Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Department of Information Networks

Efficient FOG Offloading Optimized Approach to Improve IoT Devices Performance

A Thesis

Submitted to the Council of the College of Information Technology for Postgraduate Studies of University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology - Information Networks

By

Ishraq Madi Jbour Jassim

Supervised by

Asst. Prof. Dr. Hilal Abdul Hussein Abood

2022 A.D 1444 A.H

بسم الله الرحمن الرحيم

صدق الله العظيم

Certification of the Examination Committee

We, the undersigned, certify that (**Ishraq Madi Jbour**) candidate for the degree of Master in Information Technology-Software, has presented her thesis of the following title "**Efficient FOG Offloading Optimized Approach to Improve IoT Devices Performance**" as it appears on the title page and front cover of the thesis that the said thesis is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: October 29,2022

Signature:
Name: ***Dr. Saad Talib Hasson***
Title: Professor
Date: / / 2022
(**Chairman**)

Signature:
Name: ***Dr. Wael Jabbar Abed***
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name: ***Dr. Mohammad Hussein Jawwad***
Title: Assistant Professor
Date: / / 2022
(**Member**)

Signature:
Name: ***Dr. Hillal Abdul Hussein***
Title: Assistant Professor
Date: / / 2022
(**Member and Supervisor**)

Signature:
Name: **Dr. Hussein Atiya Lafta**
Title: Professor
Date: / / 2022
(**Dean of Collage of Information Technology**)

Supervisor Certification

I certify that the thesis entitled " **Efficient FOG Offloading Optimized Approach to Improve IoT Devices Performance** " was prepared under my supervision at the department of Networks / College of Information Technology/ University of Babylon, by "**Ishraq Madi Jbour**" as partial fulfillment of the requirements of the degree of Master in Information Technology

Signature:

Supervisor Name: **Asst. Prof. Dr. Hilal Abdul Hussein Al Libawy**

Date: / /2022

The Head of the Department Certification

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:

Prof. Dr. Saad Talib Hasson Aljebori

Head of Networks Department

Date: / /2022

Declaration

I hereby declare that this Dissertation, submitted to the University of Babylon in partial fulfillment of requirement for the degree of Master of Information Technology-Information Networks has not been submitted as an exercise for a similar degree at any other university. I also certify that this work described here is entirely my own except for experts and summaries whose sources are appropriately cited in the references.

Signature:

Name: **Ishraq Madi Jbour**

Date: / /2022

Acknowledgement

In the name of Allah, Most Gracious, Most Merciful, Praise and thanks be to Him and the satisfaction of parents and conciliation only from Him greatest praise is to Allah for His assistance in coping with the difficulty that I met in my study, and for always helping me to achieve my aims, also for His great graces all the time.

I would like to express my deepest thanks to my supervisor **Asst. Prof.Dr. Hilal Abdul Hussein Al Libawy** for his valuable advice, motivation, guidance and for so many fruitful discussions throughout the preparation of this thesis.

I would like to extend my respect and deepest gratitude to the College of Information Technology.

Sincere appreciation and love go to my family; my father who was always with me and my dear mother, nothing could reward her, they provide me with optimism pure affection. they give me great hope and encouragement. As a palm tree, they stand with me in every step in this research. I dedicate this work and special thanks to those who encouraged me to continue my scientific career.

Finally, sincere thanks and appreciation to all friends, colleagues and loved ones.

Abstract

Fog Computing refers to a computing architecture that offers computing, storage, control, and networking capabilities to enable IoT applications. By shifting IoT workloads to Fog computing instead of the cloud, response times for data analytics can be slashed through the use of low latency. Because of the exponential increase in IoT-connected devices, high quality of service (QoS) must be guaranteed. Fog computing achieves this by intelligently incorporating IoT device improvements and expanding cloud computing's potential. To improve computation and QoS, Fog node selection is crucial. QoS requirements for smart cities, smart energy, healthcare, automobiles, agriculture, and so on are met by Fog computing. The limitations of cloud computing are circumvented by Fog computing. Although Fog computing is now a decade old, it is still in its infancy and has ways to there has been a lot of published work looking into ways to reduce latency or boost efficiency. Low latency and low power consumption are two areas where this study excels. For the first time, the effects of mobile device mobility on network performance, as well as the rise in latency and power consumption with varying degrees of mobility, were discussed in the context of Fog computing offloading. This strategy for managing network resources is recommended to use the meta heuristics multi-objective Gray Wolf Optimization (MO-GWO) algorithm (latency and power consumption). For testing purposes, the well-known "FogNetSim++" simulator is used to setup an experiment and to build a case study network in Fog layer based. The conclusive evidence of successful implementation.

Table of Contents

Table of Figure

List of Abbreviation

Abbreviation	Description
IoT	Internet of Thing
FC	Fog Computing
QoS	Quality of Service
DC	Data Center
PSO	Particle Swarm Optimization
GWO	Gray Wolf Optimization
MO-GWO	Multi-objectives Gray Wolf Optimization
IP	Internet Protocol
TCP	Transmission Control Protocol

List of Symbols

Abbreviation	Description
$CompE$	Computation Energy
$CommE$	Communication Energy
μ_{device}	CPU Energy
$Od_{u,i}$	Offloading discussion factor
CPU_f	CPU resources to execute the task
$Ol_{u,i}$	Offloading location factor
i	Index of task
$prei$	Index of previous task
N	Total number of task
ω_{prei}^{u}	Size of transmitted data
U	Index of task
$r_{u,s}$	Transmission rate between device and Fog
Tp	Transmission Power of end device
$CommL_{fog}$	Communication Latency
$CompL_{fog}$	Computation latency
S_f	Service rate of Fog
A_f	Arrival rate of data
B_f	Fog network bandwidth
X_p	Size of data packet
C	CPU Speed of Fog

List of Tables

Chapter One

Introduction

1.1 Introduction

The Internet of Things (IoT) attracts attention from both businesses and universities in recent years, which is great because it has real-world applications. Data collected by smart sensors is typically sent to cloud data centers, where applications are then run on server-side computers. [1]. Cloud Computing is an appearing technology, also the subsequent great stage in the evolution of virtual computing in the information technology basis in past several years. This presents the flexibility and scalability on request services, and also it presents the virtualization kind of services along the internet to the user [2]. although the paradigm of cloud computing is effective in providing resources such as storage and computation for applications in IoT, However, how to conduct IoT applications has become an urgent issue due to the ever-increasing resources required by such applications, which in turn causes a volatile increase in consumed energy along with a loss of computing nodes' performance due to data transmission and the movement of computing nodes. [3]. The IoT 's development requires a specific infrastructural basis that can meet all of its needs, Fog computing is currently on the rise and seems to be the most viable alternative.

A simple concept of Fog is 'nearer to the ground' that provides us an understanding of how Fog Computing works. Fog Computing is a middleware plain positioned amidst cloud layer and IoT devices to reduce latency in some cloud scenarios. The terminology relates notably in data management and analysis to a new range of applications and services. In Cloud Computing data must be accessed from the central mainframe while Fog Computing provides quick and local access to edge devices. Fog Computing is a decentralized computing infrastructure known as Fog Networking and Fogging, where data, computation, storage and apps are

spread in most sensible and efficient position between data-source and cloud. Cloud Computing services can be simply extended by Fog Computing for using these services in network edge. "Sending data and computing these data in cloud could be catastrophic. Any network latency and processing delay might end with bad result. For example, your automated car is traversing through busy street. Suddenly a person comes in front of automated car. In this scenario, any network latency, slowness of computation and analysis effects the decision and subsequent action (Apply brake on car)".

The routers in a Fog network may act as the actual servers that supply the Fog's edge services with data and processing power. It is possible that the routers' ability to improve compute and storage performance will allow them to be used more effectively as computing nodes. In the era of big data, IoT applications, especially real-time apps, have various performance needs, hence they prefer to be hosted on the edge computing nodes. Similar to how cloud customers access and use computing, storage, and network resources, users in a Fog environment could do the same, and virtualization technology could be used to dynamically provision these on-demand resources. Fog computing nodes and remote cloud data canter installed physical resources could run Internet of Things applications. [3]

The impact of low latencies in edge network's domain plays a vital role in a smooth function of the real-time applications and proving to be the energy efficient as it saves much of active network sessions Once the info from the top users is being generated, and therefore, the routers within the local network will be directed it to the specified network. The workloads (workflows) are being offloaded to the intermediate Fog storage layer for a little time and keep the network active for other processes and reduce the network congestion. The Fog intermediate layer will lower hops count in

the network, it will automatically lower the focused workloads and will enhance the data mobility

1.2 Problem Statement

One of the important issues in offloading topic with low power consumption which should be considered, that tasks are scheduled in a way that no Fog node be overloaded and also not be underutilized. As a result, a solution should be explored as trade-off between energy use and efficiency Because if the servers are overburdened, there will problem of network efficiency, furthermore, if the servers are underused, energy usage will increase. Mobility is another non-discussed problem that will effect on latency, power consumption, the mobility plays an important role, it creates a new challenge for researchers in the form of optimum use of resources and resources handover

1.3 Research Objectives

1- The main objective of this thesis is to optimize the latency and power consumption simultaneously.

2- Finding the Mobility effect on offloading performance in Fog computing

3- Identify the appropriate offloading probability and transmit power to minimize energy consumption and delay

4- Formalize a multi-objective optimization problem.

1.4 Contribution of the research

1- To reduce the amount of power needed to run mobile and Internet of Things devices by maintain the QoS.

2- To speed up computing at Fog Nodes and reduce response times in result the latency was reduced.

3- To use a multi-objective meta-heuristic method, optimally decide if the job is to be computed while also offloading tasks.

1.5 Related works

In this section, the most related works associated with offloading and optimization in IoT-Fog computing are explained in the following paragraph

In [4] It was determined how much electricity could be saved without negatively impacting service latency by balancing the workload across Fog and cloud nodes. In the research work that was done, the author made the assumption that cloud and Fog nodes are coupled via a singular communication point. They came up with a system for the most efficient distribution of tasks between the Fog and the cloud, with the least amount of power consumption and service delay possible.

In [5] An easy-to-understand framework for decoupling data-centric IoT apps has been developed. This framework was designed on the idea of categorizing and computing tasks. There is no consideration for the role of communication delay in the suggested architecture.

In [6] seeks to compare the energy consumption and latency of services using the novel Fog paradigm applied to the Internet of Things with those using the conventional Cloud. However, at this time, model [6]

is being used exclusively for deployment of software behavior via the Fog infrastructure.

In [7] The offloading framework provided by Yousefpour et al. is based on an IoT-Fog application, and it takes into account a delay minimization policy. The author made an effort to lessen the service latency by thinking about the workloads placed on Fog nodes by Internet of Things programmers. They deliver the service based on the comparison's findings, and if the incoming job exceeds the threshold for cloud offloading, it is sent there instead of to the Fog. They compared the service to a minimum required quality in their policy. They compared the results, and then delivered the service accordingly. When formulating the delay-minimizing policy, the authors fail to account for communication lag.

In [8] It was pointed out that Chang et al. highlighted the challenges of optimizing energy utilization and tracking delay performance. They demonstrated an approach to computing the potential for work offloading while accounting for power consumption constraints. For the processing that occurred in mobile devices and Fog nodes, they used two distinct queuing models.

In [9] As part of their theoretical work on the topic, they present a layer model for interfacing devices and clouds with the web. The authors propose putting the programs in the cloud and in the Fog, cloud as alternate protections. Further, it has been demonstrated that a Fog computing structure is required, one that facilitates communication between the cloud and Fog, within the Fog, and between the Fog and the IoT devices.

In [10] Using energy consumption as a metric, Zaho et al. suggested an offloading technique for mobile devices that might be based on cloud computing or Fog computing. In order to maximize efficiency in terms of

energy use, they developed an algorithm for the computing set out above. When time is of the essence, it is possible to offload work from mobile devices to Fog rather than cloud computing. If that isn't an option, cloud storage can be used to back up information gathered from mobile devices. The writer took into account both energy use and total delay.

In [11] developed an algorithm to facilitate the sharing of resources between Fogs. They define a utility metric for Fog nodes in their Fog layer that takes into consideration the advantages of sharing resources through communication. Based on this criterion, the first generates a prioritized set of Fog nodes that can be used to couple preferences. Each Fog node will then send a pairing request to the other nodes in the Fog that it thinks might be a good match for it. On the receiving end, a target node will either accept or deny the request based on its preferences and the value it sees in similar requests it has already received. The main parameters on which Fog node make judgments are the communication cost between nodes, which can be influenced by time and location of the pairing, which is a limitation of this work. Also, while deciding how to divide up resources, they don't factor in quality of service considerations.

In [12] Hasan et al. introduced a plan to offload work for Internet of Things applications using a mobile ad hoc cloud computing architecture. Aura, an incentive and contract system, formed the basis of the proposed program. Using Aura, clients on mobile devices can build an impromptu Internet of Things cloud.

In [13] The control architecture for the available resources was presented by Zahoor et al. to be based on a Smart Grid (SG) scenario. Requests from smart devices like smart metres are scheduled by Fog node virtual machines using several approaches, including round robin, ant colony optimization (ACO), particle swarm optimization (PSO) [48], and

artificial hybrid bee colony. The ACO algorithm, inspired by bee colonies, performs better than competing methods. However, the proposed algorithms and load balancing mechanism are not detailed because the study's primary focus is on the Cloud-Fog architecture of the smart grid scenario.

In [14] suggested an architecture for migrating resources, with special attention paid to moving VMs around among Fog nodes. Maintaining VM availability during user migration is key to preserving QoS, and this can only be achieved by supplying the necessary management components to enable Fog's operation. They take it as read that the user's data and app components are stored in the VM (s). Users will experience no downtime or other performance issues with their applications throughout the migration. However, there is no explanation for how Fog can accommodate several migrations to the same Fog node, which could result in excessive load and the subsequent suspension of Fog services.

In [15] In their discussion of scheduling problems in cloud and Fog computing, Nguyen et al. To remedy the situation, they suggested implementing a time-cost-aware scheduling system. The work's goal was to find a good balance between the time and money needed to carry out the various tasks.

In [16] The Fog-cloud computing system's tradeoff between power consumption and transmission delay is studied. By breaking down the primary problem into three sub-problems of corresponding subsystems, the authors are able to formulate a workload allocation problem that suggests optimal workload allocations between Fog and cloud toward the minimal power consumption with the constrained service delay and solve it using an approximate approach.

In [17], the purpose of containers was taken into account by Yin et al. when they presented a method for scheduling tasks. To check if the Fog node's synchronized activities were selected appropriately and completed on schedule, a task-scheduling method was developed. Finally, a reallocation technique was introduced to reduce task/delay times, which improved container capabilities. Results show that the introduced reallocation strategy may effectively increase the number of jobs in Fog nodes while simultaneously reducing task delay.

In [18], to facilitate the timely preparation of various IoT and non-IoT based projects, Adhikari and Srirama had created a novel energy-efficient container-based scheduling (EECS) approach. The primary goals of EECS were to reduce the computational complexity and overall energy consumption of productive resource-using endeavours. In order to determine an appropriate compartment for each work with the least amount of delay, the suggested solution makes use of the accelerated particle swarm optimization (APSO) technique. The results of the experiments also reveal that the new methods are superior to the old ones in terms of resource use, computing time, temperature discharge, energy usage, and CO2 outflow.

In [19] have proposed a scheduling approach to cut down on both the processing time and delays experienced by Fog computing. Three policies are used to determine their schedules. The first policy has the Fog node randomly carry out an operation. The second strategy is based on choosing the Fog node that causes the least latency given the current condition of the system, and the third policy chooses the Fog node that still has the greatest available resources.

In [20] Hussein and Mousa suggested using an Ant Colony Optimization -based method to delegate work at Fog setting. The work is

developed with only the cost of transmission in mind. Previous researchers have tended to pay more attention to power usage and delay in transmission than to communication costs, as evidenced by the relevant literature. In our work, we have taken into account the time it takes to perform the computation itself, as well as the time it takes to offload the task in a Fog environment. The computational expense was overlooked.

In [21] Razaq et al. use service providers to offload the work that is caused by Internet of Things devices. They concentrated their efforts primarily on the transmission delay as well as the task's level of safety. However, they failed to take into account the delays that would be caused by processing and communication when offloading the work.

Author name	Proposed technique	Advantages	Limitations of the techniques
Deng et al.[4]	Power consumption and delay tradeoff for fog environment	Minimized power consumption in task offloading	Used only single point connection between cloud and fog
Chen et al. [5]	Mechanism for scheduling of job for IoT paradigm	Minimized the cost offloading the task from IoT applications	Communication delay is not considered for reduction of cost
Yousefpour et al. [7]	Delay minimization technique	Reduced delay for IoT applications	Computation and communication delay is not discussed
Change et al. [8]	Alternating direction method of multipliers technique	Optimized the energy for offloading the task in fog environment and considered the computational delay	Not discussed the communication delay in optimizing the energy
Zaho et al. [10]	Optimal energy consumption	Energy consumption minimized	Only single user system model is used

Hasan et al.[12]	Aura technique for task offloading	Memory consumption and energy consumption is minimized	No discussion about the communication and computation delay in reduction of energy consumption
Zahoor et al. [13]	Smart grid resource management	Analyzed the architecture for smart grid	No discussion about delay and scheduling
Nguyen et al. [15]	Time cost aware scheduling technique	Minimizes the operational and completing cost	Communication cost is not discussed
Canali & Lancellotti [16]	Data mapping on fog nodes through load and Latency	Reduces the communication latency	Computational latency is not clearly discussed
Hussein and Mousa [20]	ACO based task offloading	Minimizes the communicational cost	Computational cost is not considered
Razaq et al. [21]	Offloaded the task using service provider	Reduces the transmission latency	Computational and communicational latencies are not discussed

Table 2-1: Summery of Related Work

1.6 Thesis Outlines

In addition to chapter one, this thesis contains four chapters:

Chapter Two: present the theoretical background and basic concepts. This part contains review of Internet of Things, framework of Fog computing, benefits of Fog computing, Fog computing applications, latency and power consumption with their measurements, optimization techniques and algorithms.

Chapter Three: This chapter presented the proposed system and illustrates the practical stages of the system and explains the proposed algorithms system.

Chapter Four: This chapter show the results and it evaluated the used system methodology.

Chapter Five: This chapter presented the results' conclusion, also it described future works.

Chapter Two

THEORETICAL BACKGROUND

2.1 Introduction

The Internet of Things (IoT) is an emerging and challenging field for researchers. IoT is a network of general objects which are embedded with technologies that helps to communicate and interact within themselves and external environment. This in-turn provides intelligence to the objects to make people life comfortable [22].

Fog Computing is a new architecture to migrate some data center's tasks to the edge of the server. The Fog computing, built on the edge servers, is viewed as a novel architecture that provides the limited computing, storing, and networking services in the distributed way between end devices and the traditional cloud computing Data Centers. It provides the logical intelligence to the end devices and filters the data for Data Centers. The primary objective of Fog computing is to ensure the low and predictable latency in the latency-sensitive of Internet of Things (IoT) applications such as the healthcare services [23].

2.2 Concepts of Internet of Things (IoT)

The Internet of Things (IoT) is denied as a paradigm in which objects equipped with sensors, actuators, and processors communicate with each other to serve a meaningful purpose. Today the Internet has become ubiquitous, has touched almost every corner of the globe, and is affecting human life in unimaginable ways. However, the journey is far from over for this intelligence and interconnection, IoT devices are equipped with embedded sensors, actuators, processors, and transceivers [24].

IoT is not a single technology; rather it is an agglomeration of various technologies that work together in tandem. Sensors and actuators are devices, which help in interacting with the physical environment. he data collected by the sensors has to be stored and processed intelligently in order to derive useful inferences from it. Note that the term sensor; a mobile

phone or even a microwave oven can count as a sensor as long as it provides inputs about its current state (internal state + environment) [24].

2.2.1 Internet of Things (IoT) Architecture

There is no single consensus on architecture for IoT, which is agreed universally.

A Three- and Five-Layer Architectures

The most basic architecture is a three-layer architecture as shown in Figure (1-1). It was introduced in the early stages of research in this area. It has three layers, namely, the perception, network, and application layers.

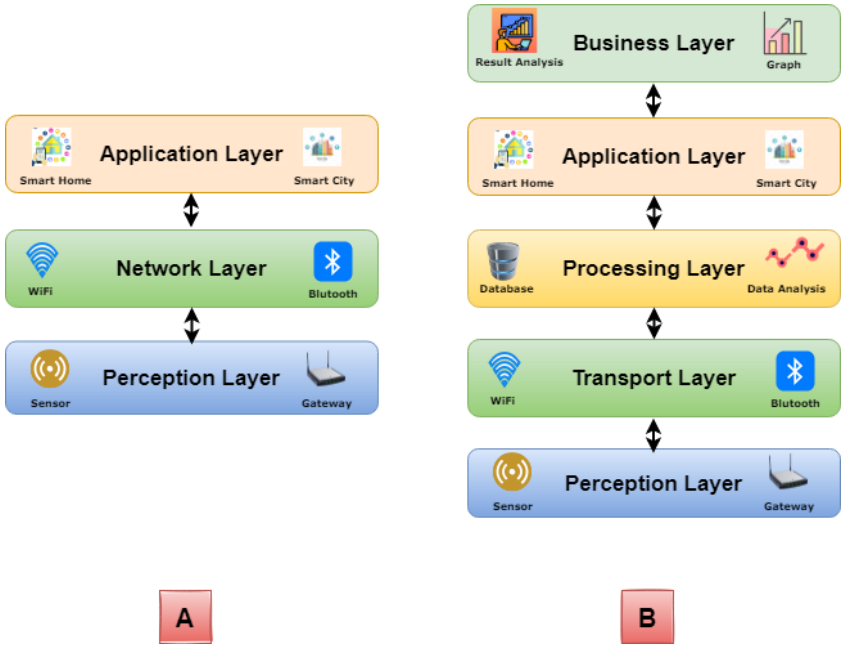


Figure 2.1: Architecture of IoT (A: three layers) (B: Five layers)

i. The perception layer is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identities other smart objects in the environment [25].

ii. The network layer is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data [25].

iii. The application layer is responsible for delivering application specific services to the user. It defines various applications in which the Internet of things can be deployed, for example, smart homes, smart cities, and smart health [25].

