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

A PROPOSED HYBRID LOAD BALANCING APPROACH TO IMPROVE THE FOG COMPUTING PERFORMANCE

A Thesis Submitted

to the Council of the College of Information Technology for Postgraduate Studies at the University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology / Information Networks.

By

Abrar Saad Kadhim Hassan

Supervised by

Asst. Prof. Dr. Mehdi Ebady Manaa Mehdi

2022 A.D. **1444 A.H**

بسم الله الرحمن الرحيم

يرفع الله الذين آمنوا منكم والذين أوتوا العلم درجات والله بما تعملون خبير

صدق الله العظيم

سورة المجادلة . أية 11

Certification of the Examination Committee

We hereby certify that we have studied the thesis entitled (**A PROPOSED HYBRID LOAD BALANCING APPROACH TO IMPROVE THE FOG COMPUTING PERFORMANCE**) presented by the student (**ABRAR SAAD KADHIM HASSAN**) and examined her in its content and what is related to it, and that, in our opinion, it is adequate with (**Very Good**) standing as a thesis for the degree of Master in Information Technology-Information Networks.

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

Signature:
Name: Raad A. Muhajjar
Title: Asst. Prof. Dr.
Date: / / 2022
(**Member**)

Signature:
Name: Saba M. Hussain
Title: Lecturer
Date: / / 2022
(**Member**)

Signature:
Name: Mehdi Ebady Manaa
Title: Asst. Prof. Dr.
Date: / / 2022
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

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

Supervisor Certification

I certify that the thesis entitled (**A PROPOSED HYBRID LOAD BALANCING APPROACH TO IMPROVE THE FOG COMPUTING PERFORMANCE**) was prepared under my supervision at the department of Information Networks/ College of Information Technology / University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology-Information Networks.

Signature:

Supervisor Name: **Asst. Prof. Dr. Mehdi Ebady Manaa**

Date: / /2022

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "**A PROPOSED HYBRID LOAD BALANCING APPROACH TO IMPROVE THE FOG COMPUTING PERFORMANCE"** for debate by the examination committee.

Signature:

Prof. Dr. **Saad Talib Hasson Aljebori**

Head of Information Networks Department

Date: / /2022

Declaration

I hereby declare that this thesis, 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 : **Abrar Saad Kadhim Hassan**

Date: **/ / 2022**

Dedication

To my affectionate mother, may God have mercy on her, affection of my childhood, it is embodied in this thesis, guiding her with pride, love and pride.

My dear father, Dr. Saad, is the pillar of wisdom, the source of knowledge, literature, and knowledge, an inexhaustible help throughout my life. Father, God help him.

My brothers

Dr. Aqeel is pure and my support. May God grant him peace and care teachers.

(Amir, Karrar, Hussein), Affection ensures that they live a long life.

Researcher

Acknowledgement

In the name of Allah most merciful:

“And We had certainly given Luqman wisdom [and said], "Be grateful to Allah ." And whoever is grateful is grateful for [the benefit of] himself. And whoever denies [His favor] - then indeed, Allah is Free of need and Praiseworthy.” Luqmān :12

I extend my sincere thanks and appreciation to **Asst. Prof. Dr. Mehdi Ebady Manaa**. May the honorable one separate him by supervising the thesis and honoring him with my advice and guidance, and who spared no effort in preparing it, may God prolong his life with great pride, I address my distinguished professors in the College of Information Technology in general and the Networks Department, especially the distinguished **Prof. Dr. Saad Talib Hasson Aljebori**, Head of the department those who were a great help to me in completing this thesis, may God bless them all.

I thank the brothers and sisters working in the central library of the University of Babylon and the Library of the College of Information Technology for helping me by providing me with references for review in filling the gaps in this thesis. I wish them success, and praise is [due] to Allah, Lord of the worlds.

Researcher

Abstract

The Internet of Things (IoTs) applications could choose fog computing nodes for responding to the resource requirements, and load balancing is one of the key factors to achieve resource efficiency and avoid bottlenecks, overload, delay and low load. However, it is still a challenge to realize the load balance for the computing nodes in the fog environment during the execution of IoT applications. The load balancing has some goals in fog environment, including throughput maximization, response time minimization, and traffic optimization. Consumption optimization in the server-side resources, request processing time minimization, and scalability improvement in the distributed environment are some other purposes of the technique of fog computing load balancing.

To reduce the impact load of traffic nodes and fog node failure on the fog computing architecture, each node should use a distributed structure for better results. For the aspect of node load. The service node uses the distributed technology, which integrate the computing resources of the whole network, improve the utilization of computing resources. The proposed methodology is based on hybrid load balancing with Least Connection (LC) and Weighted Round Robin (WRR) algorithms combined together in fog nodes to take into consideration the traffic load, and response time to distribute requests to the available servers.

The results showed improving network performance by it ensures the efficient servicing of the IoT requests, as in distributed Hybrid approach the response time is 131.48 ms, total packet loss rate is 15.670%, average total channel idle is 99.55 %, total channel utilization is 77.44 %, average file transfer speed is 260.77 KB/seconds, average time of file transfer is 19.27 seconds.

Table of Contents

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

Abbreviation	Description
CF-EE	Candidate Fog Node based Energy Efficient
CPU	Control Process Unit
DLBM	dynamic load balancing mechanism
DLS	Distributed Load Sharing
EE	Energy Efficient
EM	Expectation Maximization
ERAA	Efficient Resource Allocation Algorithm
FCFS	First Come First Serve
FNs	Fog Nodes
HEFT	Heterogeneous Earliest Finish Time
IoT	Internet of Things
LAN	Local Area Network
LB	Load Balance
LBSSA	Load Balanced Service Scheduling Approach
LC	Least Connection
M2F	Mist-assisted Fog computing-based load balancing
MDs	Mobile Devices
MPSO	Modified Practical Swarm Optimization
MQTT	Message Queue Telemetry Transmission
NFV	Network functions virtualization
OPT	Optimization of Processing Time
ORT	Optimize Response Time
QoS	Quality of Service
QTCS	Queuing Theory based Cuckoo Search
RDLB	Reconfigure Dynamically with Load Balancing
RFID	Radio Frequency Identification
RR	Round Robin
SJF	Shortest Job First
SLA	Service Level Agreement
TRAM	Technique for Resource Allocation and Management
VFC	Vehicular Fog Computing
VM	Virtual Machines
WOA-BAT	Whale Optimization Algorithm With The Bat Algorithm
WRR	Weighted Round Robin

List of Thesis Related Publications

1st paper Name of Conference:

- **Fifth International Iraqi Conference on Engineering Technology and its Applications (5th IICETA).**

2nd Paper Published on the journal:

- **Bulletin of Electrical Engineering and Informatics (BEEI), ISSN: 2089-3191, e-ISSN: 2302-9285، http://beei.org. This journal is indexed by Scopus (Elsevier)/ScimagoJR(SJR), SNIP 2020: 0.954, CiteScore 2020: 1.7, SJR 2020: 0.251.**

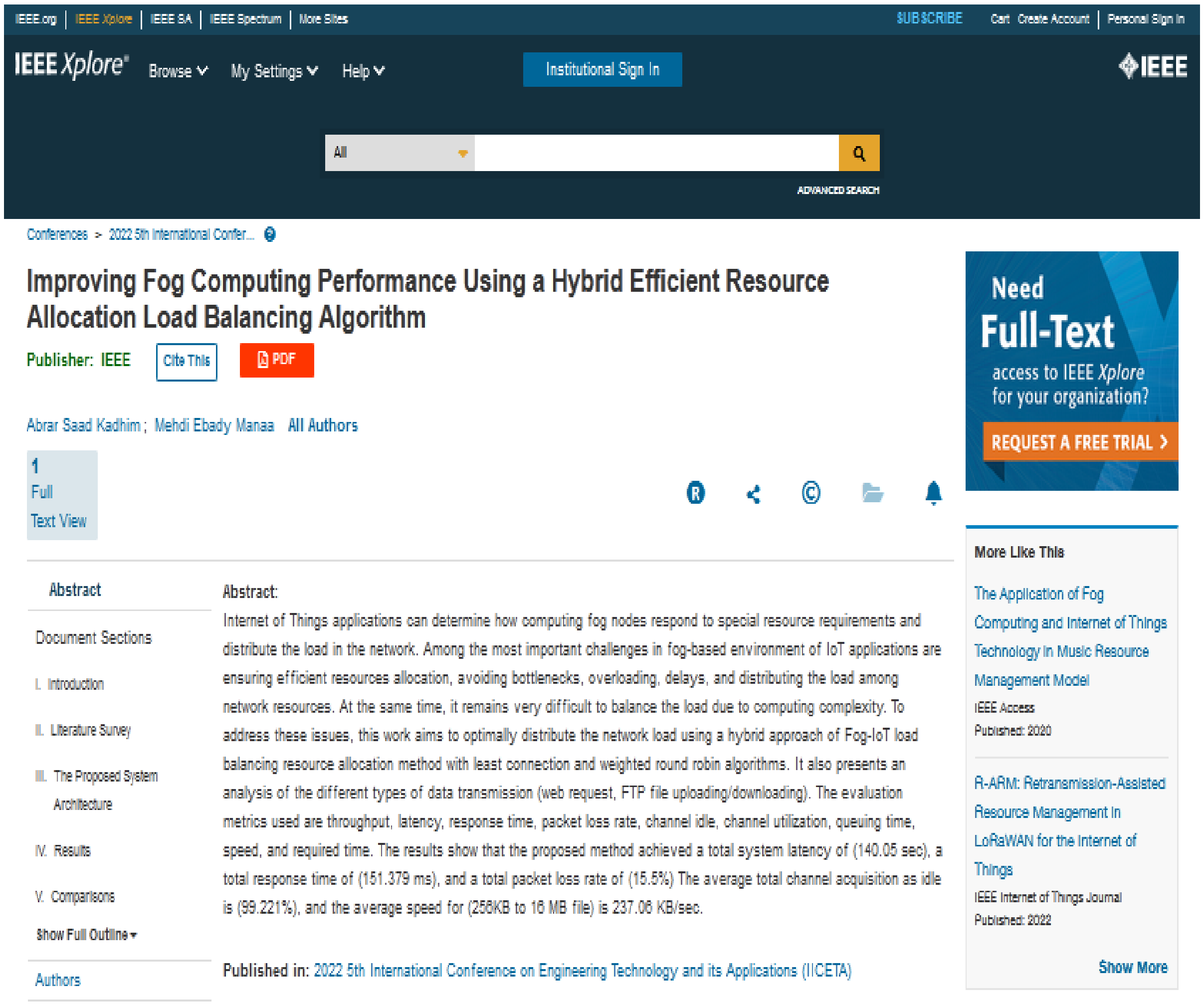
- **1st Paper Title: Improving Fog Computing Performance using Hybrid Efficient Resource Allocation Load Balancing Algorithm.**
- **2nd Paper Title: Hybrid Load-balancing Algorithm for Distributed Fog Computing in Internet of Things Environment.**

- **Authors:**
    - Abrar Saad Kadhim Hassan
    - Asst. Prof. Dr. Mehdi Ebady Manaa Mehdi

Information Network Department, Information Technology College, Babylon University.

Email: abrar.kadhim@student.uobabylon.edu.iq

Email: It.mehdi.ebadi@itnet.uobabylon.edu.iq


IEEE.org | IEEE Xplore | IEEE SA | IEEE Spectrum | More Sites
SUBSCRIBE
Cart Create Account | Personal Sign In
IEEE Xplore
Browse
My Settings
Help
Institutional Sign In
All
ADVANCED SEARCH
Conferences > 2022 5th International Confer...

