tasks to Internet gateway devices of healthcare sensors and handle data analytics tasks. This method is beneficial because it produces faster results by processing data near its source. Additionally, it reduces outgoing communication costs by using gateway devices for most tasks [10].

2.9 CLOUD COMPUTING DEFINITIONS

Cloud computing is an emerging technology that revolutionizes the way we develop applications, construct computer systems, and utilize existing resources to create software. The concept is rooted on "dynamic provisioning," which can be applied to services, computing power, storage, networking, and IT systems in a broader sense. Cloud computing can be elucidated by various means, however, the following are the three most prevalent explanations:

i. The definition of the National Institute of Standards and Technology (NIST)[11]: Cloud computing is "a Cloud computing is an emerging technology that revolutionizes the way we develop applications, construct computer systems, and utilize existing resources to create software. The concept is rooted on "dynamic provisioning," which can be applied to services, computing power, storage, networking, and IT systems in a broader sense. Cloud computing can be elucidated by various means, however, the following are the three most prevalent explanations".

ii. The definition of SUN microsystem[12]: "Cloud computing can be defined as the gradual transfer of computer and data resources to the Internet. However, there is a clear differentiation: cloud computing represents a new milestone in the importance of

network computing. It offers enhanced effectiveness, extensive capacity for growth, and expedited, streamlined program creation. The topic encompasses new programming paradigms, advanced information technology infrastructure, and the promotion of creative business models".

iii. Barrie Sosinsky defines cloud computing[11]: " Cloud computing encompasses the usage of virtualized resources on a distributed network, accessed using common Internet protocols and networking standards, for running programs and providing services. It is characterized by the concept that resources are virtual and infinite, and that specific information about the physical systems on which software operates is hidden from the user".

2.10 CLOUD DEPLOYMENT MODELS

The function of the Cloud and the specifics of its location are described in a cloud deployment model.
The following are the four deployment models:

2.10.1 Public Cloud

The technology of the public cloud is run by a company that offers cloud services. Anyone or a large group of businesses can use the services. The design of the public cloud is shown in Figure 2.1. There are different public clouds, such as Amazon Web Services, HP Cloud Services, Google App Engine (GAE), and Windows Azure [11], [13], [14].

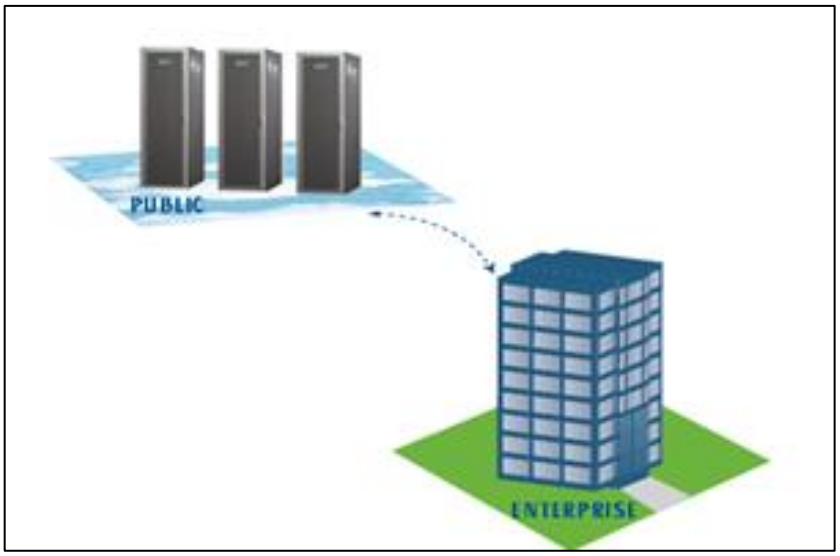


Figure 2.1: Public Cloud [12].

2.10.2 Private Cloud

The infrastructure that makes up a private cloud is developed specifically for the purpose of serving the needs of an enterprise. The Private Cloud could be operated by that organization or by a third party. It is possible to construct private clouds either on-premises or off-premises[11], [13], [14].

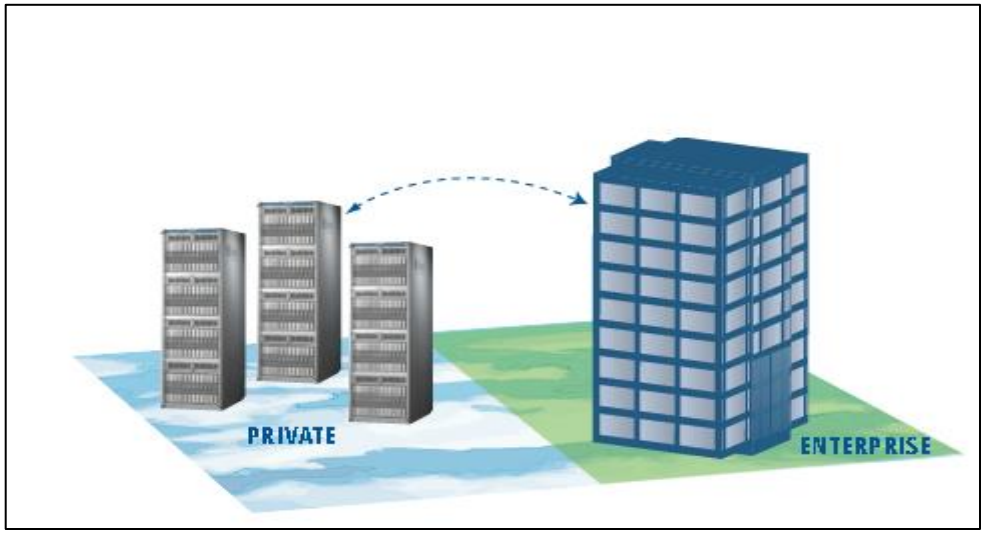


Figure 2.2: Privat Cloud [12].

2.10.3 Hybrid Cloud

A hybrid cloud is a type of cloud computing system that combines multiple clouds, including private, community, or public clouds while maintaining their individual identities. These clouds are connected together to form a cohesive unit. A hybrid cloud can provide access to data and applications through standardized or proprietary means and also allow for the transfer of applications from one Cloud to another [11], [13], [14].

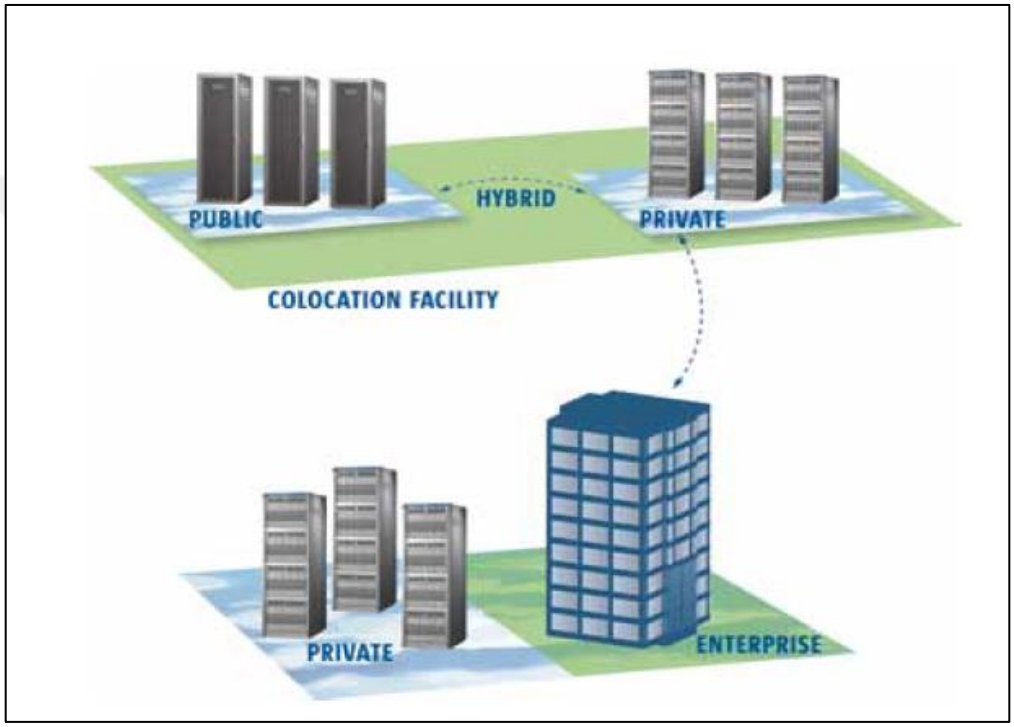


Figure 2.3: Hybrid Cloud [12]

2.10.4 Community cloud

A "community cloud" is a type of cloud computing system that is made to help a group of companies with a certain task. These groups usually care about the same things, like policies, security, following the rules, and their purpose. The groups in the community can run the community cloud, or a third-party provider can make it work [11], [13], [14].

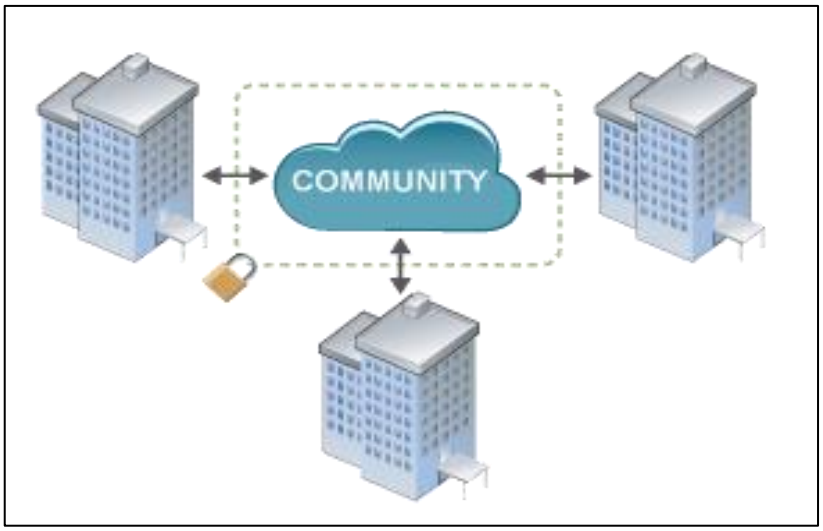


Figure 2.4: Community Cloud [12]

2.11 CLOUD EVOLUTION

Some people have thought about cloud computing for a long time, but it is only recently becoming popular in the IT and communication technology fields. In the past few years, it has had a lot of economic success. A lot of people agree that cloud computing will be very important in the next ten years. This part looks at the past of how cloud computing has changed over time and tries to figure out why it has only recently become popular even though it has been around for a while.

2.11.1 Getting Ready for Cloud

2.11.1.1 Datacenter

All you have to do to make a big datacenter out of cheap computers is put these building blocks together in a parking lot and connect them to the Internet. In the past few years, there have been a lot more computers and data centers. The data center has changed its name to the mainframe [15].

2.11.1.2 Internet

In recent times, there has been a rapid improvement in network performance. This is mainly due to the availability of wired, wireless, and 5th-generation mobile communication technologies that have made the Internet accessible to a larger audience. In addition, most

cities and towns now have hotspots for Internet access, and even transportation services such as air, train, and ship offer Wi-Fi connections through satellite-based or undersea fiber-optic cables. That means anyone can now connect to the Internet at any time and from anywhere. Widespread high-speed broadband Internet has made it possible for cloud computing to be used by many people [15].

2.11.1.3 Terminals

The PC is not the only device used for computing. Many other electronic devices like smartphones, tablets, MP3 players, PDAs, and notepads also require personal computing capabilities. However, synchronizing data among these devices is often time-consuming and can lead to errors. To address this issue, a solution is needed that enables people to access their data from anywhere and on any device [15].

2.12 SERVICES AND MODES OF CLOUD COMPUTING

Cloud services are divided into various types, including infrastructure, platforms, and applications. These services are provided and used instantly over the Internet, and we'll be talking about them in a more general context, as shown in Figure (2.1).

2.12.1 Software as a Service (SaaS)

An instance of the software operates on the service provider's systems and caters to multiple client corporations; this is known as multitenancy. SaaS (Software as a Service) is a type of service that delivers a complete application on demand. A well-known example of SaaS is Salesforce.com, along with other services like Google Apps, which provide essential business functions such as email. Before "cloud computing" became a popular term, Salesforce.com had a multi-tenant app that was among the first SaaS services. Today, Salesforce.com operates at various levels of the cloud, such as with Force.com, a recently released tool for creating custom applications[16], [17].

2.12.2 Platform as a Service (PaaS)

PaaS is a layer in the middle that gives you a place to work on your projects and a set of services. Most of the time, the services are delivered as a Xen image that includes a web

server, a Linux distribution, and a computer language like Perl or Ruby. PaaS services can be used at different steps of making and testing software, or they can be tailored to specific tasks, like managing content. Google App Engine is an example of a PaaS service because it runs apps on Google's servers. It's possible for PaaS services to be flexible, but they may have limits based on what the provider can do[16], [17].

2.12.3 Infrastructure as a Service (IaaS)

It's the base layer that makes storage and computing resources over the network consistent. With virtualization, storage systems, computers, routers, switches, and other systems can work together to handle different kinds of work. This can be anything from processing large groups of files at once to handling heavy traffic by adding more servers or storage space. One example of an IaaS service is Amazon Web Services EC2 and S3, which provide basic computing and storage functions. Another is Joyent's line of virtualized servers for running web apps written in different programming languages [16], [17].

2.13 ESSENTIAL CHARACTERISTICS

2.13.1 On Demand

Cloud computing stands out as a type of computing that can be accessed on demand without being a permanent fixture in an IT infrastructure. This offers a significant benefit for the use of cloud computing in the IT industry. With this model, users can access shared resources anytime, anywhere, using their connected devices, without having to wait for availability [11], [13], [16], [17], [18], [19].

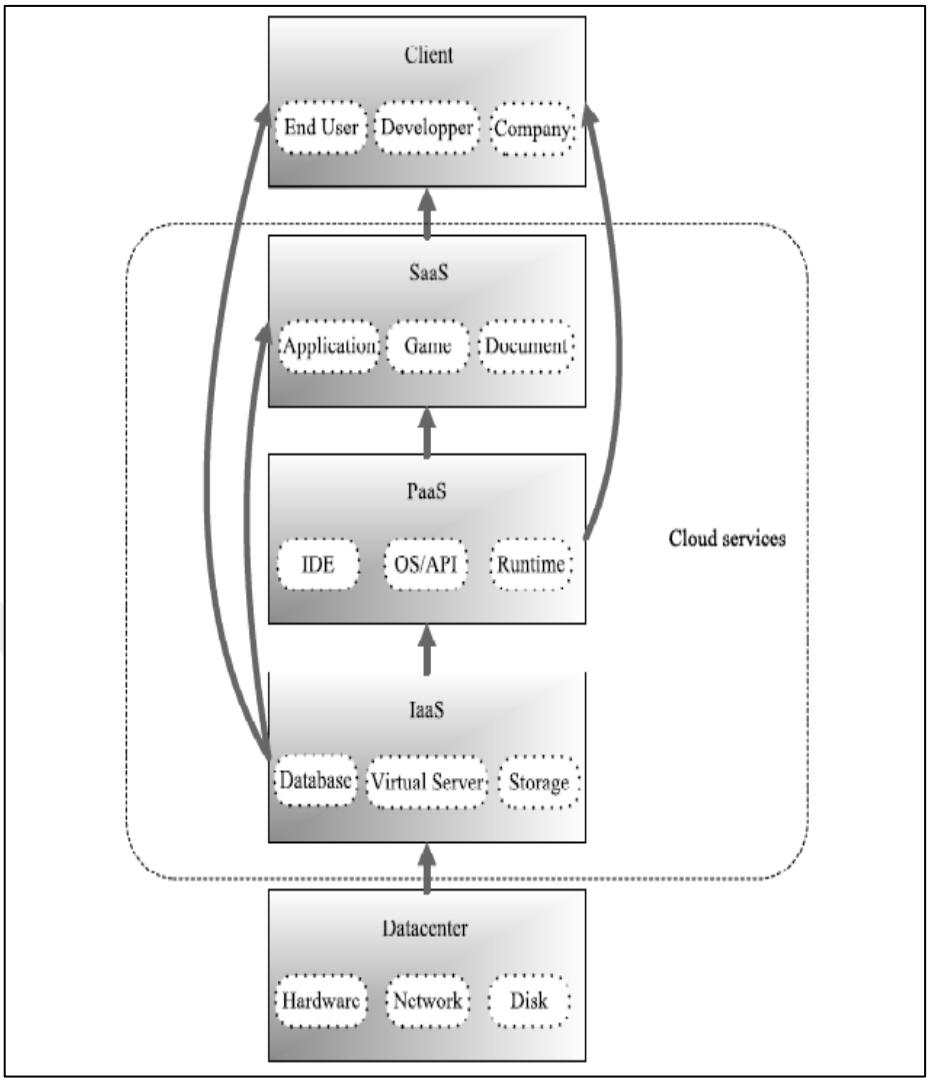


Figure 2.5: Cloud Services [12].