The three-layer architecture defines the main idea of the Internet of things, but it is not sufficient for research on IoT because research often focuses on finer aspects of the Internet of things. that is why, we have many more layered architectures proposed in the literature [25].

One is the Five-layer architecture, which additionally includes the processing and business layers. The Five layers are perception, transport processing, application, and business layers [26]. he role of the perception and application layers is the same as the architecture with three layers. We outline the function of the remaining three layers.

i. The transport layer transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC [27].

ii. The processing layer is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules [27].

iii. The business layer manages the whole IoT system, including applications, business and profit models, and user's privacy [26].

2.3 Fog computing

In the last few years, cloud computing has provided many opportunities for companies to supply a wide range of computer services to their customers. Currently, the cost model in return for use cloud computing for the owning and management of private data centers that customers encounter with web applications and batch processing, it is an efficient alternative.

Cloud computing releases companies and end-users from detailed specifications, such as storage resources, computational constraints, and network communication costs. However, this feature makes it difficult for time-sensitive applications. When IoT technologies and devices bring into people's lives, current cloud computing paradigm can hardly compensate for the need for movement support, location awareness and lack of time [28].

The computing of Fog presented to address the upper network, those computing present the time shortage, situation notice, expand the QoS to run and also real time of applications. Common illustrations contain artistic automatism, transportation, trigger and sensor networks. Also, its fresh genesis protects discrepancy like systems of Fog that contain the system of last client, reach cases, marchers of edge, options. The sample of Fog is good-shaped to assay great real-time data and protect for dense distributed data set cases, also presents the entertainment, private computers, advertising, any longer applications [29]

In summary, the Fog computing has the following characteristics:

Sensitive data is analyzed on the same source or destination device, and close to the collection location, which is used to prevent large amounts of data from being sent over the network.

Operation is performed on data in milliseconds.

A specific section of the data based on policy defined will be sent to the cloud computing part. There are several challenges in the design and implementation of IoT systems, IoT devices are constantly creating data, and, data analysis and processing should be done quickly in most cases.

For instance, imagine that if the temperature of the environment exceeds a certain limit, then the necessary measures should be taken quickly, therefore it is possible the time spent sending and analyzing data in data center, users incur costs and risks. Considering various types of IoT services, the computational requirements can be categorized as follow:

a. Very little Delay

Data analysis at a location near the sensors and IoT devices will help to reduce the delay in executing commands and data analysis.

b. Optimal use of bandwidth

Due to the large amount of data sent to such systems, performing many calculations and analyzes near the place of production and collected data, it will prevent large amounts of data sent to the network, which will play a significant role in reduces network costs.

c. Reduce security concerns

Considering the minimal data transmission on the network, there will be fewer security risks for the system. Existing cloud computing systems cannot remove many of the concerns about IoT systems. And as noted, moving a large amount of data in a network apart from cost and security

risks will cause relatively long delays. In addition, many cloud computing systems use the IP protocol to exchange data while some existing devices and sensors do not support this protocol, adding an IP protocol stack increases the price of many devices and terminals. Therefore, it is logical that the data be analyzed near the device. This computing method is in fact a kind of development of cloud computing [30].

The data that are sensitive to delay instead of sending to the center, they are analyzed at the same location and if necessary, you can run certain commands, For example, imagine that you want the command to close the door of a building in case you see some security issues or when viewing a danger on the rails export brake train command, even You can turn on air conditioning system when increase the temperature of your home engine, All examples mentioned are commands to be executed without spending time and sending data to cloud computing centers and waiting to get the response will bring irreparable damage[27].

2.3.1 The architecture of Fog computing

Fog computing is a technique that moves a few data center operations to the network edge. It provides less storage, processing, and network services in a shared approach among cloud computing data centers and end devices. The primary goal of Fog computing is to provide minimal and predictable latency for time-sensitive IoT functions [31]. Distinct academics have produced various Fog reference architectures, which are based on diverse topologies appropriate to user applications and services.

This research covers FC designs, addresses the most fundamental proposed architectures to improve FC, and identifies appropriate answers to most issues. Figure (2.2) depicts the general architecture of Fog Computing,

which includes three levels that are commonly used to define Fog Computing's perspective.

d. The general architecture of Fog Computing

The first layer is the IoT layer, which is responsible for implementing numerous IoT applications/services in various scenarios such as smart cities, smart homes, streaming films, online games, and the smart grid.

The second layer is the FC layer, which is where the FC servers and devices are located due to the high demand for processing and storing data.

The third layer is the core cloud server, which stores all of the necessary data to serve future customers' requests.

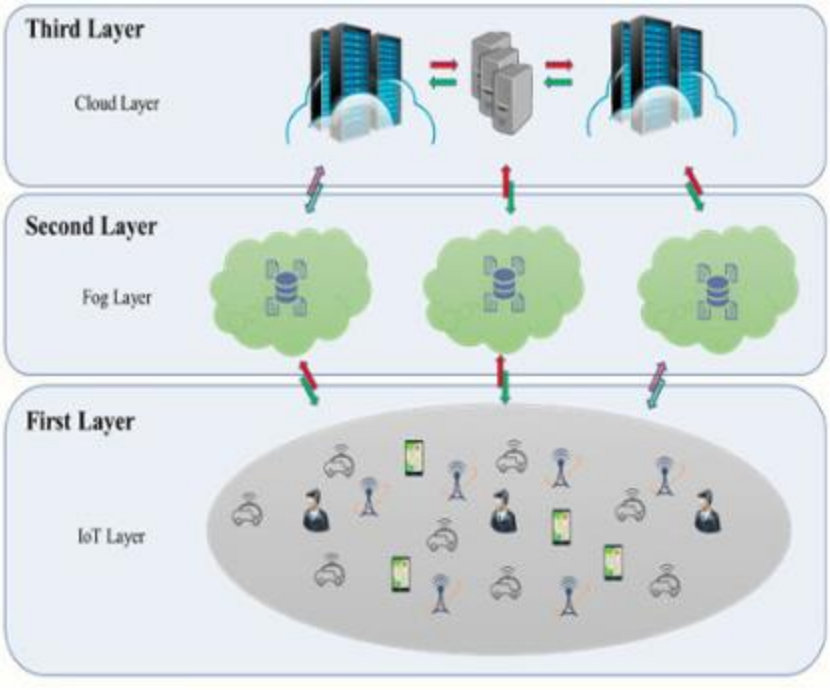


Figure 2.2: Fog Computing Architecture[25]

e. Fog Computing architectural or local architecture

A Fog computing reference framework is being developed, which consists of seven tiers, namely layer1: virtualized and physical sensor, layer 2: Fog devices, servers, and gateway, layer 3: monitoring, layer 4: preprocessing and post-processing, layer 5: storage and resource management, layer 6: security and layer 7: application [32] Figure (3) shows the Fog architecture

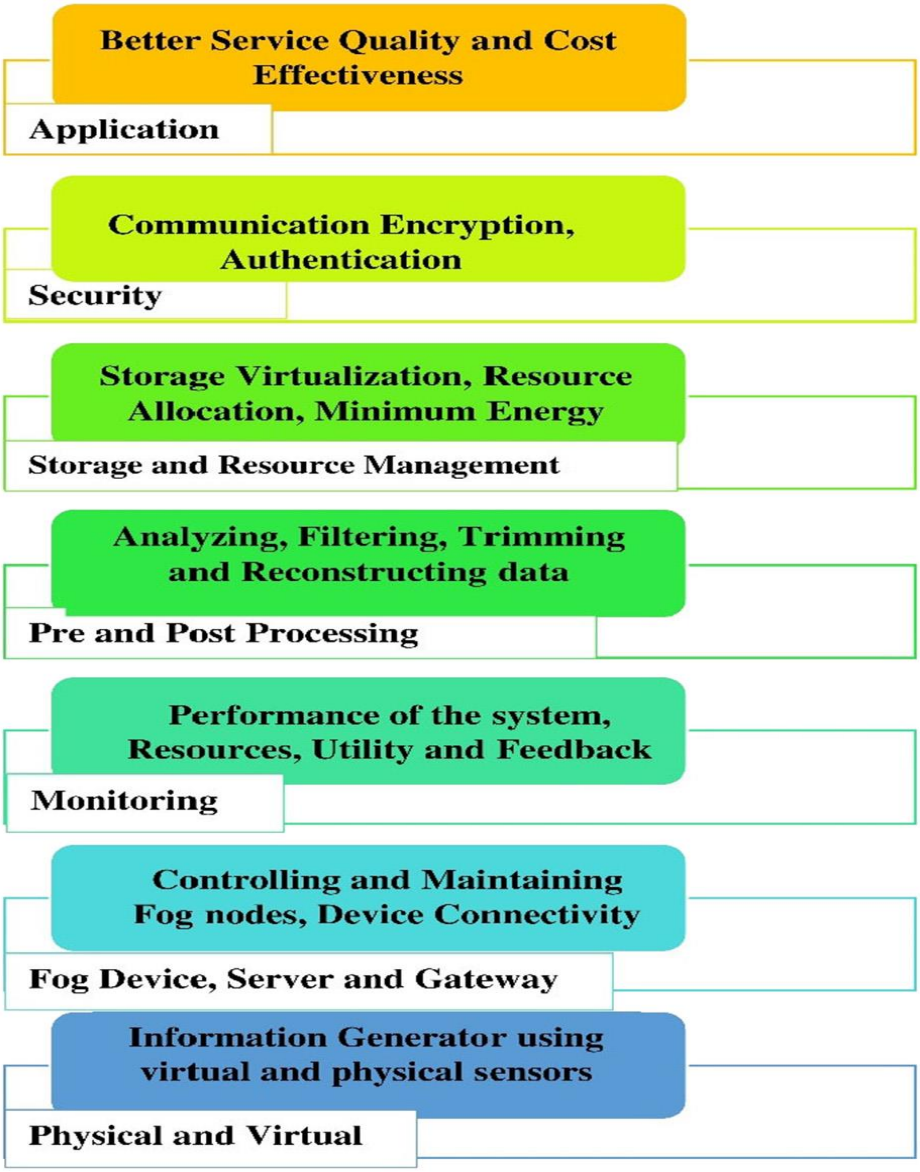


Figure 2.3 Fog Compting Local Architecture [30]

plane 1: Physical and virtualized sensor

Many types of information supplied by the sensors are the primary information generators in Fog Computing [33]. This data could be generated by a variety of technologies, including intelligent houses and devices, CCTV and traffic surveillance systems, automated driving vehicles, humidity and temperature sensors, and so on. For example, in a traffic intelligent surveillance system, we need to acquire continuous traffic status of all routes from various sensors, devices from pathways, and roadside CCTV monitoring to help manage traffic lights. It is critical to forecast future traffic demand by collecting data from various GPS sensors. When there is a traffic collision, virtual sensors, in addition to physical sensors, play a significant role [34]. It seems unlikely that a single sensor will be able to determine whether to close the road or allow traffic to continue. The path may have numerous lanes, one of which may be impacted by this incident, while the alternative path may allow road traffic movement to continue. Nonetheless, the capacity to regulate traffic will be hampered as a result of this occurrence. In this case, a virtual sensor might provide an instant resolution on traffic rerouting, multiplexing, road environments, and so on. As a result, the physical level incorporates both virtual and physical sensors.

plane 2: Fog devices servers and gateway

A Fog server, Fog device, or gateway can be an IoT or an independent device [35]. Nonetheless, because it oversees a large number of Fog, the Fog server must have a better setup than the Fog gateway and devices. A Fog Computing device framework. Many aspects are involved in making a Fog server run, such as hardware configuration, network connectivity, devices it can manage, and so on. The Fog server's role is dictated by its IoT component. A network of virtual and physical sensors

is linked to Fog devices. Similarly, a group of Fog devices will be linked to the Fog server. With Fog devices, the Fog server must have improved computing and storage capabilities. It should also have superior compute and storage capabilities in comparison to the Fog device. When necessary, a specific set of Fog devices connected to the same server can broadcast with one another. Other Fog groups may determine the computation of a few applications in an intelligent transport use case. For example, if a request is required to search for a fuel-efficient path, data regarding various Fog device groups or sensor groups may be required. Computation on various servers and Fog devices is required to reach a good choice. The device level and Fog server are responsible for controlling and maintaining data on software and hardware setup, as well as server and device network connectivity. This level also governs the processing requirements of numerous apps. Processing requirements are determined by information movement and the total number of devices linked to IoT associated with Fog devices, as well as the total number of Fog devices linked to Fog servers. [33]

At this level, connectivity between Fog servers is maintained. For example, a Cisco router can be used as a Fog device, and the Cisco information service of Fog devices can be used as a Fog server. [36]

plane 3: Monitoring

The monitoring level [33] keeps track of the system's and resources' performance, as well as utility and feedback. As the relevant resources during an operating system monitoring, facilities are chosen. Various procedures take place in scenarios involving smart transportation systems. A scenario could arise in which the availability of resources negates the need for calculations or storage on the Fog device. Similarly, the identical scenario can occur on the Fog server. To deal with such conditions, Fog-

side devices and servers will seek assistance from other peers. As a result, the system monitoring components will make these decisions efficiently. The component of resource demand analyzes current resources and forecasts future resources based on user behaviors and consumption. The procedure must manage any potentially dangerous conditions in which a failure could occur. The performance of the Fog system can be signaled by the prediction monitor based on network load and resource availability. This is essential since it is utilized to maintain the required QoS qualities in SLAs (Service Level Agreement). If SLAs are consistently violated, the system's cost will rise as a result of the penalty. This cannot be avoided by performance prediction, but it will reduce total SLA breaches by projecting the structure's utilization and performance.

Plane 4: pre and post processing

This level focuses on data analysis of basic and advanced data and includes a number of components. Its purpose at this level is to gather data by analyzing, filtering, trimming, and rebuilding the data as needed. Once the data has been analyzed, the data flow component determines whether the data should be kept locally at the Fog or in the cloud for long-term storage [35]. The extreme challenge in Fog computing is that information is processed at the edge and only a little quantity of information is saved. The fundamental idea is to transfer data that is regularly used to Fog servers and data that is rarely or not used for an extended length of time to the cloud. Data is generated from various sensors in a smart transportation application. This data will be investigated and processed in order to obtain the generated data. This generated data may or may not be useful. In a few cases, storing all of this generated data is not optimal. For example, depending on the application requirements, the sensor produces information every second, from which the mean value of the data within a

minute and hour is saved. In this manner, data is reduced while retaining a great amount of storage. In another example, if the information values do not change over time but performance suffers, the number of readings collected is lowered. A vast amount of data could be manipulated in this manner. Although the accuracy would not be perfect, the requirements would be met. Another component in this level is data reconstruction, which handles imperfect data generation from sensors and incorrect tolerance. This component also ensures that if one or more sensors fail, the information is reformatted based on the information pattern, so preventing application failure and other interruptions.

Plane 5: storage and resource management

The storage module is in charge of data storage using storage virtualization. The component known as data backup is in charge of ensuring data availability and data loss. Storage virtualization refers to a group of devices in a network that are responsible for storage and function as if they were individual storage devices. This individual storage device is simple to manage and maintain. The fundamental advantage of storage virtualization is that it reduces the cost of hardware and storage, resulting in improved corporate functionality. It also reduces the complexity of storage. There is a potential that the storage will fail [37], thus it is critical to backup data. The data backup module is in charge of customizing data backup strategies on a regular basis. The resource management level includes components that are in charge of distributing resources, scheduling them, and dealing with energy-saving issues. There is a component called dependability that ensures the application's scheduling and resource allocation is reliable. When the demand for resources is great during peak hours, the scalability of the Fog resources is ensured. The cloud platform achieves horizontal scalability, whereas the Fog platform focuses on both vertical and

horizontal scalability [38]. As storage is to be carried out in a distributed resource system, a major issue arises in the distributed resources where the process of allocation is involved. The component of resource allocation allocates, de-allocates, and reallocates all related concerns. Another critical issue is that numerous programs use the Fog environment at the same time, which necessitates proper application scheduling. The application scheduling component is in charge of the application's numerous goals. This level comprises the energy-saving component that is utilized to efficiently manage all resources. This manages to keep operational expenditures to a minimum. The system's dependability is handled by the reliability component, which focuses on various reliability measures and metrics. The metrics might be evaluated based on the following parameters: data center redundancy, Fog node redundancy, mean time between Fog node failures, and critical failures of IoT applications in the meantime. The Fog system is a complicated architecture that focuses on the upkeep of all IoT devices, Fog nodes, Fog servers, and clouds.

Plane 6: Security

The security level maintains all security issues, such as communication encryption and secure information storage. This level also protects the Fog users' personal information. The Fog environment is planned to be established as a utility system, similar to a cloud environment. In the cloud environment, the user requests all services from the cloud by connecting to it, whereas in the Fog environment, the clients request all services from the Fog system, while the Fog's middleware controls and maintains all connections with the cloud. As a result, the user who want to associate with a service must be permitted. As a result, the validation component is in charge of sending authentication requests to all users in the Fog [39].

It is critical to maintain security by encrypting different communications in order to minimize hostile user intrusion. The encryption component will encrypt various connections between IoT devices and the cloud. Because the majority of the Fog components are linked via a wireless connection, security is critical. Users' data should not be leaked in the Fog environment; it is critical to protect user data privacy. A few smart city or smart house services have difficulties since they contain user-related data that should not be published. In most cases, users accept security policies without giving them a thorough reading. As a result, it is critical to evaluate these types of services in which the user's privacy is involved.

Plane 7: Application

When Fog computing was first established to serve IoT applications [40], various apps based on Wireless Sensor Network (WSN) began to support it. Almost all applications that suffer from latency began to take advantage of the Fog environment. These included any utility services that could be integrated with Fog to give better service while reducing costs. In an application where the system uses Augmented Reality, Fog infrastructure can be used because it will modify the existing environment in the future. The requirements of real-time processing using augmented reality can be met by a Fog environment, which can lead to long-term improvements in many augmented reality systems.

2.3.2 Benefits of Fog computing

a. Reducing network traffic:

Cisco estimates that there are currently 50 billion devices in the world, the number will reach 90 billion by 2025. Billions of mobile devices, such as smartphones and tablets, are currently used to generate, receive, and send data and provide an option to bring computing capabilities closer to where

the tools are located, instead of sending all the network data to the centralized data centers

Depending on the frequency adjusted, sensors may collect data for a few seconds once. Therefore, sending these small data to the cloud is neither efficient nor reasonable. So cloud computing here is useful by providing a platform for filtering and analyzing the production data of these near-edge devices, as well as creating local views of the data. This greatly reduces traffic to the cloud [41].

b. Suitable for IoT tasks and queries:

By increasing the number of smart devices, most requests belong to tools. Therefore, such requests can be serviced without the help of global information in the cloud. For example, the Edomondo sports follower application allows a user to do this which will put together similar sports players. Due to the local nature of the typical requests created by this program, it creates the sense that requests are processed instead of cloud infrastructure in Fog. An example can be a smart device that records only 100 meters' distance events [42].

In Fog computing shortens communication distance by reducing the physical distance, by processing closer to the edge of the network. Low Delay Requirement: Mission critical applications require real-time data processing. One of the best examples of such programs is the robotic cloud that controlling fly-by-wire spaceships or auto-brakes. For a robot, motion control is dependent on data collected by sensors and control system feedback. By executing the control system on the cloud, the sensor loop of the stimulation process is slowed down or it will be unavailable in the event of communication failures. This is where the computing in the Fog plays a role with performing the required processing for a control system that is

very close to the robots, and therefore enables real-time response, and can be useful. Scalability: Even with unlimited virtual resources, the cloud may be bottlenecks if all of the production data continues to be sent by the final tools. Since the Fog computing goal is processing input data in its vicinity, it reduces processing loads in the cloud, and therefore solves the scalability concerns caused by the increase of the end points [43].

2.3.3 Fog computing applications

There is a wide range of applications that use the computing model in the Fog:

a. Health care

Cao et al. suggested FAST, which, with the help of the Fog computing, is a distributed analytical system for monitoring the fall of hit patients (apoplexy). The researchers have developed a series of real-time fall detection algorithms, such as speed-based algorithms and time series analysis methods, and also filtering techniques to facilitate the process of fall detection. The investigators expanded a set of real-time fall detection algorithms, like speed - based algorithms and the methods of time series analysis, filtering methods for facilitate the process of fall detection. They planned a real-time fall detection system according to the computing of Fog that distributed the fall detection tool among the systems of cloud and edge. The offered device presents upper sentimentality, proper characteristic while checking by real information. At the same time, both energy consumption and period of reply are near the techniques of existing effective [44].