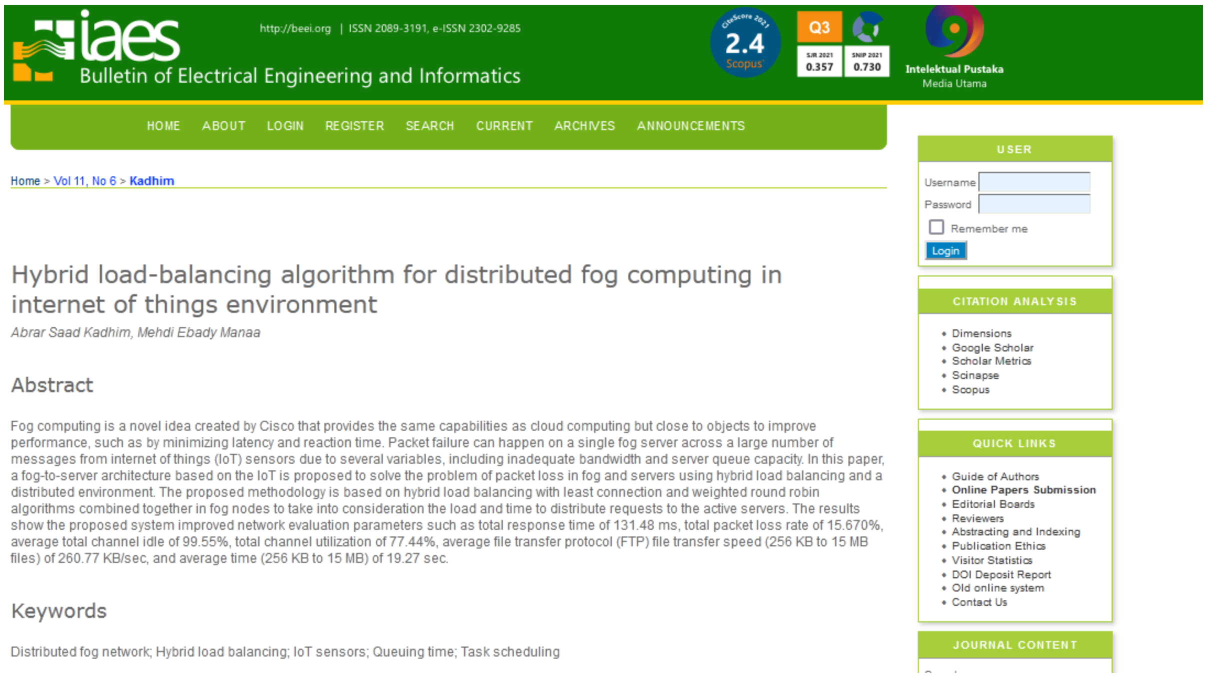
Improving Fog Computing Performance Using a Hybrid Efficient Resource Allocation Load Balancing Algorithm

Abrar Saad Kadhim ; Mehdi Ebady Manaa All Authors

Abstract:
Internet of Things applications can determine how computing fog nodes respond to special resource requirements and distribute the load in the network. Among the most important challenges in fog-based environment of IoT applications are ensuring efficient resources allocation, avoiding bottlenecks, overloading, delays, and distributing the load among network resources. At the same time, it remains very difficult to balance the load due to computing complexity. To address these issues, this work aims to optimally distribute the network load using a hybrid approach of Fog-IoT load balancing resource allocation method with least connection and weighted round robin algorithms. It also presents an analysis of the different types of data transmission (web request, FTP file uploading/downloading). The evaluation metrics used are throughput, latency, response time, packet loss rate, channel idle, channel utilization, queuing time, speed, and required time. The results show that the proposed method achieved a total system latency of (140.05 sec), a total response time of (151.379 ms), and a total packet loss rate of (15.5%) The average total channel acquisition as idle is (99.221%), and the average speed for (256KB to 16 MB file) is 237.06 KB/sec.

Published in: 2022 5th International Conference on Engineering Technology and its Applications (IICETA)

More Like This
The Application of Fog Computing and Internet of Things Technology in Music Resource Management Model
IEEE Access
Published: 2020
R-ARM: Retransmission-Assisted Resource Management in LoRaWAN for the Internet of Things
IEEE Internet of Things Journal
Published: 2022
Show More



http://beei.org | ISSN 2089-3191, e-ISSN 2302-9285

Home > Vol 11, No 6 > Kadhim

Hybrid load-balancing algorithm for distributed fog computing in internet of things environment

Abrar Saad Kadhim, Mehdi Ebady Manaa

Abstract

Fog computing is a novel idea created by Cisco that provides the same capabilities as cloud computing but close to objects to improve performance, such as by minimizing latency and reaction time. Packet failure can happen on a single fog server across a large number of messages from internet of things (IoT) sensors due to several variables, including inadequate bandwidth and server queue capacity. In this paper, a fog-to-server architecture based on the IoT is proposed to solve the problem of packet loss in fog and servers using hybrid load balancing and a distributed environment. The proposed methodology is based on hybrid load balancing with least connection and weighted round robin algorithms combined together in fog nodes to take into consideration the load and time to distribute requests to the active servers. The results show the proposed system improved network evaluation parameters such as total response time of 131.48 ms, total packet loss rate of 15.670%, average total channel idle of 99.55%, total channel utilization of 77.44%, average file transfer protocol (FTP) file transfer speed (256 KB to 15 MB files) of 260.77 KB/sec, and average time (256 KB to 15 MB) of 19.27 sec.

Keywords

Distributed fog network; Hybrid load balancing; IoT sensors; Queuing time; Task scheduling

Chapter One

Introduction

1.1 Introduction

Internet of Things has been growing, due to which the number of user requests on fog computing layer has also increased. Fog works in a real-time environment and offers from connected devices need to be processed immediately. With the increase in users requests on fog layer, virtual machines (VMs) at fog layer become over loaded. Load balancing mechanism can distribute load among all the VMs in equal proportion[1]. In addition, IoT networks are embedded with technologies that helps to communicate and interact within themselves and external environment [2].

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[3]. 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 [4].

Fog computing is based on the idea of locating data processing and storage closer to the nodes that are really producing the data at the lowest perceptual layer[5]. A few examples include application placement, scheduling, provisioning, offloading allocation, and load balancing. Control Process Unit(CPU) utilization, reaction time, bandwidth, power consumption, delay, data traffic, service migrations ratio, and number of active nodes are among the most important performance measures studied in the literature[6].

1.2 Problem Definition

Because of rising requirements and demands for high quality in IoT applications, the fog computing situation requires for improved output, decreased latency, enhanced performance, etc. Effective resource allocation in the fog computing environment is another key challenge in fog computing [7].

To reduce latency and computing requirements, the most pressing issue is developing resource management algorithms for deciding which parts of analytics applications should be distributed to which fog load balancer nodes[8].

However, the compute device's resources may not always be readily available since they are shared with other tasks outside of the Fog environment. The execution of apps may be impacted by this in addition to user behavior that is always changing [9]. The proposed model of Internet of Things, Fog, and Cloud is applied and the effect of resource management strategies to deal with delay, response time, consumption of resources, bandwidth, execution time, and resource availability are measured to accommodate users' unpredictable behavior in this environment.

1.3 Thesis Contribution

The main contributions include:

- Implementing a method for resource management in a fog environment that uses a hybrid load balancing strategy and a weight value to better determine which server will be used to handle requests.

- To guarantee a balanced network loading within the restrictions of the transmission performance, it is necessary to analyze the resource allocation issue in fog computing networks using a hybrid load balancing approach with minimum delay.

1.4 Aims of the Thesis

The main aims of the proposed system are :

- Enhancement resource allocation, and traffic optimization in fog computing environment by using a hybrid load balancing method by combination of the Least Connection load balancing algorithm and the Weighted Round Robin load-balancing algorithm to implement a server load balancer approach.
- Improving the resource management based on the used evaluation parameters through designing an efficient architecture and a resource allocation algorithm with the main goals:
  - A- Bandwidth maximization by increasing channel utilization by wireless nodes in distributed case study.
  - B- Response time minimization, and request processing time minimization.
  - C- Scalability improvement, and decrease end-to-end latency.

1.5 Related Work

In this section, the most related works in term of improve Fog computing performance using efficient resource allocation algorithms have been discussed and overviewed as follow:

The researchers in [10] at 2020 proposed a Fog-to-Cloud architecture based on the Internet of Things to address the problem of packet loss on Fog servers via the use of load balancing and virtualization. The fog layer is separated into a fixed number of fog servers using Network Simulation Simulator-3 and VirtualBox on the underlying physical server. The Server Load Balance router is configured to use the Weighted Round Robin allocation technique, and the Message Queue Telemetry Transmission (MQTT)control protocol to disperse the very high amount of data. By increasing the efficiency with which the Fog layer stores and processes messages, they also able to double the layer's throughput and cut packet loss in half.

An effective exploitation of the Public cloud has concentrated in [11] at 2020 by dividing the application modules across Fog devices and the cloud data centers. This was done in order to maximize the consumption of the Cloud-Fog resources. The performance of several performance criteria, including reaction time, latency, energy usage, and so on, are improved when application modules are placed on Fog devices. They suggested two load balancing algorithms, the performance of which was tested using the iFogSim simulator, and the performance of which was compared with the performance of the cloud-only solution. Fog computing has been proven to be able to lower the amount of delay and network bandwidth usage by around 90 %.

The energy efficient (EE) allocation of resources problem in fog computing networks has been evaluated in [12] at 2020 with the candidate fog nodes (FNs) process to ensure the connectivity loading balance under the transmission constraints. EE stands for energy efficient, and resource allocation refers to the process of distributing available computing resources in an effective and efficient manner. The results of the simulations show that the Candidate FN based EE allocation of resources (CF-EE) technique that was presented might produce a noteworthy performance increase in comparison to the algorithms that are already available in the literatures.

An explanation of the challenges associated is provided by [13] at 2020 with the allocation of tasks and the placement of virtual machines while using a single fog computing platform. In order to examine how well the suggested method operates in a fog environment, it was developed in the CloudSim tool. According to the findings, the suggested technique is capable of allocating resources in a manner that is superior to the strategy that handles resource allocation by default. When contrasted to the other approaches, the overall processing time required by the model that has been presented is shorter. The experiment has been completed, and the results demonstrate that the suggested framework enhances Quality of Service(QoS) in an environment characterized by fog.

The challenge of assigning limited fog resources to vehicular applications, which was stated by [14] at 2020, in such a way that the operation latency is reduced, by making use of parked automobiles via the use of a heuristic algorithm. The proposed algorithm is also combined with supervised learning in order to make decisions regarding resource allocation that are more effective. These decisions are made by utilizing the information regarding the movement of vehicles and their parking status that is obtained from the intelligent environment and infrastructure. When compared to traditional resource allocation algorithms, the findings indicate that the Vehicular Fog Computing (VFC) resource allocation algorithm is capable of achieving greater levels of service satisfaction.

A optimal power flow scheme for the Internet of Things fog computing network of interconnected multiple mobile devices (MDs)has been suggested by [15] at 2020. The purpose of this scheme is to minimize the scheme costs associated to latency, energy consumption, and weights of MDs, and it is dependent on a Lyapunov optimization-based algorithm for the joint processing offloading and radio allocation of resources. by reducing as much as possible the upper limit that has been calculated for the Lyapunov dispersion function. The viability of the strategy that has been presented may be established via the use of performance assessment.

To reduce total delay, time loss, and Service Level Agreement (SLA) breaches, a multi-criteria-based resource allocation strategy was presented [16] at 2020 with capacity reserve. This method takes into account objective functions and dynamic changes in user requirements in order to locate suitable resources for implementing time-sensitive tasks in a Fog environment, taking into account characteristics unique to Fog computing such as device diversity, resource constraints, and mobility. As shown by the experiments, the suggested approach is superior than the current one, cutting down the overall delay by 51%. Time-critical applications may benefit from the suggested algorithm's reductions in process time and SLA violations in the Fog environment.

For the purpose of handling heterogeneity, the authors of [17] at 2020 have proposed two decentralized load balancing algorithms: Sequential Forwarding and Adaptive Forwarding. They also use theoretical models and simulations in their investigation. The suggested methods significantly improve upon the status quo in which no balancing is implemented, as shown by a 19 improvement in drop rate reduction and a 19% improvement in response time. It can virtually half the reaction time in a practical environment, and the loss rate can be reduced from 13% to the less than 0.2%.

In [18] at 2020, the authors have implemented both static and dynamic load balancing methods through the POX controller and examine how they affect the average network performance as the workload requests increase from 0 to 180 per second (req/sec). Compared to static load balancing systems like random, round robin, and weighted round robin, dynamic least bandwidth-based load balancing has been demonstrated to significantly increase average network performance by up to 8%, 3.3%, and 2.56 percent, respectively. In contrast, Dynamic Least Bandwidth showed a marginally inefficient improvement. When comparing the performance of static and dynamic least connections, we found that they differed by less than 1 %.

A cloud-to-fog architecture with fog nodes has been developed by in[19] at 2020 to handle tasks such as large-scale storage, low latency, heterogeneity, resource allocation and interaction with a small number of Internet of Things devices, and security. Factors including bandwidth use, location awareness, reaction time, cost maintenance, intrusion detection, fault tolerance, and maintainability are just some of the areas that have been analyzed in depth to compare and contrast various scheduling algorithms' approaches to load balancing.

The resource allocation problem in Cloud-Fog situations has been addressed by [20] at 2021. Load Balanced Service Scheduling Approach (LBSSA) takes into account load balancing across resources when allocating requests to them, categorizing them as either real-time, essential, or time-tolerant. The method employs a collection of algorithms for processing various requests. CloudSim is used to execute simulation studies that evaluate the LBSSA strategy with regards to the quantity and usage of computing resources, the load balance variation, and the amount of time required to complete a given task.