2.13.2 Resilience

In cloud computing, if a resource crashes, the user cannot be left with an idle service. To ensure uninterrupted access to cloud services, a resilient cloud service offering will automatically switch the user to another physical resource without causing any inconvenience to the user [11], [13], [16], [17], [18], [19].

2.13.3 Virtualization

Cloud computing heavily relies on virtualization as a fundamental aspect. This involves the virtualization of various layers such as hardware, operating systems, storage devices,

network resources, and more. Its main advantage lies in hiding the technical complexities from users, enhancing the autonomy of cloud services. Additionally, virtualization enables efficient configuration and utilization of physical resources, allowing multiple applications to run on the same machine. It also provides quick recovery and fault tolerance, ensuring uninterrupted service. Lastly, users can easily enter and exit the virtual environment without experiencing any service disruptions [11], [13], [16], [17], [18], [19].

2.13.4 Multi-Tenancy

Several individuals have the ability to utilize public cloud services concurrently, using the same infrastructure. Public cloud service providers may separate servers and storage units in a physical or virtual manner based on the specific needs of each user [11], [13], [16], [17], [18], [19].

2.13.5 Pay-Per-Use

Cloud computing Enables users to pay only for the cloud services they use, whether it's for a short-term need like CPU time, or a long-term need like storage or cloud-based vault services. In other words, users are charged based on their actual usage of the cloud services, rather than a fixed amount regardless of usage [11], [13], [16], [17], [18], [19].

2.13.6 Energy-Efficiency

The reason why clouds are effective in reducing the use of unused resources is because they can centrally manage and control energy consumption and carbon emissions, which can be difficult to regulate in situations where cooperation is lacking. Furthermore, it's important to consider green IT concerns at both the software and hardware levels [11], [13], [16], [17], [18], [19].

2.14 THE BENEFITS OF CLOUD COMPUTING

2.14.1 Reduce Run Time and Response Time

Cloud computing makes it possible to utilize a multitude of resources to accomplish a task much faster than using a single server. This is illustrated by the successful completion of a group project, where the run time was significantly reduced by leveraging the Cloud. For instance, when the New York Times had to convert 11 million articles and images to PDF format, their internal IT team projected that it would require seven weeks to finish the task. However, a developer was able to complete the task in just 24 hours using 100 Amazon Elastic Compute Cloud instances running Hadoop, at a cost of less than $300 (excluding data upload and storage fees) [20].

2.14.2 Minimize Infrastructure Risk

Organizations can mitigate the risk associated with purchasing physical devices by leveraging cloud computing. Cloud computing enables organizations to avoid the uncertainty of whether a new application will be successful or not, as well as the need to quickly deploy additional servers as workload demands increase. By leveraging cloud computing, the responsibility of scaling infrastructure to meet workload demands is shifted to the cloud provider. This helps to avoid over or under-provisioning, which can lead to wasted resources or performance issues. Another benefit of cloud computing is surge computing, which enables enterprises to handle workload spikes by leveraging public cloud resources. With cloud computing, organizations can better manage the application lifecycle, optimize resource utilization, and minimize infrastructure risks [21].

2.14.3 Increased Pace of Innovation

Cloud computing can speed up innovation by making it easier for new businesses to enter new markets. This lets startups release new goods quickly and cheaply. This makes it easier for small businesses to compete with bigger ones, whose deployment processes can take a lot longer in company data centers. Because there is more competition, new ideas come up faster. Open-source software is a big part of making these improvements possible [21].

2.14.4 Lower Cost of Entry

Cloud computing offers various benefits that allow us to enter markets at a minimal cost. The cost is kept under control as the infrastructure is rented rather than purchased, which means there is no need for significant capital investment. Additionally, cloud providers benefit from economies of scale, resulting in lower expenses for compute cycles and capacity, thereby reducing costs even further.

Furthermore, the development of applications is primarily done through assembly rather than programming, resulting in faster application development and reducing the time to market. This approach can potentially give companies that deploy applications in a cloud environment a competitive advantage[21].

2.15 CHALLENGES IN CLOUD-CENTRIC INTERNET OF THINGS (CIOT)

These section detail the challenges in Cloud-centric Internet of Things architecture development.

2.15.1 Bandwidth

The volume of data generated by IoT devices is seeing tremendous growth and is on track to exceed the available bandwidth in the near future. For example, a connected vehicle has the capability to provide a substantial volume of data each second, including details about its trajectory, velocity, state, driver's condition, and the surrounding surroundings. Furthermore, autonomous vehicles demand a greater amount of data due to their necessity to transmit live video feed. Consequently, depending exclusively on the Cloud to manage all this data becomes impractical [22].

2.15.2 Latency

There are issues with the Cloud's ability to meet the need of controlling end-to-end latency in a few tens of milliseconds. Due to the detrimental effects of CioT latency, industries including smart grids, self-driving cars, virtual and augmented reality, real-time finance, healthcare, and eldercare face enormous hurdles [22].

2.15.3 Uninterrupted

The far distance between the cloud and IoT devices that are located in the front-end may encounter problems caused by the inconsistent and intermittent network connection. This means that if a connected vehicle that is based on CIoT technology experiences a disconnection between itself and the Cloud, which occurs at an intermediate node, it won't be able to operate as intended [22].

2.15.4 Resource-constrained

In general, a lot of front-end devices don't have enough resources to perform complicated computations. This means that most CIoT (Connected Internet of Things) systems need these devices to send their data to the Cloud constantly. But, this approach isn't practical for devices that rely on batteries because sending data over the Internet can use up a lot of energy [22].

2.15.5 Security

Many constraints front-end devices lack the necessary resources to defend themselves against attacks, especially outdoor-based devices that rely on cloud updates for security. Attackers can target these devices and carry out malicious activities at the edge network, which is beyond the Cloud's control. This can result in the attacker damaging or controlling the front-end device and sending false data to the Cloud [22].

2.16 SCHEDULING OBJECTIVE

Typically, schedulers allocate tasks to resources based on a set of guidelines designed to meet certain goals. These rules rely on a formula that takes into account the desired goals to achieve optimal outcomes. In cloud computing, the goals of scheduling revolve around completing tasks swiftly, making efficient use of resources, and ensuring cost-effectiveness. However, many scheduling methods in cloud computing are fixed and don't adapt to changes in resource availability. In contrast, dynamic scheduling considers the current status of the system and can create effective schedules that enhance task completion speed and overall system efficiency.

2.17 RELATED WORK

2.17.1 Delay-aware algorithms

The article's authors discuss scheduling tasks in a complex environment with numerous sectors. To expedite task completion, they convert the issue into an integer program. Additionally, they devised a heuristic approach based on priority to address the problem [23].

This excerpt suggests implementing an improved version of the non-dominated sorting genetic algorithm II (NSGA-II) to decrease the time it takes to complete tasks and enhance the reliability of task execution [24].

Liu and his team have developed a cutting-edge task scheduling technique for fog networks called Dispersion-stable Task Scheduling (DATS). This approach utilizes stable matching theory to minimize service delays in various fog networks. The design of DATS takes into account both computation and communication delays, making it a decentralized algorithm [25].

Develop a scheduling system to manage IoT service requests using Integer Linear Programming (ILP), and propose a personalized genetic algorithm (GA) designed to reduce service time, including transmission, propagation, queueing delays, and processing time [26].

This passage describes a research study on task offloading in fog networks enabled by software-defined networking (SDN). The main goal of the study is to reduce both the latency of IoT tasks and the energy consumption of IoT devices. Researchers tackle this issue using an Integer Linear Programming (ILP) model and solve it with a greedy-heuristic method [27].

To fulfill the needs of applications that demand minimal latency, the authors propose the utilization of a method that is founded on fuzzy logic for the purpose of arranging the tasks that have been offloaded in edge-cloud environments [28].

The authors in [29] Discuss a method for implementing load-balancing in fog computing to enhance the performance of healthcare applications. They suggest using a probabilistic

approach to decide whether a task should be handled by the node in the fog or sent to a fog master for additional processing. The fog master effectively assigns the job to the suitable fog node for processing, utilizing Ant Colony Optimization. Simulation tests have shown that the proposed method significantly reduces delay compared to not offloading at all or offloading all tasks. As per the expert in artificial intelligence, it is indeed feasible to adjust the threshold number for task transfers.

The authors in [30] they have used the edge data center for real-time services that need to avoid latency and the Ford-Fulkerson algorithm and a priority-based queue to balance load and schedule tasks in fog devices, which are network-based systems that quickly look at huge amounts of data, or "big data."

2.17.2 Energy-efficient Algorithms

This chapter outlines a suggested analytical framework for evaluating the energy efficiency of fog networks that consist of homogeneous fog nodes. The authors also present a work scheduling system dubbed MEETS, which aims to maximize energy efficiency based on their model [31].

The research focused on studying assigning tasks and energy scheduling challenges in green edge computing, with the goal of reducing brown energy consumption. At first, the researchers approached the problem as a Mixed-Integer Linear Programming (MILP) problem. However, they later developed a heuristic algorithm based on relaxation to effectively tackle the issue [32].

To tackle the issue of assigning tasks in high-performance fog computing, the focus has been on reducing energy usage while ensuring that the total execution time, or makespan, remains within constraints. An innovative heuristic algorithm has been devised to address this optimization model [33].

The author in [34] researche introduced a heuristic algorithm known as the Priority-aware Genetic Algorithm (PGA) to optimize the makespan and energy usage in the fog-cloud environment.

2.17.3 Joint Delay-Aware and Energy-Efficient Algorithms

The authors introduce a model called MASM, which stands for "multiple algorithm service model". This model allows virtual machines to use different AI algorithms to process the same type of task. They also create an algorithm called tide ebb algorithm (TEA) to produce strong solutions for their MASM model [35]

The authors in [36] By leveraging a cloud-fog computing architecture and a task scheduling algorithm, this study tackles the scheduling challenges in smart manufacturing lines by prioritizing jobs according to their importance. The proposed method, named the HMA algorithm, improves upon the techniques used in monarch butterfly and ant colony optimization algorithms. It has shown superior performance in terms of task completion time, power consumption, and efficiency compared to previous approaches. The researchers aimed to reduce latency and power consumption for time-sensitive operations. Through computer simulations, it has been demonstrated that the proposed method can complete tasks efficiently and effortlessly.

The authors[37] suggest a good way to deal with the issue of service timing in fog-cloud computing systems. The policy's goal is to make sure that IoT requests are answered quickly while still saving cloud service providers energy. In order to do this, the writers divide IoT services into two groups: normal and critical. For important services, they suggest a way called MinRes that aims to cut down on response time as much as possible. For normal services, they offer a method called MinEng that aims to make Fog Nodes (FNs) use less energy.

The study[38] focuses on using deep reinforcement learning (DRL) to optimize task completion before their deadlines while also reducing the energy consumption of edge servers.

2.18 SUMMARY

Internet of Things (IoT), fog computing, and cloud computing are initially defined in this chapter. We attempt to improve a few examples and provide an impartial and general description of IoT, fog computing, and cloud computing, despite the fact that there is

substantial debate about their meaning. The system architecture, deployment model, and critical features are all described in these definitions, which go beyond simply describing an overarching notion. As ever-changing paradigms, the Internet of Things (IoT), fog computing, and cloud computing incorporate a wide range of current technology. To put it simply, fog-cloud computing is a model for providing services that allows for the direct delivery of software, platforms, infrastructure, data, and hardware to end users. Presented here are the features of the service from a technical, qualitative, and financial perspective. The groundwork for future progress is laid by the present endeavors. Several difficulties with middleware, programming models, resource management, and business models are brought to light after reviewing current commercial products and research initiatives. We are motivated to address these limitations in fog-cloud computing through our future research. Both the micro and macro views of the resource management problem are covered in the subsequent chapters. Allocation of resources and scheduling of tasks are two areas that receive special attention.

3. SYSTEM ARCHITECTURE AND OUR PROPOSED ALGORITHM

3.1 INTRODUCTION

Three emerging technologies are having a profound impact: the internet of things (IoT), fog computing, and cloud computing. A wide variety of definitions have been proposed for them. There are famous definitions for all of them:

i. NIST which defines Cloud Computing as follows[7], [14]: “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”.

ii. Cisco defines Fog Computing as[3] “a paradigm that extends Cloud computing and services to the edge of the network. Similar to Cloud, Fog provides data, compute, storage, and application services to end-users. The distinguishing Fog characteristics are its proximity to end-users, its dense geographical distribution, and its support for mobility”.

iii. IBM define[39] “The Internet of Things (IoT) refers to a network of physical devices, vehicles, appliances and other physical objects that are embedded with sensors, software and network connectivity that allows them to collect and share data. These devices — also known as “smart objects” — can range from simple “smart home” devices like smart thermostats, to wearables like smartwatches and RFID-enabled clothing, to complex industrial machinery and transportation systems. Technologists are even envisioning entire “smart cities” predicated on IoT technologies.”

In every computer system, one of the most important tasks is scheduling jobs. Therefore, one of the primary ways to improve the efficiency of fog-cloud settings by decreasing the makespan and boosting resource utilization is through work scheduling, same as in other computing environments[30]. It is also the goal of certain fog-cloud job scheduling efforts to minimize power consumption[40] .

3.2 SYSTEM MODEL

The architecture of the Internet of Things (IoT) and fog-cloud computing is described in this part, along with the job characteristics that are required to process this system.

3.2.1 Architecture

Figure 3.1 shows how the (IoT-Fog-Cloud) setup is put together. IoT devices are at the base of the system stack and are where it all starts. In this group are things like cell phones, UAVs, medical gadgets, industrial gadgets, smart houses [5] [5], [41]etc. At this stage, data is transformed into jobs that are then forwarded to the subsequent Various tiers of fog or cloud computing. Fog computing is an intermediary concept that bridges the gap between the Internet of Things (IoT) and the cloud.. The fog could be thought of as a small part of the cloud. It's a bit like the cloud in some ways, but it doesn't have as much computer power [42]. This is the cloud, which has a lot more space and power than the fog. To keep the network from being slow, tasks with short due dates will be sent to the fog. However, we can use the cloud to send jobs with longer due dates and make good use of how the resources can spread the work. It is near the fog and is known as a fog trader [43][44], it’s installed in the fog layer. The fog controller is a component of three important sections:

Task Director (TD).

Resource Monitoring (RM).

Task Scheduler (TS).

The task director is the one who receives the task and determines what its characteristics are. It is the responsibility of the resource monitoring job to collect data regarding the resources that are within reach. The task director and resource monitoring job gather data on the task and resources, which is subsequently transmitted to the task scheduler via the job scheduler program.. Once the task planner has gathered every necessary data from the TD and RM, they will proceed to allocate the work to the suitable resource. It is important to return all completed work back to the sender as the final step [45].