The Fog computing other application offered by Stantchev and his colleagues in healthcare [45]. They presented an architecture for intelligent of smart healthcare that contained a role model, the architecture of cloud

layer, a layer of Fog computing for presenting an effective architecture for applications of health and healthcare to the elderly. The layer of Fog has expanded architecture with presenting low latency, substitution protect, notice of location, security mensuration. The healthcare application process flow is designed by utilizing the business process modeling notation, after that located to tools by a service-oriented method. The model's accuracy by a use-case is presented as a template to the infrastructure of sensor-based healthcare.

b. Augmented Reality

The applications of augmented reality are too sentimental for delay, therefore, even short delays in reply are able to interrupt the act of client. So, computing of Fog has ability for doing a big role in the augmented reality. Zao and his colleagues made a game that has effect on each other of augmented brain computer in order to computing of Fog and information communicated. While someone do the play, the raw data, which are gathered with the sensors of EEG is produced and graded for fixing the player brain's location. The classification of the brain state is one of the tasks which is associated with the high computational load of signal processing tasks. But this should be done in real time. It will use cloud servers as well as Fog servers, a hybrid that enables the system to perform continuous real-time classifications on Fog servers, while classifying models adjusts regularly collected by sensors on cloud servers based on EEG readings. In [42] He et al. proposed a coverable cognitive auxiliary system based on Google Glass tools that helps dimwit people. Due to the inherent nature of cognitive tools with limited resources, the workloads with high computing this program required to transfer to the external server. However, this transfer should provide contraction and real-time responses, and there is no possibility of doing tasks that disrupt the task of

the user. Transferring high-computing tasks into the cloud causes significant delays, so researchers use the nearby tools. These tools can communicate with the cloud for to do tasks with different delays, such as error reporting and submit events. The above works are common uses of Fog computing, in which Performing an analysis of the critical delay time at the edge and calculating the delay time in the cloud. So the description of the Fog is performed as an expanded format from the cloud [46].

c. Caching and preprocessing

Zhu and his colleagues suggested the edge servers' usage for expanding the networks performance. Clients are joining to net by utilizing the boxes of Fog. So, clients gave every demand of HTTP, who are using an equipment of computing of Fog. The equipment of computing of Fog do the optimizations amount, which reduce the period number that each client must hold up for the needed pages of web for load. Aside the common optimizations, like caching the elements of HTML, reforming the structure of page of the web, declining the size of the pages of the web, equipment of edge and present optimizations, which considered of client manner and situations of net. Such as, in point of net congestion, equipment of edge might present low- resolution draws to reach adaptation times of reply. Furthermore, the equipment of edge is able to control the user machine performance, based on the browser's time, forwards the draws at a suitable resolution [47].

The way of joining computing of cloud and Internet of Things is one of the main applications of computing of Fog. This aggregation is necessary, also occasions by these challenges.

Sorting the information is one of the serious important challenges. This sorting or information pre-progressing in Internet of Things surroundings

is essential in previous of sending them to cloud through the great number of information produced in these surroundings. Sending a great number of raw information to the cloud guides to the congestion in the center of net and also the core of information.

In order to cope with the challenges of preprocessing, Aazam et al. proposed a smart gateway based on communications for the development of IoT for cloud computing.

Data generated by IoT tools are sent to the smart gateway, either one-hop or via sink nodes (multi-hop). Smart gateway manages the preprocessing needed before sending data to the cloud. In the proposed architecture of the researchers, Fog computing services will assist in intelligent gateways to operate on IoT data in a content-aware and sensitive to delay method. Such a communication method facilitates the creation better performance and more powerful method in IoT applications [48]

2.3.4 Fog computing challenges

1. Quality of Service (QoS)

It is the important metric of Fog computing that can be of quality of service was determine by four aspects: reliability, capacity, connectivity and delay

a. Reliability: Madsen et al. examine the reliability requirements of clustering computing, grid computing, cloud computing, and sensor networks in preparation for a discussion of Fog computing dependability [49]. Normally, dependability can be increased by performing frequent check-points to resume after failure, rescheduling failed tasks, or replicating to use parallel execution. However, check pointing and rescheduling may not be suitable for the extremely dynamic Fog computing environment due to latency and inability to adapt to changes.

Replication appears to be more promising, although it requires numerous Fog nodes to collaborate.

b. Capacity: Capacity is divided into two categories:
1) network bandwidth
2) storage capacity.

It is critical to explore how data are stored in Fog networks in order to achieve high bandwidth and effective storage utilization, as data locality for computing is critical. Similar efforts exist in the contexts of cloud [50] and sensor network [51]. However, in Fog computing, this problem faces new obstacles. A Fog node, for example, may need to compute on data scattered over several adjacent nodes. The computation cannot begin before the data aggregation is completed, which adds significant latency to services. To address this, we may use the user mobility pattern and the service request pattern to place data on appropriate Fog nodes in order to either reduce operational costs, latency, or increase throughput. Data placement in a Fog and cloud federation necessitates rigorous thinking as well. The difficulties stem from determining how to design the interaction between Fog and cloud to meet diverse workloads. Because of the dynamic data placement and big total capacity volume in Fog computing, we may need to adapt the search engine to handle search queries of material spread across Fog nodes [52] [53]. It would also be very interesting to rebuild the cache on the Fog node to take advantage of temporal locality and larger coverage in order to conserve network bandwidth and reduce latency, given the existing work on cache on end device [54] and cache on edge router.

c. Connectivity: Network relaying, segmentation, and clustering provide new potential for cost reduction, data reduction, and connection expansion in a heterogeneous Fog network. Because of the coverage of rich-resource Fog nodes, an ad-hoc wireless sensor network, for example, can be partitioned into many clusters (cloudlet, sink node, powerful smartphone, etc.). Work [54] provides an online Access Point association technique that achieves not only low throughput but also low computational cost. Similarly, the end user's choice of Fog node has a significant impact on performance. With restrictions such as delay, throughput, connectivity, and energy consumption, we can dynamically pick a subset of Fog nodes as relay nodes for optimization goals of maximum availability of Fog services for a specific area or a particular user.

d. Delay: Delay Applications that require real-time streaming processing rather than batch processing, such as streaming mining or complicated event processing, are examples of latency-sensitive applications that require Fog computing. To address the latency requirement, K.Hong et al. [55]offer a Fog-based opportunistic spatio-temporal event processing system. Their method predicts future query regions for travelling consumers and initiates event processing early in order to provide timely information when consumers arrive at the future areas. Work [56] introduces RECEP, which uses overlapping data interests and tolerable errors to reuse computation and reduce resource requirements. RECEP improves scalability and amortizes delay in mobile CEP systems.

2. Energy Minimization

Because Fog environments need the deployment of a large number of Fog nodes, computation is fundamentally distributed and may be less energy-efficient than the centralized cloud model. As a result, reducing energy consumption in Fog computing is a significant task. Deng et al. [57]investigate the trade-off in a Fog computing system between power consumption and delay. They formalize the challenge of dividing workloads between the Fog and cloud by modelling the Fog system's power consumption and delay functions. Simulation results suggest that Fog computing can drastically reduce communication latency while consuming slightly more energy. Do et al. [58]investigate a related issue, namely cooperative resource allocation and reduction. In Fog computing, energy consumption for video streaming service. Because there are so many Fog devices, a distributed solution has been proposed to alleviate performance and scalability difficulties. The approach is based on proximal algorithms, which are an effective way for dealing with distributed convex optimization problems. The suggested approach has a high convergence rate and a high solution quality.

3. Interfacing and programming model

We need a uniform interfacing and programming architecture to make it easier for developers to move their apps to the Fog computing platform. The reasons for this are as follows:

1) application-centric computing will be an important Fog computation model, in which components in the environment will be application aware and allow appropriate optimizations for different types of applications [59].

2) it is difficult for developers to orchestrate dynamic, hierarchical, and heterogeneous resources to build compatible applications on diverse platforms. Hong et al. [60]offer a high-level programming model for future Internet applications that are large-scale geographically dispersed and latency sensitive, with on-demand scalability. Their method, however, is built on a tree-based network architecture in which Fog nodes have fixed placements. As a result, more generic designs for varied networks where Fog nodes are nodes with dynamic mobility may be required.

4. Computation offloading

Computation offloading can help mobile devices overcome resource limits since some computation-intensive processes might benefit from offloading in application speed, saving storage and battery life. Existing work on mobile cloud computing computation offloading can be categorized into six metrics: objectives, granularity, scheme, adaptation, distributed execution, and communication [61]. While there has been a lot of research towards compute offloading in the context of cloud computing and mobile computing [62] [63] [64] [65], we will only look at a few of them in this work. MAUI [63] propose code offloading and profile offloading methods for making judgments on future invocations based on network connectivity, bandwidth, and latency changes. It necessitates developers manually annotating methods that can be offloaded. CloudCloud [64]use a static code analyzer to identify potential migrate/merge points in program bytecode. ThinkAir [46] advances to cloud elasticity and scalability, increasing the potential of mobile cloud computing by parallelizing method execution over numerous virtual machine (VM) images. COMET [66] makes use of distributed shared

memory and VM synchronization primitives to enhance smartphones or tablets with network-accessible machines. The key issues in Fog computing offloading are dealing with dynamic. The dynamic comprises three components.

1- Access to radio/wireless networks is highly dynamic.

2- The Fog network's nodes are highly dynamic.

3- The Fog's resources are highly dynamic.

The Fog and cloud federation truly presents us with a three-layering structure: Device-Fog-Cloud. In such infrastructure, computation offloading faces new difficulties and opportunities. There are questions such as which granularity to choose for offloading at different Fog and cloud hierarchy levels; how to dynamically partition an application to offload on Fog and cloud; and how to make offloading decisions to adapt to dynamic changes in network, Fog devices, and resources, among others [67].

5. Security and privacy

There are currently few works that focus on security or privacy issues in Fog computing. However, some areas have been intensively researched, such as virtual machines and hypervisors [68] and cloud computing [69].

a. Authentication: As biometric authentication, such as fingerprint authentication, face authentication, touch-based or keystroke-based authentication, and so on, becomes more prevalent in mobile computing and cloud computing, using biometric-based authentication in Fog computing will be advantageous. The biggest security concern of Fog computing, according to I. Stojmenovic et al [70], is authentication at multiple levels of Fog nodes. While a PKI-

based solution could overcome this problem, we believe a trusted execution environment (TEE) technique has potential in Fog computing [71] [72]. We can also utilize measurement-based methods to filter fraudulent or unqualified Fog nodes that are not in the neighborhood of end users to reduce authentication costs [73] [74]

b. Access control: Control of access has shown to be a reliable solution on smart devices [75] and in the cloud [76], maintaining system security. S. Yu et al. [76] extend data owner access control into the cloud by combining approaches from different encryption systems to construct an efficient fine-grained data access control in the setting of Cloud Computing. Work [77] proposes policy-based resource access control in Fog computing to facilitate secure collaboration and interoperability among heterogeneous resources. In Fog computing, we can also ask how to build access control that spans Client-Fog-Cloud to fulfil the goals and resource restrictions at various levels.

c. Intrusion detection: Intrusion detection techniques have been used to cloud infrastructures in order to combat attacks such as insider assaults, flooding attacks, port scanning, and VM or hypervisor attacks [78]. These intrusion detection systems can be placed on a host machine, a virtual machine, or a hypervisor to detect malicious behavior by monitoring and analyzing log files, access control policies, and user login information. They can also be used on the network to detect malicious activity like denial-of-service (DoS), port scanning, and so on. It opens up new avenues for research into how Fog computing might aid in intrusion detection on both the client

and centralized cloud sides. Implementing intrusion detection in a geo-distributed, large-scale, high mobility Fog computing system is one of the problems.

d. Privacy: Nowadays, users are concerned about the risk of data, location, or usage leaks on the Internet. Many privacy-preserving techniques have been proposed, including the cloud [79] [71], smart grid [80], wireless network [81], and online social network [82]. Because processing and storage are sufficient for both sides of the Fog network, privacy-preserving algorithms can be executed between the Fog and cloud, but those algorithms are typically resource-prohibited at end devices. Edge Fog nodes typically collect data generated by sensors and end devices. Techniques such as homomorphic encryption can be used to enable privacy-preserving aggregation without decryption at local gateways [83]. Differential privacy [84]can be used for aggregation and statistical inquiries to assure non-disclosure of private.

2.4 Latency

Latency is a measure of delay. Latency is the time it takes for data to travel across a network. It is typically measured as a round trip delay the time it takes for information to travel to its destination and back. The round trip delay is important because a computer using a TCP/IP network sends a limited amount of data to its destination and then waits for an acknowledgement before sending any more. As a result, the round trip delay has a significant impact on network performance. Typically, latency is measured in milliseconds (ms) [85].

Network latency is a fundamental feature of any network channel, including end-user broadband Internet connections, roads between

two servers in a Content Delivery Network, paths between two enterprise network locations, and so on. This research focuses on latency as it relates to broadband service end consumers, although many of the concepts apply to other network channels as well. The less latency a network or application has, the more "responsive" a service appears to an end user. The more delay (or lag) there is, the worse it feels. Low latency is critical in every application that involves users interacting with each other, a device, or an application, and it will become much more vital as new cloud-based applications, augmented reality, virtual reality, and other new application types develop. However, minimizing delay significantly improves all existing user apps [43].

Because a data packet cannot be sent from its origin to its destination in an instant, the network latency metric measures the total delay experienced by a minimal packet as it travels through many different network nodes along an end-to-end path to reach its destination. This metric can often be narrowed down for ISPs to separately measure downstream latency and upstream latency to gauge network delays encountered when sending data in the downstream direction through the network to the end user. or in the opposite manner, upstream from the end user to the network's core. Understanding the network's operational latency (described further below) often entails measuring the delay across different network segments for network operators in general (e.g., the home network, the access network, the metro core, etc.)

2.4.1 Latency Time Module

$CommL_{fog}$: is the communication time taken to offload the data from sensor Si to Fog node fi and it is calculated as

$$CommL_{fog} = \frac{1}{S_f} + \frac{A_f}{2S_f(S_f - A_f)} + \frac{X_p}{B_f} \qquad \textbf{(2.1)}$$

$CompL_{fog}$: is the computation time for data at Fog node fi and it is calculated as:

$$CompL_{fog} = \frac{1}{S_f - A_f} + \frac{A_f * CPU_i * X_p}{C} \qquad \textbf{(2.2)}$$

The total latency time (TotL) taken to offload task calculated is calculated from (1) and (2)

$$TotL = CommL_{fog} + CompL_{fog} \qquad \textbf{(2.3)}$$

When*:*

***commL*$_{fog}$:** Communication Latency

***compL*$_{fog}$:** Computation latency

S_f : Service rate of fog

A_f : Arrival rate of data

B_f : Fog network bandwidth,

***X_p*:** Size of data packet

***C*:** CPU Speed of fog

2.5 Power Consumption

The amount of energy used per unit of time is called "power consumption." In digital systems, the amount of power used is very important. The amount of power used by portable systems like cell phones and laptops determines how long their batteries will last. A typical smartphone battery packs around 10 watt-hours (W-hr) of energy, enough to power the device at 1 watt (W) for 10 hours, 2 W for 5 hours, and so on [86] .

The difference between energy and power must first be established. Power is the rate of energy use over a given time period. The use of electricity to produce heat. However, energy usage is the single most important factor in determining battery life. Most of the time, we will just refer to energy and electricity use collectively as "power," only making a distinction when absolutely essential. The official word for low power applications is power consumption measured in watts (often milliwatts, mW), while current consumption measured in amperes (typically milliamperes, mA) is sometimes used instead [87]. As the power is just the operating voltage multiplied by the operating current, this is simple for fixed voltage operations but becomes more difficult to predict when employing batteries whose voltage varies over time and under varying loads. Frequently, power consumption is not what concerns. Joules, or micro joules (J), are the unit of measurement for energy consumption, and they show how much juice is truly needed to get something done [88]. The amount of energy used will be calculated by summing the power consumed during the time required to complete the task at hand. Again, this would be a straightforward multiplication of power usage and time if the signal were static, but dynamic signals necessitate more nuanced analysis. In the case of a current-limited power source, such a Lithium-ion coin cell battery, power consumption becomes very important. These batteries are common in handheld sensors and smart devices but can only source a few mA current peaks before they are broken. If you try to draw a greater peak, the battery's capacity will decrease over time, which could affect the battery's output voltage. In situations where the available current can handle the peak demand, the power draw won't be a concern [89].

All developed and emerging countries' economies now depend heavily on the Internet. Many countries have benefited greatly from the virtual cycle

of telecommunications advancements enabling economic growth, which in turn encourages growth in telecommunications infrastructure. This cycle, however, must come to an end because all telecommunications networks need resources to run, most notably (electrical) electricity. The network uses more energy as it grows larger (both in terms of capacity and physical size). In today's industrialized national economies, the information and telecommunications sector accounts for about 5% of all electrical power usage [86].

Information and communications technology (ICT) energy consumption is a constant problem since network operators must pay high energy costs and customers experience a similar problem with device operation times. We anticipate a significant increase in the number of devices in the upcoming years, as was previously noted. Even with the most advanced communication architecture, simply connecting every gadget to a base station will result in unsustainable levels of energy usage. Therefore, it is necessary to create new communication structures and protocols to drastically reduce energy consumption [90].

One of the major issues facing our modern computer facilities is the energy usage of cloud computing solutions. Major energy consumers in cloud computing are data centers (DCs). A 2016 analysis from Lawrence Berkeley National Laboratory (LBNL) stated that by 2020, DCs are expected to consume over 73 billion kWh in the United States, which is 1.8% of the country's total power usage. At the same time, the expansion of new Internet of Things (IoT)-related uses necessitates the use of more distributed cloud-related architectures that rely on resources placed throughout and at the network's edge. These new virtualized architectures, also known as Fog, Edge, and occasionally P2P computing infrastructures strive to meet the low latency and high bandwidth demands demanded by

IoT-based applications. While there is no longer any disagreement on the deployment of such infrastructures [84].

2.5.1 Energy Consumption Module

When the task is executed in the cloud server, the energy required is often negligible compared to the abundant computing resources of the cloud service nodes themselves, so our research mainly considers the energy consumption of three parts, local computing energy, Fog node computing energy and communication transmission energy. Among them, the computing energy consumption of the task u can be expressed as:

$$CompE = \sum_{i=0}^{N}(\mu_{device}.\ (1 - Od_{u,i}).\ CPU_f + \mu_f.O_{u,i}.(1 - Ol_{u,i}).CPU_f) \quad \textbf{(2.4)}$$

$$CommE = \sum_{i=i}^{N+1}\sum_{pre}^{p_i^u}(Od_{u,i} \oplus Od_{u,pre_i})(Ol_{u,i} + Ol_{u,pre_i})^2 .\ \frac{\omega_{prei}^u}{r_{u,s}} .\ Tp \quad \textbf{(2.5)}$$

the total energy consumption required during the computation and offloading performed by task u can be expressed as:

$$TotE = CompE + CommE \quad \textbf{(2.6)}$$

CompE: Computation Energy

CommE: Communication Energy

μ_{device}: CPU Energy

$Od_{u,i}$: Offloading discussion factor

CPU_f: CPU resources to execute the task

$Ol_{u,i}$: Offloading location factor

i: Index of task

pre_i: Index of previous task

N: Total number of task

ω_{prei}^u Size of transmitted data

U: Index of task

$r_{u,s}$**:** Transmission rate between device and fog

T_p : Transmission Power of end device

2.6 Mobility

Mobility is very important feature of end node devices. In FogNetSim wireless, and wired nodes/devices are supported. The mobility can play an important role and effecting on latency time and power consumption of network. There is different type of mobility models. Normally mobility models are classified into two categories, one type is called "entity model" where movement of each node is independent of other nodes; whereas, the second category is called a "group model". In this model, movement of one node is dependent on the movement of other nodes [87]. The most commonly used entity and group models are listed here:

Random Waypoint – entity model

• Random Walk – entity model

• Gauss-Markov – entity model

• Column Mobility – group model

• Reference Point Group – group model

• R Random Waypoint model – Introduces the pause times between variations in speed and destination.

• Mass Mobility – Introduces a mass point with momentum and inertia.

• Deterministic Motions models– The mobility of fixed point nodes as well as moving nodes around linear and rectangular paths.

• Chiang Mobility – Where probabilistic transition matrix is used to alter the state of motion.

• Gauss-Markov – Where a turning parameter is used to change the amount of randomness in the pattern

In this search we finding the effective of mobility parameter on our evaluation metrics (Latency and Power Consumption)

2.7 Metaheuristic Algorithms

In most cases, metaheuristic algorithms are developed through seeing the wonders of nature. The advancement of metaheuristic computations has made it possible to find solutions to difficult problems. When accurate and exploratory methods fail to produce better solutions within a reasonable amount of computational time, metaheuristic streamlining calculations have demonstrated their ability to locate close to ideal solutions [91].

Over the past two decades, meta-heuristic optimization strategies have exploded in popularity. Some of them, including the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO), are surprisingly well-known among experts from a variety of domains, not only computer scientists. Many different academic disciplines have made use of optimization strategies, adding to the vast body of theoretical works on the topic. The question of why meta-heuristics has become so widespread is raised here [92]. The answer to this issue can be distilled into four key reasons: ease of implementation, adaptability, the absence of a need for derivation, and the desire to avoid local optimums.