There is a need for a reliable system to allocate fog layer resources and tolerate failures has been presented by [21] at 2021. Utilizing Nash equilibrium as a starting allocation method and feeding that information on to a reinforcement learner is a win for game theory. The allocation is predicated on the current and past activity of the network. The suggested work provides an enhanced 32ms recovery time and an increased average processing times in the event of a breakdown. In terms of increased service time, decreased delay, and optimum energy utilization, the experimental findings make sense.

In order to effectively allocate and manage resources, a new method has been developed by [22] at 2021 by Technique for Resource Allocation and Management (TRAM), a method for managing and allocating resources, to make sure the fog layer makes use of its available resources. This method employs an Expectation Maximization (EM) algorithm to keep tabs on the stress level of ongoing projects and determine where things stand in terms of available resources. For the purpose of resource grading in the fog computing setting, it offers a scheduling technique. The findings of the iFogSim simulator were compared to those of Shortest Job First (SJF), First Come First Serve (FCFS) and Modified Practical Swarm Optimization (MPSO) simulations. The experiments showed that TRAM significantly reduces task execution time, network use, energy consumption, and average loop delay.

A load balancing technique called Mist-assisted Fog computing-based load balancing (M2F) balancer has been proposed in [23] at 2021 for each server load using dynamic resources allocation method. The proposed method is being compared with the Least count, Round Robin, and Weighted Round Robin. In the end, the results demonstrate that the solutions enhance QoS in the mist assisted cloud environment concerning maximization resource utilization and minimizing the makespan. Consequently, it improves performance even at peak times

In order to effectively allocate resources in Fog computing, a [24] at 2022 have used a Queuing Theory based Cuckoo Search (QTCS) model to enhance Quality of Service will lead to more efficient use of available resources. The suggested model is simulated using CloudSim, a cloud computing simulator. The findings demonstrated a 20.39 % improvement in average reaction time and a 12.55 % decrease in average energy usage when compared to conventional methods.

In order to improve the efficiency of the fog computing environment, a load-balancing technique that takes use of the optimization of processing time (OPT) algorithms has been presented by [25] at 2022. The performance of the suggested algorithm is analyzed by comparing it with other algorithms. The results show that the suggested load-balancing algorithm with optimized processing time provides faster response and process time for user requests and lower total cost of ownership for data centers.

In order to effectively balance workloads, a cloud services system has been suggested by [26] at 2022. Load balancing is improved by combining the Whale Optimization Algorithm with the Bat Algorithm (WOA-BAT). The model's processing and reaction time are compared to those of state-of-the-art load balancing methods including throttled, round robin, whale, and particle swarm algorithms. According to the findings, the suggested WOA-BAT outperforms the other three algorithms by 4.3 percentage points when comparing reaction time. Moreover, it has the fastest processing time of any algorithm by at least 22.3%.

Table (1.1) illustrated the aims of previous researchers and the main methods which are used to achieved the works.

Table 1.1: The literature survey summary.

Ref, Year	Methods	Tool	Aims	Results
[10], 2020	Weighted Round Robin	Ns-3	solve the problem of packet loss on Fog server using Load Balancing and virtualization	Avg Speed 110 KB/sec, Packet Loss Rate (%) is 180 %
[11],2020	Proximity Algorithm, Cluster Algorithm	Cloud-Fog load balancing, iFogSim	Improving performance like response time, latency, energy consumption	Total Time Taken of Proximity Algorithm is 90 sec, and Cluster Algorithm is 100 sec.
[12], 2020	FNs based EE resource allocation (CF-EE) algorithm	cloud-fog analytics	Energy Efficient Resource Allocation in Fog Computing Networks	It achieved a highly considerable performance improvement

Ref, Year	Methods	Tool	Aims	Results
[13], 2020	efficient resource allocation algorithm (ERAA)	CloudSim	Improving quality-of-service in fog computing through efficient resource allocation	the processing time of proposed model is reduced by 61%
[14], 2020	vehicular fog computing (VFC) algorithm	Python 3.7, Tensorflow	Resource Allocation for Vehicular Fog Computing	VFC resource allocation algorithm can achieve higher service satisfaction compared to conventional resource allocation algorithms.
[15], 2020	Lyapunov optimization	Fog cloud	Dynamic Resource Allocation and Computation Offloading for IoT Fog Computing System	the effectiveness of the proposed scheme
[16], 2020	Multi-Criteria-based resource allocation policy	CloudSim	address dynamic changes in user behaviour in a resource limited Fog device	reducing the total delay by 51%.
[17],2020	sequential forwarding and adaptive forwarding algorithms	Omnet++	address the problem of resources management	decrease drop rate, and Response time
[18],2020	dynamic least connections	Mininet	improve the load performance of the server	Least bandwidth load balancing scheme better handled the network traffic
[19],2020	FCFS, SJF, PS, RR and WRR	Cloud analytics	allocating the devices to respective fog nodes based on the number of devices connected	Enhanced throughput, burst time, average waiting time

Ref, Year	Methods	Tool	Aims	Results
[20],2019	Load Balanced Service Scheduling Approach (LBSSA)	CloudSim	reduce load of computing in data centers and reduce latency of requests	Enhanced computing resources, utilization of resources, load balance variance and running time.
[21],2020	Game theory, and Nash equilibrium allocation strategy	Mininet	improve the efficiency of task scheduling in dynamically changing network resource allocation.	Improved service time, lower delay, and optimal energy utilization
[22],2021	SJF, FCFS and MPSO	iFogSim	Resource allocation and management in fog computing environment	Minimized execution time, network consumption, energy consumption and average loop delay of tasks.
[23],2021	A mist-assisted fog computing-based load balancing (M2FB)	MATLAB	increase resource utilization	Enhancing QoS, maximization resource utilization and minimizing the makespan.
[24],2022	Queuing Theory based Cuckoo Search (QTCS) model	CloudAnalyst, CloudSim	Allocating resources to ensure Quality of Service (QoS)	Time Taken for Resource Allocation is 8 sec
[25], 2022	Optimizing Processing Time (OPT) , First Come First Serve (FCFS), Priority algorithm	Hybrid load-balancing-Fog, iFogSim	Decreasing processing and the response time	Total Response Time of OBT is 309.5 ms, FCFS is 326.7 ms, Priority algorithm is 323.7 ms.
[26], 2022	Round Robin (RR), Whale Optimization algorithm with bat algorithm (WOA-BAT)	cloud-fog, Java Netbeans and cloud analyst	To overcome latency and processing time	Total Response Time of RR is 270.14 ms, and WOA-BAT is 256.59 ms

1.6 Thesis Outline

The thesis is divided into five chapters in addition to chapter one :

Chapter Two: It presents fundamental concepts of Fog computing, fog layers, interfaces, packet forwarding, Fog-load balancer signaling. Also, Fog Architecture Issues, Limitations and challenges. In addition to the load balancing techniques as static load balancing, dynamic load balancing and Hybrid Load Balancing, finally evaluation of Load Balancing algorithms have been sorted.

Chapter Three: It presents the used system and illustrates the working steps of the system as the used architecture for designing and implementing the proposed method.

Chapter Four: It shows the achievement of the suggested algorithm with the result explanations.

Chapter Five: It summarizes the findings and makes recommendations for future research.

Chapter Two

Theoretical Background

2.1 Introduction

The term "Internet of Things" refers to the general concept of things, participants took things, that are intelligible, readily identifiable, locatable, directly accessible through sensors and other application specific device and/or easily controlled via the Internet, regardless of the effective communication (whether via Radio Frequency Identification (RFID), connectivity local area network (LAN), public networks, or other means). [27] Internet of Things is a term that was coined by Kevin Ashton.

Fog computing is a computing architecture in which a series of nodes receives data from IoT devices in real time. These nodes perform real time processing of the data that they receive, with millisecond response time[28]. The nodes periodically send analytical summary information to the cloud. On the other hand, due to the resource limitations, resource heterogeneity, dynamic nature, and unpredictability of fog environment, it necessitates the resource management issues as one of the challenging problems to be considered in the fog landscape [29].

2.2 Fog Computing

Fog is a new development in the world of communication that uses "fog nodes," which are located on the edge of a cloud, to perform cloud services. Each fog node has a certain number of users to whom they may provide services, like in the examples below[30], [31]:

- Data storage and retrieval
- Network using
- Tasks Relating to Command and Control
- Data server coordination and resource allocation

- Capabilities in computation

Fog computing system models will be presented, including network, computation, analysis, and communication models[32]. Figure (2.1) presented a systemic view of the interconnections between fog computing and IoT applications [33].

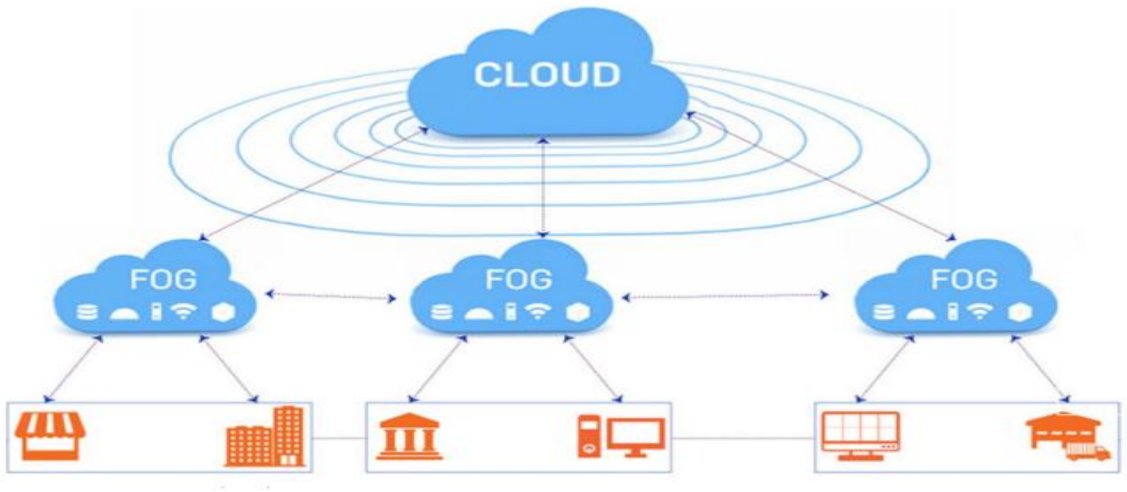


Figure 2.1: System model for Fog Computing[33].

Methods are needed so that a network model for evaluating fog computing may be established. Fog layer interpreter, resource selection, processing effectiveness and analysis, history analyzer, resource picker, task scheduler, IoT device allocation and management, etc, are all possible solutions based on the premise of the fog computing environment, these features may be interpreted in a way that meets the needs of a fog computing environment[34].

Fog computing have huge benefits in the real time applications. It is broadly used in IOT applications which involves real time data. It acts as an extended version of cloud computing. It is an intermediate between the cloud and end users (closer to end users).It can used in both the ways, that can be between machine and machine or between the human to machine [35].

2.2.1 The architecture of Fog Computing

A- **The physical and virtualization layer** : It is the layer that is the most fundamental, and it is concerned with each and every "thing" that is able to connect to the internet or a network and produce data. It is made up of nodes, devices, virtual and physical sensors, vehicles, and other things like that[36].

B- **The monitoring layer** : It dictates the next job that will be carried out to monitor generated data. The amount of energy that nodes use is also monitored in order to ensure that appropriate actions, such as job offloading, are done at the appropriate moment in accordance with the condition of the sensor[37].

C- **The storage layer** : The storing of data inside a fog is the responsibility of this component. The majority of the data that is kept in fog is only there temporarily. The cloud is preferable for long-term storage due to the fact that it offers much more resources[37].

D- **The fog security layer** : Before delivering the data across a public or potentially susceptible route, it protects the information by implementing the necessary safety and privacy safeguards[38].

In fog computing, resource management is one of the most essential topics that need be taken into mind in order to spread load across servers node based on load balancing algorithms. This is one of the reasons why fog computing was developed. Whether it be over one or more computers, network ports, hard disks, or other computer resources, load balancing techniques are responsible for distributing the burden. These risks include equipment failure, energy and/or network disruptions, and insufficient resources during times of high demand. In fog computing, the load is balanced using the primary two approaches, which are known as static collectively called and dynamic load balancing, respectively [39].

As mentioned earlier, fog computing spreads the standard cloud computing to the edge, and therefore, it is also referred to as edge computing. The concept of fog computing is to bring net-working resources near the nodes that are generating the data residing at the lowest perception layer. Fog is a highly virtualized platform that is responsible for providing computation, storage, and networking services between the end nodes in an IoT environment and traditional clouds, as showed in Figure (2.2) [40].