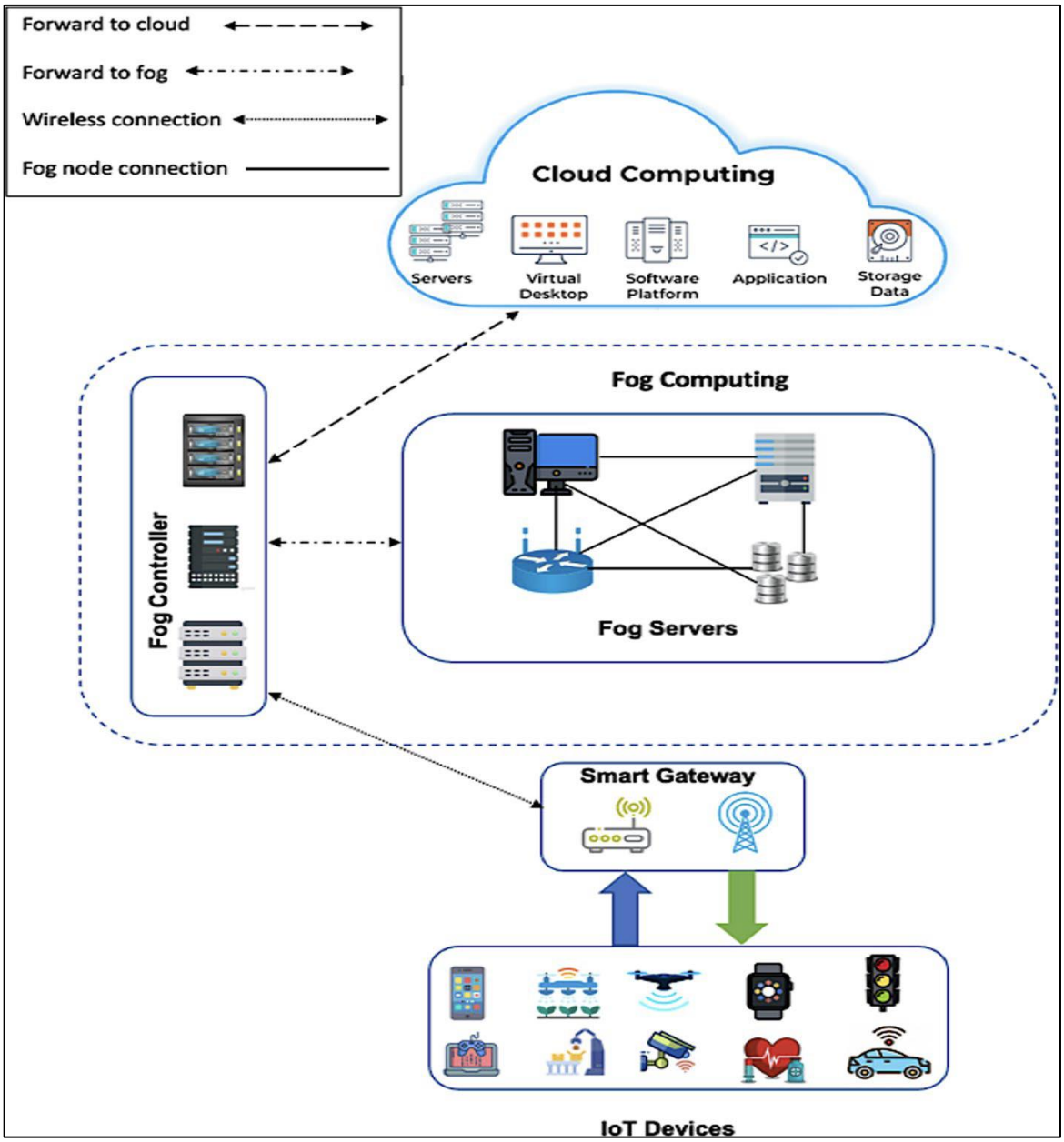


Figure 3.1: System Architecture [45].

3.3 PROBLEM DEFINITIONS

The majority of conventional algorithms used for fog-cloud scheduling send the work to any cloud or fog resource as soon as it arrives without considering its bandwidth or deadline. This leads to some problems, such as unbalanced resource utilization and high latency time in execution tasks, in case most IoT applications need real-time services. today the huge number of IoT devices need a numerous of resources which causes to high energy consumptions. By 2022, there is an anticipated surge in global energy consumption, with a projected increase of 7%. [46]. In this paper, we will be discussing algorithms that focus on Delay-aware, Energy-efficient, and Collective Delay-aware and Energy-efficient approaches. We will compare and analyze these algorithms to determine their effectiveness.Since all tasks within the fog-cloud share the same traits. The following is a mathematical description of the task's characteristics:

3.3.1 Execution Time

Is that time that the task Ti take it to execute on the resource Rj, we can calculate by using equation 3.1:

$$EX_{(i,j)} = \frac{t_{s(i,j)}}{r_{p(i,j)}} \tag{3.1}$$

The size of the job that needs to be executed is represented by $t_{s(i,j)}$, while The calculation of resource j power is denoted by $r_{p(i,j)}$.

$\mathrm{R} = \text{ Collection of resources}, |\ \mathrm{R}\ | = \mathrm{m}$
$\mathrm{T} = \text{ Collection of tasks}, |\ \mathrm{T}\ | = \mathrm{n}$
$\mathrm{j} = \text{ Index of resources}, \mathrm{j} \in \mathrm{R}$
$\mathrm{i} = \text{ Index of tasks}, \mathrm{i} \in \mathrm{T}$

3.3.2 Completion Time

Does the resource Rj need that much time to finish the task? Ti, equation (3.2) figures out how long it will take to finish:

$$CT_{ij} = EX_{(i,j)} + R_j \quad (3.2)$$

The ready time of a resource is represented by Rj; when resources in a cloud or fog are not yet ready to execute a task, the job must wait for the resource to become ready; this waiting time is added to the execution time to determine the total time it takes to complete the operation.

3.3.3 Makespan

It is the duration between the beginning and completion of a series of tasks or assignments. You can use Equation (3.3) to figure it out:

$$Makespan = max(CT_i)_{t_i \in MetaTask} \quad (3.3)$$

In general, scheduling is better with the shorter the makespan.

3.3.4 Difference Time

We use equation to find The temporal disparity between the two instances due date and the completion date so that we can decide where to process the leftover tasks on the metatask (3.4):

$$\nabla T = D_t - C_t \quad (3.4)$$

When evaluating the time frame for task ti deadline and completion on a specific resource rj, it is important to take into account the D_t and C_t values.

3.3.5 Task Probability

The following equation can be used to express the chance that the m^{th} ant will upload the job (Ti) to process on a fog or cloud device (Rj):

$$P_{ij}^{m}(t) = \frac{(\tau_{ij})^{\alpha} \ (\eta_{ij})^{\beta}}{\sum_{k=1}^{n} (\tau_{ik})^{\alpha} \ (\eta_{ik})^{\beta}} \quad (3.5)$$

Where:

$\tau_{ij} = \ pheromone\ trail\ on\ node\ j\ .$

$\eta_{ij} = represent\ the\ computing\ power\ of\ the\ resource\ on\ fog\ or\ cloud\ .$

Can be calculated using equation 3.6:

$$\eta_{ij} = \frac{1}{\nabla T} \tag{3.6}$$

$\alpha = \ pheromone\ trail\ persistence.$

β

$= \ heuristic\ weight\ of\ difference\ time.$

$k =$ all possible next nodes.

3.3.6 Pheromone Update

Ant Colony Optimization (ACO) techniques rely on ant communication via pheromones to solve problems. Two types of pheromones are utilized in ACO codes: global and local.

3.3.6.1 Global pheromones

Are the hormones that are left behind by every single ant, and they are utilized in order to gather information about the entire ant population. To determine the pheromones that are present on a global scale, the following phrase might be utilized:

$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \ \Delta\tau_{ij} \tag{3.7}$$

$\boldsymbol{\tau_{ij}} = pheromone\ value\ on\ the\ edge$

$between\ node\ i\ and\ node\ j$. Can be calculated using equation 3.8:

$$\Delta\tau_{ij} = \ {}^{1}/_{CT_{ij}} \tag{3.8}$$

$\boldsymbol{\rho} = \ pheromone\ evaporation\ rate,$

a is $value\ between\ 0\ and\ 1.$

$\Delta\boldsymbol{\tau_{ij}} = route\ of\ newly\ produced$

$pheromones\ from\ node\ I\ to\ node\ j.$

3.3.6.2 Local pheromones

Are chemicals that are only left behind by one ant while it is exploring. They help the algorithm's local search and give information about the exact path an ant took. The following expression can be used to figure out the local pheromones:

$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \tau_0 \tag{3.9}$$

The pheromone evaporation rate, denoted by ρ, is a predetermined quantity, and the starting pheromone concentration, denoted by τ_0, is also predetermined.

3.3.7 Calculate the Priority Depending on the Task Cost

At the beginning of the process, we use Equation (10) to determine the cost of each action for each resource that is freely available

$$Cs\ (t_{i_{R_j}}) = NI(t_i)\ \times\ CI(t_{i_{R_j}}) + TB(t_i)\ \times\ CPBW(t_{i_{R_j}}) \tag{3.10}$$

Where:

$NI(t_i)$ is the number of job instructions (ti),

$CI\left(t_{i_{R_j}}\right)$ is the expense of ti on resource per instruction Rj,

$TB(t_i)$ is the data for a task (ti),

and $CPBW\left(t_{i_{R_j}}\right)$ is the cost of performing the job per bandwidth Ti on resource Rj.

The profit is then determined for each job using Equation (3.11) for the resource with the most significant cost.

$$Profit\ =\ CsU(T_i) -\ CO\ \left(T_{i_{R_{max}}}\right) \tag{3.11}$$

Where:

$CsU(T_i)$ is the cost paid by the user to run task Ti,

$CO\left(T_{i_{R_{max}}}\right)$ is the actual cost of running task Ti on the resource Rmax.

(Rmax is the resource that has the highest cost).

Following this, the jobs are arranged in the appropriate order and assigned to the appropriate queues. Regarding our project, we established three tiers of queues, which we referred to as Q1, Q2, and Q3, each with the same ranges of priorities. What this indicates is that the most recent jobs are distributed uniformly across these queues in terms of the broad priority range that they fall under. As a result of this, We can calculate the maximum size of the queue range (Q_r) of priority for each queue by making use of Equation (3.12). This is because there are a total of four queues.

$$Q_r = Pr_{max}(T_i)/3 \tag{3.12}$$

where $Pr_{max}(T_i)$ can be paraphrased in the text as follows:

PCA distributes one-third of the maximum reward obtained after finishing task Ti equally among the three queues, taking into account their presence. The technique utilizes three queues, each representing distinct priority regions: the High queue, indicating the most critical priority sector; the Middle queue, indicating a region of intermediate importance; and the Low queue, indicating the least essential priority region. As previously stated, this enables us to carefully examine:

IF ($Pr(T_i) \leq Q_r$), Next, the task Ti is included in the Q3. (3.13)

F ($Q_r < Pr) \leq 2Q_r$) Next, the task Ti is included in the Q2.

IF ($2Q_r < Pr(T_i) \leq 3Q_r$) Next, the task Ti is included in the Q1.

3.4 GWO ALGORITHM

The gray wolf optimizer (GWO) is an algorithm for metaheuristic optimization that draws inspiration from Understanding the framework of society and hunting patterns techniques employed by gray wolves. In their natural habitat, gray wolves exhibit a structured social order, led by an alpha wolf with beta and delta wolves as subordinates. The alpha leader holds the most elevated position in the hierarchy. This pack of wolves cooperates as a unified unit to efficiently chase and capture prey. The objective of the GWO technique is to replicate the cooperative and communicative actions of wolves, while also addressing optimization problems. The various strategies for resolving the problem are depicted by a group of wolves, and the locations of the wolves within the solution space are intended to represent their degree of effectiveness. Currently, The alpha wolf represents the most optimal and effective strategy, while the beta and delta wolves are also viable alternatives to contemplate [47].
In the next paragraphs, you will find an explanation Regarding the field of mathematics formulation of the GWO (gray wolf optimization) algorithm.

3.4.1 Encircling Prey

In this stage, the gray wolves will encircle their prey in order to protect it. In terms of mathematics, this behavior can be represented by the equations (3.13) and (3.14) that are provided below:

$$\vec{D} = \vec{C} \,.\, \overrightarrow{X_p}(t) - \overrightarrow{X_w}(t) \tag{3.14}$$

$$\overrightarrow{X_w}(t+1) = \overrightarrow{X_p}(t) - \vec{A} \,.\, \vec{D} \tag{3.15}$$

The vectors $\vec{A}$ and $\vec{C}$ represent the coefficients, the vectors $X_p$show the position of the prey, and the vector $\overrightarrow{X_w}$ indicates the position of a gray wolf. The symbol t denotes the current iteration. Furthermore, the values of $\vec{A}$ and $\vec{C}$ are determined by employing the formulas that are presented in Equations (3.15) and (3.16).

$$\vec{A} = 2\vec{a} \,.\, \overrightarrow{r_1} - \overrightarrow{a} \tag{3.15}$$

$$\vec{C} = 2 \,.\, \overrightarrow{r_2} \tag{3.16}$$

where r_1 and r_2 Illustrate a collection of arbitrary vectors. in [0, 1], and $\vec{a}$ is linearly decreased from 2 to 0.

3.4.2 Hunting

Currently, the alpha wolf, who holds the dominant position, together with the beta and delta wolves, who serve as counselors, possess extensive knowledge regarding the whereabouts of the prey.. Because of this, the other wolves have to move based on where the smartest agent is, which is shown by mathematical formulae (3.17) and (3.18):

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}.\overrightarrow{X_\alpha} - \vec{X}\right| \quad \overrightarrow{D_\beta} = \left|\overrightarrow{C_1}.\overrightarrow{X_\beta} - \vec{X}\right| \quad \overrightarrow{D_\delta} = \left|\overrightarrow{C_1}.\overrightarrow{X_\delta} - \vec{X}\right| \tag{3.17}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}.\overrightarrow{D_\alpha} \quad \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.\overrightarrow{D_\beta} \quad \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.\overrightarrow{D_\delta}$$

$$\overrightarrow{X_w}(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{3.18}$$

3.4.3 Attacking Prey

Wolf hunting involves two main parts: exploitation and exploration. Exploration involves chasing and looking for food. Wolves' ability to chase and fight their prey shows how well they can catch it, which could be thought of as their exploitation skill. Wolves can reach global optima with this exploitation skill, which makes them better hunters. In this process, the number of parameter A is very important. If the value of |A| is less than 1, the gray wolves have to go after their present prey very hard. If, on the other hand, |A| is greater than 1, the gray wolves have to leave their current food and look for other possible prey. The pseudocode of the GWO algorithm [48] is depicted in algorithm1.

3.4.4 Equations Power Consumption Models

A server's energy use can be broken down into two groups: idle and active. When the server is not doing anything, it uses power. This is called "idle consumption." On the other hand, "dynamic consumption" is the amount of power used when the server is doing work. We can

use Equations (3.19) and (3.20) to figure out how much power is used when nothing is being done and when something is being done, respectively [48][49].

Algorithm 1: Pseudocode of GWO Algorithm [47]

"Initialize Population
Initialize a, A, and C
Calculate the fitness of each search agent.
X_α = the best search agent
X_β = the second-best search agent
X_δ = the third-best search agent
While (t < maximum number of iterations)
For each search agent
Update the position of the current search agent by.
End for
Update a, A, and C
Calculate the fitness of the current search agent.
Update Best Solution.
Update X_α, X_β and X_δ
t= t + 1
End While
Return X_α"

$$P_{C_{idle}} = \sum_{i=1}^{n} P_{r_i} \tag{3.19}$$

where Pri represents the reduced power consumption of Ri.

$$P_{C_{dynamic}} = \sum_{i=1}^{n} P_i' \tag{3.20}$$

where Pi denotes the dynamic power consumption of the Ri having a utilization (load) of Li.