1- Simplicity: it is easy to understand meta-heuristics. Typically, they've been sparked into action by rather elementary ideas. Nature and its workings, animal behavior, and theories of evolution are common sources of motivation. Because of its ease of use, computer scientists are able to simulate a wide variety of natural concepts, propose brand new meta-heuristics, hybridize existing ones, and make improvements to those already in use. Additionally, the ease

of use facilitates the learning and application of metaheuristics by other scientists [31].

2- The term "flexibility" is used to describe how meta-heuristics may be adapted to new issues with few tweaks to the underlying algorithm design. Since meta-heuristics tend to treat issues as black boxes, they may be easily adapted to suit a wide variety of scenarios. In other words, a meta-heuristic cares exclusively about the system's inputs and outputs. For meta-heuristics to work, all a designer has to know is how to express their issue [93].

3- Most meta-heuristics can be implemented without resorting to lengthy derivations. In contrast to gradient-based optimization methods, meta-heuristics optimize problems in a random way. Since the optimization process starts with a random solution or solutions, there is no need to compute the derivative of search spaces to determine the optimal solution. Because of this, meta-heuristics work well in real-world scenarios where access to accurate derivative information is either prohibitively expensive or highly uncertain [93].

4- When compared to more traditional optimization methods, meta-heuristics are much better at avoiding local optima. As a result of their stochastic nature, meta-heuristics are able to avoid getting stuck on a small subset of solutions and instead thoroughly explore the entire search space. Meta-heuristics are useful tools for optimizing the difficult real-world problems for which they were designed. The search space of such problems is typically unknown and very complex, with a large number of local optima [94].

2.7.1 Single Objective Optimization

Single-Objective Optimization (SOO) issue manages the boost or minimization of the target work dependent on a solitary variable given a requirement or an unconstrained issue. The SOO issues have a solitary variable in the given target work. The capacity may fluctuate as indicated by the distinctive estimations of that variable. The capacity may have:

i) Relative or Local Minimum
ii) Relative or Local Maximum
iii) Absolute or Global Minimum
iv) Absolute or Global Maximum.

The utilizations of SOO are identified with less unpredictable continuous issues. Notwithstanding, at little dimensions to advancement is required [95].

2.7.2 Multi Objective Optimization

In Multi-objective (or multi-criteria or multi-quality) streamlining (MOO), at least two clashing goals are all the while improved as for a given arrangement of requirements. In spite of the fact that, in true issues commonly improvement in one target prompts the debasement of another. The utilizations of MOO can be effectively found in the field of system investigation, flying machine structure, bioinformatics, oil and gas industry, vehicle plan, item and procedure structure, and a lot more fields. Summing up, the advancement issues have the accompanying attributes [95].:

- Availability of various choice options.
- Number of accessible choice choices restricted by extra limitations.
- Different impact by every choice option on the assessment criteria.

However, Multi Objective issues are more challenging to answer than Single Objective problems since there isn't a single optimum solution; rather, there is a group of options that achieves acceptable trade-offs. The Pareto front describes these options. By identifying all Pareto-optimal solutions to the Multi Objective issue, Multi Objective optimization is often thought of as the analytical phase of the MCDM process [90]. The designer or DM chooses the solution from the Pareto set that maximizes his or her utility. There are various benefits to generating the Pareto set. In contrast to SO optimization, which may neglect this trade-off perspective, the Pareto set provides the DM with a comprehensive picture of all feasible alternatives, allowing for more deliberate deliberation and a more well-informed choice. This is helpful from the standpoint of a systems engineer because it allows for a more thorough examination of the system's state and behavior in relation to all of the goals.

2.7.2.1 Pareto Solution

Multi-objective optimization having conflict objective functions gives rise to a set of optimal solutions, instead of one optimal solution because no solution can be considered to be better than any other with respect to all objective. These optimal solutions called Pareto-optimal solutions. With a set of feasible solutions and different objective function to purse, Pareto improvement is a movement from one feasible solution to another that can make, at least one objective function to return a better value. Asset of feasible solution are Pareto efficient or optimal when no further Pareto improvements can be made [96].

2.8 Some of metaheuristic Algorithms

Metaheuristic algorithms includes several types of algorithm that inspire their activity from nature such as butterfly optimization algorithm, bee

optimization algorithm, Ant Colony Optimization, Particle Swarm Optimization, Whale Optimization, Gray Wolf Optimization and so on.

2.8.1 Particle Swarm Optimization

PSO is one the optimization algorithms rely on population randomness. PSO first proposed by Kennedy and Eberhart by implementing the algorithm on the simple social sample, start with a number of random solutions (particles), each particle has a velocity in a range defined by the user, to optimize the value of the cost function which is evaluated at the position of the particle. Each particle developed repeatedly in the search space trying to get a better solution in the following method:

$$\boldsymbol{vk(i+1) = wvk(i) + c1r1[Pbestk - xk(i)] + c2r2[Gbest - xk(i)]} \quad \textbf{(2.7)}$$

$$\boldsymbol{xk(i+1) = xk(i) + vk(i+1)} \quad \textbf{(2.8)}$$

In equations (7 and 8) the (x) represents the position and (v) represents the velocity of particle which represented in (k) in the iteration (i), c1 and c2 are stable acceleration equal to 2, r1 and r2 are independent random numbers in the range (0, 1), and $Pbest$ is the best position of a particle and $Gbest$ is the best position of the swarm. $Pbest$ represents the best position where the cost is lower value in its search history [41]. In this work these equations have been used to change the position of the particle until the best cost have been obtained which leads to the best solution.

2.8.2 Gray Wolf Optimization

In 2014, Seyedali Mirjalili and colleagues proposed the grey wolf optimizer, a novel heuristic swarm intelligent optimization algorithm. As the top predator in the food chain, the wolf has a strong capacity for prey capture. Wolves typically enjoy social interactions, and they have a rigid social structure inside their bodies. The wolves are divided into four types: alpha, beta, delta, and omega in order to mimic the internal leadership

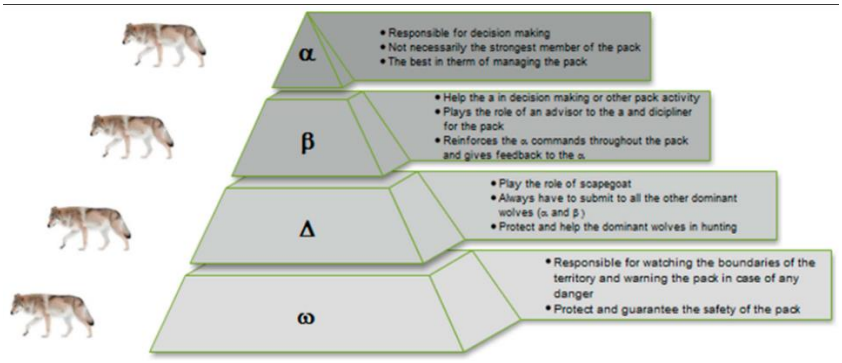


Figure 2.4: GWO Algorithm Social hierarchy[42]

hierarchy of wolves. The best individual, second-best individual, and third-best individual are recorded as alpha, beta, and delta, and the remaining individuals are considered to be omega. In the GWO, alpha, beta, and delta serve as the hunting (optimization compass). They direct other wolves (W) to the most advantageous location for searching.

Grey wolf, is a member of the canine family. The grey wolf is the top predator on the planet. Pack life is highly preferred by grey wolves. On average, there are five to twelve people in each group. Particularly intriguing is the fact that, as can be seen in Figure (2.4), they adhere to a very strict social dominant hierarchy. A male and female alpha are in charge.

2.8.2.1 Social hierarchy

a. The alpha typically makes the most crucial choices, such as where to sleep, when to wake up, what to hunt, and so on. The pack must obey the alpha's orders. On the other hand,

democratic behavior in which an alpha is submissive to the pack's majority has been documented. When the pack is together, the alpha is recognized by everyone lowering their tails. The alpha wolf is also known as the dominant wolf because the pack is expected to follow his or her lead [46]. Only within the pack is the mating of the alpha wolves permitted. The alpha is not necessarily the strongest animal in the pack, but rather the one who is best at keeping the others in line. This demonstrates the importance of a pack's structure and discipline over its raw physical power.

b. In the social structure of grey wolves, beta represents the second tier. Betas are wolves below the alpha in the pack hierarchy who assist the alpha with decision making and other pack duties. It is likely that if one of the alpha wolves were to die or become very old, the position of alpha would fall to the beta wolf, which could be either a male or female. While deferring to the alpha, the beta wolf is also in charge of the pack's subordinates. It advises the alpha and enforces order within the pack. The beta relays information from the pack to the alpha and helps spread the alpha's orders.

c. The grey wolf is the lowest-ranking species. The omega is used as an innocent bystander. The alpha wolves must always take a back seat to the pack's other alphas. Only these wolves are given food now. To outsiders, the omega may not seem like a crucial member of the pack, but studies have shown that losing the omega can cause major problems for the entire pack. This is because the alpha wolf acts out violently and releases all the wolves' pent-up frustrations (s). In addition to keeping the pack happy, this also helps preserve the pack's

established hierarchy. A pack's omega may double as the designated caretaker for young members.

d. A wolf is subordinate if it is not an alpha, beta, or omega (or delta in some references). While the delta wolf must respect the alpha and beta, they are the dominant pack member among omega wolves. This group includes watchmen, guardians, elders, hunters, and caretakers. Scouts keep an eye on the territory's fringes and sound the alarm if they spot any trouble. Sentinels are responsible for ensuring the wellbeing of the pack. The elder wolves are the seasoned former pack leaders. Hunters assist the alphas and betas in hunting for the pack's food. One last duty of the caretakers is to tend to the pack's sick, injured, and frail members.

2.8.2.2 Encircling the Pray

During the hunting process, grey wolves often encircle their prey. The following equations can be used to mathematically model this behavior:

$$X(t+1) = X_p(t) - A \cdot | C \cdot X_p(t) - X(t) | \quad \textbf{(2.12)}$$

$$A = 2a \cdot r_1 - a \quad \textbf{(2.11)}$$

$$C = 2 \cdot r_2 \quad \textbf{(2.9)}$$

$$a = 2 - 2\frac{t}{Max\_iter} \quad \textbf{(2.10)}$$

2.8.2.3 Attacking the Prey

When searching for food, grey wolves rely heavily on the guidance of α, β and δ wolves, who are able to pinpoint the exact location of potential prey. Every time the population is re-evaluated, the top three wolves (α, β, δ) are kept and the rest of the search agents' positions are adjusted accordingly. The GWO behavior was shown in Figure (2.5) GWO Flowchart. Specifically, the following equations are proposed: **(2.13)**

$$X_1 = X_\alpha - A_1 \cdot | C_1 \cdot X_\alpha - X | \quad \textbf{(2.16)}$$

$$X_2 = X_\beta - A_2 \cdot | C_2 \cdot X_\beta - X | \quad \textbf{(2.14)}$$

$$X_3 = X_\delta - A_3 \cdot | C_3 \cdot X_\delta - X | \quad \textbf{(2.15)}$$

$$X(t+1) = \frac{X_1(t) + X_2(t) + X_3(t)}{3}$$

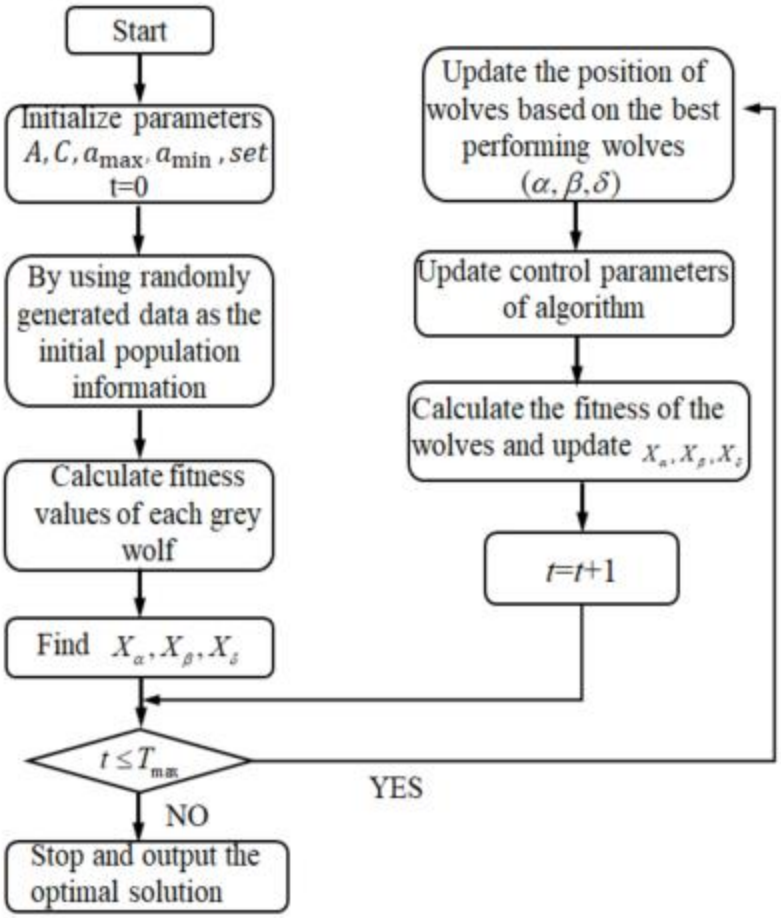


Figure 2.5: GWO Flowchart

Pseudocode of the Grey Wolf Optimizer (GWO) Algorithm

Step 1: Initialize the a, A, C, maximum number of iterations (lmax), and search agents (n)
Step 2: Set iteration counter l = 0
for i = 1:n do
Calculate the fitness(solutions)
end for
Step 3: Find the optimal solution of each search agent (Xα, Xβ, and Xδ)
for i = 1:n do
Update A and C by (10) and (11) and each search agent by (6)to (8);
Decrease a from 2 to 0; Calculate the fitness(solutions)
end for
Update the solutions (Xα, Xβ, and Xδ) and Set l = l + 1.
Step 4: Produce the optimal solution and return Xα.

Figure 2.6:Pseudocode of (GWO)

2.9 Tasks Offloading framework

The basic idea of computation offloading is to delegate some or all of a computationally intensive task to a more capable entity (like a cloud or edge server), have it performed, and then receive the results. The main goals of computation offloading are to increase application performance by decreasing their execution time and decreasing power consumption in end devices [95].

The two main challenges to computation offloading are as follows [96] [97]:

- The decision-making process for offloading, which includes determining which tasks should be offloaded and where, is the first challenge.
- The allocation of computing and communication resources is the second.

When discussing Fog computing, we assume a simple case in which data centers are deployed at the network's periphery, and that if a request is

received by an overburdened data center, it is probabilistically redirected to a nearby data center. It is assumed that data centers employ a huge amount of servers, and that heavy usage of some of those servers will eventually lead to overload. To deal with the overload, the other data centers may begin to take some of the requests that were previously denied.

Tasks requiring real-time and low-latency services could be served at the Fog layer in an IoT-Fog-Cloud architecture, while those requiring extensive computations or persistent storage could be offloaded to the remote cloud server [98]. The backhaul network's workload can be significantly reduced and latency-sensitive delivery can be guaranteed by shifting computation to Fog nodes. Fog computing's ubiquitous and adaptable services will give end users the Quality of Service (QoS) they need [99].

Computation offloading has been the subject of numerous studies aimed at increasing energy efficiency or delaying performance at end devices. However, the majority of these studies either concentrated on offloading decision-making or on resource allocation, such as transmit power allocation and communication and computation resource allocation, without taking both of them into account simultaneously [100]. When offloading tasks, an increase in transmit power will result in a higher transmission rate and a shorter transmission delay, but the energy consumption of the devices will increase due to the power limitations of the devices and the latency requirements of the tasks. In contrast, lowering the transmit power results in energy savings for the device, though the increased transmission delay will result in a breach of the task's required delay [96]. GWO chooses the best way to offload tasks, which saves energy and cuts the time it takes to finish all tasks.

2.10 Working Environment

There are a number of simulation tools for Fog computing environments, and each one works in a different way. Before we look at how they do what they're supposed to, we'll talk about the current problems in Fog computing that lead to new, specialized methods [101] . With the right simulation tools, these new ideas can be tested, evaluated, and their costs can be estimated. It's not easy to make Fog computing simulators that meet the needs for generality, scalability, efficiency, support for mobility, and low-latency. Finding out what the simulator is made of is one of the hardest parts of this process. Every choice is based on how much it will actually cost to set up and run the Fog computing scheme and all of its parts, including infrastructure, computing, and applications [102].

a. SimpleIoTSimulator [103] :A paid simulation tool, with an emphasis on Internet of Things use cases. The simulator sets up testing environments with multiple sensors and gateways. The goal is to facilitate the testing of gateways from different manufacturers in a variety of Internet of Things environments. Plus, the simulator works with a wide variety of popular IoT protocols like MQTT, CoAP, MQTT-SN, and MQTTBroker.

b. FogTorch [104]: FogTorch, a Fog-based simulator, tests IoT application deployment, execution, and testing. The framework can check an application's resource and network constraints before deployment. The proposed framework lacks mobility and multi-Fog support.

c. EdgeCloudSim [105]: Edge CloudSim. The simulator's CPU utilization models limit VM tasks. The simulator evaluates edge computing scenarios by modelling network linking, and

edge service. It lacks mobility, handoffs, and Fog federation support.

d. FogBus [106]: A framework for integrating IoT systems with Fog and cloud. The framework lets many apps run simultaneously and provides platform-independent IoT interfaces. The solution allows service providers to customize, develop, and test cross-platform IoT applications. The suggested system uses data authentication, privacy, and block chain for data integrity in sensitive applications. The proposed mobility framework is limited. The system doesn't support federated Fogs.

e. iFogSim [107]: A simulation of a Fog infrastructure that can coordinate the allocation of resources to running applications. The suggested simulator, an expansion of CloudSim, can measure the resources' impact on energy use, bottlenecks in the network, and response times. To evaluate their simulation framework, the authors provide two case studies. The proposed simulator includes Fog devices that receive data from Internet of Things (IoT) sensors and process that information to produce insights and send orders to the simulator's actuators. Yet the simulator does not account for Fog federation use cases. Not only that, but the simulator uses antiquated APIs that are incompatible with modern versions of Java.

f. FogNetSim++ [108]: is an expansion of CloudNetSim++ that makes advantage of OMNeT++'s capabilities. The OMNet++ is a popular open-source program that mimics the behavior of real network devices through the use of a library with a wide

variety of pre-built modules. FogNetSim++ makes it simple to incorporate any of OMNeT++'s many available modules. To run their own simulations, researchers need only tweak a few settings in FogNetSim++'s modular system. FogNetSim++ was created since most existing frameworks are geared at handling a relatively small number of sensors. But they haven't taken into account the network features that are essential to latency-sensitive application modelling. Likewise, current simulators lack mobility assistance or only offer extremely rudimentary mobility models [109]. It's an innovative system that works with both moving and stationary components. Figure (6) depicts FogNetSim++'s modular architecture. The core components of FogNetSim++ are the end devices and the broker.

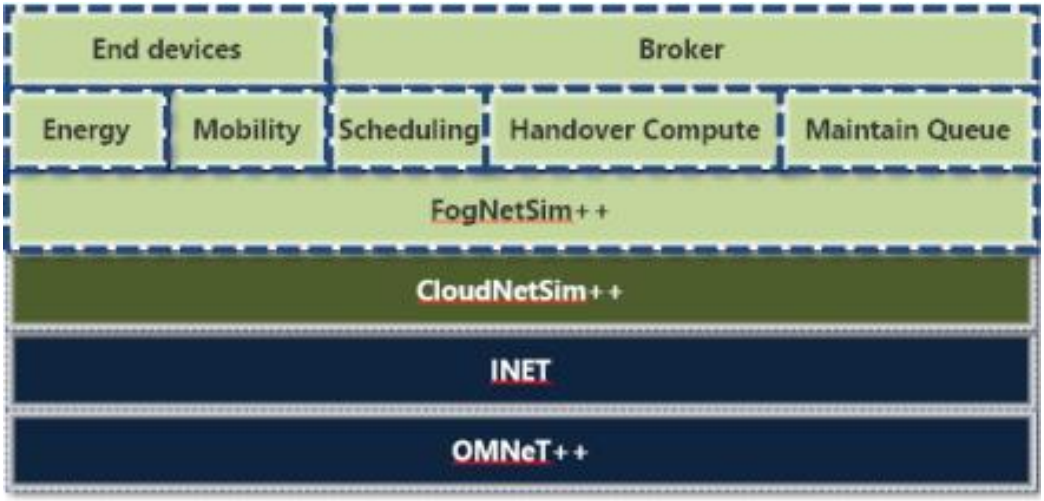


Figure 2.7: FogNetSim++'s Modular Architecture [110]

Table 2-1: Comparison of different Fog simulator

Simulator	Platform independent	Open source	Customized mobility model	Fog federation	Device handover	Developed by (R/A/I)	Distributed Fog location
SimpleIoTSimulator	x	x	√	x	x	I	-
MobIoTSim	x	√	x	x	x	A/R	-
Google cloud IoT	Cloud based	x	x	x	x	I	-
iFogSim	√	√	x	x	x	A/R	single
Cooja	x	√	x	x	x	A/R	single
FogTorch	√	√	x	x	x	A/R	-
EmuFog	√	√	x	x	x	A/R	-
EdgeCloudSim	√	√	√	x	x	A/R	single
Mobile Fog	-	x	x	x	x	A/R	-
FogBus	x	√	x	x	x	A/R	single
Virtual Fog	x	√	x	x	x	A/R	limited
FogRoute	x	x	x	x	x	A/R	-
FogBed	√	√	x	x	x	A/R	-
D2D Fogging	-	-	x	x	x	A/R	Single
FogNetSim++	√	√	√	x	√	A/R	Single

2.10.1 FogNetSim++ Characteristics

Using multi-objective optimization to balance price, availability, and performance throughout the Fog federation, the simulator bolsters latency-sensitive applications at the Fog layer. The following are some of the most important contributions of the proposed framework [108] :

1- It uses an allocation mechanism to facilitate resource sharing in the Fog so that SLAs can be met for end users (SLA). The performance,

pricing, and energy/workload availability of Fog resources are tracked in a table that is updated by each broker. This data is distributed to other brokers in a federated system so that resources can be shared openly and efficiently.