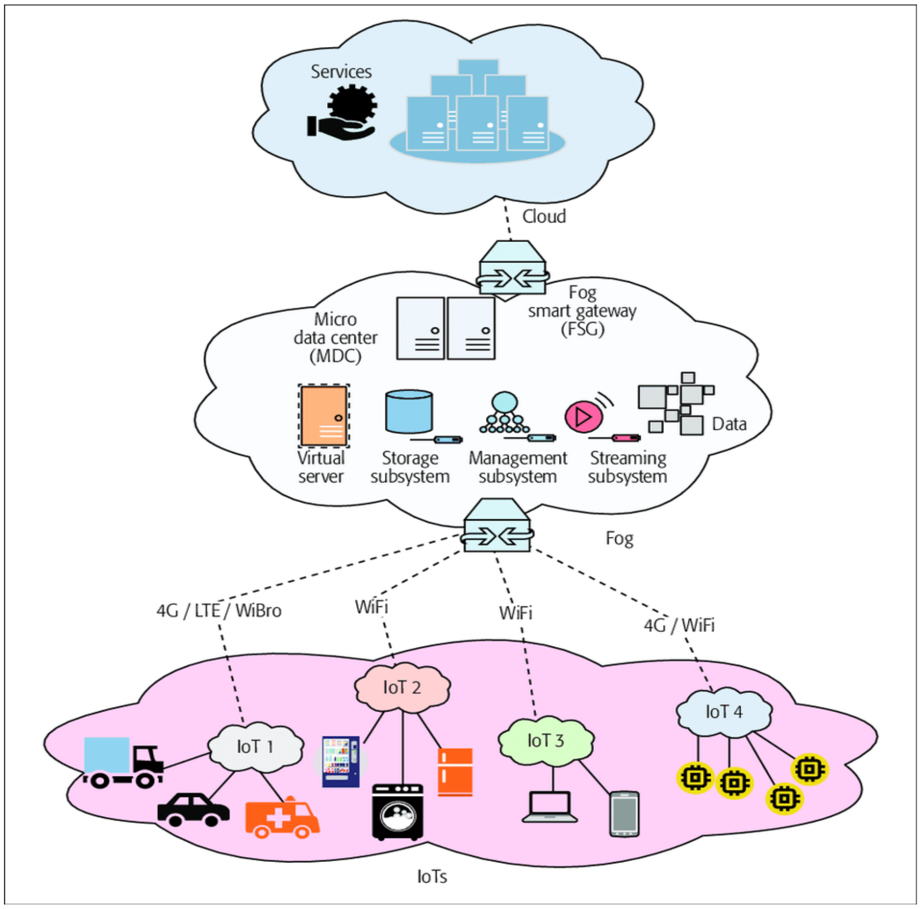


Figure 2.2: Overall architecture and positioning of fog [40].

2.2.2 Benefits of Fog Computing

Some advantages of using Fog Computing are as follows:

- As an alternative to transmitting data to the cloud, it processes them on-premises. As a result, it helps save data transfer rates across networks. Thus, operating expenses are reduced[41].
- For this reason, decisions may be taken rapidly with less delay. When done, this aids in the prevention of accidents [41].
- Since user data is processed locally rather than uploaded to a remote server, this solution provides increased security and privacy for the user's information. In addition, the IT department has complete control over all the hardware.
- Using the correct technologies, creating fog apps that can drive equipment based on client needs is simple.
- The fog nodes are always on the mobility. Thus, they are not restricted in any way in terms of entering or leaving the network[42].
- Fog nodes are resilient enough to function in extreme environments including train tracks, moving vehicles, the under ocean, and industrial settings. Additionally, it may be set up in outlying areas.
- Since data is evaluated at the edge nodes, latency is decreased with fog computing. This is because there is less data capacity and shorter round trip time[43].

2.2.3 Fog Computing Issues

Fog computing is a kind of Internet of Things-enabled cloud computing. The so-called fog nodes are portable and may be set up wherever an internet connection is available. When it comes to processing needs, the fog has supplementary edge storage. Because of this, the Fog server will need to adjust its offerings, which will need more time and money to operate and maintain. Furthermore, the operator must deal with the following difficulties [44]:

A- Network Management

There will be a lot of effort involved in controlling fog nodes, networks, and connections between each node if heterogeneous devices are used [45].

B- Placement of Fog Servers

Optimal placement of a cluster of fog servers to meet the needs of a given area is a challenge. The price of server upkeep may be minimized by analyzing the workload of each node before deciding where to place it [46].

C- Delay in Computing

Problems with data collection, When fog servers are overworked, they can't perform as well, which slows down data processing. Prior to processing, data should be gathered, and scheduling for resource-constrained fog nodes should be based on a priority and mobility model [47].

D- Energy Consumption

Computation in fog settings is scattered among a large number of fog nodes, which may reduce efficiency. Energy efficiency is so crucial in fog computing [48].

2.2.4 Fog-computing in IoT applications

In a conventional network, a single hub would struggle to handle the massive amounts of data generated by various devices and transactions.

Techniques based on older data warehouse models might be slower to respond and fail to provide the low latency rate demanded by end users. Fog computing provides infrastructure with reduced latency and fewer bottlenecks. At the end of the system, geo-distributed fog servers do calculations. Each fog server is equipped with similar processing power, allowing for massive workloads to be handled locally [49].

As a result, fog computing has become an important force behind the Internet of Things at academic institutions. With the proliferation of Internet-enabled gadgets comes an explosion of data, and with that comes the possibility of uploading that information to a centralized cloud. Data may be examined locally, and users can then decide what information needs to be sent to a centralized cloud (2.3) [50].

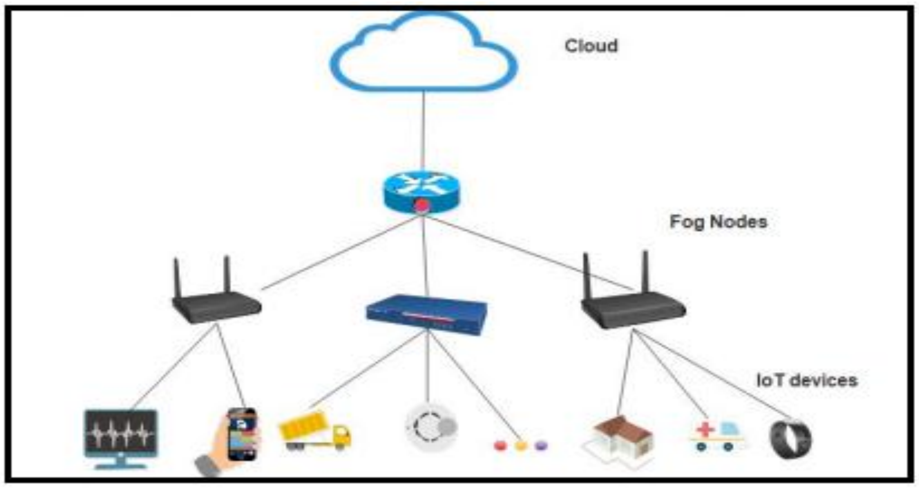


Figure 2.3: Edge Computing and Fog Computing Together[50].

2.2.5 Resource management approaches in Fog-IoT Environment

The following difficulties related to managing network resources or effecting fog node load balancing are mentioned in[51] as a result of the exponential increase of fog node end-users:

- Controlling the load of numerous Internet of Things devices on fog nodes
- Assigning resources to multiple Internet of Things (IoT) devices through fog nodes
- Identification or tracking of different IoT's; Estimation of substantial bandwidth owing to device mobility; Location awareness; Intrusion Detection; Preservation of data integrity, conviction, and privacy; Authentication of various IoT's via fog nodes [51].

2.3 Load Balancing Techniques

Load balancing is the technique of distributing the workload evenly among many processors to improve overall performance. Workload refers to the entire amount of time a machine must spend processing in order to complete its given work. With load balancing, idle Virtual Machines are used by transferring their workload to other, less busy VMs. While an application is running, load balancing may slow down or speed up the system. Since the time it takes for an application to complete its work on a given set of resources is highly variable depending on the circumstances, load balancing is used to improve performance in this regard[52]. In fog computing, load balancing is performed across both the real and virtual nodes. The load balancing technique divides the workload uniformly across the available processing nodes. Both the starting point and the current condition may be used to categorize load balancing algorithms. The first category has three subtypes: those that are initiated by the sender, those that are initiated by the recipient, and those that rely on both the sender and the receiver [53]. In addition, Figure (2.4) that follows presents a classification of the load-balancing techniques used in fog computing.

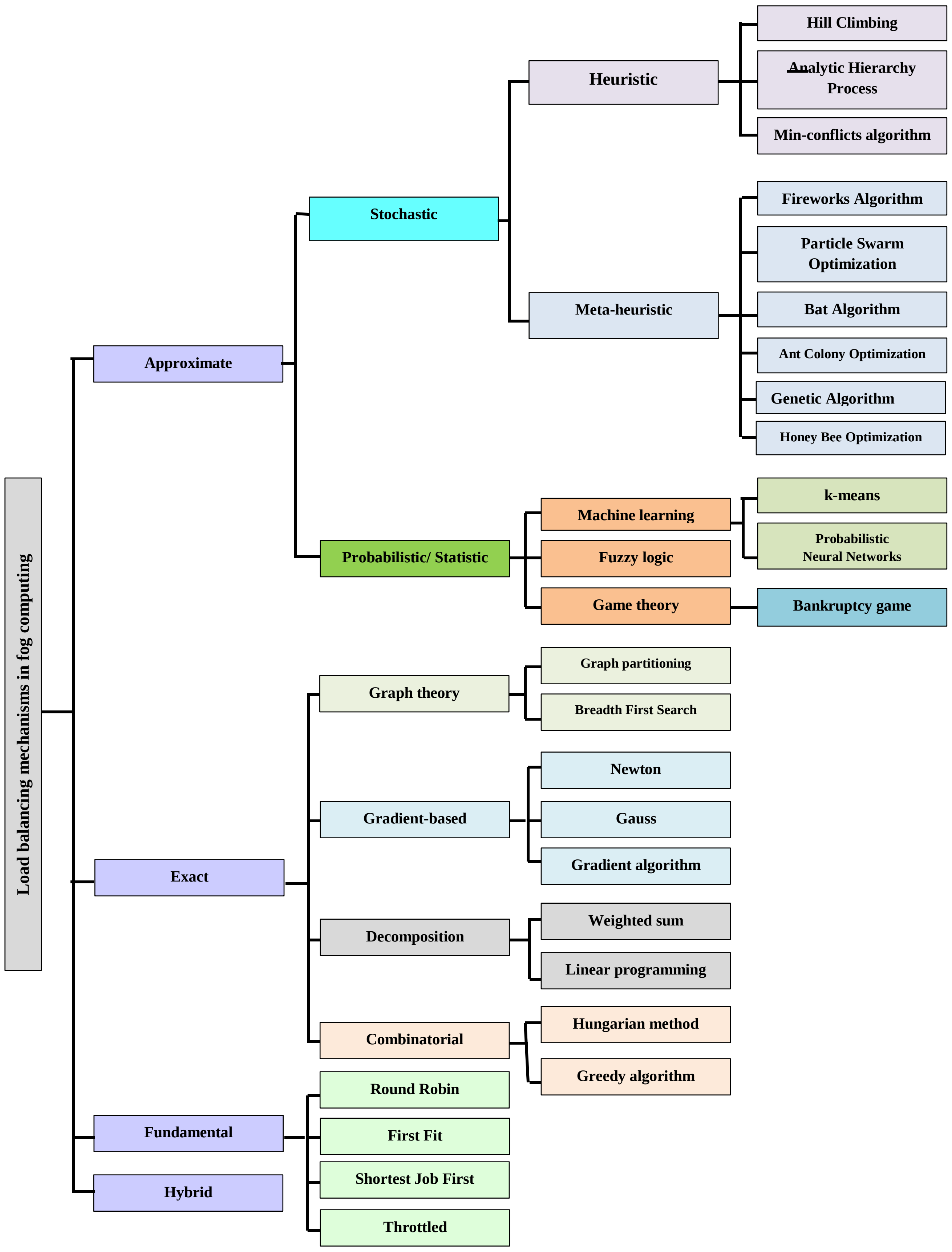


Figure 2.4: In fog computing, a classification of load-balancing techniques[53].