3.4.5 Total power Consumption

The overall power consumption of a multi-core CPU is determined by Equation (3.21):

$$P_c = P_{C_{idle}} + P_{C_{dynamic}} \tag{3.21}$$

3.4.6 Fitness Function

This paper looks at two parameters: the first is the shortest time it takes to finish, and the second is the amount of power it uses. Equation (3.22) can be used to find the fitness function (FF).

$$FF = CT_{ij} + P_c \tag{3.22}$$

3.5 BANDWIDTH VS. LATENCY

The efficiency of a communication network is commonly assessed based on broadband and latency, which are critical factors to consider. These factors can be further categorized into bandwidth and latency. Bandwidth refers to the maximum data capacity that can be transmitted through a network within a specific time frame, typically measured in bits per second (bps). With improved bandwidth, users can experience faster download and upload speeds, resulting in improved data transmission. Latency, measured in milliseconds (ms), indicates the amount of time that it requires for an information packet to travel from one point to another within a network after being sent. Minimizing latency is essential for time-sensitive applications such as Voice over Internet Protocol (VoIP), video conferencing, and online gaming. It ensures minimal transmission delay, enabling seamless real-time communication and smooth gameplay [50], [51].

3.6 OUR PROPOSED ALGORITHMS

Here, we have introduced two algorithms:

The Bandwidth Deadline Algorithm, or BDA for short, is the initial algorithm used. The Fog-Cloud environment provides a solution to the problem of job scheduling. As a heuristic algorithm, it takes into account both the tasks and resources it is provided with. In addition, when arranging the process, it considers both the job priority and the due date of each task. Subsequently, it allocates the work to the suitable resource with the aim of identifying the most efficient solution to an optimization problem. In the subsequent section, we shall elaborate on the methodology that we put up.

3.6.1 Bandwidth Deadline Algorithm (BDA)

One of the primary objectives of the algorithm that we have proposed is to schedule Internet of Things jobs according to their due dates and priorities, and then to assign those tasks to the appropriate resource in a fog or cloud environment. Due to the fact that tasks that require a great deal of processing power but not a great deal of bandwidth perform better in a cloud environment than in a fog environment [52], this strategy sends tasks that require a high bandwidth to the fog environment and tasks that require a low bandwidth to the cloud environment. Reducing the latency of the system is the objective here. in order to guarantee that the approach operates as efficiently and effectively as feasible in real time. In our proposed method, there are four stages to be completed. This is how we will explain it to you

i. Step 1: Determine which of the tasks in the metatask has the highest bandwidth value, and then divide that value by three using equation (3.23).

$$LB = \lceil BW_{max}(T_i)/3 \rceil \tag{3.23}$$

The LB represents the up-rounded length bandwidth, whereas the BW_{max} represents the maximum bandwidth that is available for the tasks that are part of the meta task.

Assign the work to the appropriate lists determined by the result, taking into account the bandwidth that he has available. C$List$, F$List$, and FC$List$. C$List$ are the assignments that

are separated into three distinct groupings. It is the cloud list that is denoted by C, the fog list is denoted by F, and the fog-cloud list is denoted by FC.

ii. Step 2: The jobs should be stored in the lists in ascending order, with the order determined by the deadline.

iii. Step 3: Send the task to the fog computing which is be in the fog list and send the task to cloud comouting which is be in the cloud list, other tasks send to the fog-cloud list after use the ACO algorithm.

iv. Step 4: Return all steps above.

Algorithm 1

Step 1- FOR all available tasks DO

calculate the LB using Eq (9)

IF ($T_b \leq$ LB)

C$List \leftarrow$ Ti

Else IF ($2LB <$ T$_b$ $\leq$ 3LB)

FC$List \leftarrow$ Ti

Else

F$List \leftarrow$ Ti

END FOR

Step 2- Sort the tasks in each list according to the deadlines of tasks in ascending order

Step 3- if (CList or FList) = Ø and (FCList) ≠ Ø then

Call ACOE

Else

If (CList or FList) ≠ Ø and (FCList) = Ø then

Call MF-MC

Else

Call MF-MC and ACOM (Ti, Clist – Flist)

End IF

Step 4- return all steps above.

Where T_b is the task bandwidth.

3.6.2 Algorithm 2

The FM-CM algorithm, or Fog Max – Cloud Min, is a way for remote systems to schedule tasks and share resources. There are several steps to how the program works:

Step 1: For every job in the (Clist and Flist) is assigned a finish time, which is denoted by the letter Ti. In addition, this time is determined for each individual task on each resource (Rj). What time is it expected to be finished? You will be informed of the amount of time that the resource will require to complete the task.

Step 2: The one that has the earliest due date is selected from the list of tasks that are available. When it comes to the task at hand, the target is a predetermined length of time that must be completed. The work that has been selected is then assigned to the resource that is capable of completing it in the shortest amount of time.

Step 3: In the event that the work is stored in either the Flist or the Clist, it is then transmitted to either the fog or the cloud in order to be processed.

Step 4: Once the work has been allocated, it is taken off the task list, and the estimated time needed to complete each resource is modified correspondingly.

Step 5: This process is repeated until all of the jobs have been assigned to the resources, at which point the algorithm moves on to the next stage.

To sum up, the FM-CM algorithm is a way to schedule tasks and divide up resources. It does this by giving tasks to resources based on their minimum due date and finishing time. The algorith aim is to make sure that targets are met while also reducing the time it takes to complete tasks.

Algorithm 2 (FM-CM algorithm)

Step 1- FOR all tasks T_i in the lists (Clist and Flist) DO
FOR all available resources R_j DO
Calculate the competition time using Eq 2

END FOR
END FOR

Step 2- Get the task T_i which has the minimum *deadline* in the lists and allocate it to the resource which has the minimum completion time.

Step 3- IF task T_i in the Flist assign it to the fog
Else
Assign it to the cloud.
END IF.

Step 4-Remove task T_i from the list set and update r_j for the selected R_j and Update CT_{ij} for all j.

Step 5- return all steps above.

3.6.3 Algorithm 3

How to Improve Ant Colonies ACO is a meta-heuristic program that is based on how ant colonies work.[53], [54]. It is utilized for resolving decision-making and optimization problems. [55]. We changed the (ACO) method to deal with problems that come up when you try to schedule big tasks. This made it easier to figure out the best way to finish tasks. This code shows how to use the improve ACO method in method 3.

Algorithm 3 Improved Ant Colony Optimization

Input: ACO parameters and child node list

Step 1: Initialize all ACO parameters.

Step 2: For each node in the FClist, calculate (6)

Step 3: For each ant, repeat the following:

A. For each node in the FClist, perform the following:

i. Calculate $\tau ij(t)$ using equation (8)

ii. Calculate the probability using equation (5)

iii. Add the selected node to a list.

iv. Update local pheromone using equation (9)

B. Update global pheromone using equation (7)

Step 4: Choose the optimal path.

Output: Best fog node for offloading the load

End.

3.6.4 The Second Algorithm is

The present energy consumption of various data centers accounts for 2% of the total world power usage, which is equivalent to 416.2 terawatt hours. It is projected that this consumption will significantly rise in the future. Projections indicate that by 2022, renewable energy will account for 7% of global energy consumption, reflecting a substantial increase in demand [56]. We propose an enhanced technique that incorporates the cost and energy considerations of task processing in addition to the Makespan in the PCA algorithm. The algorithm commences by employing the PCA algorithm, as illustrated in the pseudocode. The number 5 is used in algorithm2 [57]. The GWO algorithm is invoked during step four of the method, as depicted in algorithm3, which represents the pseudocode of the PC-GWO algorithm.

Algorithm 2: Pseudocode of PCA Algorithm [57]

"1- FOR all available tasks DO

Calculate the priority of each task.

END FOR

2- Sort the tasks according to the priorities in the scheduler's queues.

3- FOR all tasks Ti in meta-task DO

FOR all resources Rj DO

Calculate the completion time:

$(CT)\_ij = (EC)_{ij}+r_j$

END FOR

END FOR

4-Find task Tk which has the highest Priority and assign this task Tk to the resource which has the minimum completion time.

5-Remove task Tk from Meta-tasks set and update rj for the selected Rj and Update CTijfor all j.

6-IF the waiting time of any task in the lower queues has exceeded the threshold THEN

Move this/these tasks to the next upper queue.

END IF.

7-IF there is a new task has arrived THEN Calculate its priority and sort it in the end of appropriate queue and repeat the above steps

END IF"

Algorithm 3: Pseudocode of PC-GWO Algorithm

Step 1-FOR all tasks in meta task DO
Calculate the priority of each task.
END FOR

Step 2-insert the tasks according to the priorities in the scheduler's queues.

Step 3-FOR all tasks Ti in meta-task DO
FOR all resources Rj DO
Calculate the completion time:
(CT)ij = (EC)ij+rj
END FOR
END FOR

Step 4-Call the GWO with the fitness function using equation (17)

Step 5-Strip task T_k from Meta-tasks set and update rj for the selected Rj and update CTij for all j.

Step 6-IF the waiting time for any task in the lower queues has gone over the threshold, THEN shift this/these tasks to the next upper queue.
END IF.

Step 7-IF there is a new task has arrived THEN Calculate its priority and place it at the end of the right queue and repeat the steps above.
END IF

Where:

Proceed 1: Assess the significance of each job within the meta-task by utilizing the Equation (5).

Proceed 2: involves arranging the items in the queue based on their priority, as determined by Equation (6).

Proceed 3: Determine the total time required to complete wholly jobs within the meta task in evey resource that is currently available.

Proceed 4: Invoke the GWO algorithm to select the optimal solutions for executing the task, utilizing the fitness function defined by Equation (17).

Proceed 5: Delete the job as it was overseen in the main task, modify the duration for resource j to get prepared, and update the completion time for all resources.

Proceed 6: In this time, we concider the job time spend it in the queue if it will be in the deadline. By shifting the task to the next level, we take into account its anticipated arrival time.

Proceed 7: If a new task is introduced, the algorithm will repeat all the preceding steps.

3.7 SUMMARY

In this chapter, we first defined all of IoT, fog computing, and cloud computing. Then we explain the system hierarchy of IoT, fog computing, and cloud computing with the fog controller and how it works. After that, we define the problem definitions of the tasks with a description of the mathematical representation for all algorithms that we used to improve our proposed work. At the end of the chapter, we explained our proposed algorithms and displayed his specifications in detail.

4. PERFORMANCE EVALUATION AND RESULT

4.1 INTRODUCTION

The scheduling problem is defined in a previous chapter as scheduling a set of jobs on available processors, and our proposed algorithms are defined too. This chapter presents the results of employing several application samples as the output of the proposed scheduling method. The results are derived from the mathematical expressions discussed in the previous chapter.

4.2 PERFORMANCE EVALUATION

4.2.1 Bandwidth Deadline Algorithm (BDA)

This section evaluates the performance of the suggested algorithms by considering three criteria: Makespan, resource consumption, and the number of completed jobs within the specified deadline. It assesses the effectiveness of these algorithms when compared to other algorithms currently in use. The simulation runs on a machine powered by an Apple M1 CPU and 8GB of RAM, utilizing the Java programming language. Here are three scenarios that have been created for experimental testing:

i. Scenario A: In this scenario, there are more jobs that require a large amount of bandwidth compared to tasks that require a moderate or low amount of bandwidth.
ii. Scenario B: In this situation, there are more jobs with a moderate level of bandwidth compared to tasks with high and low levels of bandwidth.
iii. Scenario C: In this scenario, there is a higher proportion of jobs that require less bandwidth compared to tasks that require medium or high bandwidth.

The simulation and algorithm Table 4.1 displays three parameters. In order to demonstrate the effectiveness of our proposed algorithms, we compare them with the following benchmarks.

a. First Come First Serve (FCFS): The objective of this task scheduling method is to achieve a balanced distribution of the workload across the computer nodes in the environment. The fog controller prioritizes and schedules the first Internet of Things

(IoT) job based on the First-Come-First-Serve (FCFS) criteria. Tasks are executed by either a fog node or a cloud node, selected randomly [58].

b. Earliest Deadline First (EDF): This scheduling system takes into account the constraints of schedules for projects and gives priority to tasks with shorter deadlines. The selection process for nodes located in fog or cloud node locations is determined using a random mechanism, much like the First-Come-First-Serve (FCFS) method [59].

c. Max-Min Algorithm: Unlike the (Min-Min) algorithm, This strategy gives priority to assigning the task with the longest completion time over the resource with the shortest duration of execution. The Max-Min method aims to optimize the allocation of resources by assigning jobs with longer completion times to resources with shorter execution times [60].

Table 4.1: Simulation and Algorithm3 Parameters.

Parameter	Value
The number of tasks.	[30,60,90,120,150]
Cloud nodes number.	5
Fog nodes number.	[10,20,30]
processing rate of the cloud nodes.	[5000,15000] MIPS
processing rate of fog nodes.	[500,2000] MIPS
cloud nodes Bandwidth	[50,100] MBPS
Fog nodes Bandwidth	[200,300] MBPS
latency from fog to cloud (ms)	100
latency from device to fog layers (ms)	20
Deadlines of tasks (ms)	[50-150]
$\boldsymbol{\alpha}$	1.0
$\boldsymbol{\beta}$	3.0
$\boldsymbol{\rho}$	0.5
$\boldsymbol{\tau_0}$	1.0

4.2.2 PC-GWO Algorithm

The aim of this section is to explore the effectiveness of the proposed algorithms by comparing them with other existing ones. The assessment revolves around three crucial metrics: Makespan, cost, and energy consumption. To carry out the suggested approach, a detailed simulation is essential. This involves utilizing various programming tools to run multiple scenarios and thoroughly analyze the outcomes.

The chapter commences with a comprehensive exploration of the execution of the suggested resolution., known as PC-GWO. We crafted a basic version of the algorithm using Java, running the simulation on a personal computer powered by a Mac Apple M1 chip, with 8 GB of RAM, operating on macOS Ventura 13.2.

For the experimental testing, we designed three distinct scenarios, each with varying levels of importance. This approach aimed to test the system's performance across different situations and determine its susceptibility to random initial conditions.

i. Scenario 1: There are multiple tasks that have high priority, along with a smaller number of jobs that have medium and low importance.
ii. Scenario 2: In addition to a few high- and low-priority tasks, there are a lot of medium-priority tasks.
iii. Scenario 3: In addition to a small number of high and medium priority jobs, there are a lot of low priority chores.

Table 4.2 displays the properties of clouds, while Table 4.3 presents the characteristics of fog. The simulation and algorithm of GWO parameters can be found in Table 4.4.

Table 4.2: Cloud Parameters.

Parameter	**Value**
Cloud nodes number	5
Processing rate of the cloud nodes	[20,000, 40,000] MIPS
Cloud nodes bandwidth	[1500, 4000] MBPS
Latency from fog to cloud (ms)	100
Power consumption in idle mode	(65–75) % of the active dynamic mode
Power consumption in dynamic mode	(100–200) Watts

Table 4.3: Fog Parameters.

Parameter	**Value**
Fog nodes number	(10, 20, 30)
Processing rate of the fog nodes	[5000, 20,000] MIPS
Fog nodes bandwidth	[5000, 10,000] MBPS
Latency from the device to fog layers (ms)	20
Deadlines of tasks (ms)	[50–150]
Power consumption in idle mode	(65–75) % of the active dynamic mode
Power consumption in dynamic mode	(75–150) Watts

Table 4.4: Simulation and GWO Algorithm.