2- Facilitates both fixed and mobile network nodes. The framework dynamically handles the handover method, keeping tabs on mobile devices and their accompanying hop counts to ensure uninterrupted service delivery. As a result, subscriber devices placed everywhere on the network can receive results quickly and efficiently.

3- Evaluated in terms of energy, latency, scalability, and resource use.

4- Allows users to simulate complex Fog scenarios with various network properties through a graphical user interface. Figure (7) shows GUI of FogNetSim++

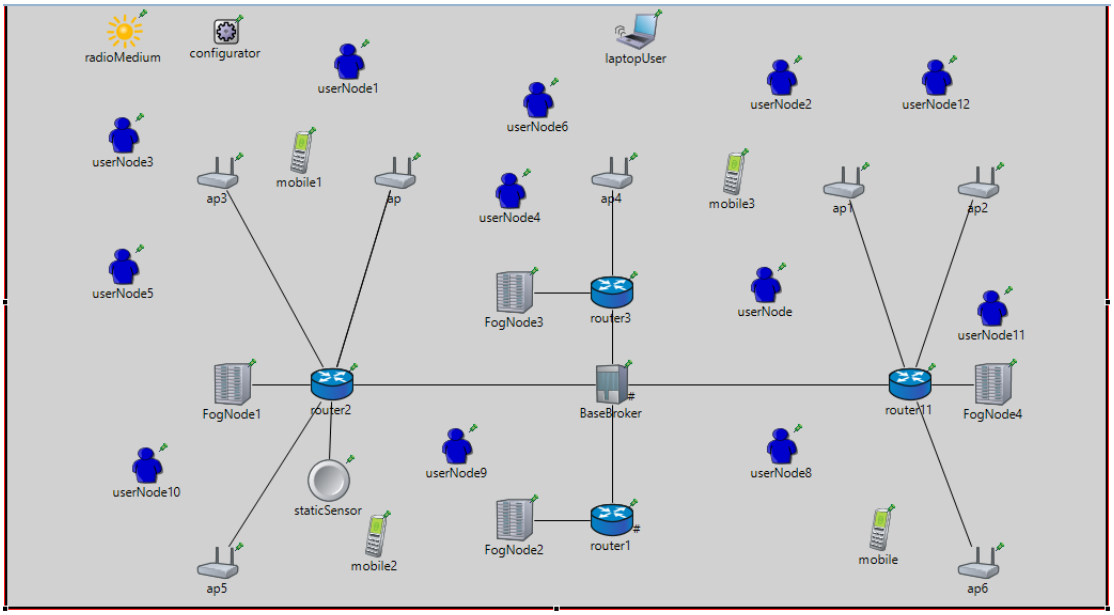


Figure 2.8: GUI in FogNetSim++

2.10.2 FogNetSim++ Components:

1- User devices: the real end node/sensor that may move around while communicating and making requests of computer nodes. It's an innovative system that works with both moveable and immobile nodes, gadgets, and gateways.

2- The Fog node is a type of network node that offers computational services. As a rule, it is employed to collect information from sensors, with only the most pertinent details being transmitted to the data center.

3- The broker node or (Base Broker) is in charge of the Fog nodes; it is the one that takes in service requests and distributes them to the available nodes while keeping each one's capacity in mind. Because of this, the broker is in charge of task scheduling, execution, and handover [110].

2.11 Summery

This chapter summarized and explained theoretical background for a robust Fog-computing for IoT applications with explanations of the Internet of Things (IoT) concepts, architecture, applications, the advantages and challenges of IoT. In addition, the Fog computing explained with the main issues and how it implemented in IoT applications, Fog computing architecture, advantage and applications. alongside, Furthermore, latency and Power Consumption is explained in general, and the impact of these metrics in Fog computing. The used metaheuristic algorithms to find optimal solution of task offloading in Fog computing. As well as, the used offloading mechanism explained. Finally, the working environment and simulation tools was discussed in the last section.

Chapter Three

The Proposed System and methodology

3.1 Introduction

In this chapter, design of efficient Fog offloading optimized approach to improve IoT devices performance are presented. The practical partition includes two phase: Fog computing and Optimization. Generating Fog scenarios using OMNET++ simulator, calculating performance metrics (Latency and Power consumption). The second phase include optimization used to select best solution using proposed multi-objective Gray Wolf Optimization algorithm. This algorithm finds best solution depending on performance metrics (lowest latency and power consumption at the same time).

3.2 The Proposed System

The proposed system consists of: IoT layer that contains static and mobile (devices, sensors and actuators). Fog layer consist from one or more administrative node that named Base Broker and Fog nodes that called Compute Brokers any device can be a Fog Node if it supports processing, data storage and networking (examples: Switches, Gateways, Access Point and Surveillance Cameras). Cloud layer which represents unlimited service. The main objective of this work is load managements among Fog nodes to be processing and response the user in less time and resource consumption. When static or mobile user sending requests, the Fog node received this request and sending it to base broker to decision making where this request will be processed. Using metaheuristic Multi Objective GWO to optimized our problem in optimal way.

Figure (3.2) explain dataflow in proposed system, when user sending request, the Fog nodes (compute Broker) receive this request and send it to Base Broker with its capability information (MIPS, Memory size and queue), base broker in turn decomposition request to task and calculate power and latency for each compute broker optimized result using MO-

GWO then offload tasks to compute broker to process it in minimum cost of latency and power consumption, after compute broker complete processing send the task back to base broker to allocate it and send response either directly or through Fog node according to routing table. Figure (3-1) show the proposed system.

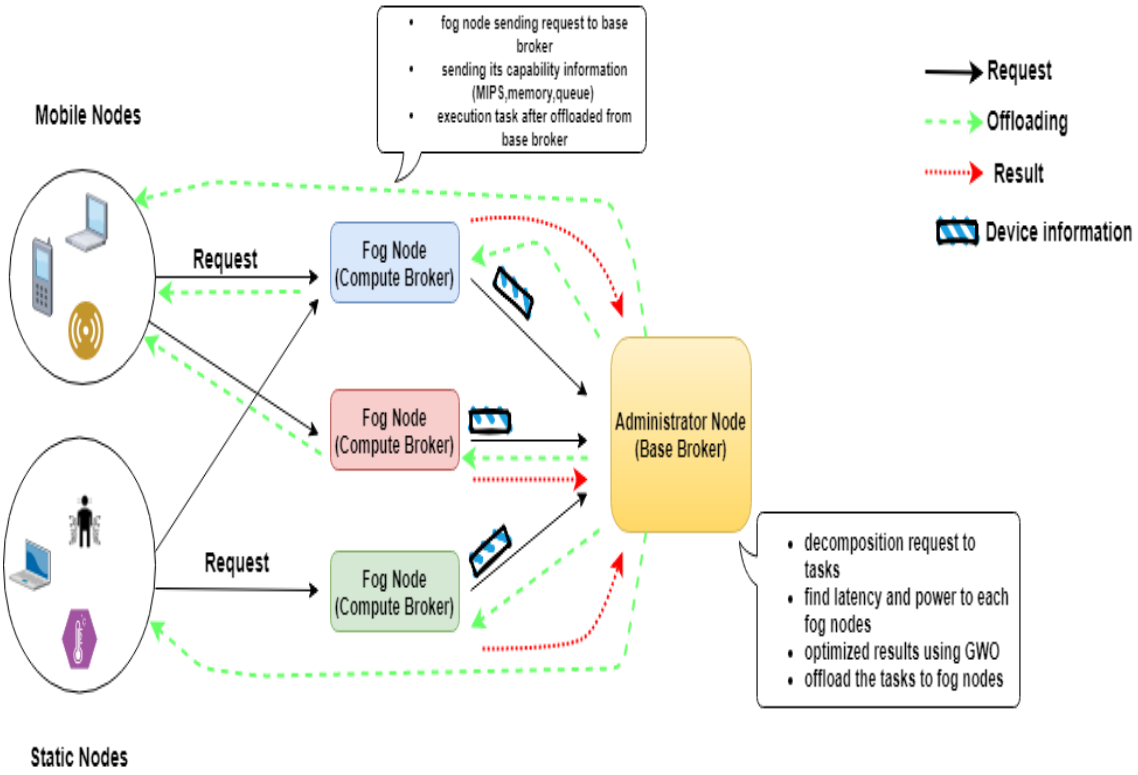


Figure 3.1: Proposed System Module

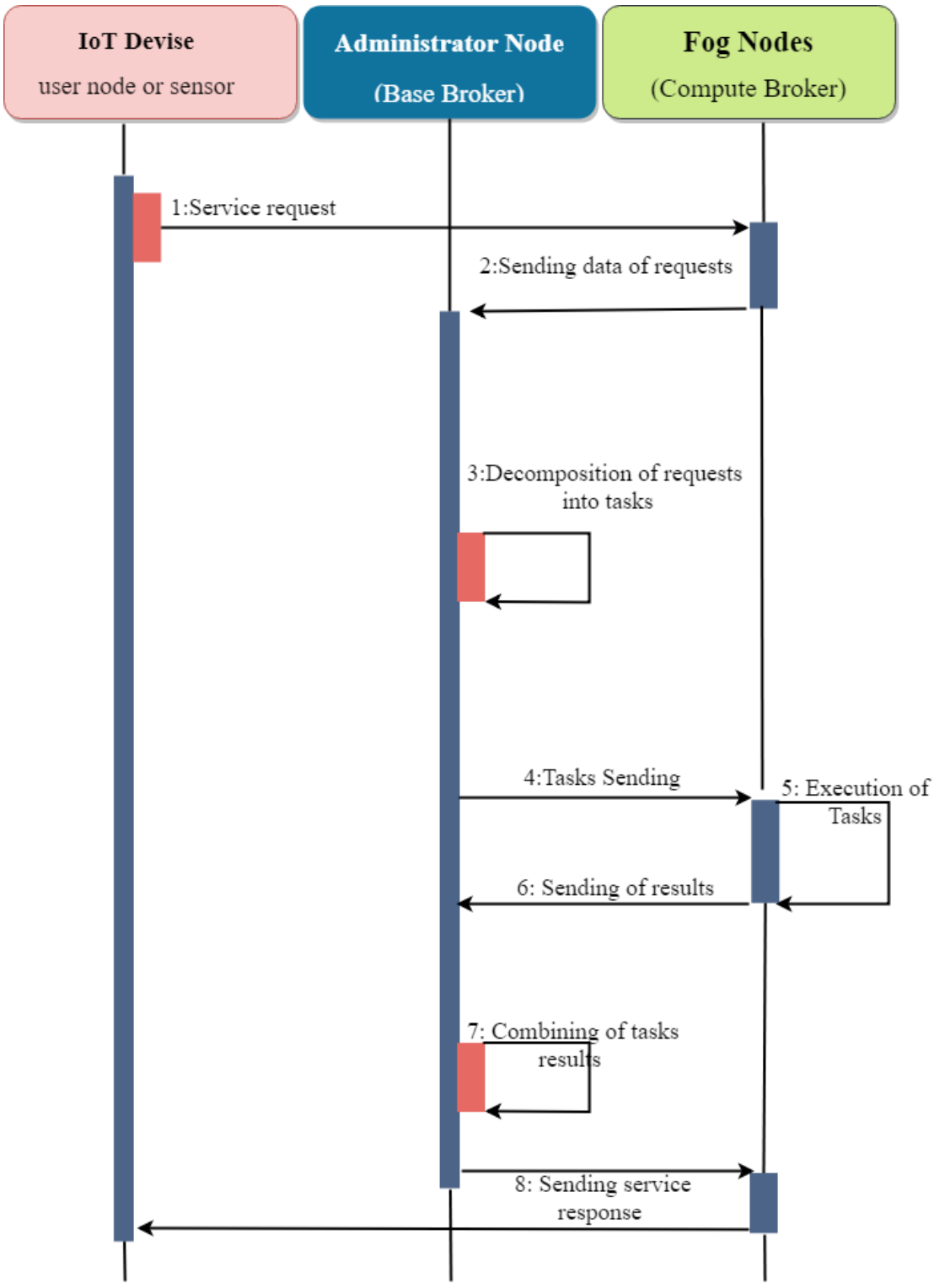


Figure 3.2:Dataflow in Fog Framework

3.3 The Proposed System Requirements

3.3.1 Fog Scenarios

To create Fog computing scenarios, do the following:

1- Start OMNET++ and create project from:
File ⟶ New ⟶ OMNET++ Project

2- Right click on created project > Properties > Project References > choose "INET" from list , and press OK

3- Then Setting up the project:
- Adding .NED (Network Description File) file
- Adding C++ file
- Adding .ini file

4- In these files the details of scenario and optimization will be added:
- Create Network
- Create Nodes
- Create Message
- Create Program

wireless5.ini | wireless5.cc | wireless5.h | final.cpp | wireless5.ned

```
network WirelessNetwork5
{
    parameters:
        int numb;
        int numbUsers;
        @display("bgb=1083,679");
    types:
        channel C extends DatarateChannel
        {
            datarate = 100Mbps;
            delay = 0.1us;

        }


    submodules:
        ComputeBroker1: StandardCompute {
            parameters:
            @display("p=412,616;i=device/router");
        }
        ComputeBroker2: StandardCompute {
            @display("p=582,616;i=device/router");
        }
        BaseBroker: StandardCompute {
            @display("p=582,424;i=device/server;q=Tasks");
        }
        user[numbUsers]: WirelessCompute;
        router1: Router {
            @display("p=582,552;q=Tasks");
        }
        configurator: IPv4NetworkConfigurator {
            //config = xml("<config><interface user='*' address='145.236.16.2' netmask='255.255.255.0'/></config>");
```

Figure 3.3:Network.ned file configuration-1

wireless5.ini | wireless5.cc | wireless5.h | final.cpp | wireless5.ned

```
# mqtt apps
**.user[*].numUdpApps = 1
**.user[*].udpApp[*].typename =  "mqttApp2"
**.user[*].udpApp[*].messageLength = 2028B
**.user[*].udpApp[*].localAddress = ""
**.user[*].udpApp[*].localPort = 1000
**.user[*].udpApp[*].destAddresses = "BaseBroker"
**.user[*].udpApp[*].destPort = 1001
**.user[*].udpApp[*].sendInterval = 1.5s
**.user[*].udpApp[*].startTime = 0.0s
**.user[*].udpApp[*].stopTime = 300s
**.user[*].udpApp[*].publishToTopics = "test topic 1"
**.user[*].udpApp[*].publish = true
**.user[*].udpApp[*].taskSize = 1500

# basebroker apps
**.BaseBroker.numUdpApps = 1
**.BaseBroker.udpApp[*].typename =  "BrokerBaseApp3"
**.BaseBroker.udpApp[*].localPort = 1001
**.BaseBroker.udpApp[*].MIPS = 1000

# computeBroker apps
**.ComputeBroker*.numUdpApps = 1
**.ComputeBroker*.udpApp[*].typename =  "ComputeBrokerApp3"
**.ComputeBroker*.udpApp[*].localPort = 1001
**.ComputeBroker1.udpApp[*].MIPS = 1000
**.ComputeBroker2.udpApp[*].MIPS = 2000
**.ComputeBroker3.udpApp[*].MIPS = 3000
**.ComputeBroker4.udpApp[*].MIPS = 4000
**.ComputeBroker*.udpApp[*].destAddresses = "BaseBroker"
**.ComputeBroker*.udpApp[*].destPort = 1001
**.ComputeBroker*.udpApp[*].messageLength = 1028 Byte
**.ComputeBroker*.udpApp[*].sendInterval = 0.1s
**.ComputeBroker*.udpApp[*].startTime = 0.0s

*.configurator.config = xml("<config> \
```

Form | Source

Figure 3.4:Network.ned file configuration-2

3.4 Performance Metrics Evaluation

Some of the factors that affected the mechanics of offloading and thus affect the performance of the network in general, which we aim to improve during this thesis, as shown below:

3.4.1 Latency Time

The total latency time (TL) taken to offload task is being calculated from the summation of communication time and computation time to offload the data. Communication time means the time that spend while transmitting through the network propagation time and queuing time. Communication latency is time that taken while waiting and processing task according device capacity and task size.

TotL is calculated from (2.1) and (2.3)

$$TotL = ComL_{fog} + CompL_{fog} \quad \textbf{(3.1)}$$

3.4.2 Power Consumption

The total energy consumption required during the communication and computation and offloading performed by task u can be expressed as:

$$TotE = CompE + CommE \quad \textbf{(3.2)}$$

The energy consumption generated by data transmission during the execution of task u includes the communication energy consumption between the local device and the edge server, and Computation energy means the energy that consumed through the processing of tasks.

3.5 Optimization Implementation

Optimization issues in which more than one objective function must be optimized at the same time. When optimal judgments must be made in the context of trade-offs between two or more conflicting objectives, multi-objective optimization problems emerge in various domains, including network, engineering, and economics. In this work, we must choose the best two variables at the same time in order to discover the best solution. The fitness function is used in GWO algorithms to find multi-objective optimization in this work. The method seeks the optimal value for assessment measures such as the and the small amount of delay and power consumption. The algorithms will then decide which values are the best at the same time, and we will finally choose the optimal solution.

3.5.1 Proposed Multi-Objective GWO Algorithm

MOGWO is an algorithm that integrates two mechanisms to achieve results similar to those of multi-objective Particle Swarm Optimization (MOPSO). The first part of MOGWO is an archive, which is in charge of storing the current best solution that has not been dominated, and the second part is a leader selection strategy, which aids in selecting the solutions of, and as the top dogs in the hunting hierarchy. A non-dominated Pareto optimal solution can be stored in, or retrieved from, an archive. If the solution enters the archive or if the archive is completely used up, the archive controller takes over and manages everything inside. The need for a large number of contributors to an archive has been noted. As the iteration process continues, the non-dominated Pareto solution pairs are compared to the database participants. Therefore, the following are the three scenarios:

One or more previously archived solutions can now be managed by the newly obtained solution. As a result, the established archival solution must be discarded and the emerging solution must be integrated into the archive.

• If the storage space is at capacity, the grid method is implemented to rearrange the target space's subsections and discover the most densely populated subsection from which a solution can be removed. The optimal Pareto front can be made more diverse by incorporating the more recent solution into the less populated section.

• If the new solution and the archive members are not mutually dominant, the newer solution must be added to the archive.