Load balancing is required in fog computing to maximize the efficiency of the available resources. Below it explored some benefits of integrating resource scheduling with fog computing[54]:

- **Availability** : As a result of task scheduling, fog resources are readily accessible. If a particular resource fails to respond, there are still others that may be used. In the event of latency-sensitive applications, a load balancer monitors for both under- and over-utilization to prevent service disruptions[55].
- **Flexibility** : In the event of a failure in one resource, the load balancer ensures that the other resources can continue to process data and reply to users[56].
- **Reduced energy consumption:** All resources, whether actively performing a task or sitting idly, require energy. Load balancing ensures that all available resources work at about the same rate. Therefore, power usage in hardware nodes may be lowered by preventing both overuse and underuse of available resources[57].
- **Reduced cost** : The cost of fog load balancers is low since it is proportional to the amount of resources used. As a result of load balancing, a variety of capabilities may be employed selectively in response to user demands in fog computing while keeping hardware costs low[58].
- **Resource utilization** : By using load balancing, all of the available resources in a fog computing setting may be used effectively. Because load balancers prevent both under- and over-utilization of resources, they are becoming common[59].

The main Load balancing methods for static, dynamic, and hybrid loads were shown in Figure (2.5) along with the most popular load balancing categories.

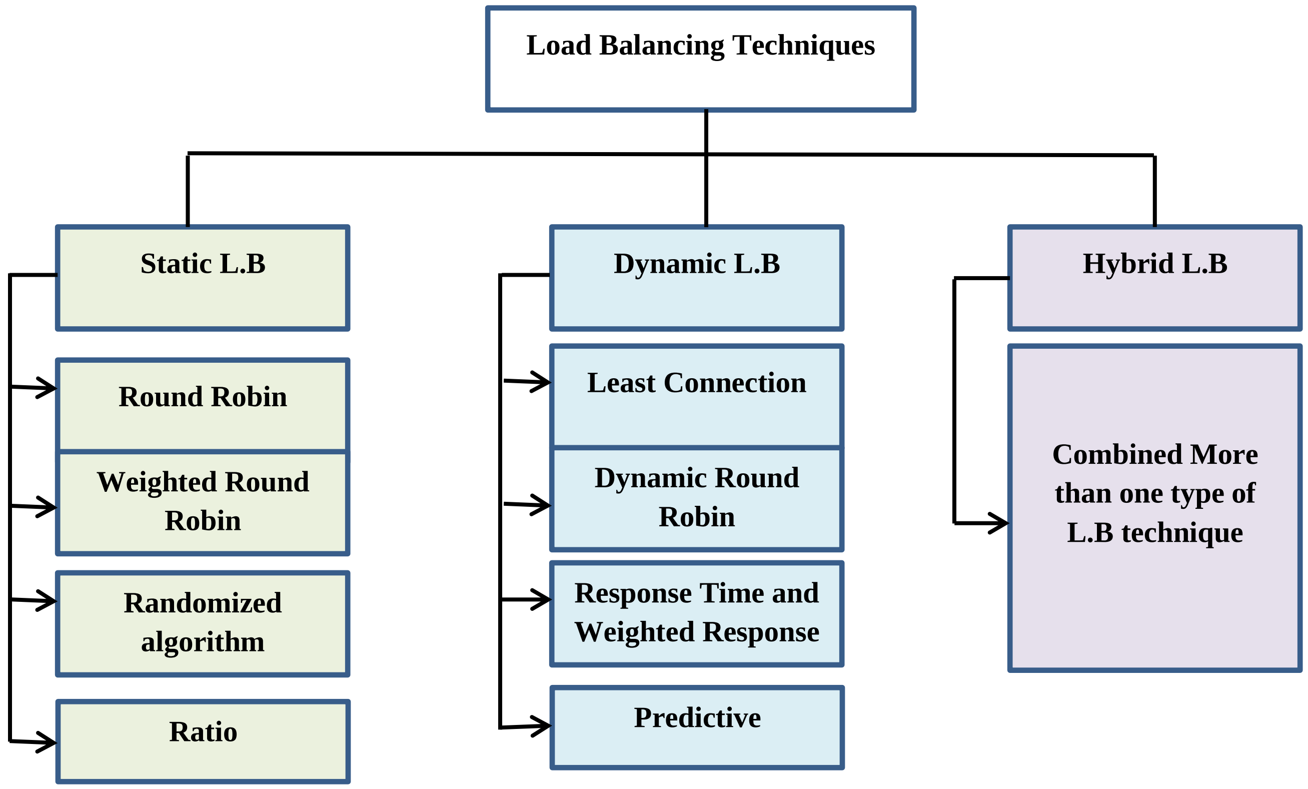


Figure 2.5: The Load Balancing technique types[60].

2.3.1 Static Load Balancing

The method of load balancing in this method is based on data collected in the past about the system. Then, the burden is split up without taking the node's health into account, based on its performance. The load cannot be moved from the node to which it was first assigned. Working on behalf of their customers, the nodes complete their tasks and report back with the findings. static load balancing algorithm characteristics[60]:

- The execution time is decreased since less communication is required, which in turn minimizes the communication delays.
- The allocation distribution is generated independently of the present system status.
- In terms of both response and processing speed, weighted algorithms perform much better. Before beginning execution on a node, load balancing algorithms choose a node based on the expected load.
- Algorithms that do not adapt to their surroundings are called "static," and they tend to perform best in contexts that are both consistent and well-defined. It's simple to put into action. Static algorithms' outcomes are often well-known in advance.
- The arrival timings of loads and the processing timeframes needed for future loads are almost impossible to forecast.

There are several static algorithms; this survey includes five of them[61]:

A- Round Robin Algorithm

It is a simple method for distributing clients' requests across set of listed servers, where an equal load is assigned to each node in a circular order without any priority. Whereas round-robin cycles over the queues or tasks and gives one service opportunity per cycle, weighted round robin offers to each a fixed number of opportunities, as specified by the configured weight which serves to influence the portion of capacity received by each queue or task [62].

B- Weighted Round Robin Algorithm

Each server's relative importance is calculated by its ability to process requests. It is a specified improvement on the round robin method that addresses the major difficulties that it presents[63].

The method may use the results of these frequent load calculations to determine which VM would be best suited to handle the request. The load is used to dynamically determine how much each virtual machine (VM) should weigh. Along with these other considerations, the addition of priority shortens the queue for requests that are more urgent.

Priority is taken into account while sorting incoming tasks, and the work queue is arranged with the tasks from major priority to low importance in a manner that distributes high priority duties to VMs with weights determined by their processing capability and load. As a result, incoming high-priority activities are handled before low-priority tasks, and the time it takes for high-priority jobs to be finished is cut down. However, this strategy can't be used in real time since requests can't be sorted in advance. As shown in Algorithm(2.1), the priority parameter is useless in this kind of task scheduling if the immediately preceded are not sorted [64].

Algorithm 2.1 : The weighted round robin algorithm

```
Input: required time for each request as speed

Output: elapsed time (speed for requests)

Require: Weight
  i = 1 # process number
  cw = 0 # counter of weight
while TRUE do
```

```
  i = (i + 1) mod n # n = total number of process
if i == 0 then
  cw = cw − gcd(W )
 if cw < 0 then
   cw = max(W )
 if cw == 0 then
 return null
  end if
 end if
end if
 if Wi ≥ cw then
 return i
 end if
end while
```

End of algorithm

For explanation Example (2.1) for Weighted Round-Robin Load Balancing:

The high weight is the best optimal server for incoming request depending on the current weight value for server and the proposed system based on the speed per seconds

Server A weight = **5 Bps** speed (Byte per second) on average

Server B weight = **2 Bps** speed (Byte per second) on average

Server C weight = **1 Bps** speed (Byte per second) on average

That are waiting to serve incoming requests behind the load balancer. The load balancer will forward:

Incoming 8 Bps as 8 requests each request size is 1 Byte

# 5 Bps requests to **Server A**

# 2 Bps requests to **Server B**

# 1 Bps request to **Server C.**

If any of the other incoming requests arrive, the load balancer will forward those requests back to **Server A** again for the next five incoming requests with (5 Bps), then **Server B** will get its turn with (2 Bps), and after that the requests will be forwarded to **Server C**. The cycle will continue in this way, as shown in Figure (2.6).

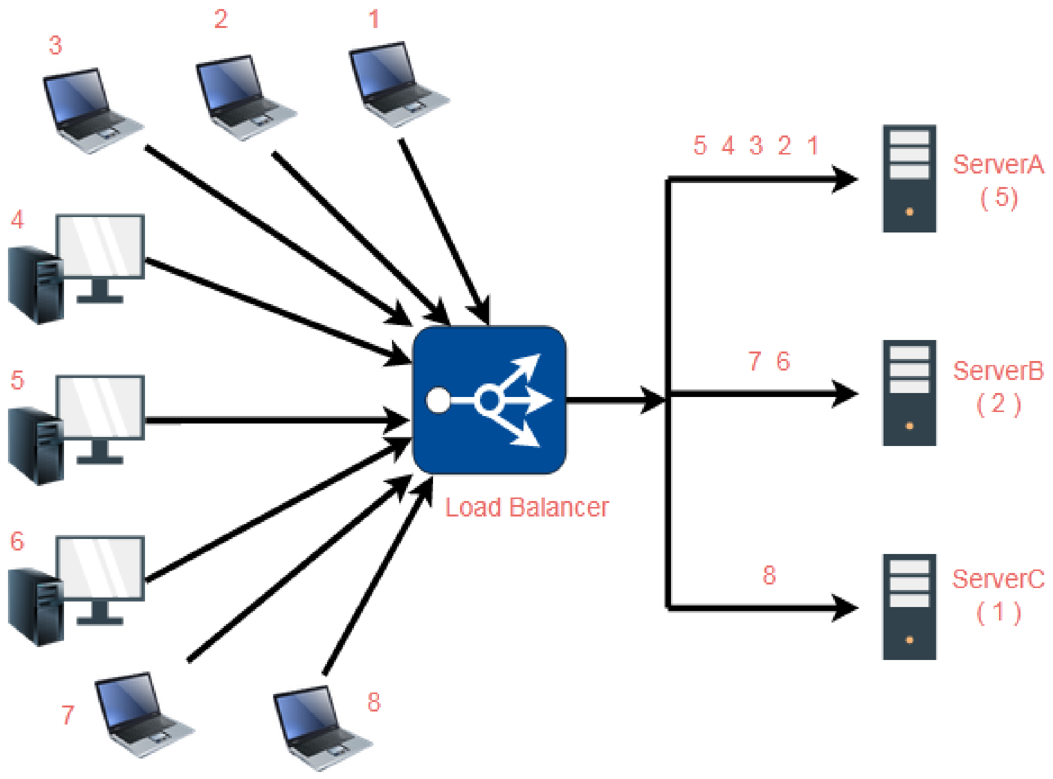


Figure 2.6 :Weighted round-robin load balancing technique[64].

C- The Randomized algorithm

In this method, each node N has a chance (probability) of handling a given process.

D- Ratio

A specified fraction (ratio) is used to decide how the algorithm will function during system startup (initial state). With this method, the workload is distributed proportionally throughout the load balancing network's servers[65].

2.3.2 Dynamic Load Balancing

These algorithms keep tabs on how much work is being done by the system and reorganize things if the load shifts. In most implementations, this technique is broken down into three sub-steps: transfer, location, and information. It is up to the transfer strategy to determine which jobs may be split up between different nodes. To carry out a delegated responsibility, the location strategy will choose a distant node. In the Task Allocation algorithm [66], the information strategy is the nerve center for all the data that flows through it.

A- The Least Connection and Weighted Least Connection algorithm

The load balancer delivers the request that is receiving to the fewest number of concurrent connection. The existing server load is included into the least connections approach. This means that the load balancer must monitor the overall number of open connections at any one time. The requests are distributed across available servers based on their numerical values in the Weighted Least Connection algorithm method. As seen in Algorithm (2.2), if servers have same list of available connections, the load balancer will direct the message to the server with the greater weight [67].

Algorithm 2.2: the pseudocode for least connection

```
Input : Active connection for each server
Output: Number of active connection
Begin
Initial servers S = {S0, S1, ..., Sn-1}, The weight = W(Si), Connection number = C(Si),
m = message request, n= total number of message requests.
for (m = 0; m < n; m++)
  if (W(Sm) > 0)
    for (i = m+1; i < n; i++)
      if (W(Si) <= 0)
        continue;
      if (C(Si) < C(Sm))
        m = i;
    return Sm;
return NULL;
```

End algorithm

For explanation Example 2.2 for Least connection Load Balancing

Let started six requests to communicate with the servers hiding behind load balancer.