Parameter	Value
The number of tasks	[50, 100, 150, 200]
Number of wolves	30
Number of iterations	25
C	[0, 2]
r_1, r_2	[0, 1]
D	Any Value

To showcase the effectiveness of our proposed algorithms, we conducted a comparison with the following benchmarks.

i. Performance and Cost Algorithm (PCA): Introducing a cutting-edge cloud service scheduling algorithm that prioritizes performance and cost efficiency, the PCA algorithm is truly exceptional. This algorithm focuses on prioritizing tasks based on their profitability, with the goal of optimizing resource utilization and minimizing the Makespan. Utilizing Principal Component Analysis (PCA), it considers the importance of completion time and cost priorities, resulting in more cost-effective services and enhanced performance for cloud users.

ii. The Gray Wolf Optimize (GWO): The gray wolf optimizer algorithm is a metaheuristic algorithm that draws inspiration from the social framework and hunting behavior of gray

wolves. Gray wolves possess an inherent hierarchical organization, in which a powerful leader assumes control and the remaining members obediently adhere to their guidance

iii. Particle Swarm Optimization (PSO): PSO is a clever approach that mimics the behavior of a flock of birds. Particles represent the birds in the context of the issue being addressed in this technique, and a multidimensional velocity governs their movement. A particle's position is impacted at each iteration by both the best position identified thus far and the best position among all particles in the whole problem space. A fitness function evaluates each particle's fitness value, showing its proximity to the desired aim in the search space and, hence, its relevance. Furthermore, each particle has a velocity that influences its path. Each particle successfully explores the issue space by continually pursuing the most promising particles at any given time[61].

4.3 RESULTS

4.3.1 Bandwidth Deadline Algorithm (BDA) Results

Here, we present the findings from the tests that we carried out. First, we thoroughly examine how the performance of our proposed algorithms impacts the outcomes. Later on, we evaluate the effectiveness of our algorithms by fine-tuning the number of tasks, fog nodes, and cloud nodes, and comparing them to other existing algorithms

Figures 4.1, 4.2, and 4.3 depict the Makespan for three different scenarios: one with 10 fog nodes, another with 20 fog nodes, and a third with 30 fog nodes. Each figure corresponds to one of the three cases described. The graphic clearly demonstrates that the suggested method surpasses the other three algorithms (FCFS, EDF, and Max-Min) in terms of effectiveness, since it consistently outperforms them in all scenarios.

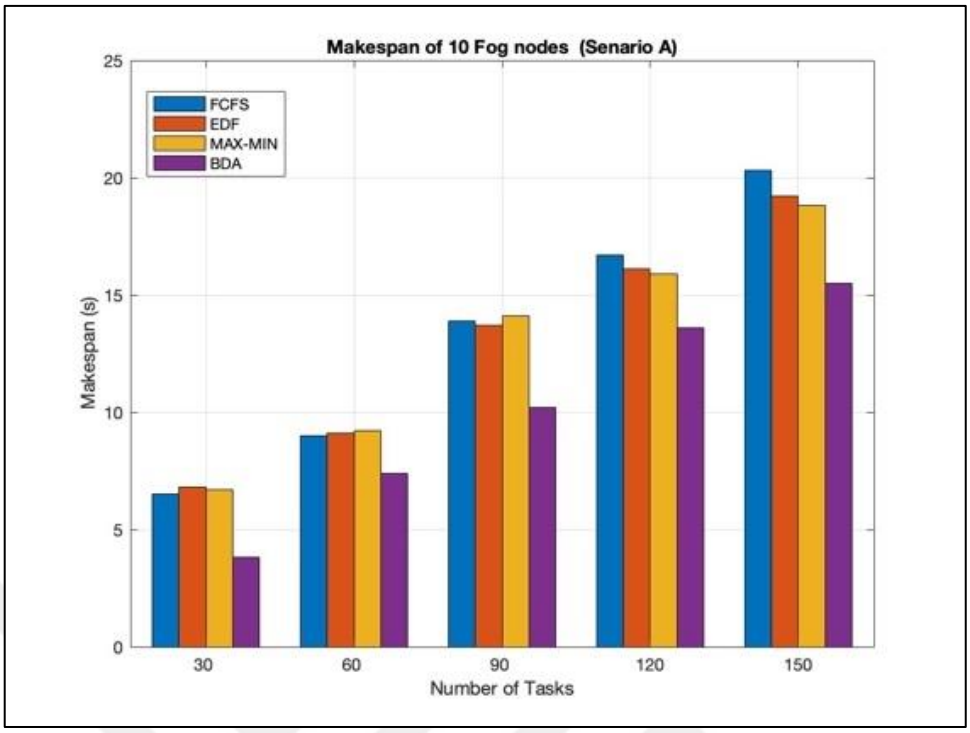


Figure 4.1 Show the Tasks Makespan of Scenario (A) for 10 fog.

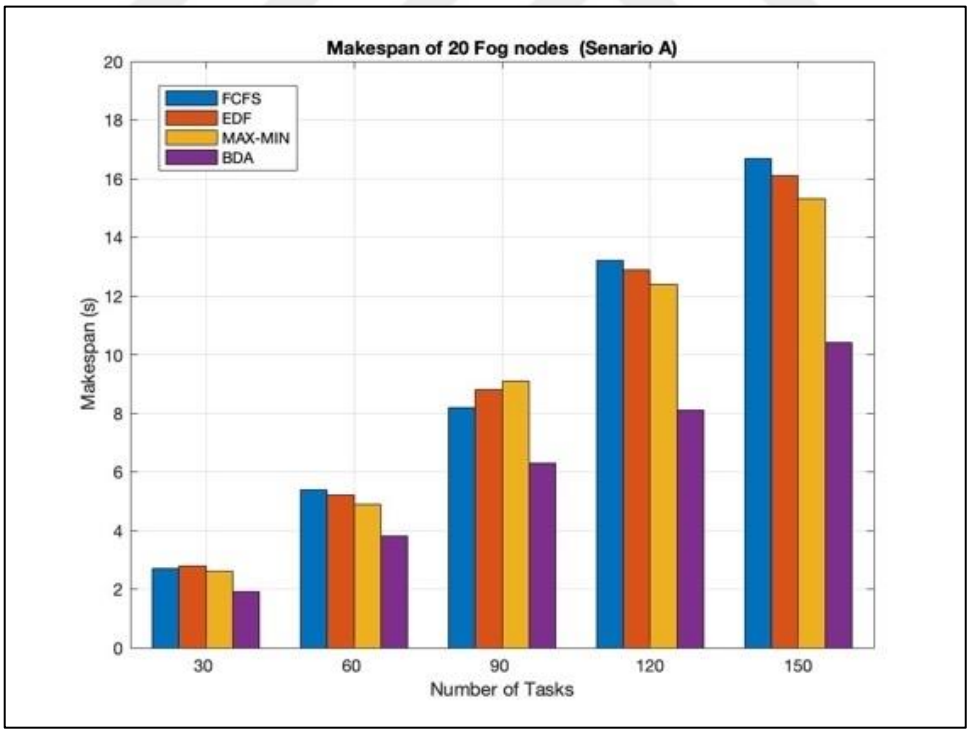


Figure 4.2: Show the Tasks Makespan of Scenario (A) for 20 fog.

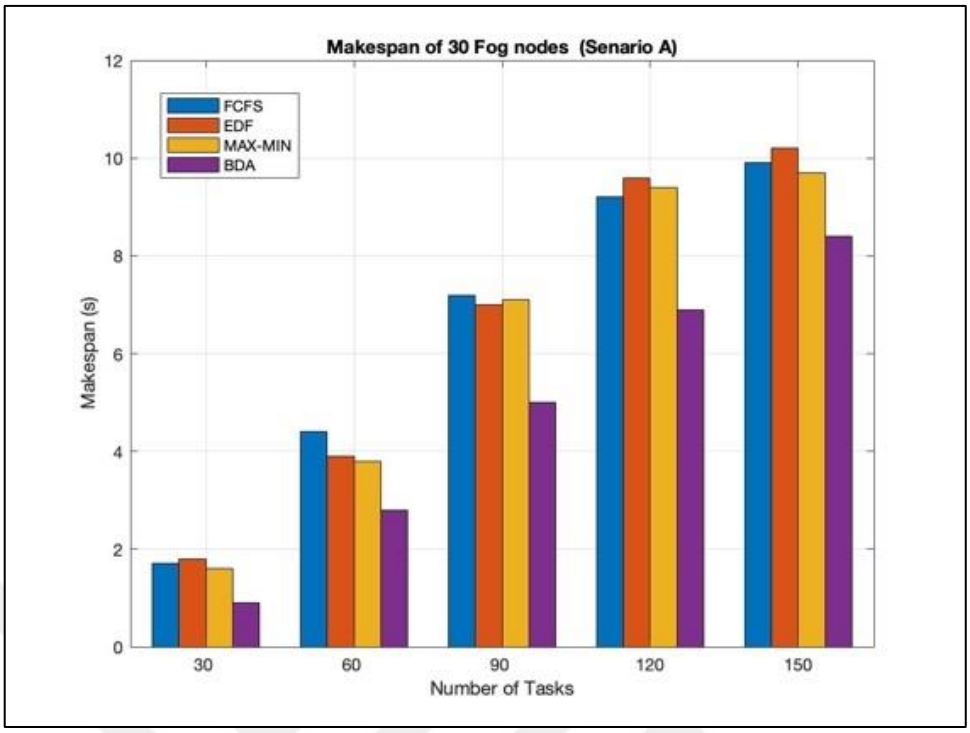


Figure 4.3: Show the Tasks Makespan of Scenario (A) for 30 fog.

Figures 4.4, 4.5, and 4.6 depict the Makespan for three different scenarios: one with 10 fog nodes, another with 20 fog nodes, and a third with 30 fog nodes. Each figure corresponds to one of the three aforementioned cases. The graphic clearly demonstrates that the suggested method surpasses the other three algorithms (FCFS, EDF, and Max-Min) in terms of effectiveness, since it consistently outperforms them in all scenarios.

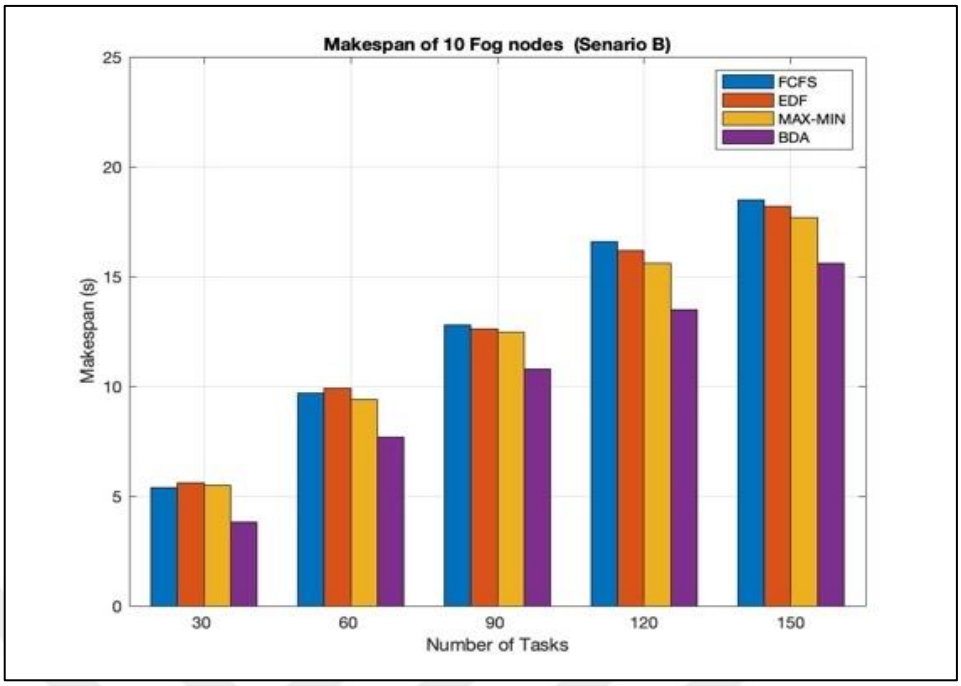


Figure 4.4: Show the Tasks Makespan of Scenario (B) for 10 fog

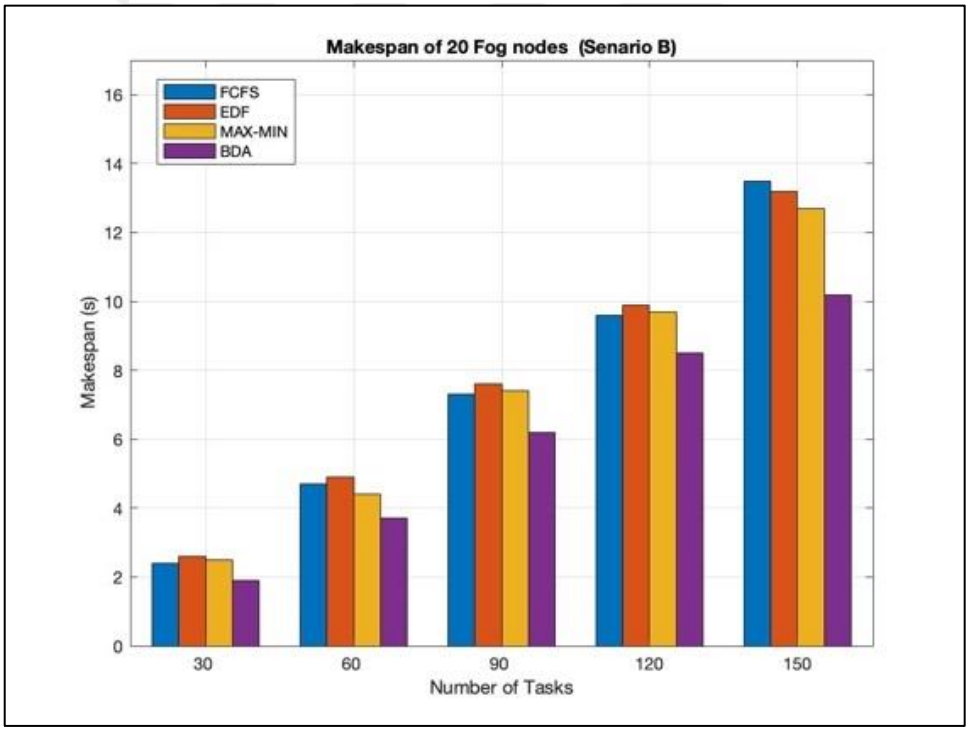


Figure 4.5: Show the Tasks Makespan of Scenario (B) for 20 fog.

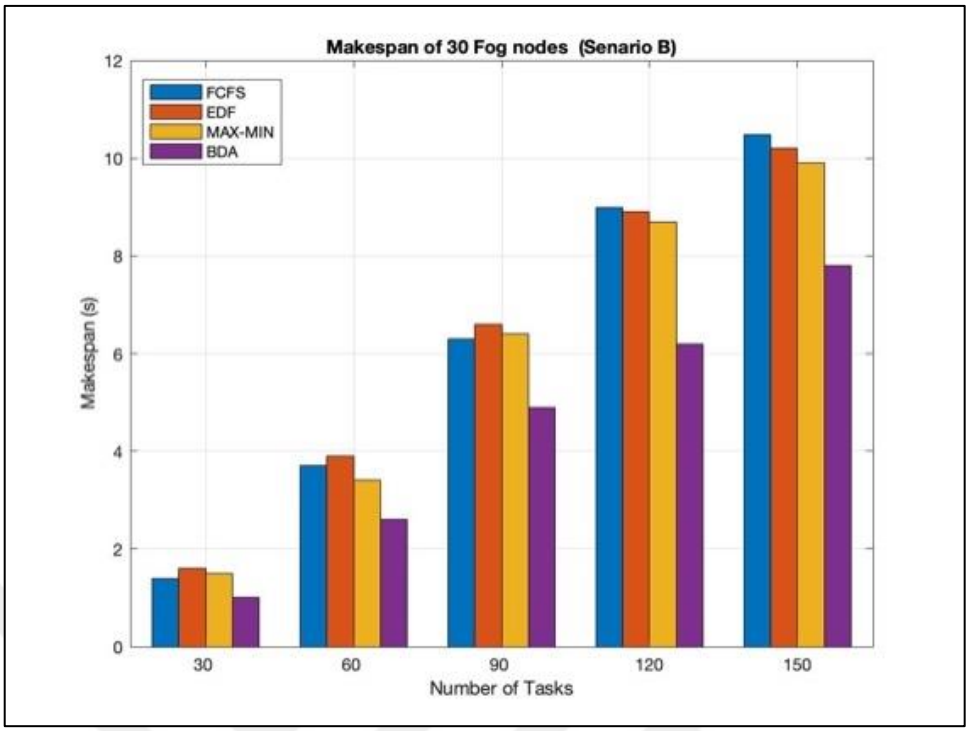


Figure 4.6: Show the Tasks Makespan of Scenario (B) for 30 fog.