Algorithm 1: MO-GWO Algorithm	
Input	**Arrays of performance metrics from scenarios**
Output	**Characteristic Matrix (M)**

Begin

- Initialize population

Set Pop_NO←20 //number of Population

Set iter-num←3 //number of iteration

Set a-gwo←2 // a decreases linearly from 2 to 0

Declare search agent

upper and lower boundaries // lw&up

Declare Array of Position Randomly by call(Set random position)

Declare P_Positon //network parameters

Declare cost_P_position Randomly // network performance metrics

Declare fitt with size 1 by 3 // saved values of fitness function

Declare P_Best with size 1 by 3 // network power

Declare L_Cost // network latency

Declare array of Temp_fitness // values of fitness function

Set Max_iter ← 1000 //maximum iteration number

Set r1 //random number [0-1]

// Optimization

1- Loop itr ∈ [0,…. , Max_iter] // iteration loop

// Calculate objective function for each search agent

2- Update Alpha, Beta, and Delta

3- Alpha_pos← Positions(i,:)

// If the fitness>Alpha_score && fitness<Beta_score

4-Update beta score&& position

// if fitness>Alpha_score && fitness>Beta_score && fitness<Delta_score

5-Update delta score&& position

6- a decreases linearly from 2 to 0

7- Loop ij ∈ [1,…. , position]

8- update A1//equation (2.10)

9- update C1 // equation (2.9)

10- D_alpha=abs(C1*Alpha_pos(j)-Positions(i,j))

11-X1=Alpha_pos(j)-A1*D_alpha;

12- Update r1=rand(); r2=rand();

13- Find C2, D_beta ,X2

14- Update r1,r2

15- A3,C3, D_delta,X3

16-Update Positions(i,j)=(X1+X2+X3)/3

// If the fitness value is better than the global value (gBest) in history then set current value as the new gBest

17- cost_P_position [i][0] ← Temp_fitness[0][0]

18- cost_P_position [i][1] ← Temp_fitness[0][1]

19- cost_P_position [i][2] ← Temp_fitness[0][2]

20- P_position [i][0] ← position [i][0]

21- P_position [i][1] ← position [i][1]

22- P_position [i][2] ← position [i][2

23- End if

13- if (Temp_fitness[0][0] > L_Cost[0][1] &&Temp_fitness[0][1] < L_Cost[0][1] && Temp_fitness[0][2] < L_Cost[0][2])

Begin

TotL= $CommL_{fog} + CompL_{fog}$

Power[ii][jj] = MIPS [ii][jj] + $c1 * Math.random() *$ $postion[ii][jj]$)**..(3.1)**//update Power

// Print optimal network parameters and network performance metrics

14- L_Cost[0][0] ← Temp_fitness[0][0]

15- L_Cost[0][1] ← Temp_fitness[0][1]

16- L_Cost[0][2] ← Temp_fitness[0][2]

17- P_Best[0][0] ← position [i][0]

18- P_Best[0][1] ← position [i][1]
19- P_Best[0][2] ← position [i][2]
20- iteration← Temp_fitness[0][3]
21- End if
22- Loop ii ∈ [0,…. , Population_NO]
23- Loop jj ∈ [0,…. , Itreation_size]
24- // Update Latency in equation (3.1)
25- // Update Power in equation (3.2)
$P\_postion[ii][jj] = postion[ii][jj]$
26- End loop
27- End loop
28- End loop
29- Print "Latency ←" + L_Cost[0][0] +
30- Print "Power ←"+P_Best[0][1]+
32- Print "no. nodes ←" + P-Position[0][0] +
33- Print "no. tasks ←"+G_ Best [0][1]+
34- Print "rate of speed MIPS←" + G_ Best [0][2]

End

Pseudocode of the MO-GWO Algorithm

1: Initialize population of GWs

2: Initialize a, A and C

3: Calculate the objective value for each gray wolf

4: Find the non-dominated solution

5: Select the best leader from an archive (Xa)

6: Temporarily exclude from an archive to avoid selection of same leader

7: Select the second best leader from an archive (Xb)

8: Temporarily exclude (b) from an archive to avoid selection of same leader

9: Select the third best leader from an archive (Xd)

10: Add (a) and (b) back to the archive

11: t = 1;

12: while (t < MaxIt) do

13: for each gray wolf do

14: Update the position of gray wolf through Equations (7)–(8)

15: end for

16: Update a, A and C

17: Calculate the objective values for all GWs

18: Find the non-dominated solutions

19: Update the archive according to the obtained non-dominated values

20: if the archive is full then

21: Grid mechanism will omit one of existing member from an archive

22: Add the new wolf to the archive

23: end if

24: if the new solutions added to the archive is outside the segment then

25: Update the whole grid to cover the new solutions

26: end if

27: Select the best leader from an archive (Xa)

28: Temporarily exclude (a) from an archive to avoid selection of same leader

29: Select the second best leader from an archive (Xb)

30: Temporarily exclude (b) from an archive to avoid selection of same leader

31: Select the third best leader from an archive (Xd)

32: Add (a) and (b) back to the archive

33: t = t + 1;

34: end while

35: return archive

Pareto Solution

a multi-objective offloading algorithm based on the improved strength Pareto evolutionary algorithm is proposed considering the parallel processing mechanism and global optimization. our algorithm is able to obtain a set of Pareto-optimal solutions for user or supplier selection in a

short time [90]. As shown in Algorithm 2, we initialize the population $P0$ with size M and create an empty external archive set ($A0$) and an empty local search set ($L0$), both with size $(M0)$. Then we calculate the fitness of all individuals in the population and external archive set and local search set, and save all the non-dominated solution sets in them to the next generation of external archive set ($At+1$) and local search set ($Lt+1$). If the number of individuals in the external archive set exceeds $M0$ at this time, truncate the operation; if the number of individuals does not reach, we select some of the dominant liberation from (Pt) and (At) into $(At+1)$ and $(Lt+1)$. After that, the local search update is performed for $Lt+1$, and the individuals from the external archive set ($At+1$) at this time are selected into the mating pool for crossover and mutation and the results are kept into the next generation population $(Pt+1)$ to recalculate the fitness, thus iterating. Finally, the non-dominated solution set in ($At+1$) is taken as the final output of the offloading policy

Algorithm 2: Pareto Algorithm	
Input	***M (population size), M0 (archive size), T (maximum number of iterations)***
Output	***Output:* Q (offloading strategy)**
Begin	
1 Initialize the first generation $P0$, empty external set $A0$ and local search set $L0$; 2 $t = 0$; 3 Calculate fitness values of individuals in Pt, At and Lt; 4 Copy all non-dominated individuals from Pt, At and Lt to $At+1$; 5 for $t < T$ do	

6 if $Pt + At > M0$ then
7 Clip $At+1$ by truncation operation ;
8 else
9 Add dominated individuals from Pt, At and Lt to $At+1$;
10 $Lt+1 = At+1$;
11 Conduct the binary tournament selection on At and $At+1$ for filling the mating
pool ;
12 Crossover and mutation operation ;
13 Perform a partial search update on Lt ;
14 Set $At+1$ as the next generation ;
15 $t = t + 1$;
16 Set Q to the set of offloading strategy vectors represented by the non-dominated
individuals in $At+1$;
17 final ;
18 return Q;

3.5.2 Objective Function

If we use the weighted sum aggregation method to assign weights to the objectives in a real-world scenario where the objectives of minimizing the average delay and minimizing the average energy consumption of task computation offload conflict, we can only obtain a single optimal solution, which is often difficult to meet the diverse needs of different users or service providers in a resource-variant environment. As a result, to better meet the needs of users and providers, as well as to maximize user experience and resource utilization, we formalize the problem as a multi-objective optimization problem, with the optimization goal of minimizing

the average delay and average energy consumption of task computation offloading.

$$\min \frac{1}{U} \sum_{u=1}^{U} TotE$$

$$\min \frac{1}{U} \sum_{u=1}^{U} TotL$$

$$S(i) = |\{j \mid j \in P_t + A_t \wedge i \succ j\}| \quad (19)$$

$$R(i) = \sum_{j \in P_t + A_t, j \succ i} S(i) \quad (20)$$

To avoid the situation that individuals have the same fitness, the MO-GWO algorithm calculates the number of dominated solutions for each individual in the population and external archive set and defines it as the strength value *S(i)*, and based on this, the original fitness *R(i)* is defined to represent the number of dominated solutions for individual *i*. The larger the *R(i)*, the more individuals dominate individual *i*

3.6 Summery

This chapter has shown the method used to implement the IoT - Fog system with the explanation of the proposed system implementation steps and installation requirements for IoT-Fog system, besides the way of execution, implementation of offloading mechanism among Fog nodes depending on latency and power in Fog devices. The main algorithm for optimization is presented to finding optimal offloading solution to improve network performance and enhanced IoT devices and application by improving QoS.

Chapter Four

Implementation and Results

4.1 Introduction

This chapter explains the main details of the implementation of Fog computing offloading optimization technique to improve latency and power consumption in Fog network. Optimization using meta-heuristic MO-GWO algorithm using FogNetSim++ simulation environment with C++ programming language.

4.2 Implementation

There is more than one scenario with different parameters that was executed with using FogNetSim++ to achieve our objectives.

4.2.1 The computer specification

This scenario was implemented in computer device with the following characteristics shown in Table (4-1):

Table(4-1): Computer Specification

Parameter	Value
Type	Lenovo ideapad S145
Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
RAM	8.00 GB
Hard Drive	265 SSD + 1T HHD
Operating System	64-bit operating system , Windows 10

Scenario 1:

The network in scenario one (Figure 4-1) consist of three layers: IoT layer contains 3 static device and (20-25) mobile device (users, sensors, actuators and so on). Fog layer contains parameters that shown in Table (4-2): one base broker, 4 compute broker and network devices like Access Points, Routers, Switch if there's need to it.

Table 4-2): Network Parametes

Parameter	Value
Router	3
Access Points	5
Base Broker	1
Fog nodes or compute broker	4
User node and sensor	20
User Mobility	Static , linear ,circular

The properties of these devices is:

```
**.user[0..5].mobilityType = "CircleMobility"
**.user[1].mobility.startAngle = 60deg
**.mobility.initFromDisplayString = true
**.user[*].mobilityType = "LinearMobility"
**.user[*].mobility.speed = 20mps
**.user[*].mobility.updateInterval = 100ms
**.radio.transmitter.power = 3.5mW
**.ComputeBroker*.udpApp[*].sendInterval = 0.1s
**.displayCommunication = false
**.displayCommunicationRange = true
**.displayInterferenceRange = false
**.drawCommunication2D = true
*.BaseBroker.energyStorage.cmdenv-ev-output = true
*.user[*].energyStorage.typename = "SimpleEpEnergyStorage"
```

```
*.user[*].wlan[*].radio.energyConsumer.typename="StateBasedEpE
nergyConsumer"
```

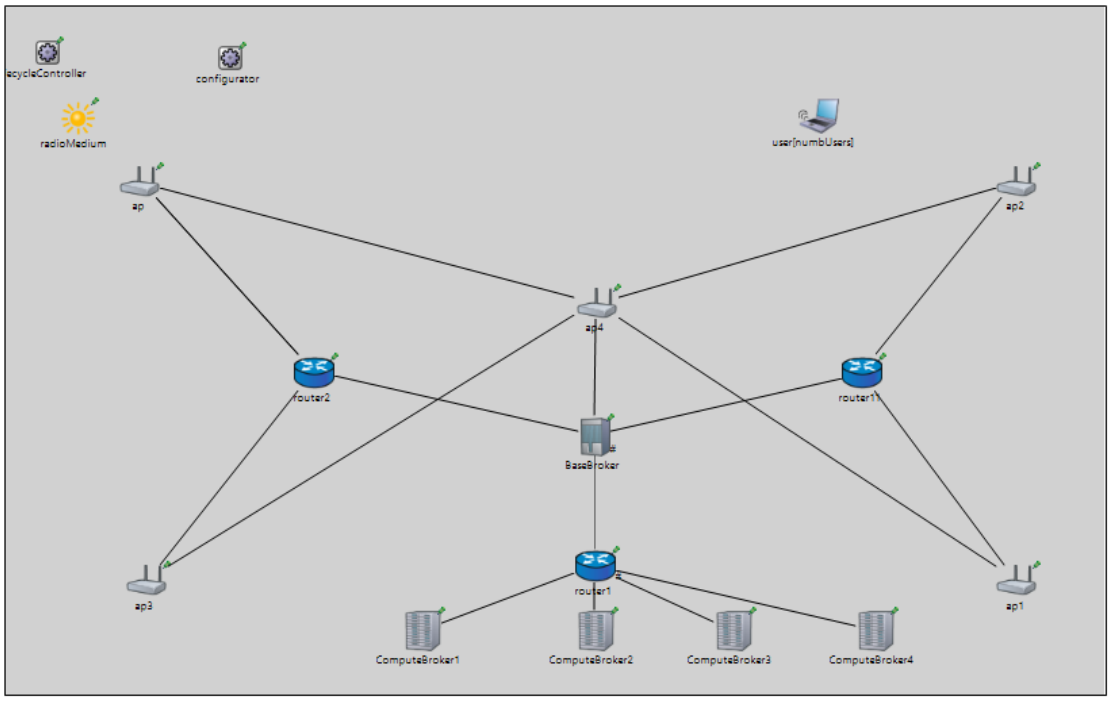


Figure 4.1: Scinario 1

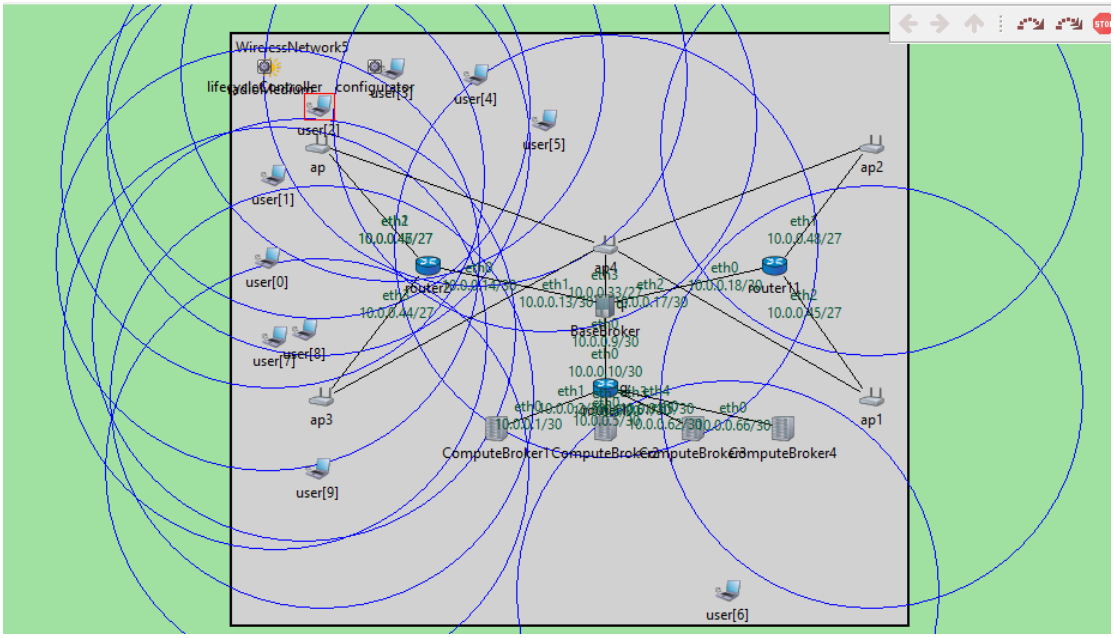


Figure 4.2: Proposed Scenario1 through implementation

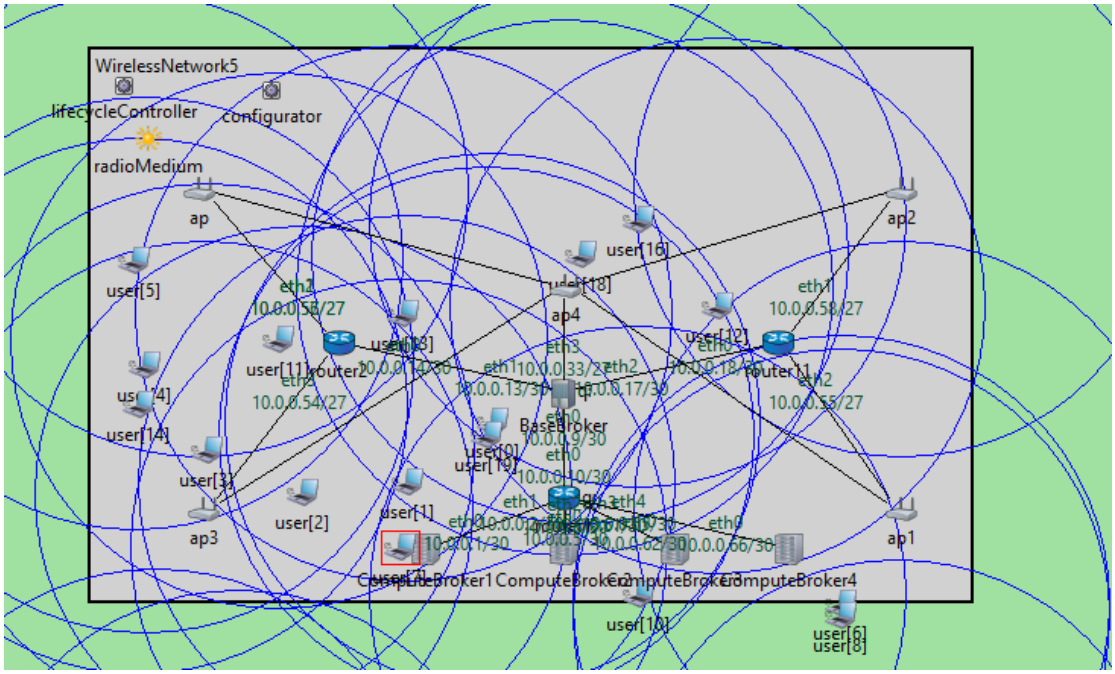


Figure 4.3:Proposed Scenario1 through implementation while mobility parameters change

When any device send request to network, the Fog nodes received it and then send acknowledgment to base broker with the request information's and it information like (MIPs. Memory, CPU capacity, message length, device mobility type, waiting queue) according this information the base broker was make decision about where is this request will be processing after decomposition this request to tasks and sending these tasks to compute broker to be processed, after processing each compute broker send result to base broker to combining the task results and send it to user.

Scenario 2:

The network in scenario two (Figure 4-2) consist of: IoT layer contains 5 static device and (10-25) mobile device (users, sensors, actuators and so on). Fog layer contains: 5 base broker, 12 compute broker and

network devices like Access Points, Routers, Switch if there's need to it.

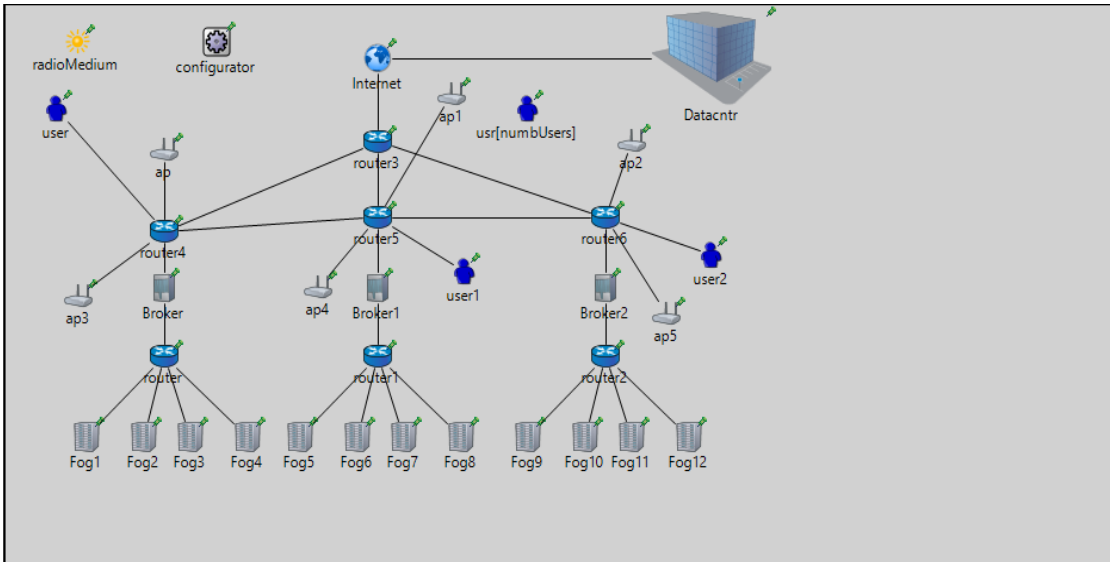


Figure 4.4: Scenario 2

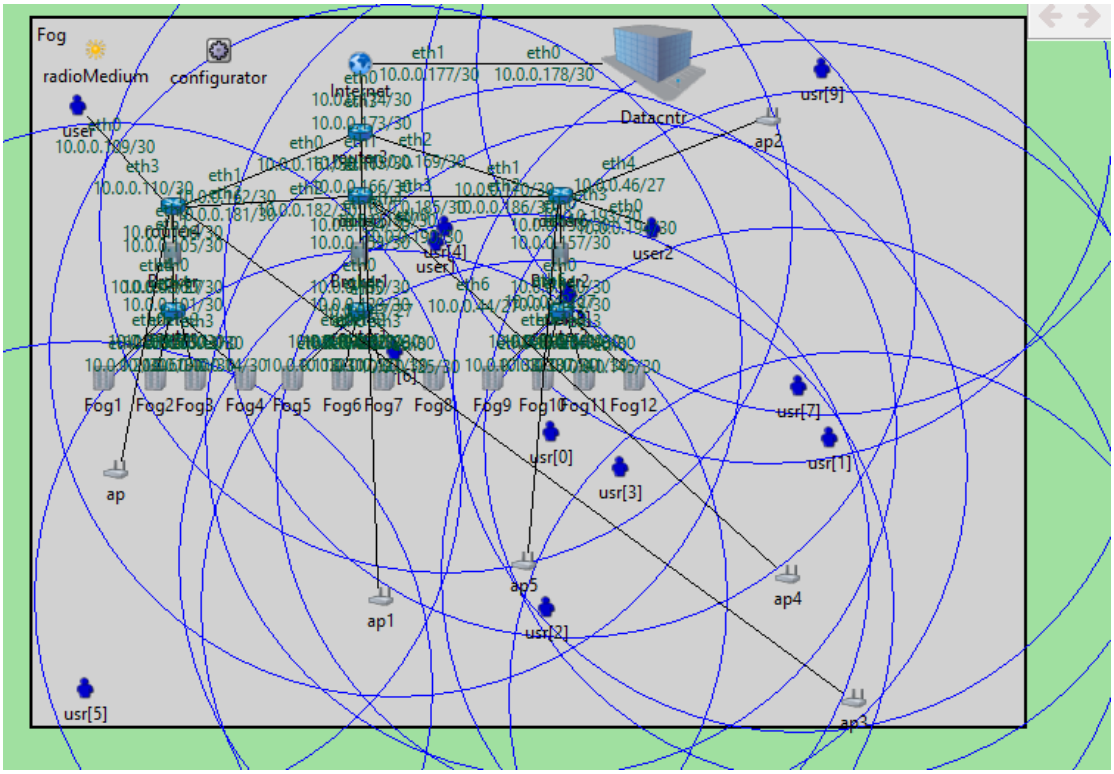


Figure 4.5:Senario2 while impelemntation with mobility parameter

After running simulation in scenario 1 the figures (4.2) and (4.3) will appears respectively and the request beginning to the fog nodes and progresses was began, while implementation, there is tracing file shown in Figure (4.6) that contains all information about connections, files, devices in network and tracking the request from source until the response from destination.