1- Requests are being handled by Server A, (1, 3, 5)
2- Requests are being handled by Server B, (2, 4, 6)
3- Requests are being handled by Server C. (7, 8, 9)

# Following a definite delay, requests 1, 2, and 3 are completed.

- In this current state, Server A is responsible for processing requests (5)
- Requests are being handled by Server B, are (4, 6)
- Server C is also handling the same load, are (7, 8, 9)

Since Server A has the fewest open connections, the load balancer should send any subsequent requests (let's say number 10) there.

- At this time, Server A is managing two requests that are (5, 10)
- Additionally, Server B is handling two requestss, which are (4, 6)
- The amount of requests handled by Server C is equivalent (7, 8, 9) . To illustrate this point, Figures (2.7), Figures (2.8), and Figures (2.9). have been included.

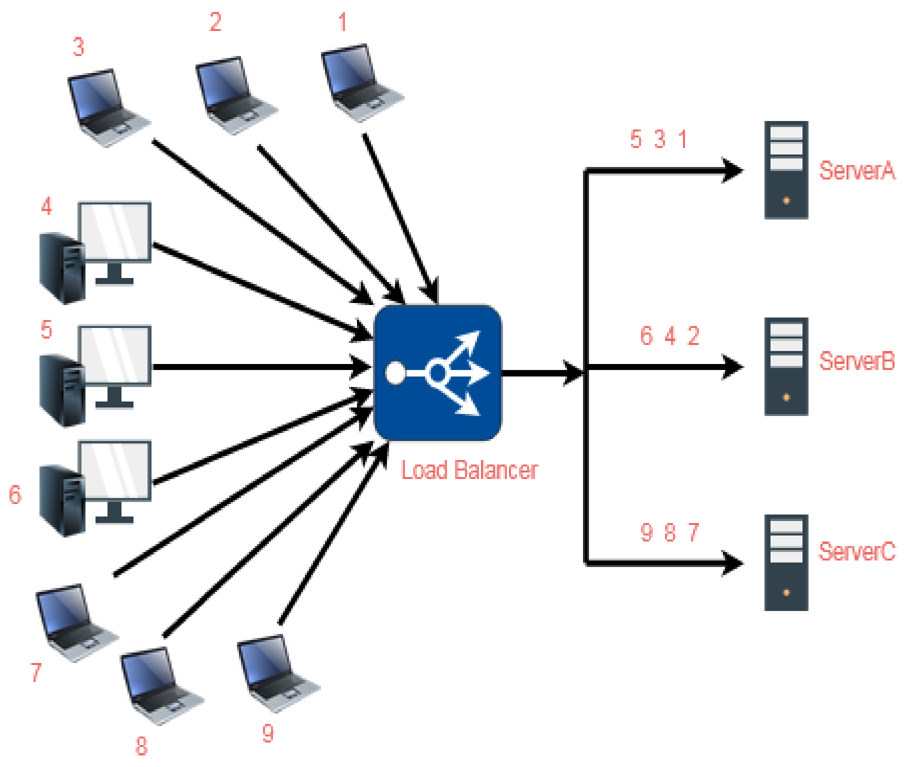


Figure 2.7: Servers managing assigned tasks[67].

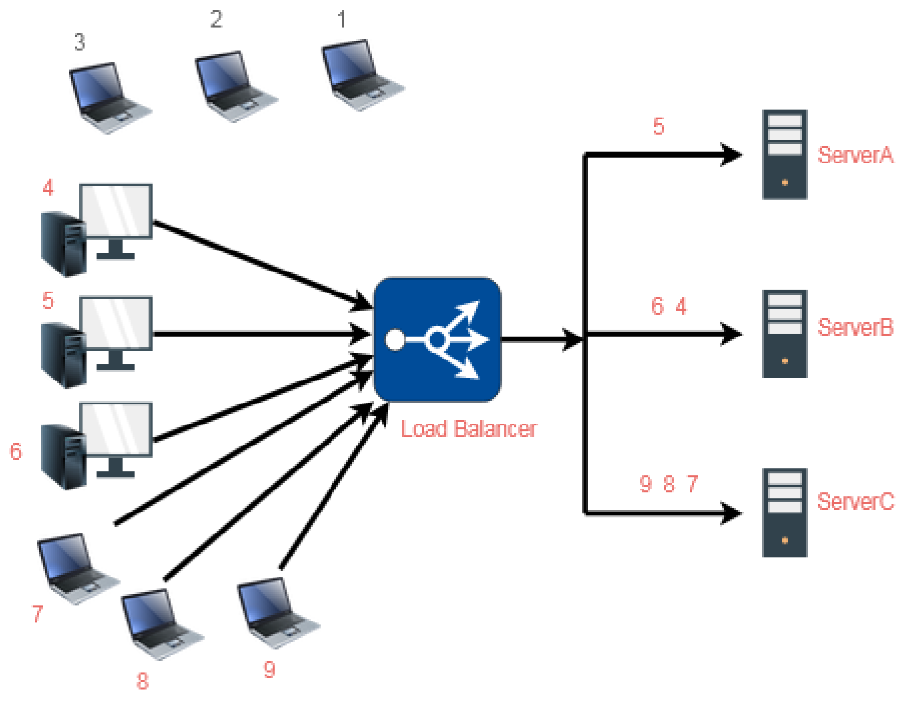


Figure 2.8 : Three of the requests were completed[67].

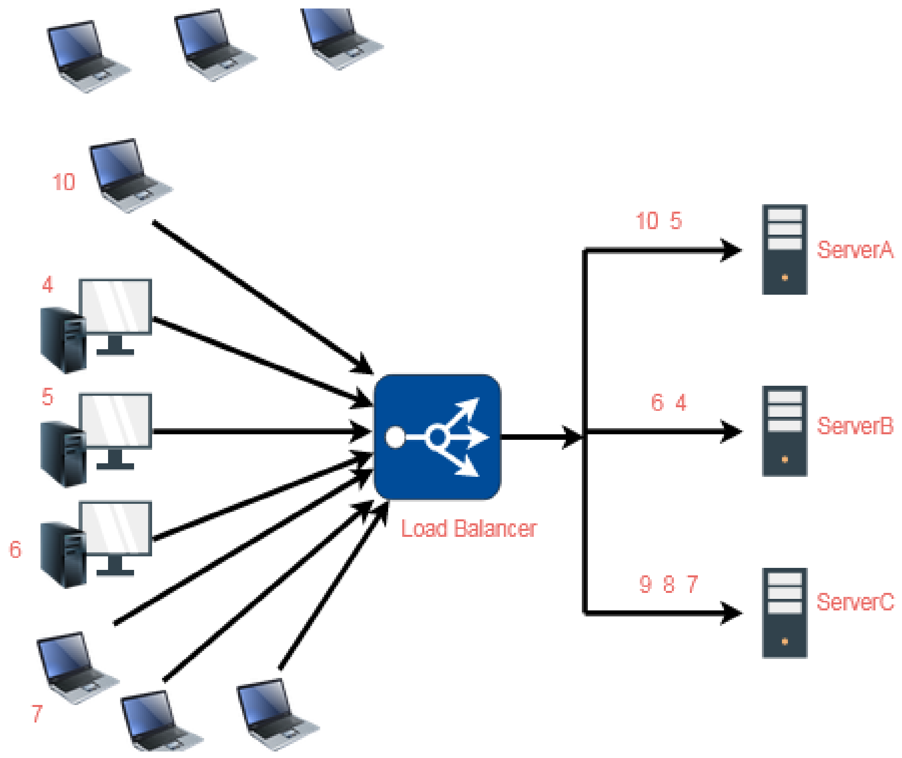


Figure 2.9: Modified the new request in order to use the load-balancing method with the fewest number of connections[67].

B- Predictive algorithm

In order to forecast the best server to route the request to the next time, this algorithm categorizes the performance of many servers in the system over a period and selects the best within them based on the increase in performance[68].

C- The Dynamic Round Robin algorithm

It's quite equivalent to the Weighted Round Robin technique; the main difference is that here, each server is given a weight depending on its actual load at any given moment. It's always different, since the weights are continuously changing [69].

D- The Response Time and Weighted Response Time algorithm

This strategy takes into account the speed with which all servers reply and provide service. Requests are sent to the server with the quickest response time by means of the load balancer [70].

2.3.3 Hybrid Load Balancing

These algorithms have been developed and suggested to address the shortcomings of both static and dynamic load balancing strategies by combining the best features of each into a single solution. This means that it is possible to create a new algorithm by combining the best features of two or more preexisting algorithms, whether they are dynamic or static[71].

For explanation about Hybrid Load balancing :

- i= Number of redirect request + total required time

1- Increased weight due to the incoming request

- New Weight = weight + i = Sum weight

2- decreased weight due to the finished request (terminated)

- i= Number of terminated request on the server + total required time.
- New Weight = weight – i = Sum weight

Initial :

- Weight value for all servers = 0.0 through the starting state

Suppose Weight value after a specific number of requests from IoT sensors changed depending on the computation process of request with required time for over all request of each session as :

- Server 1 Weight = *0.2*
- Server 2 Weight = *0.4*
- Server 3 Weight = *3.7*

A- Case 1 incoming requests

- New Requests R=2 process, Time = 0.4 seconds
- i= Number of redirect request + total required time
- Sum Weight(SW) of i= 2+0.4 = 2.4
- New Weight = for instance Server 1 Weight (less weight) + i = Sum weight
- New Weight = *0.2* + 2.4= 2.6 as the current weight value of Server 1 through processing new requests.

B- Case 2 finished requests

Decreased weight due to the finished request (terminated)

1- Suppose 1 request completed on server 1 with time 0.1 seconds so the finished sum weight value is (SW= 1+0.1 =1.1 weight), For more explanation :

- i= Number of terminated request on the server + total required time.
- i= 1+ 0.1= 1.1 completed weight
- New Weight = current weight – i = Sum weight
- New Weight = 2.6 – 1.1 = 1.5 = New updated Server 1 Sum Weight = 1.5

2- Suppose 2 request completed on server 3 with time 0.6 seconds so the finished sum weight value is (SW= 2+0.6 =2.6 weight)

- New Weight = current weight – i = Sum weight
- New Weight = 3.7 – 2.6 = 1.1 = New updated Server 3 Sum Weight = 1.1

Last updated weight value for all servers are :

- Server 1 Weight = *1.5*
- Server 2 Weight = *0.4*
- Server 3 Weight = *1.1* , So, the optimal server for the next incoming requests is Server 2 Weight = 0.4

2.3.4 Compare Between Algorithms

The contrast between the methods for static load balancing and those for dynamic load balancing, and according to characteristics in Table (2.1) [72].

The data presented in this table suggests that dynamic load balancing methods are superior than static load balancing strategies. Because it can make changes to the load balance during the process's runtime, dynamic load balancing algorithms are a superior technique to offer high availability and are suitable for use with process migration. [73] This is because dynamic load balancing algorithms can make adjustments to the load balance.

Table 2.1: Shows The Comparison Between The Load Balance Algorithms.

Parameter	Static	Dynamic
Process Migration	Limited	Better
Overhead	High	Better
Scalability	Limited	Better
Availability	Limited	Better

When compared to the dynamic load balance, the static load balance has a much higher level of overhead. Even while dynamic load balancing requires monitoring the state of each host while the executions are taking place, the costs associated with this monitoring are deemed to be quite low in comparison to the expenditures required by static load balance [74].

Static algorithms are superior than dynamic algorithms of scalability since the performance of dynamic algorithms decreases as the number of processing unit or the quantity of tasks is increased. This is because static methods do not depend on the state of the system.

2.4 Resource Allocation Algorithm

In the proposed system, the Particle Swarm Optimization (PSO) is used to balance the load across the entire system is introduced while trying to minimize the makespan and increase the processing capacity. Besides, the resource allocation process depending on the hybrid approach is based on the weight value for each server and fog load balancer node decide where request served, the main steps for resource allocation method as showed in Algorithm (2.3) [75].

Algorithm 2.3: Resource allocation algorithm

```
Input: User request as R, Resources as S.
Output: Allocate resource Ss based on priority
Begin
  if (Ri == valid) then
      if (Ri ε R) then
          call Manage Priority
          User i gets Ri
          status (Ri) = allocated
      else
          R is rejected
      else
          User R is invalid
      End if
   End if
End
```

End algorithm

The task scheduling for resource allocation applied in the Basebroker module to reserve resources based on application time scheduling submission history in the Fog environment. The main steps of PSO algorithm explained as follow:

1- Evaluating current process
2- Checking to see if the current position is an individual best
3- Updating new particle process

4- Evaluating current weight as constant inertia weight (how much to weigh the previous velocity)
5- Update the particle position based off new velocity updates
6- Adjust maximum fog position if necessary
7- Determining the best fog position for request
8- Establishing the swarm
9- Beginning the optimization loop
10- Cycle through particles in swarm and evaluate fitness
11- Determining if current fog particle is the best.
12- Cycle through swarm and update velocities and position.