Figure (4.7, 4.8, and 4.9) depicts the Makespan for the scenario using three different methods: 10 fog nodes, 20 fog nodes, and 30 fog nodes. Each figure corresponds to one of the three scenarios indicated above. The chart clearly demonstrates that the suggested method surpasses the other three algorithms (FCFS, EDF, and Max-Min) in terms of effectiveness, consistently outperforming them in all cases.

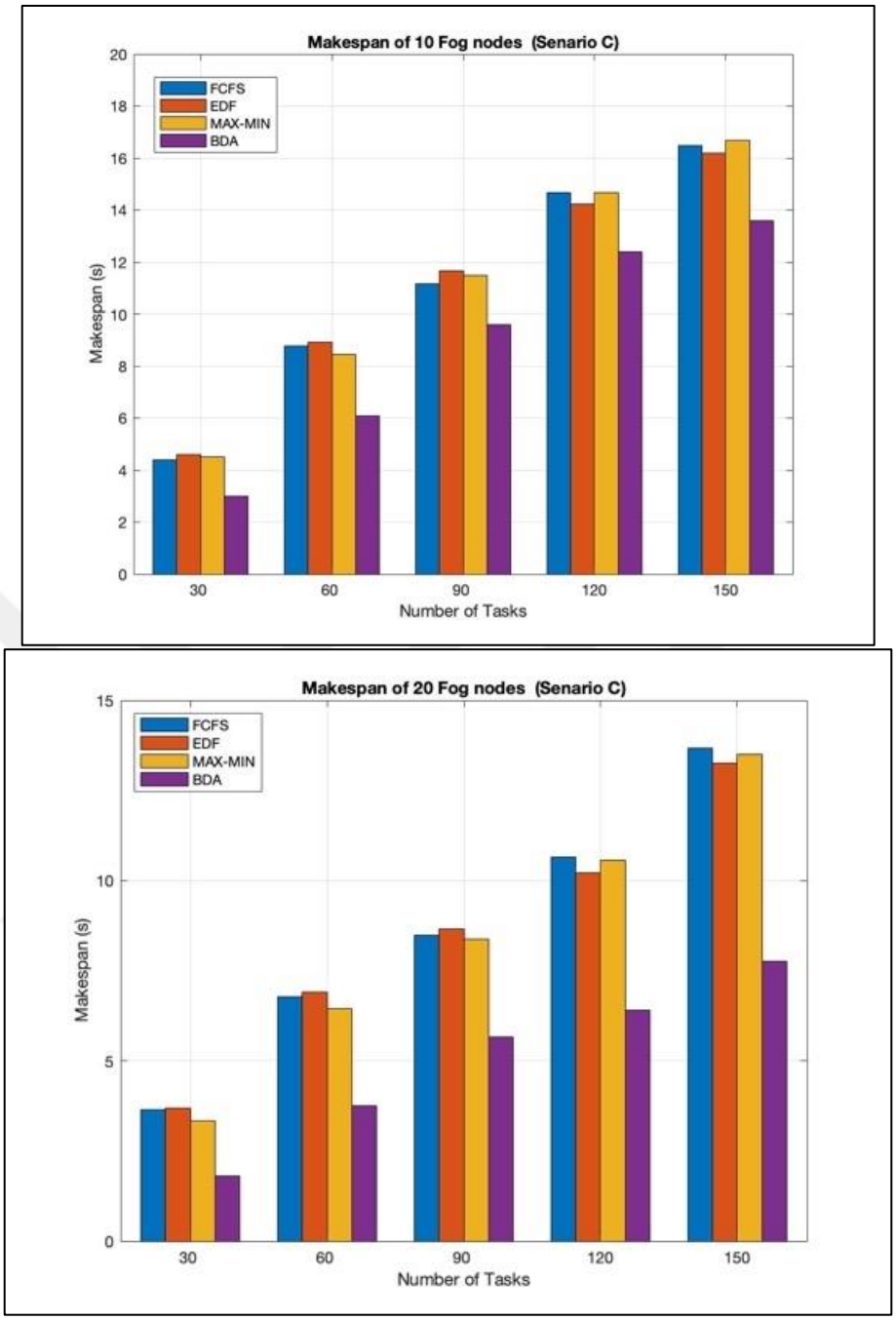


Figure 4.8: Show the Tasks Makespan of Scenario (C) for 20 fog.nodes.

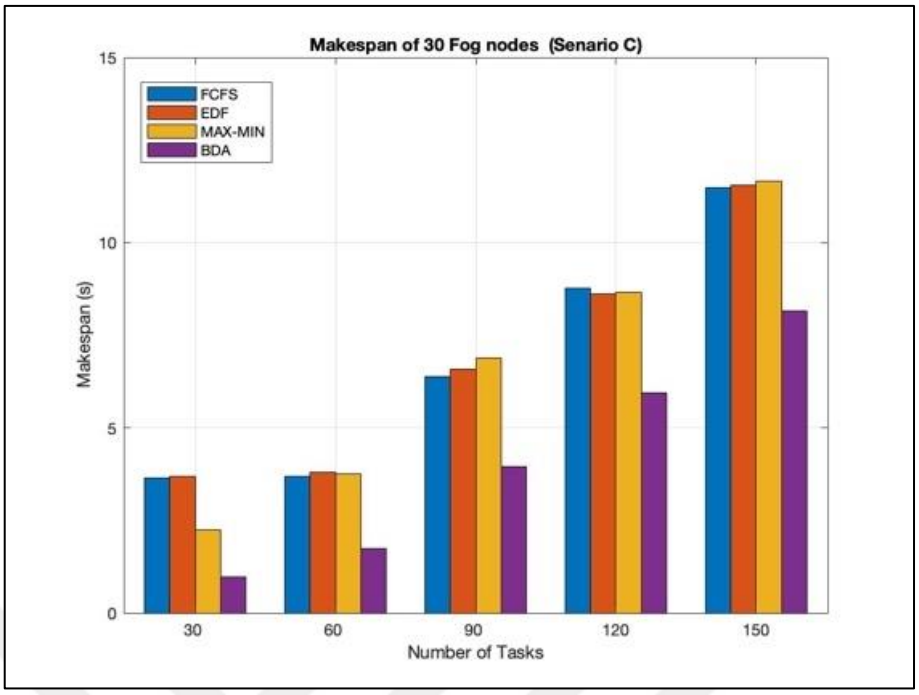


Figure 4.9: Show the Tasks Makespan of Scenario (C) for 30 fog.

Figure 4.10 displays the average resource use for the four algorithms, and it is evident that the proposed algorithm outperforms the other three algorithms in terms of resource utilization (FCFS, EDF, and Max-Min).

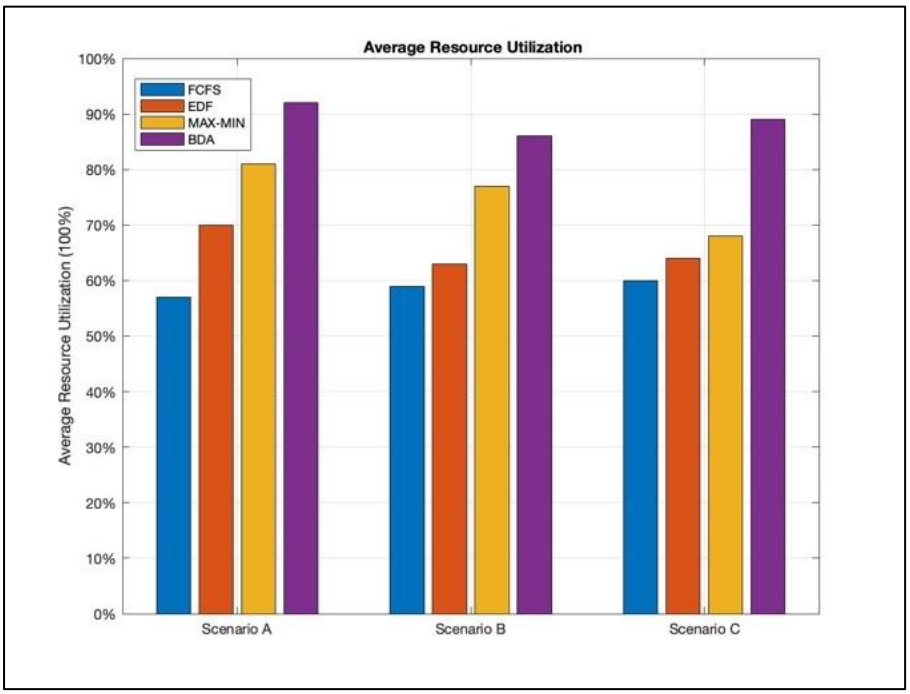


Figure 4.10: Average Resource Utilization.

Figure 4.10 illustrates the average resource use for the four algorithms, clearly demonstrating that the suggested algorithm surpasses the other three methods (FCFS, EDF, and Max-Min) in terms of resource use.

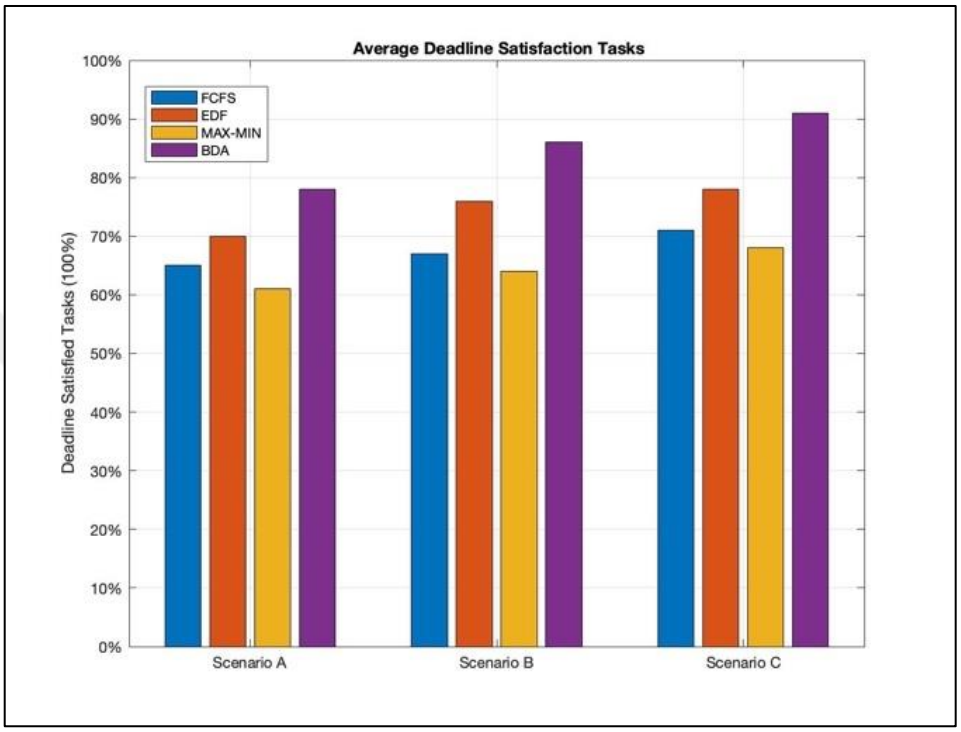


Figure 4.11: Average Deadline Satisfaction.

Figure 4.11 displays the jobs that meet the deadline for the four methods, demonstrating that our suggested approach outperforms the FCFS, EDF, and Max-Min algorithms.

For our assessment, we constructed three scenarios to evaluate the efficacy of our suggested algorithm. In scenario one, the majority of tasks required a large amount of bandwidth, whereas a few processes required low or medium bandwidth. Scenario two consisted of numerous activities with moderate bandwidth, as well as a few tasks with high and low bandwidth. Scenario three consisted of numerous tasks that required a modest amount of bandwidth, whereas just a few tasks required a high or medium amount of bandwidth.

We employed a three-step progression in node quantities, commencing with 10 nodes in the first stage, followed by 20 nodes in the second stage, and concluding with 30 nodes in the

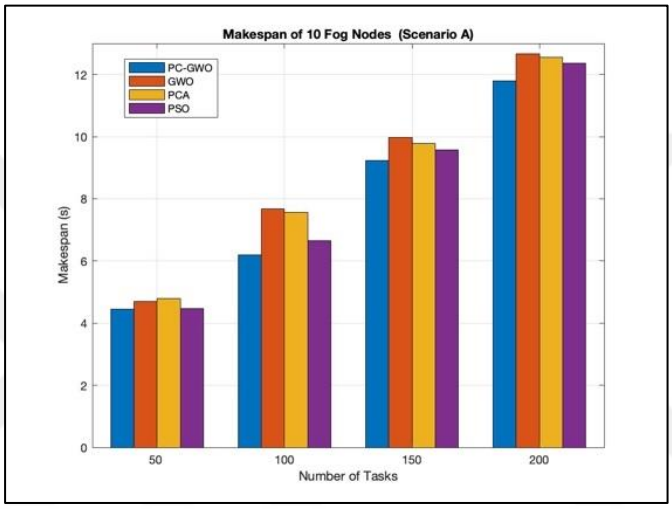
third stage. There were 5 cloud nodes in each stage, with a delay of 100ms in the cloud and 20ms in the fog. The jobs were distributed among the scenarios in the following quantities: 30, 60, 90, 120, and 150.

Our testing demonstrated that our algorithm outperformed the other algorithms we examined, exhibiting superior performance. Figures 4.12-4.14, 4.15 and 4.16 demonstrate that our solutions outperformed other algorithms in all three circumstances regarding makespan, average resource use, and deadline satisfaction. The reason for this is that our proposed algorithm takes into account the priority of tasks based on the deadline field and the minimal completion time. Additionally, it considered the allocation of jobs to the suitable environment (fog or cloud) based on the available bandwidth, and determined the most efficient route between the IoT devices and the fog-cloud environment.