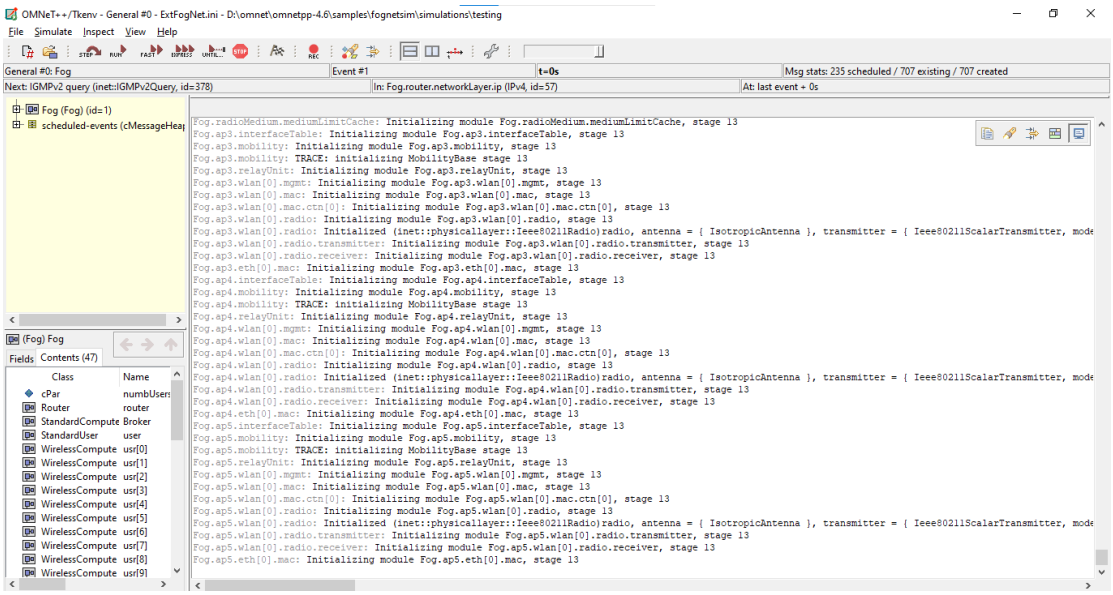


Figure 4.6: Tracing File

4.3 Results

The results (shown in tables below) that include comparison between the result in using GWO algorithm and MO-GWO algorithm in latency and power consumption at the same time and with effect of mobility

- Latency and power consumption with GWO algorithm and MO-GWO (Latency in microsecond, Power consumption in mill watt)

Case1:

In case1 (Table 4-3) the message length (2028B) and base broker MIPS (Million Instruction Per Second) was (1000) and compute broker MIPS was (1000) also, the results show that the message length was effected on delay and energy consuming in implementation, and MO-GWO enhanced the results than traditional GWO.

Table 4-3: Results of case1

Message Length	**Base Broker MIPS**	**Compute Broker MIPS**
2028 B	1000	1000

No.tasks (requests)	Latency (µs)		Power consumption(mw)	
	GWO	MO-GWO	GWO	MO-GWO
926	5.20E+05	1.50E+05	2.98E+02	1.94E+02
1919	7.40E+05	4.29E+05	7.13E+02	2.94E+02
32701	9.64E+05	6.52E+05	3.99E+03	1.05E+03
163218	1.08E+06	7.59E+05	5.47E+03	2.97E+03

Case 2:

In case 2 reducing the message length led to reduce the power and latency in each request, and also that enhanced in MO-GWO rather traditional GWO. The results shown in (Table 4-4)

(Table 4-4): Results of case2

Message Length	**Base Broker MIPS**	**Compute Broker MIPS**
1024 B	1000	1000

No.tasks (requests)	Latency (µs)		Power consumption(mw)	
	GWO	MO-GWO	GWO	MO-GWO
926	2.00E+04	1.95E+04	2.07E+02	1.37E+02
1919	3.30E+05	2.29E+05	2.50E+02	1.84E+02
32701	5.01E+05	6.21E+04	4.65E+02	3.91E+02
163218	6.40E+05	9.58E+04	6.32E+02	5.72E+02

Case3:

In case 3 the message length was (1024 B) and Base Broker MIPS (1000) and the changing occur on Compute Broker MIPS with different values, when the MIPS increasing the energy consumption and latency decrease. The results shown in (Table 4-5)

Table 4-5: the Results of case 3

Message length	Base Broker MIPS	Compute Brokers MIPS
1024B	1000	Compute Broker1 1000 Compute Broker2 2000 Compute Broker3 3000 Compute Broker4 4000

No. tasks (requests)	Latency (µs)		Power consumption(mw)	
	GWO	MO-GWO	GWO	MO-GWO
926	1.03E+04	4.67E+03	2.81E+02	1.50E+02
1919	2.65E+05	2.09E+05	3.04E+02	1.24E+02
32701	4.17E+05	1.25E+04	4.65E+02	2.42E+02
163218	5.79E+05	9.58E+04	7.02E+02	5.52E+02

Apply changing Mobility parameter and make it (massive, circle) on case 3. The result after applying different type of mobility shows increasing in power consuming while latency was decrease. And the enhancement of MO-GWO give acceptable results that qualify the equipment capacity.

Table 4-6: case 3 after Mobility

No. tasks (requests)	Latency (µs)		Power consumption(mw)	
	GWO	MO-GWO	GWO	MO-GWO
926	9.51E+03	3.47E+03	3.81E+02	1.90E+02
1919	1.75E+05	4.04E+04	4.56E+02	2.16E+02
32701	3.92E+05	1.05E+04	5.66E+02	3.24E+02
163218	5.01E+05	8.58E+04	8.14E+02	6.52E+02

To achieve to our recommendation, minimum latency and low power consuming the Pareto was implement to give us responsible results at the same time.

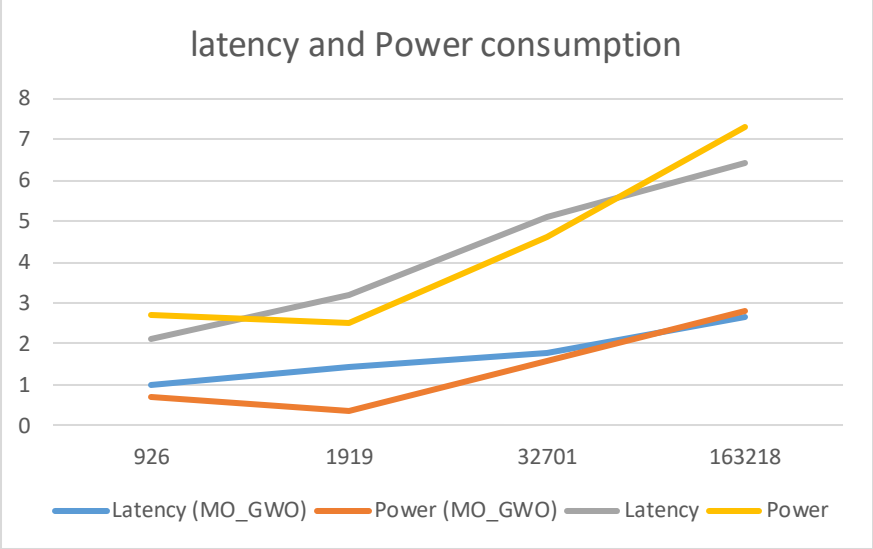
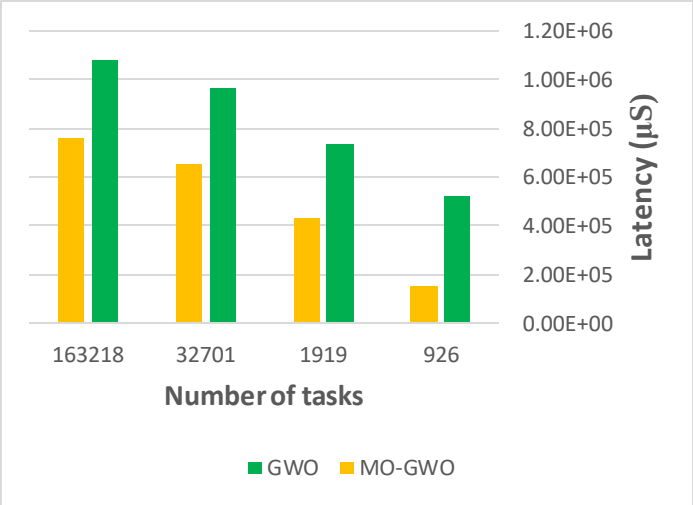
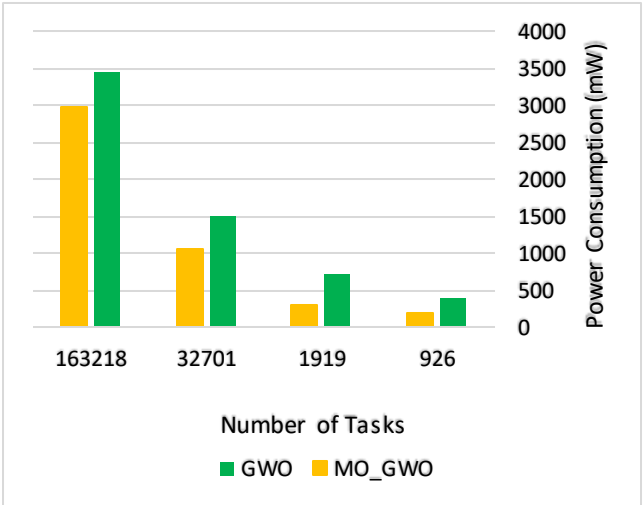


Figure 4.7:Chart of Latency and Power in Network



(a)



(b)

Figure 4.8:Variation of the Latency and the energy consumption (a) Latency; (b) energy consumption

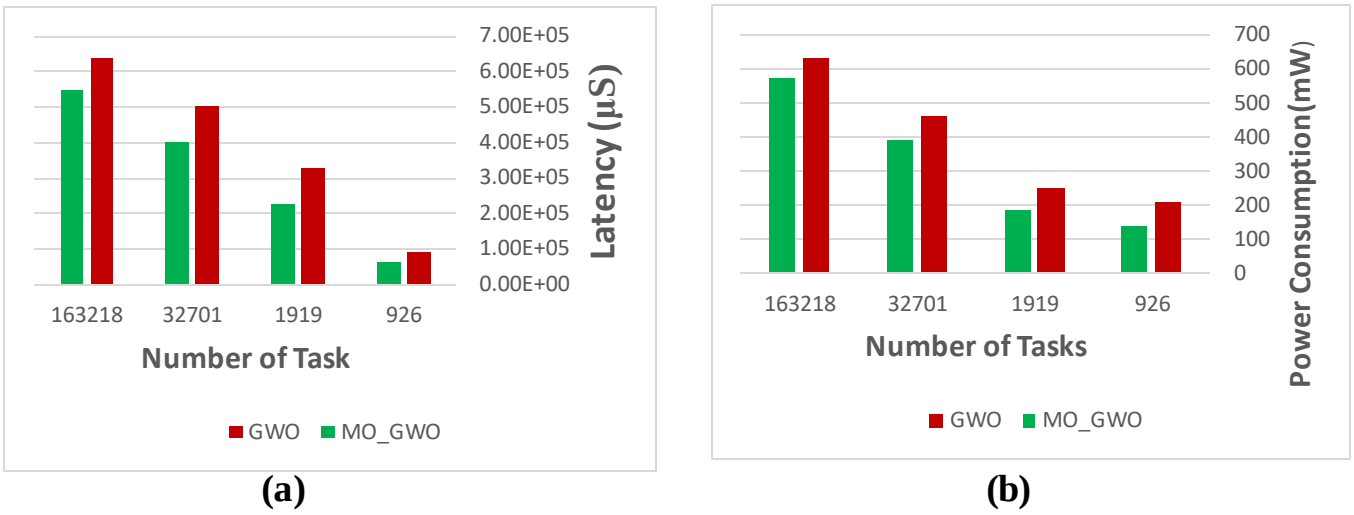
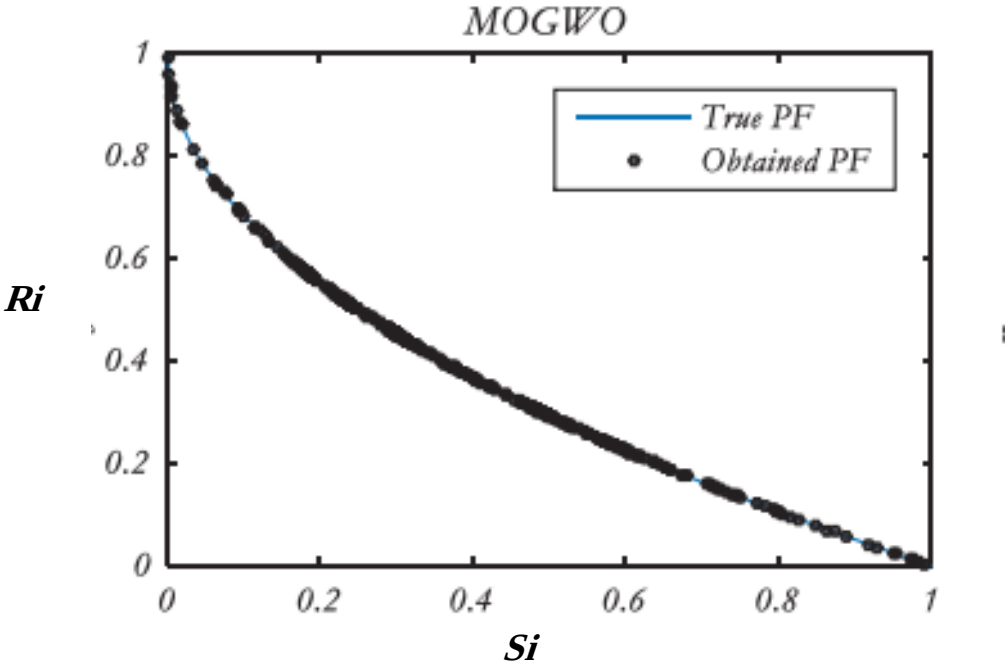


Figure 4.9:Variation of the Latency and the energy consumption respectively; (a) Latency; (b) energy consumption

with the number of Fog when the number of end devices is {25}and message length (1024B)



Figure(4.10):Pareto Optimal Soulution

Chapter Five

Conclusion and Future works

5.1 Introduction

This chapter explains the proposed system conclusions and the main suggestions for future works can be summarized as shown below.

5.2 Conclusion

1- Regarding the Fog computing, the proposed model is a well-known fact that computationally intensive tasks are beyond the capabilities of mobile devices due to their low battery life and processing speed.

2- Developing model to achieve these goals while selecting the optimal combination of offloading tasks with minimal energy consumption and execution latency.

3- The proposed meta-heuristic-based task offloading model in this work. The simulation results demonstrate that MO-GWO outperforms conventional GWO by a large margin.

4- The results show the improvement using MO_GWO in 0.5% than traditional GWO in latency and power consumption at the same time with mobility effect in Fog network.

5- There is no comparison with previous studies due to the lack of use of this simulator, as well as the discussion of latency and the power at the same time under the influence of mobility, instead of that there is a comparison between GWO and MO-GWO at the same scenarios

Successful offloading task selection is the driving force behind the performance advantage. However, real-time systems cannot afford the time required for these metaheuristics to evolve before making offloading decisions. For each evolutionary algorithm, we also provide a graphical representation of the tradeoff analysis between energy and delay, both for small and large task sizes. When it comes to choosing which tasks to delegate, you can trust that they will be handled by GWO. It carefully

chooses which computations to offload, which in turn saves power on all nodes. The figures also show that the energy required to complete a task grows proportionally with its size, whether the task is run locally or offloaded. Furthermore, energy consumption rises as the number of tasks expands. Both of these factors contribute to a general rise in the amount of work required of an Internet of Things (IoT) device or an edge server. Energy consumption from processing unit (CPU) utilization is a given if the computation is done locally. Alternatively, energy consumption for an offloaded task is incurred due to the transmission of data and reception of results to/from the edge server.

5.3 Future works

- It is possible to explore how to get offload decisions faster in dynamically changing heterogeneous scenarios by combining the feature of deep reinforcement learning that builds on prior experience to significantly accelerate the learning of new tasks.
- It is possible to use Software-Defined Networks to Enable Highly Programmable Fog Computing on a Wide Range of Different Types of Computing Hardware

REFERENCES

[1] *S. C. a. S. M. S. Sarkar, "Assessment of the Suitability oFog Computing in the Context of Internet of Things," IEEE Transactions on Cloud Computing, vol. 6, no. 1, pp. 46-59, Jan.-March 2018.*

[2] *Mohit Kumar, S.C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," Procedia Computer Science, vol. 115, pp. 322-329, 2017.*

[3] *Xiaolong Xu , Shucun Fu, Qing Cai, Wei Tian,Wenjie Liu , Wanchun Dou ,Xingming Sun , and Alex X. Liu, "Dynamic Resource Allocation for Load Balancing in Fog Environment," Wireless Communications and Mobile Computing, vol. 2018, 26 Apr 2018.*

[4] *. Deng, R., Lu, C., Lai, T. H. L., & Liang, H., "Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption.," IEEE Internet Things J, vol. 3, no. 6, pp. 1171-1181, 2016.*

[5] *C. Chen, Y.-C. Chang, C.-H. Chen, Y.-S. Lin, J.-L. Chen, andY.-Y. Chang, "Cloud-fog computing for information-centric Internet-of-Things applications," in inProc. Int. Conf. Appl. Syst. Innov. (ICASI), 2017.*

[6] *S. Sarkar and S. Misra, "Theoretical modelling of fog computing: a green computing paradigm to support IoT applications," IET Networks, vol. 5, no. 2, pp. 23-29, 2016.*

[7] *Yousefpour, G. Ishigaki, and J. P, "Fog computing: Towards minimizing delay in the Internet of Things," in inProc. IEEE Int. Conf. EdgeComput. (EDGE), Jue (2017).*

[8] *Chang, Z., Zhou, Z., Ristaniemi, T., & Niu, Z., "Energy efficient optimization for computation offloading in fog computing system," in In GLOBECOM 2017–2017 IEEE Global Communications Conference IEEE, (2017, December).*

[9] *Bonomi, Flavio, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. , ""Fog computing: A platform for internet of things and analytics." In Big data and internet of things: A roadmap for smart environments," Springer, Cham, pp. 169-186, 2014.*

[10] *Zhao, X., Zhao, L., & Liang, K. , "An energy consumption oriented offloading algorithm for fog computing," in . In International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness Springer, Cham., 2016.*

[11] *S. F. Abedin, M. G. R. Alam, N. H. Tran, and C. S. Hong, " A Fog based system model for cooperative IoT node pairing using matching theory," Network Operations and Management Symposium (APNOMS), pp. 309-314, 2015.*

[12] *Hasan, R., Hossain, M., & Khan, R., "Aura: An incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading," in Future Generation Computer Systems, 2018.*

[13] Zahoor, S., Javaid, S., Javaid, N., Ashraf, M., Ishmanov, F., & Afzal, M. K. , " Cloud–fog– based smart grid model for efficient resource management. Sustainability,," 2018.

[14] L. F. Bittencourt, M. M. Lopes, I. Petri and O. F. Rana, "Towards Virtual Machine Migration in Fog Computing,," in International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015.

[15] Nguyen, B. M., Thi Thanh Binh, H., & Do Son, B, "(2019). Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment.," Applied Sciences, vol. 9, no. 9, p. 1730, 2019.

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

[17] Yin, L., Luo, J., Luo, H, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," IEEE Trans. Ind.Inf, vol. 14, no. 10, p. 4712–4721, 2018.

[18] Adhikari, M., Srirama, S.N., "Multi-objective accelerated particle swarm optimization with a container-based scheduling for Internet of-Things in cloud environment," Netw. Comput. Appl., pp. 35-61, 2019.

[19] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5g cellular networks," in Pervasive Computing and Communication Workshops (PerCom Workshops)," in IEEE International Conference , 2016.

[20] Hussein, M. K., & Mousa, M. H., " Efficient task offloading for iot-based applications in fog computing using ant colony optimization.," IEEE Access, vol. 8, p. 37191–37201, 2020.

[21] Razaq, M. M., Tak, B., Peng, L., & Guizani, M., "Privacy-aware collaborative task offloading in fog computing," IEEE Transactions on Computational Social Systems, 2021.

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

[23] Shi, Y., Ding, G., Wang, H., Roman, H. E., & Lu, S., "The fog computing service for healthcare," 2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare, pp. 1-5, 5 2015.

[24] Kaur, S., & Bawa, R. K., "Future trends of data mining in predicting the various diseases in medical healthcare system," International Journal of Energy,Information and Communications, vol. 6, no. 4, pp. 17-34, 2015.

[25] Singh, A., Payal, A., & Bharti, S., "A walkthrough of the emerging IoT paradigm: Visualizing inside functionalities, key features, and open issues," Journal of Network and Computer Applications, vol. 147, pp. 111-151, 2019.

[26] Singh, A., Payal, A., & Bharti, S., "A walkthrough of the emerging IoT paradigm: Visualizing inside functionalities, key features, and open issues.," Journal of Network and Computer Applications,, Vols. 143,, pp. 111-151, 2019.

[27] Irmak, E., & Bozdal, M, " Internet of Things (IoT): The most up-to-date challenges, architectures, emerging trends and potential opportunities," . Foundation of Computer Science, (2018, May).

[28] Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50-58, 2010.

[29] Bonomi, Flavio, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli, "Fog computing and its role in the internet of things," in In Proceedings of the first edition of the MCC workshop on Mobile cloud computing.