Algorithm (2.4) showed the used PSO algorithm to scheduling the incoming request tasks.

Algorithm (2.4): Task-scheduling based on Particle Swarm Optimization (PSO) algorithm

Input : Incoming processes from IoT nodes through access point to pass to the fog nodes
Output: Process scheduling for ach fog node, and fog node balance the load for each server.

```
1. Define a set of n parallel tasks of an application $A = \{T_1, T_2, \dots , T_n\}$, to be distributed over a set of resources in the fog network $Fnet = \{R_1, R_2, \dots , R_n\}$.
2. Initialize the importance weights of the criteria $W^{make}$ , $W^{CPU}$, $W^{link}$, of the task scheduling.
3. Repeat:
- Define $W^k$ with $W^k = W_{max} - k\,\frac{W_{max}-W_{min}}{K\ max}$ For each particle $i = \{1, \dots , q\}$ do:
- Determine the speed of request
$Velocity\ of\ practicle\ i\ at\ time\ k+1 = v_{k+1}^{i} = w\ v_k^i + c_1\ rand\,\frac{(p^i - x_k^i)}{\Delta t} + c_2\ rand\ \frac{(p_k^g - x_k^i)}{\Delta t}$
-   Determine the Fog position to optimize requests.
Go to next iteration k=k+1
```

```
Until k=kmax or a stop condition is reached
4. gbest is chosen as the optimal fog solution for the task scheduling
End of algorithm
```

2.5 Resource Allocation Evaluation Metrics

The proposed method is based on different evaluation metrics as follows:

A- **Response Time**: It is characterized by the amount of time that passes between the accepting of a response (or task) and the delivery of a response to requests made of a server operating in a fog environment [76].

Response Time =Total round trip required time from source event node to destination node (2.1)

B- **Channel Resource Utilization**: It refers to the condition in which all of the fog system's readily accessible resources are used up to their full potential [77].

C- **Latency**: It is the amount of time between the load balancer receiving a request and returning a response [78].

$$\mathbf{d_{trans}} = \mathbf{L/R} \quad (2.2)$$

Where d represents as delay time in seconds and L is the packet length in bits and R is the rate of transmitted data in bits per time unit[78].

D- **Packet Loss rate**: It represents the ratio of the number of lost packets to the total number of sent packets. Each packet has a deadline before which it must be executed, and if this is not possible, the scheduler tries to minimize the number of lost packets due to deadline expiry [79].

$$\textbf{Loss rate} = \textbf{Total Number of sent } - \textbf{Total number of received packets}(\mathbf{2.3})$$

E- **Throughput**: It is the measurement of a successfully completed job which, in the case of the load balancer, means the number of requests successfully completed per unit time. It’s an insightful metric to measure since higher throughput indicates higher efficiency of load balancers[80].

$$\textbf{Throughput} = \frac{\textbf{Total Number of sent and received packets}}{\textbf{Time}} \quad (2.4)$$

2.6 Simulation Tools

The common simulation tools in this thesis are :

2.6.1 iFogSim Simulation

A necessary framework of the OMNET++ Fog modelling tool. It is built atop the essential structure of CloudSim, which served as its basis. The CloudSim simulator has become one of the most popular tools for modeling cloud computing infrastructures[81]. Extending the abstraction of the fundamental CloudSim classes, iFogSim gives scopes to simulate a personalized fog computing platform with a significant number of fog nodes and Internet of Things (IoT) devices (such as sensors and actuators)[82].

However, in iFogSim, the classes are analyzed in such a way that even users who are not familiar with CloudSim can quickly define the facilities, service placement, and resource allocation regulations for cloud computing[83]. This is because iFogSim is designed to work with users who have no previous knowledge of CloudSim. Sense-Process-Actuate and a distributed dataflow model are both used by iFogSim in the process of modeling any given application scenario inside a fog computing environment[84].

It makes it easier to evaluate end-to-end latency, network problems, power use, operating expenditures, and the level of satisfaction with quality of service. iFogSim is already used in a significant number of research works for the purpose of simulating resource management, mobility management, latency management, Quality of Experience (QoE), energy management, security management, and Quality of Service (QoS) management in fog computing environments[85].

2.6.2 OMNET++

As the primary open-source simulator tool with a GUI, it makes the simulator's internal events easy to understand. In addition to providing a modular framework, Omnet++ also has built-in debugging and tracing facilities. By piggybacking the Fognetsim or iFogSim framework onto an existing one, it is possible to simulate Fog computing network devices with a variety of configurations and apply load balancing models within the context of fog computing. Furthermore, it was used in the creation of the Fog nodes architectural [86].

In this thesis the used tool is iFogSim(FogNetSim++) framework in OMNET++ simulation tool and the main comparison among common fog tools showed in Table (2.2)[86].

Table 2.2: A comparison of the current simulation tools in fog environment

Ref.	**Main ideas**	**Simulation**	**Advantages**	**Limitations**
[87]	Resource management approach based on the dynamic scheduling of multi-user devices using classification methods for cloud servers and heterogeneous devices to minimize latency for IoT applications.	iFogSim	The proposed approach guarantees a decrease in the network usage and hence substantiates its efficiency in streamlining the patient information flow and its accessibility for health care providers.	Not consider the monetary cost of cloud resources and power consumption. Not apply in large scale applications.
[88]	Heuristic-based algorithm to achieve the balance between the make span and	CloudSim	The proposed algorithm achieves better cost make-	High workload execution time. Low

	the monetary cost of cloud resources. Determine the task priority and choose a suitable node to execute each task.		span trade-off value than Greedy, Heterogeneous Earliest Finish Time (HEFT), and Distributed Load Sharing (DLS) algorithms.	scalability. Not consider the important parameters like deadline constraints of workflow execution and Fog provider's budget.
[89]	Efficient Resource Allocation (ERA) for resource provisioning in fog computing environments by using the virtualization technique.	Cloud Analyst (part of Cloud Sim)	Proposed strategy can be allocated resources in an optimized way and better than existing algorithms [Reconfigure Dynamically with Load Balancing (RDLB) and Optimize Response Time (ORT)] in terms of overall response time, data transfer cost and bandwidth utilization in the fog computing environment.	Not satisfying the requirement of resources during the execution of the request. They have only allocated those resources to the clients who are requesting before processing. Applicable for only reserved instances. Low availability. High complexity

Chapter Three

The proposed Hybrid Load Balancing in FOG Computing

3.1 Overview

This chapter explains the main steps of the proposed system implementation, configuration and installation. The proposed algorithms are explained as Least Connection(LC), Weighted Round Robin(WRR), and Hybrid algorithm, in addition task scheduling algorithm is implemented to optimize incoming requests(tasks) to an existing fog nodes from the list of connected active fog node.

3.2 The proposed implementation Requirements

The proposed system installation is based on OMNET++ and FOGNETsim++, and C++ programming language. The proposed algorithms installation are Least Connection, Weighted Round Robin, and Hybrid algorithm, the proposed system based implemented with two main models are centralized and distributed.

A- Centralized Fog computing

The architecture of the centralized fog node system involving three layers. The first layer is IoT sensors as the clients to transmit data. the second layer is Fog load balancer, which is control on the network component, and resource allocation through network transmission. The third layer is the servers as network services elements to provide the response for each request by IoT sensors. The proposed method, which is based on fog load balancing, has as its primary objective the distribution of load across a number of servers in order to optimize the consumption of the processing capacity of each server as well as reducing the typical response time of tasks, which is the same as increasing the amount of work that can be done by the system. The Fog node that receives and distributes all task requests to every server in the pool according to some conditions based on the nature of scheduling algorithms used as weighted round robin algorithm, least connection algorithm, and hybrid approach.This node is responsible for receiving and distributing all task requests. In fog environment, the architecture has three layers: the first layer is IoT sensor, the second layer is Fog, and the last layer is the servers resources, as it showed in Figure (3.1) .

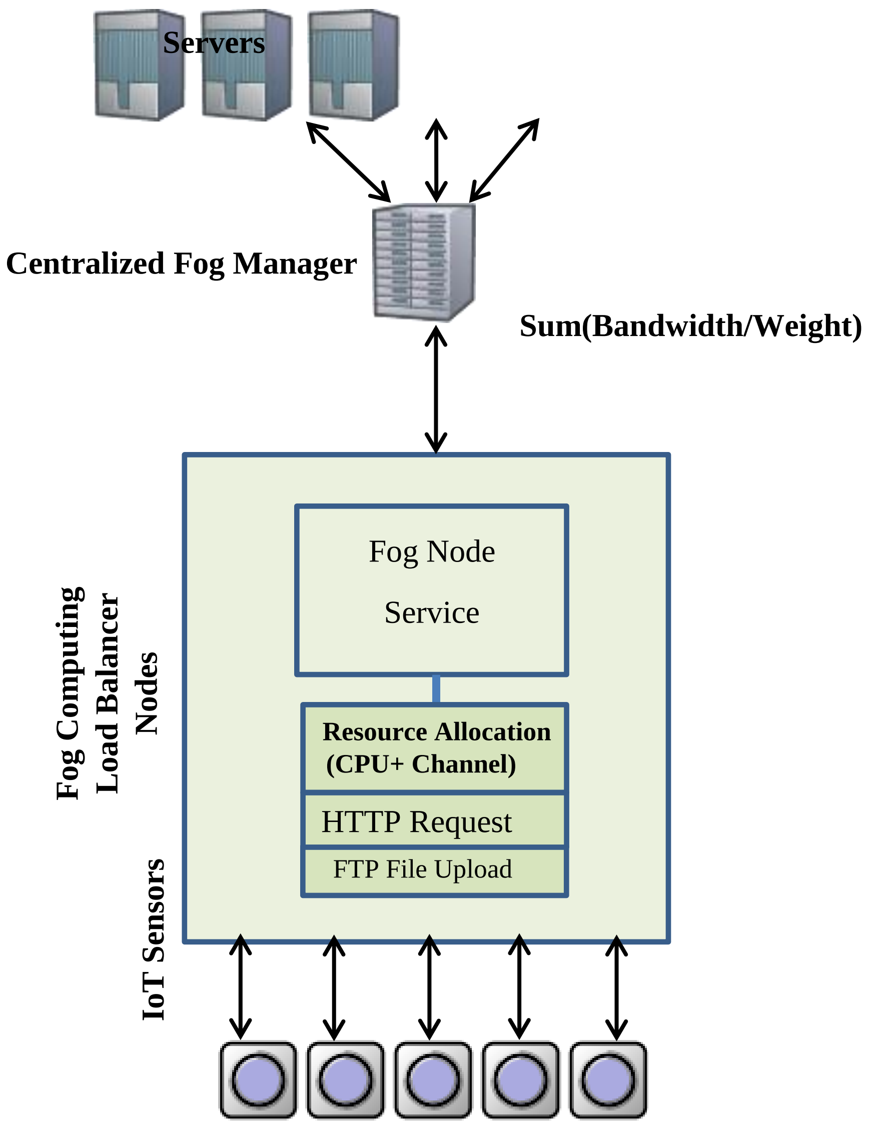


Figure 3.1: The proposed resource allocation system architecture in centralized Fog Computing environment.

The IoT sensors could choose Fog computing nodes for responding to the resource requirements, and load balancing is one of the key factors to achieve resource efficiency and avoid bottlenecks, overload, and low load. However, it is still a challenge to realize the load balance for the computing nodes in the fog environment during the execution of IoT applications. Due to the diversity of execution duration and specifications for the computing nodes in the fog, the resources could not be fully utilized. The proposed system steps showed in Figure (3.2)..