4.3.2 PC-GWO Algorithm

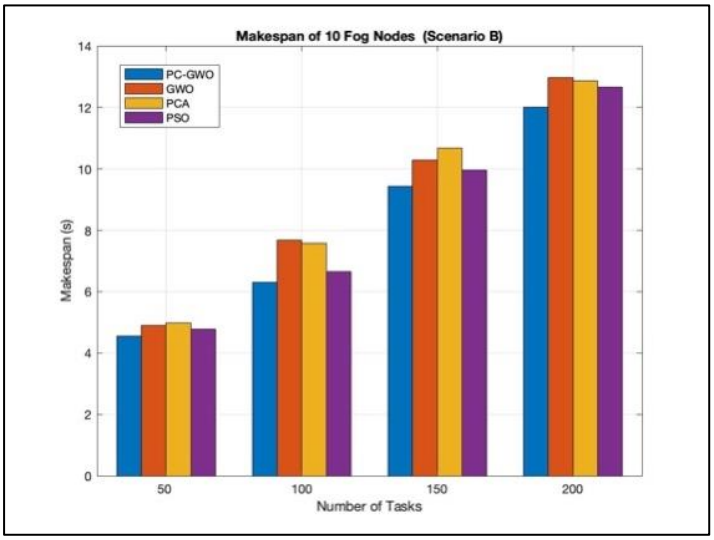
On this page, we have presented the results of our experiments. Initially, we evaluate the effect of the algorithms we implement on performance. As a network architect, we carefully evaluate the efficiency of these algorithms by fine-tuning the number of tasks, fog nodes, and cloud nodes, in comparison to existing techniques. The figures 4.12-4.14 illustrate the Makespan for the four operations with the given parameters. Our approach demonstrates superior performance compared to the GWO, PCA, and PSO algorithms in terms of Makespan, achieving reductions of 16.72%, 16.38%, and 14.107%, respectively. Our proposed technique consistently demonstrates superior performance compared to other methods (GWO, PCA, and PSO) in all scenarios. Figure 3 displays the average resource consumption of four algorithms (PC-GWO, GWO, PCA, and PSO) in scenarios A, B, and C. Each scenario consists of 10, 20, and 30 fog resources, accompanied by five cloud nodes. Below are the percentages of resource consumption for each scenario: These are the percentages: Here are the percentages: 94%, 72%, 78%, and 85%; 90%, 70%, 75%, and 83%; and 92%, 74%, 76%, 84%. Our proposed technique demonstrates superior performance in terms of resource utilization compared to GWO, PCA, and PSO. Figure 4 shows the average energy consumption, measured in kilojoules, for four algorithms (PC-GWO, GWO, PCA, and PSO) in scenarios A, B, and C. Every scenario consists of 10, 20, and 30 fog nodes, in addition to five cloud nodes. These are the energy consumption values

for each scenario: Scenario A consists of the following values: 22.75, 24.75, 26.25, and 23.55. Scenario B Includes the values 32.5, 35.89, 34.7, and 33.88. Scenario C is represented by the values 39.6, 47.25, 46.32, and 44.77. In this specific domain, our proposed methodology clearly surpasses GWO, PCA, and PSO. For this study, three distinct scenarios were devised to evaluate the efficacy of our algorithm. In one scenario, There was an increased quantity of high-priority tasks and a decreased quantity of low- and medium-priority assignments.

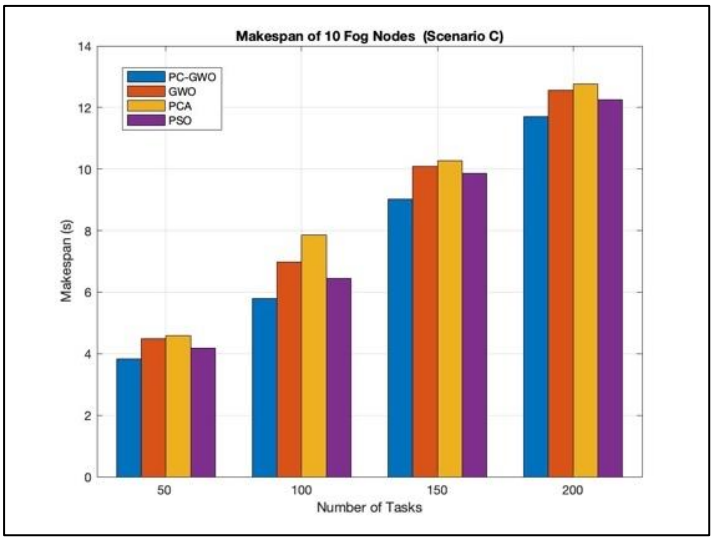


(a)

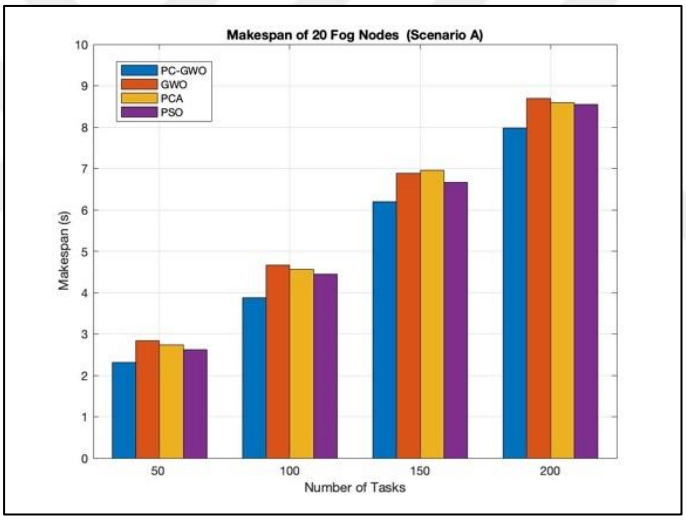
Figure 4.12: Show the Tasks Makespan of Scenario (a, b and c) for 10 fog nodes respectively.



(b)

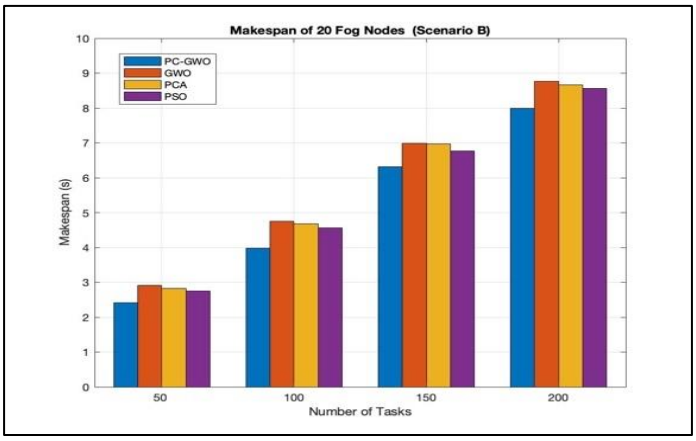


(c)

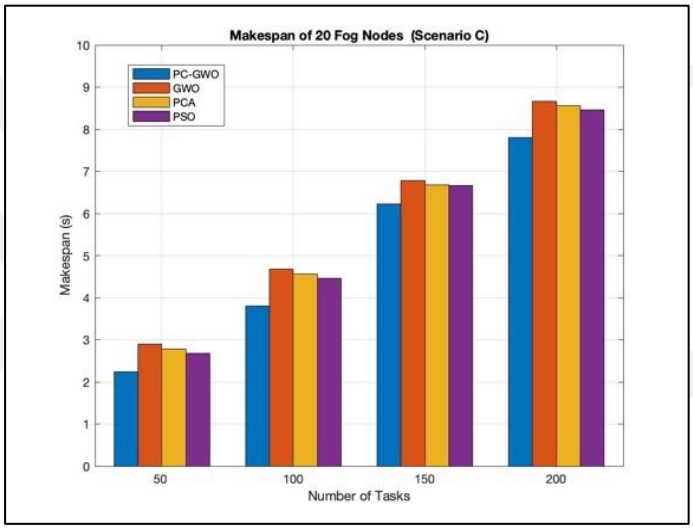


(a)

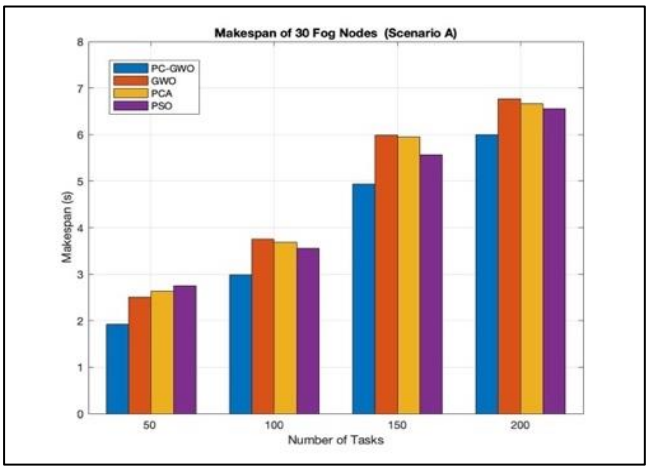
Figure 4.13: Show the Tasks Makespan of Scenario (a, b and c) for 20 fog nodes respectively.



(b)

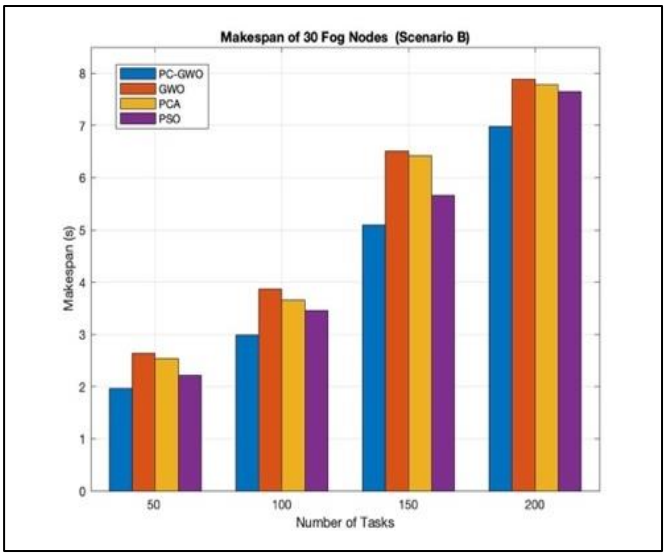


(c)

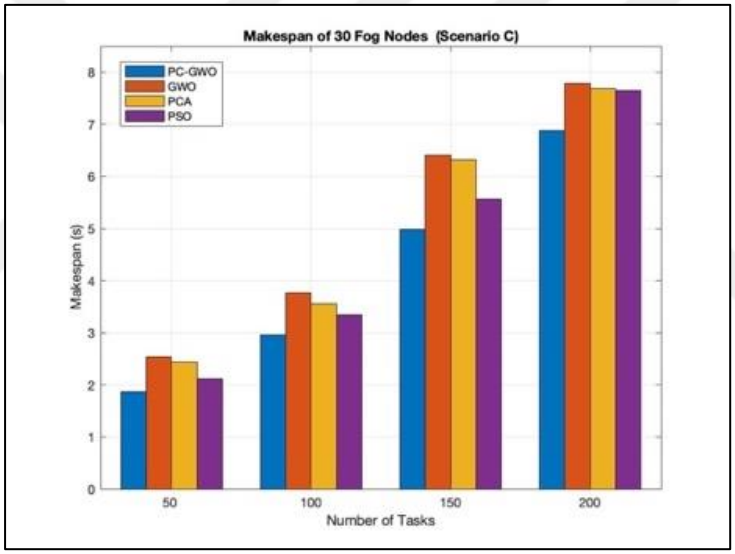


(a)

Figure 4.14: Show the Tasks Makespan of scenario (a, b and c) for 30 fog nodes respectively.


Makespan of 30 Fog Nodes (Scenario B)
PC-GWO
GWO
PCA
PSO
Makespan (s)
Number of Tasks
0
1
2
3
4
5
6
7
8
50
100
150
200

(b)


Makespan of 30 Fog Nodes (Scenario C)
PC-GWO
GWO
PCA
PSO
Makespan (s)
Number of Tasks
0
1
2
3
4
5
6
7
8
50
100
150
200

(c)

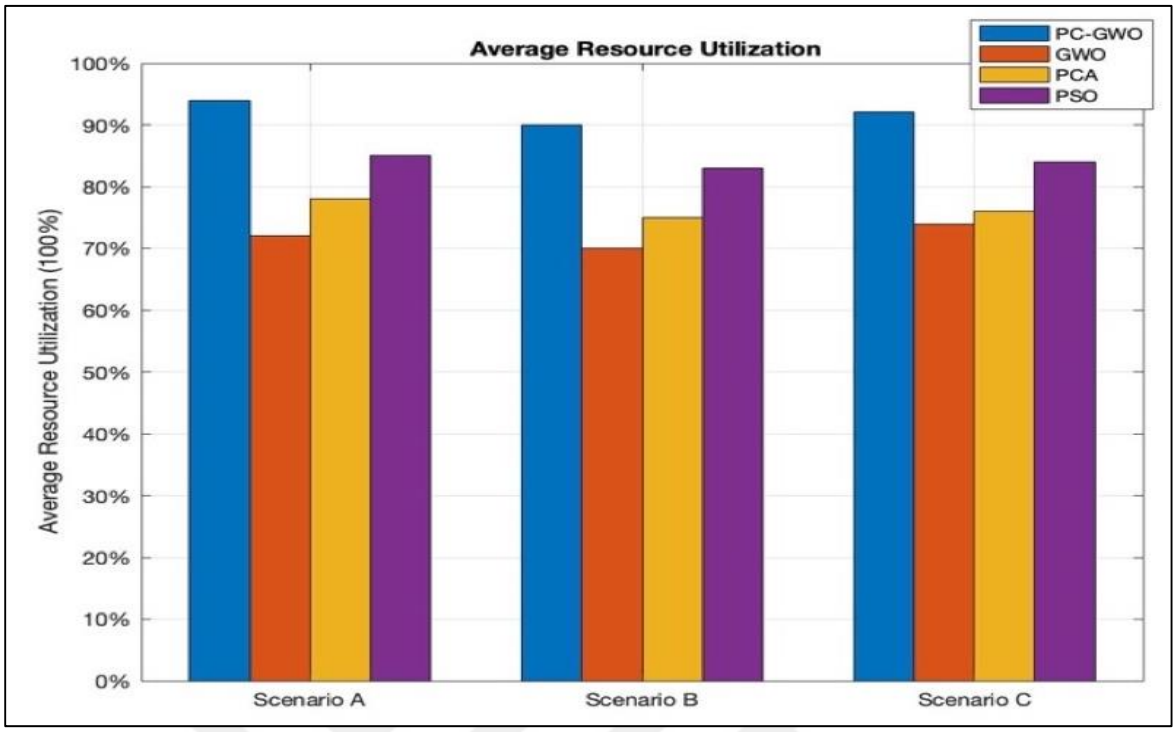


Figure 4.15: Average Resource Utilization.

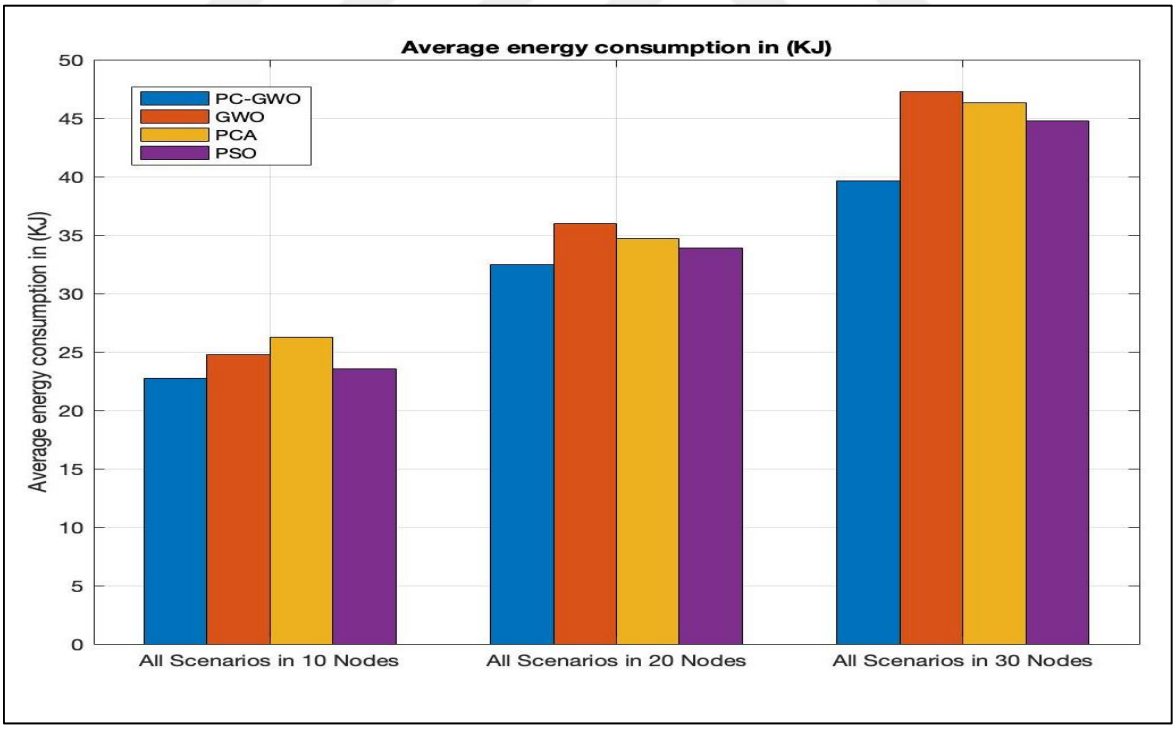


Figure 4.16: Average Energy Consumption (KJ).