[30] Stojmenovic, Ivan, Sheng Wen, Xinyi Huang, and Hao Luan., "An overview of fog computing and its security issues.," Concurrency and Computation: Practice and Experience , vol. 28, no. 10, pp. 2991-3005, 2016.

[31] S. Kabirzadeh, D. Rahbari and M. Nickray, "A Hyper Heuristic Algorithm for Scheduling of Fog Networks," in 2017 21st Conference of Open Innovations Association (FRUCT), 2017.

[32] M. Mukherjee, L. Shu and D. Wang, "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges," IEEE Communications Surveys & Tutorials thirdquarter 2018,, vol. 20, no. 3, pp. 1826-1857, 2018.

[33] Mohit Taneja , Alan Davy, "Resource Aware Placement of Data Analytics Platform in Fog Computing," Procedia Computer Science , vol. 97, pp. 153-156, 2016.

[34] M. A. a. E. Huh, "Dynamic resource provisioning through Fog micro datacenter," in 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), 2015.

[35] N.K. Giang, M. Blackstock, R. Lea, V.C.M. Leung, "Developing IoT applications in the fog: A distributed dataflow approach," in Proc. 5th Int. Conf. Internet Things (IOT), 2015.

[36] "Cisco Fog Director," cisco, [Online]. Available: http://www.cisco.com/c/en/us/products/cloud-systems-management/fog-director/index.html. [Accessed 10 1 2022].

[37] G. Albeanu, F. Popentiu-Vladicescu, "A reliable e-learning architecture based on fog-computing and smart devices,," in Proc. Int. Sci. Conf. eLearn. Softw. Edu, 2014.

[38] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar and J. H. Abawajy,, "Fog of Everything: Energy-Efficient Networked Computing Architectures, Research Challenges, and a Case Study," IEEE Access, vol. 5, pp. 9882-9910, 2017.

[39] Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, Limin Sun, "Fog Computing: Focusing on Mobile Users at the Edge," pp. 1-11, 2016.

[40] S. Sarkar, S. Misra, "heoretical modelling of fog computing :A green computing paradigm to support IoT applications," IET Netw, vol. 5, no. 2, pp. 23-29, 2016.

[41] Alhaidari, F.; Rahman, A.; Zagrouba, R., " Cloud of things: Architecture, applications and challenges.," Ambient Intell. Humaniz. Comput., pp. 1-19, 2020.

[42] Bjerkevik, H.B.; Botnan, M.B.; Kerber, M., " Computing the interleaving distance is NP-hard," Found. Comput. Math. , vol. 20, pp. 1237-1271, 2020.

[43] A.V. Dastjerdi, H. Gupta, R.N. Calheiros, S.K. Ghosh, R. Buyya,, "Fog Computing: principles, architectures, and applications," Internet of Thing, pp. 61-75, 2016.

[44] Cao, Yu & Chen, Songqing & Hou, Peng & Buhl-Brown, Donald., "FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigatio," in Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on. IEEE, 2015.

[45] Stantchev, Vladimir & Barnawi, Ahmed & Ghulam Muhammad, Sarfaraz & Schubert, Johannes & Tamm, Gerrit., "Smart Items, Fog and Cloud Computing as Enablers of Servitization in Healthcare," Sensors & Transducers, vol. 185, no. 2, pp. 121-128, February 2015.

[46] Kosta, Sokol, et al, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in NFOCOM 2012 Proceedings IEEE, 2012.

[47] Zhu, Jiang, et al, ""Improving web sites performance using edge servers in fog computing architecture." Service Oriented System Engineering (SOSE)," in 2013 IEEE 7th International Symposium , 2013.

[48] Aazam, Mohammad, and Eui-Nam Huh, "Fog computing and smart gateway based communication for cloud of things," in Future Internet of Things and Cloud (FiCloud), 2014 International Conference , 2014.

[49] H. Madsen, G. Albeanu, B. Burtschy, and F. Popentiu-Vladicescu., "Reliability in the utility computing era: Towards reliable fog computing," IWSSIP. IEEE.

[50] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman,and H. Bhogan., "Volley: Automated data placement for geo-distributed cloud services.," in NSDI, 2010.

[51] B. Sheng, Q. Li, and W. Mao., " Data storage placement in sensor networks.," in ACM, Mobihoc, 2006.

[52] H. Wang, C. C. Tan, and Q. Li.Snoogle, " A search engine for the physical world.," in In INFOCOM. IEEE , 2008..

[53] H. Wang, C. C. Tan, and Q. Li. Snoogle, " A search engine for pervasive environments.," in TPDS, 2010.

[54] F. Xu, C. C. Tan, Q. Li, G. Yan, and J. Wu., "Designing a practical access point association protocol," in INFOCOM, 2010.

[55] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenw¨ alder, and B. Koldehofe., " Opportunistic spatio-temporal event processing for mobile situation awareness.," in DEBS, 2013.

[56] B. Ottenw¨ alder, B. Koldehofe, K. Rothermel, K. Hong, and U. Ramachandran., "selection-based reuse for distributed complex event processing.," in DEBS, 2014.

[57] Dsouza, Clinton, Gail-Joon Ahn, and Marthony Taguinod. , ""Policy-driven security management for fog computing: Preliminary framework and a case study." Information Reuse and Integration (IRI)," in 2014 IEEE 15th International Conference, 2014.

[58] Misra, Prasant, Yogesh Simmhan, and Jay Warrior, ". "Towards a Practical Architecture for the Next Generation Internet of Things.," " arXiv preprint arXiv:1502.00797, 2015.

[59] De Brito, M.S.; Hoque, S.; Steinke, R.; Willner, A.; Magedanz, T., " Application of the fog computing paradigm to smart factories and cyber-physical systems.," Trans. Emerg. Telecommun. Technol., vol. 29, 2018.

[60] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenw¨alder, and B. Koldehofe., " Mobile fog: A programming model for large-scale applications on the internet of things," in In ACM SIGCOMM workshop on Mobile cloud computing, 2013.

[61] N. I. M. Enzai and M. Tang, " A taxonomy of computation offloading in mobile cloud computing.," in In MobileCloud. IEEE, 2014.

[62] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies., " The case for vm-based cloudlets in mobile computing.," in Pervasive Computing, 2009.

[63] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl., "Maui: making smartphones last longer with code offload. In Mobisys.," ACM, 2010.

[64] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. , "Clonecloud: elastic execution between mobile device and cloud," ACM, 2011.

[65] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang., "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading.," in INFOCOM, 2012.

[66] M. S. Gordon, D. A. Jamshidi, S. A. Mahlke, Z. M. Mao, and X. Chen., " Comet: Code offload by migrating execution transparently.," in OSDI, 2012.

[67] Shahidinejad, A., Ghobaei-Arani, M., " Joint computation offloading and resource provisioning for edge-cloud computing environment: a machine learning-based approach.," vol. 50, no. 12, p. 2212–2230 , 2020.

[68] Z. Hao, Y. Tang, Y. Zhang, E. Novak, N. Carter, and Q. Li., " SMOC: a secure mobile cloud computing platform.," in INFOCOM, 2015.

[69] H. Takabi, J. B. Joshi, and G.-J. Ahn., " Security and privacy challenges in cloud computing environments.," in IEEE Security and Privacy, 2010.

[70] I. Stojmenovic and S. Wen. , "The fog computing paradigm:Scenarios and security issues. In Computer Science and Information Systems (FedCSIS)," in 2014 Federated Conference , 2014.

[71] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. , "Privacy-preserving multi-keyword ranked search over encrypted cloud data.," in TPDS, 2014.

[72] C. Marforio, N. Karapanos, C. Soriente, K. Kostiainen, and S. CE¨Gapkun., " Smartphones as practical and secure ˇ location verification tokens for payments.," in NDSS, 2014.

[73] H. Han, B. Sheng, C. C. Tan, Q. Li, and S. Lu., " A measurement based rogue ap detection scheme.," in INFOCOM. IEEE, 2009.

[74] H. Han, B. Sheng, C. C. Tan, Q. Li, and S. Lu., " A timing-based scheme for rogue ap detection.," in TPDS, 2011.

[75] S. Smalley and R. Craig. , "Security enhanced (se) android: Bringing flexible mac to android," in NDSS, 2013.

[76] S. Yu, C. Wang, K. Ren, and W. Lou., " Achieving secure, scalable, and fine-grained data access control in cloud computing," in INFOCOM, 2010.

[77] C. Dsouza, G.-J. Ahn, and M. Taguinod., " Policy-driven security management for fog computing: Preliminary framework and a case study.," in IRI. IEEE, 2014.

[78] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan., " A survey of intrusion detection techniques in cloud.," in JNCA, 2013.

[79] C. Wang, Q. Wang, K. Ren, and W. Lou., "Privacy-preserving public auditing for data storage security in cloud computing.," in INFOCOM, 2010.

[80] A. Rial and G. Danezis, " Privacy-preserving smart metering," in In Proceedings of the 10th annual ACM workshop on Privacy in the electronic society,, 2011.

[81] Z. Qin, S. Yi, Q. Li, and D. Zamkov, " Preserving secondary users' privacy in cognitive radio networks," in In INFOCOM, 2014 Proceedings IEEE,, 2014.

[82] E. Novak and Q. Li. , "Near-pri: Private, proximity based location sharing.," INFOCOM, 2014.

[83] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen. , "Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications.," TPDS, 2012.

[84] C. Dwork., "Differential privacy.," in Encyclopedia of Cryptography and Security, 2011.

[85] Kishor, A., Chakraborty, C. H., & Jeberson, W., "A novel fog computing approach for minimization of latency in healthcare using machine learning," Int J Interact Multimed Artif Intell, vol. 6, no. 6, pp. 10-20, 2021.

[86] D.K. Bui, T.N. Nguyen, A. Ghazlan, N.T. Ngo, T.D. Ngo, " Enhancing building energy efficiency by adaptive façade: a computational optimization approach," Appl. Energy 265, 2020.

[87] D.K. Bui, T.N. Nguyen, A. Ghazlan, N.T. Ngo, T.D. Ngo, " Enhancing building energy efficiency by adaptive façade: a computational optimization approach," Appl. Energy 265, vol. 114797, 2020.

[88] K. Bamdad, M.E. Cholette, J. Bell, " Building energy optimization using surrogate model and active sampling,," Build. Perform. Simulat., vol. 13, no. 6, pp. 760-776, 2020.

[89] K. Bamdad, M.E. Cholette, J. Bell, "Building energy optimization using surrogate model and active sampling,," J. Build. Perform. Simulat, vol. 13, no. 6, p. 760–776, 2020.

[90] B. Chegari, M. Tabaa, E. Simeu, F. Moutaouakkil, H. Medromi,, " Multi-objective optimization of building energy performance and indoor thermal comfort by combining artificial neural networks and metaheuristic algorithms, Energy Build," no. 239, 2021.

[91] Gharehchopogh, F.S.; Shayanfar, H.; Gholizadeh, H. , "A comprehensive survey on symbiotic organisms search algorithms.," in Artif. Intell. Rev, 2020.

[92] Mishra, S.K.; Puthal, D.; Rodrigues, J.J.; Sahoo, B.; Dutkiewicz, E., " Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications.," IEEE Trans. Ind. Inform., vol. 14, p. 4497–4506., 2018.

[93] I. Boussa¨ıD, J. Lepagnot, and P. Siarry,, "A survey on optimization metaheuristics," Information sciences, vol. 237, pp. 82-117, 2017.

[94] V. Machairas, A. Tsangrassoulis, K. Axarli, , "Algorithms for optimization of building design: a review, Renew. Sustain. Energy," Rev. 31 (2014), p. 101–112., 2014.

[95] K. Kumar, J. Liu, Y.-.H. Lu, B. Bhargava,, "A Survey of Computation Offloading for Mobile Systems," Mobile Networks and Applications, vol. 18 , no. 1, p. 129–140, Feb 2013.

[96] F. Guo, H. Zhang, H. Ji, X. Li, V.C.M. Leung, , "An Efficient Computation Offloading Management Scheme in the Densely Deployed Small Cell Networks With Mobile Edge Computing," IEEE/ACM Transactions on Networking , vol. 26, no. 6, p. 2651–2664., Dec. 2018.

[97] P. Mach, Z. Becvar, " Mobile Edge Computing: a Survey on Architecture and Computation Offloading," IEEE Communications Surveys & Tutorials , vol. 3, no. 19, p. 1628–1656, 2017.

[98] K. Bilal, O. Khalid, A. Erbad, S.U. Khan,, " Potentials, trends, and prospects in edge technologies: fog, cloudlet, mobile edge, and micro data centers," Computer Networks, p. 94–120, Jan. 2018.

[99] A. Yousefpour, G. Ishigaki, R. Gour, J.P. Jue, , "On Reducing IoT Service Delay via Fog Offloading," IEEE Internet of Things Journal, vol. 5, no. 2, p. 998–1010, Apr. 2018.

[100] J. Du, L. Zhao, J. Feng, X. Chu, " Computation Offloading and Resource Allocation in Mixed Fog/Cloud Computing Systems With Min-Max Fairness Guarantee," IEEE Transactions on Communications, vol. 66, no. 4, p. 1594–1608, Apr. 2018 .

[101] Margariti, S. V., Dimakopoulos, V. V., & Tsoumanis, G., " Modeling and Simulation Tools for Fog Computing—A Comprehensive Survey from a Cost Perspective.," Future Internet, vol. 12, no. 5, p. 89, 2020.

[102] Hagras, H., Callaghan, V., Colley, M., & Carr-West, M., "A behaviour based hierarchical fuzzy control architecture for agricultural autonomous mobile robots., control and au," in In The International Conference on computational intelligence for modelling , control and automation (pp. 172-177), (1999).

[103] "Simplesoft simpleiotsimulator , accessed: 2020-04-10.," [Online]. Available: https://www.simplesoft.com/SimpleIoTSimulator.html. [Accessed 12 1 2022].

[104] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," IEEE IoT J, vol. 4, no. 5, pp. 1185-1192, Oct 2017.

[105] C. Sonmez, A. Ozgovde, and C. Ersoy,, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in in 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), May 2017.

[106] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya,, "“Fogbus: A blockchainbased lightweight framework for edge and fog computing," arXiv preprint arXiv:1811.11978,, 2018.

[107] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya,, " “ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," ” Software: Practice and Experience, vol. 47, no. 9, p. 1275–1296, 2017.

[108] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue,, " “All one needs to know about fog computing and related edge computing paradigms: A complete survey,”," J. Syst. Archit., 2019.

[109] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies,, " ”The Case for VM-Based Cloudlets in Mobile Computing pp.," IEEE Pervasive Computing, vol. 8, no. 4, pp. 14-23, Oct.-Dec. 2009..

[110] T. Qayyum, A. W. Malik, M. A. Khan Khattak, O. Khalid and S. U. Khan, , ""FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment,," IEEE Access, vol. 6, pp. 63570-63583, 2018.

APPENDEX A

7.1 Simulator Setup

Because FogNetSim++ is built on top of OMNeT++, the first step is to install OMNeT++. OMNeT++ requires the following packages, which are given below.

- **OMNeT++ installation**
  1- Go to download OMNET++ version 4.6 that compatible with FogNetSim++
  2- after extracting OMNET++ go to its folder and open Windows Command Line (mingwenv) to setting it up
  3- making build to simulation library (./configure) shown in figure (3)

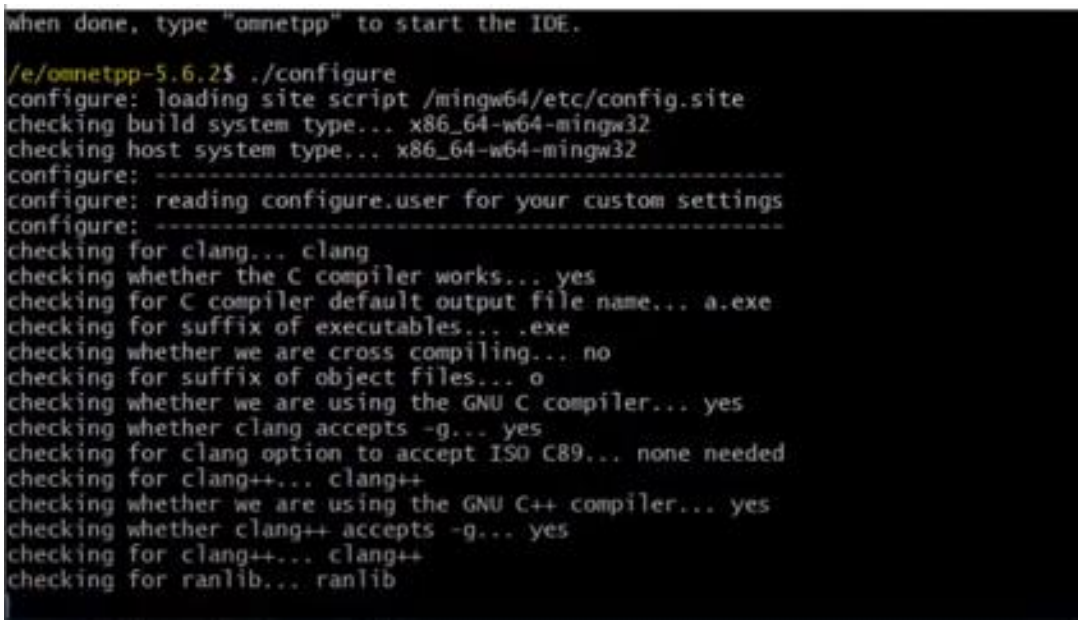


Figure 7.1: OMNET++ Installation

4- Go to inet.omnetpp.org to download INET Framework version 3.3.0 that compatible with OMNET++ 4.6 and save it in OMNET folder

- **FogNetSim++ installation**

  Visit github.com/rtqayyum/Fognetsimpp[1] on GitHub and download the appropriate installation files ('Fognetsimpp.zip' and'mqttapp.zip') in order to set up FogNetSim++. These procedures should be taken once the download is complete:

  1- If the directory isn't the same, expand the INET project, expand the source folder, and then expand the apps folder, where'mqttapp.zip' should be extracted and placed. You may see additional application directories in this directory, which is where you should copy and paste the "mqttapp" folder.

  2- Load the 'Fognetsimpp.zip' file as an OMNeT++ project. Just add a reference to INET by right-clicking on the project and selecting "Add Reference"

1 https://github.com/rtqayyum/fognetsimpp

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل كلية تكنولوجيا المعلومات
قسم شبكات المعلومات

الاسلوب الفعال لتوزيع الاحمال بين عقد الضباب لتحسين اداء اجهزة انترنت الاشياء

الرسالة

مقدمة الى مجلس كلية تكنولوجيا المعلومات جامعة بابل وهي جزء من متطلبات نيل درجة الماجستير
في تكنولوجيا المعلومات / شبكات المعلومات

من قبل الطالبة

اشراق ماضي جبور جاسم

أشراف

أ.م.د. هلال عبد الحسين عبود هادي

2022 م

1444 هـ

الخلاصة

تشير حوسبة الضباب إلى بنية الحوسبة التي توفر إمكانات التخزين والتحكم والشبكات لتمكين تطبيقات إنترنت الأشياء. توسيع خدمات السحاب وجعلها اقرب للمستخدم حيث يتم تحويل أعباء عمل إنترنت الأشياء إلى الحوسبة الضبابية بدلاً من السحابة ، يمكن تقليل أوقات الاستجابة لتحليلات البيانات من خلال استخدام زمن انتقال منخفض. بسبب الزيادة الهائلة في الأجهزة المتصلة بإنترنت الأشياء ، يجب ضمان جودة الخدمة QoS تحقق حوسبة الضباب هذا من خلال دمج تحسينات أجهزة إنترنت الأشياء بذكاء وتوسيع إمكانات الحوسبة السحابية. لتحسين الحساب وجودة الخدمة ، يعد اختيار عقدة الضباب أمرًا بالغ الأهمية. يتم تلبية متطلبات جودة الخدمة للمدن الذكية والطاقة الذكية والرعاية الصحية والسيارات والزراعة وما إلى ذلك عن طريق حوسبة الضباب . على الرغم من أن حوسبة الضباب الآن عمرها سنوات ، إلا أنها لا تزال في بدايتها ويعتبر زمن التأخير واستهلاك الطاقة المنخفض هما مجالان تتفوق فيهما هذه الدراسة لأول مرة ، تمت مناقشة تأثيرات تنقل الأجهزة المحمولة على أداء الشبكة ، فضلاً عن زيادة زمن الوصول واستهلاك الطاقة بدرجات متفاوتة اثناء التنقل ، في سياق توزيع الاحمال في الحوسبة الضبابية وللوصول الى طرقى مثلى للحصول على اقل زمن وصول باستهلاك اقل كمية من الطاقة بنفس الوقت تم استخدام خوارزمية تحسين الذئب الرمادي متعددة الأغراض MO-GWO لإدارة موارد الشبكة لاستخدامزمن الوصول واستهلاك الطاقة. تم استخدام محاكي "FogNetSim ++" المعروف لإعداد تجربة وبناء شبكة دراسة حالة في طبقة الضباب والنتائج التي ظهرت قبل وبعد استخدام خوارزمية التحسين الدليل القاطع على التنفيذ الناجح حيث تم تحسين النتائج عند استخدام ال MO-GWO على زمن الانتقال والطاقة بنفس الوقت وبتأثير ال mobility بنسبة 0.5% مقارنة بخوارزمية الذئب الرمادي التقليدية .