Scenario two involved a range of jobs, some more important than others. Scenario three consisted of a mix of low-priority tasks, as well as a few high- and medium-priority assignments. We have devised a three-step procedure for assigning numbers to nodes. Initially, there were 10 nodes, which increased to 20 nodes in the second stage, and eventually reached 30 nodes in the third stage. Each stage consists of five cloud nodes, with each node having a latency of 100 milliseconds in the cloud and 20 milliseconds in the fog. In all instances, the number of assignments varied from 50 to 200. Our technique outperformed the other algorithms that were tested, as shown by the experimental results. From the data presented in Figures 2–4, it is clear that our suggested algorithm performs better than the competing approaches in three specific cases. It is particularly impressive when it comes to the span, standard resource utilization, and energy consumption. This benefit is a result of the evaluation performed by our algorithm on different factors such as job priority, cost, minimal completion time, energy consumption, and efficient distribution of work across fog and cloud environments. We employ the GWO methodology to evaluate and identify the most beneficial option. In addition, it assesses the most efficient path between the web of Things hardware and the fog-cloud circumstances, leading to improved efficiency.

4.4 SUMMERY

In this chapter, we first explained the methods we use to evaluate the bandwidth deadline algorithm and the PC-GWO algorithm and the scenarios we use with the system we use to evaluate our methods. Because Makespan, energy consumption, and resource utilization are some of the important problems in fog-cloud computing scheduling, we used these parameters to evaluate our proposed algorithms. The results showed that our proposed algorithms are better than the other scheduling methods. In this evaluation, we used some traditional algorithms and heuristic algorithms to ensure the accuracy of our proposed methods.

5. CONCLUSION AND FUTURE WORK

A new paradigm of distributed computing called cloud computing or fog computing emerged in the past several years. This paradigm allows users to rent computing hardware or software services through the internet on an as-needed basis. Customers and service providers alike face new obstacles and needs as a result of this shift in computing paradigm, which affects their operations and profitability. One of the primary goals, in light of the intense rivalry among cloud service providers, is to provide consumers competitive offers by offering the most cost-effective services possible, which means providing high-quality services at reasonable prices. The application of a policy to accomplish one or more system objectives is the responsibility of the task scheduling process in computer systems. There are a variety of scheduling approaches, including static and dynamic, offline and online, single-objective and multi-objective, and more. A lot of effort in cloud computing research has gone on achieving a singular goal, namely reducing the Makespan. Despite its critical importance, it must be considered in conjunction with other goals in the cloud system, such as reducing energy consumption and service costs, among others. Therefore, fog-cloud computing systems necessitate a novel class of hybrid multi-objective schedulers designed to optimize a collection of objectives simultaneously. Providers of data centers may need schedulers with a variety of policies to meet the varying service requirements of fog-cloud computing customers, who demand varying degrees of service quality (i.e., different pricing offers for different service quality levels). In this thesis, we have explored the possibility of developing a hybrid scheduler that can optimize not only the load distribution across resources but also the Makespan and service cost. We found that the price of a service is directly related to its performance; in other words, if you want better results, you have to spend more. We also noticed that the profit, which is the difference between the two costs—the task cost paid by the service provider to the data center provider and the task cost paid by the service customer to the service provider—is not constant or fixed. Therefore, we implemented a scheduling policy that gives the most profitable work to the resource that can complete it the fastest, i.e., with the smallest completion time. Our results demonstrate that this policy has the potential to result in optimally performing service provisioning, with minimal Makespan and reasonable total costs. by what we can tell by comparing our algorithms to others, our Makespan can be shorter than the minimum Makespan and energy consumption of popular

single objective schedulers. While doing so, the overall cost is comparable to that which would be incurred by employing schedulers whose sole purpose is to limit costs. We found that some tasks might starve to death; thus, our scheduling method has to take the task's age into account to prevent starving. Just make the task more important as you get older, and you'll be done in no time. It has been brought to our attention that relying solely on our suggested approach could result in an uneven distribution of jobs among the accessible resources, since it would put an undue burden on the resources that perform the best. To make sure all the resources are getting their fair share of work, it's best to combine it with a load balancer. By including a load balancer into our scheduling algorithm, we were able to achieve better Makespan, reduced total cost, and fair distribution in our studies. Reduced waiting time for jobs is the direct result of higher resource utilization, which in turn reduces the Makespan. Conversely, the overall cost is reduced since, according to our suggested method, the resource with the best performance also happens to be the one with the highest cost. This is because the resource with the heaviest load is also the one with the highest performance. Thus, transferring some work from this resource to another unquestionably lowers the price. Some critical topics, which can provide fruitful avenues for future research, have escaped our attention in this study. Here are several examples: Because fog-cloud data centers are notoriously power hogs, cutting down on Many questions, such as system temperatures, will need answers in subsequent work. Therefore, it is essential and challenging to use schedulers that incorporate temperature utilization minimization as an extra target.

REFERENCES

[1] "What Is the Internet of Things (IoT)? | Oracle India." Accessed: Apr. 03, 2023. [Online]. Available: https://www.oracle.com/in/internet-of-things/what-is-iot/

[2] "What is IoT? - Internet of Things Explained - AWS." Accessed: Feb. 19, 2024. [Online]. Available: https://aws.amazon.com/what-is/iot/

[3] "IoT, from Cloud to Fog Computing - Cisco Blogs." Accessed: Apr. 02, 2023. [Online]. Available: https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing

[4] K. Siozios, D. Anagnostos, D. Soudris, and E. Kosmatopoulos, "IoT for smart grids," *Cham, Switzerland: Springer*, 2019.

[5] N. H. Mahmood, N. Marchenko, M. Gidlund, and P. Popovski, *Wireless Networks and Industrial IoT*. Springer, 2020.

[6] H. Raad, *Fundamentals of IoT and wearable technology design*. John Wiley & Sons, 2020.

[7] M. Iorga, L. Feldman, R. Barton, M. Martin, N. Goren, and C. Mahmoudi, "The nist definition of fog computing," National Institute of Standards and Technology, 2017.

[8] M. Iorga, "Fog Computing Conceptual Model-Recommendations of the National Institute of Standards and Technology. US Department of Commerce (2018)."

[9] Y. Yang, J. Huang, T. Zhang, and J. Weinman, "Fog and Fogonomics: Challenges and Practices of Fog Computing, Communication, Networking, Strategy, and Economics," 2020.

[10] A. Abbas, S. U. Khan, and A. Y. Zomaya, *Fog Computing: Theory and Practice*. John Wiley & Sons, 2020.

[11] B. Sosinsky, *Cloud computing bible*, vol. 762. John Wiley & Sons, 2010.

[12] J. Baty and J. REMELL, “Take your business to a higher level,” *Sun Cloud Computing*. pp. 45–48, 2009.

[13] S. Microsystem, “Introduction to Cloud Computing architecture,” *White Paper,* 2009.

[14] P. Mell and T. Grance, “The NIST definition of cloud computing,” 2011.

[15] T. Lynn, J. G. Mooney, B. Lee, and P. T. Endo, “The cloud-to-thing continuum: opportunities and challenges in cloud, fog and edge computing,” 2020.

[16] N. B. Ruparelia, “3 TYPES OF CLOUD COMPUTING,” 2023.

[17] N. B. Ruparelia, *Cloud computing*. Mit Press Cambridge, MA, 2016.

[18] N. K. Sehgal, P. C. P. Bhatt, and J. M. Acken, “Cloud computing with security,” *Concepts and practices. Second edition. Switzerland: Springer*, 2020.

[19] Y. E. Rachmad, R. Dewantara, S. Junaidi, M. Firdaus, and S. W. Sulistianto, *MASTERING CLOUD COMPUTING (Foundations and Applications Programming)*. PT. Sonpedia Publishing Indonesia, 2023.

[20] J. Huang, W. Susilo, F. Guo, G. Wu, Z. Zhao, and Q. Huang, “An Anonymous Authentication System for Pay-As-You-Go Cloud Computing $^* $,” *IEEE Trans Dependable Secure Comput*, vol. 19, no. 2, pp. 1280–1291, 2020.

[21] K. Hwang, J. Dongarra, and G. C. Fox, *Distributed and cloud computing: from parallel processing to the internet of things*. Morgan kaufmann, 2013.

[22] R. Buyya and S. N. Srirama, *Fog and edge computing: principles and paradigms*. John Wiley & Sons, 2019.

[23] D. Hoang and T. D. Dang, “FBRC: Optimization of task scheduling in fog-based region and cloud,” in *2017 IEEE Trustcom/BigDataSE/ICESS*, IEEE, 2017, pp. 1109–1114.

[24] Y. Sun, F. Lin, and H. Xu, “Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II,” *Wirel Pers Commun*, vol. 102, pp. 1369–1385, 2018.

[25] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, “DATS: Dispersive stable task scheduling in heterogeneous fog networks,” *IEEE Internet Things J*, vol. 6, no. 2, pp. 3423–3436, 2018.

[26] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, “Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing,” *Future Generation Computer Systems*, vol. 111, pp. 539–551, 2020.

[27] C. Bu and J. Wang, “Computing tasks assignment optimization among edge computing servers via SDN,” *Peer Peer Netw Appl*, vol. 14, pp. 1190–1206, 2021.

[28] J. Almutairi and M. Aldossary, “A novel approach for IoT tasks offloading in edge-cloud environments,” *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–19, 2021.

[29] B. K. Alotaibi and U. Broudi, “Offload and Schedule Tasks in Health Environment using Ant Colony Optimization at Fog Master,” in *2022 International Wireless Communications and Mobile Computing (IWCMC)*, IEEE, 2022, pp. 469–474.

[30] T. Islam and M. M. A. Hashem, “Task scheduling for big data management in fog infrastructure,” in *2018 21st International Conference of Computer and Information Technology (ICCIT)*, IEEE, 2018, pp. 1–6.

[31] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M.-T. Zhou, “MEETS: Maximal energy efficient task scheduling in homogeneous fog networks,” *IEEE Internet Things J*, vol. 5, no. 5, pp. 4076–4087, 2018.

[32] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, “Energy efficient task allocation and energy scheduling in green energy powered edge computing,” *Future Generation Computer Systems*, vol. 95, pp. 89–99, 2019.

[33] K. Gai, X. Qin, and L. Zhu, “An energy-aware high performance task allocation strategy in heterogeneous fog computing environments,” *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 626–639, 2020.

[34] F. Hoseiny, S. Azizi, M. Shojafar, F. Ahmadiazar, and R. Tafazolli, “PGA: a priority-aware genetic algorithm for task scheduling in heterogeneous fog-cloud computing,” in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2021, pp. 1–6.

[35] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. C. M. Leung, “MASM: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence,” *IEEE Trans Industr Inform*, vol. 15, no. 7, pp. 4216–4224, 2019.

[36] Z. Yin *et al.*, “A multi-objective task scheduling strategy for intelligent production line based on cloud-fog computing,” *Sensors*, vol. 22, no. 4, p. 1555, 2022.

[37] H. O. Hassan, S. Azizi, and M. Shojafar, “Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments,” *IET communications*, vol. 14, no. 13, pp. 2117–2129, 2020.

[38] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, “Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning,” *IEEE Trans Cogn Commun Netw*, vol. 7, no. 3, pp. 881–892, 2021.

[39] “What Is the Internet of Things (IoT)?” Accessed: Aug. 05, 2023. [Online]. Available: https://www.oracle.com/internet-of-things/what-is-iot/

[40] J. Xu, Z. Hao, R. Zhang, and X. Sun, “A method based on the combination of laxity and ant colony system for cloud-fog task scheduling,” *IEEE Access*, vol. 7, pp. 116218–116226, 2019.

[41] B. K. Tripathy and J. Anuradha, *Internet of things (IoT): technologies, applications, challenges and solutions*. CRC press, 2017.

[42] W. Chang and J. Wu, *[1] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," Computer Networks, vol. 192, p. 108040, 2021. [2] H. Raad, Fundamentals of IoT and wearable technology de*. Springer, 2021.

[43] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing," *IEEE Internet Things J*, vol. 7, no. 7, pp. 5773–5782, 2019.

[44] S. Omer, S. Azizi, M. Shojafar, and R. Tafazolli, "A priority, power and traffic-aware virtual machine placement of IoT applications in cloud data centers," *Journal of systems architecture*, vol. 115, p. 101996, 2021.

[45] N. A. Alsamarai, O. N. Uçan, and O. F. Khalaf, "Bandwidth-Deadline IoT Task Scheduling in Fog–Cloud Computing Environment Based on the Task Bandwidth," *Wirel Pers Commun*, 2023, doi: 10.1007/s11277-023-10567-1.

[46] M. I. K. Khalil *et al.*, "Renewable-aware geographical load balancing using option pricing for energy cost minimization in data centers," *Processes*, vol. 10, no. 10, p. 1983, 2022.

[47] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer Advances in Engineering Software. 69 46–61." 2014.

[48] M. Jia, W. Chen, J. Zhu, H. Tan, and H. Huang, "An Energy-aware Greedy Heuristic for Multi-objective Optimization in Fog-Cloud Computing System," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2020, pp. 794–799.

[49] W. Lin, H. Wang, Y. Zhang, D. Qi, J. Z. Wang, and V. Chang, "A cloud server energy consumption measurement system for heterogeneous cloud environments," *Inf Sci (N Y)*, vol. 468, pp. 47–62, 2018.

[50] M. Bennis, M. Debbah, and H. V. Poor, “Ultrareliable and low-latency wireless communication: Tail, risk, and scale,” *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, 2018.

[51] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, “A tutorial on UAVs for wireless networks: Applications, challenges, and open problems,” *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[52] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, “Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach,” *Journal of Network and Computer Applications*, p. 103333, 2022.

[53] J. Tang, G. Liu, and Q. Pan, “A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends,” *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 10, pp. 1627–1643, 2021.

[54] G. Li and Z. Wu, “Ant colony optimization task scheduling algorithm for SWIM based on load balancing,” *Future Internet*, vol. 11, no. 4, p. 90, 2019.

[55] Z. Yin *et al.*, “A multi-objective task scheduling strategy for intelligent production line based on cloud-fog computing,” *Sensors*, vol. 22, no. 4, p. 1555, 2022.

[56] M. I. K. Khalil *et al.*, “Renewable-aware geographical load balancing using option pricing for energy cost minimization in data centers,” *Processes*, vol. 10, no. 10, p. 1983, 2022.

[57] N. AL-Sammarraie, M. Alrahmawy, and M. Rashad, “A Scheduling Algorithm to Enhance the Performance and the Cost of Cloud Services,” *International Journal of Intelligent Computing and Information Sciences*, vol. 15, no. 1, pp. 1–14, 2015.

[58] A. S. Alfa, *Queueing theory for telecommunications: discrete time modelling of a single node system*. Springer Science & Business Media, 2010.

[59] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo, *Deadline scheduling for real-time systems: EDF and related algorithms*, vol. 460. Springer Science & Business Media, 1998.

[60] S. S. Chauhan and R. C. Joshi, "A weighted mean time min-min max-min selective scheduling strategy for independent tasks on grid," in *2010 IEEE 2nd International Advance Computing Conference (IACC)*, IEEE, 2010, pp. 4–9.

[61] D. Baburao, T. Pavankumar, and C. S. R. Prabhu, "Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method," *Appl Nanosci*, pp. 1–10, 2021.