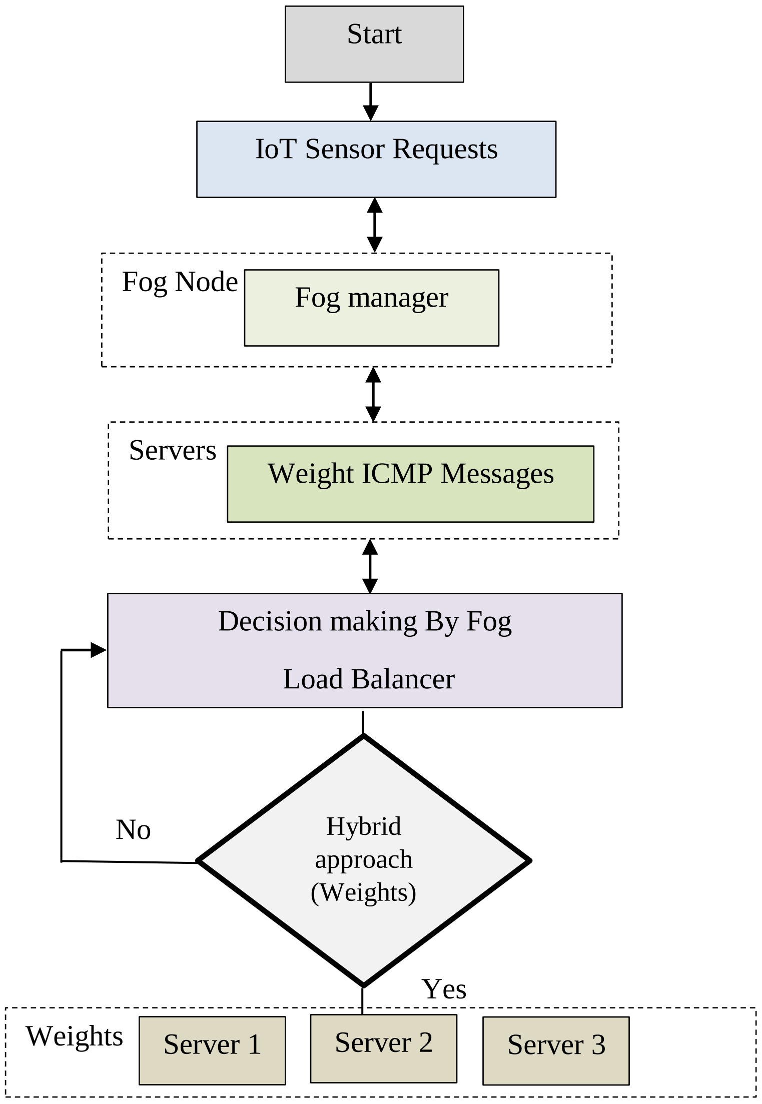


Figure 3.2 : The proposed centralized system steps.

B- Distributed Fog Node

The structure of Distributed Fog Nodes consists of three-layers. At the bottom, wireless Internet of Things sensors serve as the Hosts that send and receive information, access point, router with basebrocker as a virtualized layer. The second layer is distributed a many Fog load balancers (three Fog nodes), which are control on the network component, and resource allocation through network transmission. The third layer is the servers as network services elements to provide the response for each request by IoT sensors.

The proposed distributed process is based on Fog load balancing, and its primary goal is to evenly distribute work across multiple servers in order to maximize system throughput by maximizing the utilization of each server's computational capacity and minimizing the average response time of individual tasks. The Weighted Round Robin method, the Least Connection technique, and the Hybrid approach are only three examples of scheduling algorithms that the Fog nodes employ to accept and distribute all task requests to each server in the pool.

The basic structure of the PSO algorithm in the proposed system depending on the evaluation main function, save individual and global optimize number of request value, update rate and fog position in the active fog pool and finally verify optimization criteria for optimal fog selection to process incoming request and redirect it to the better server selection based on hybrid load balancing approach.

Figure (3.3) showed the proposed distributed fog computing with PSO task scheduling algorithm and hybrid load balancing approach.

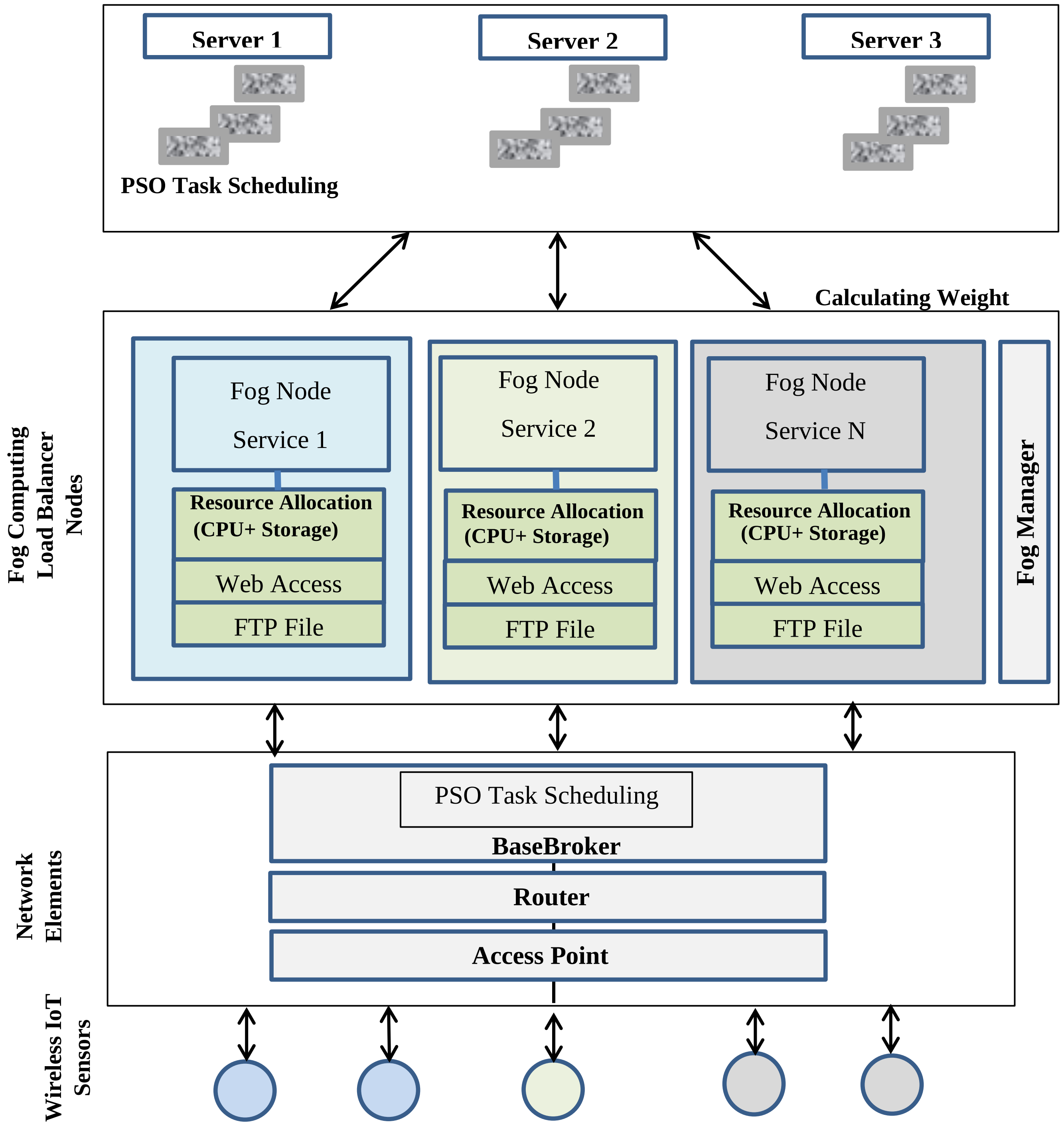


Figure 3.3: The proposed resource allocation system architecture in distributed Fog computing environment.

Fog load balancing is an important aspect in achieving resource efficiency and avoiding bottlenecks, overloading, and low load, and the basebroker device may choose Fog computing nodes to react to wireless

Sensing devices for resource needs. The proposed system steps showed in Figure (3.4).

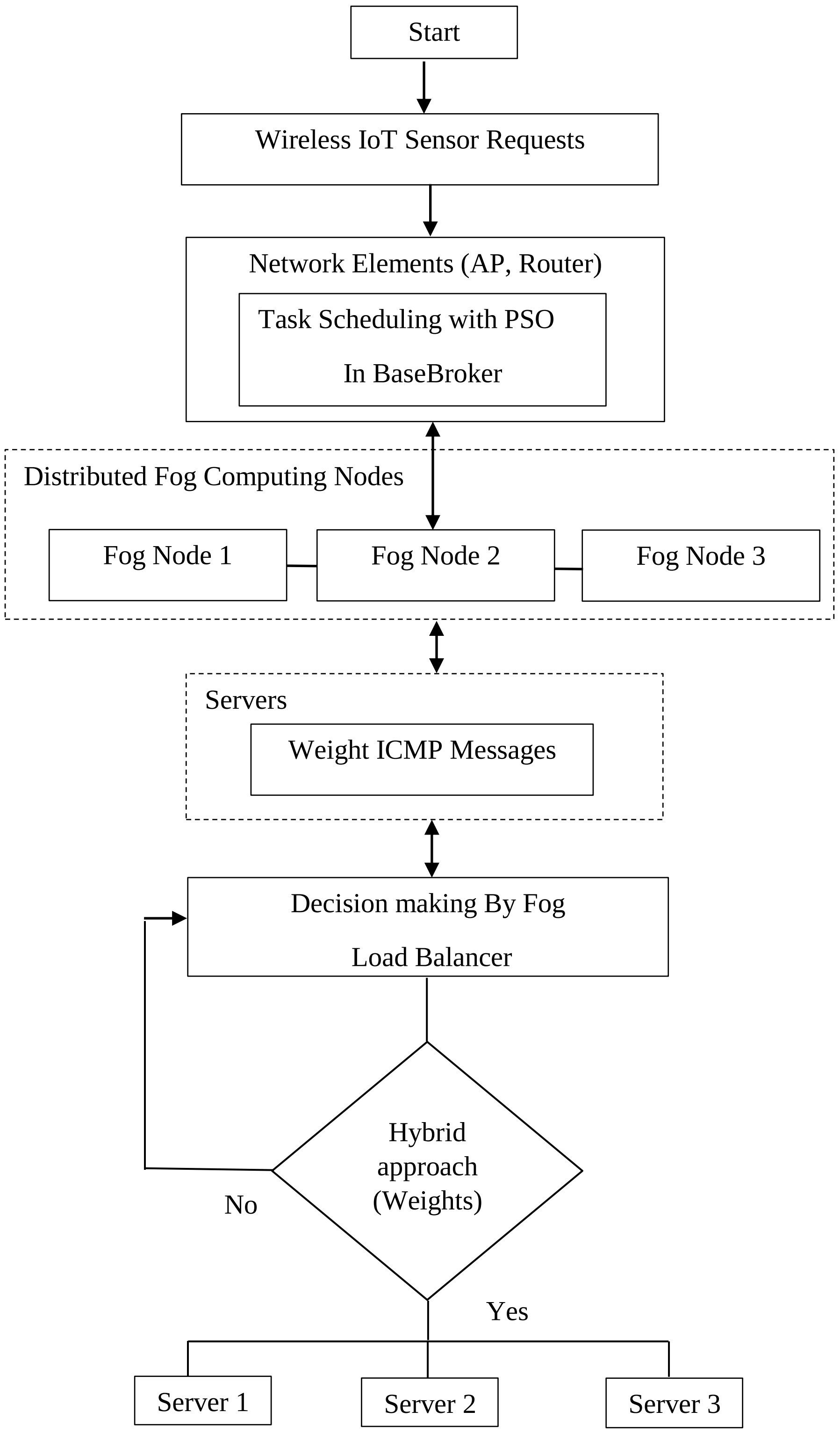


Figure 3.4 : The proposed distributed fog computing system steps.

ICMP used to calculate weight values from each servers and it sent by fog node as echo messages. In addition, because node connectivity in IoT sensors may be very unstable owing to mobility and disconnections, synchronization algorithms designed for distributed Fog networks cannot be applied directly to these sensors. As an extra, it offers a load balancing strategy for distributed fog computing that uses task scheduling as Particle Swarm Optimization (PSO) to schedule a task to an existing resource on the Fog nodes environment, as distribution can happened in fog node and among servers, each Fog node can distribute task for multiple servers, so it provides enhancement in response time and equivalent overloading high traffic data rate system. It dealt with huge amount of request and big data traffic in distributed environment and decrease waiting processes in queue with adaptively data management especially with fault tolerant state for centralize fog node.

Using criteria and goals such as the number of incoming requests and the Basebroker task scheduling needed to complete the work, the PSO event scheduling algorithm allocates tasks to the available computer resources in a distributed system. By analyzing the needs and actions of fog nodes and servers, Basebroker ranks potential allocations of such resources. Through the use of the Fog infrastructure, suitable resources are chosen. The Basebroker also makes reservations dependent on the actions of IoT devices. All of the Fog infrastructure's resource management needs, including scheduling, monitoring, and reservation, are met by this product. It also keeps an eye on the Fog environment's resource availability. Once resources have been chosen, the Fog platform will schedule applications to run on those resources and keep an eye on them while they run. The basebroker infrastructure reschedules the application to the Fog resources based on changes in load needs from task scheduling if the IoT sensor traffic with loading status initiates any dynamic changes. As an added bonus, it schedules applications and reserves resources in the Fog cloud computing infrastructure according to past submission patterns.

Besides, the proposed system of centralized and distributed based on the least connection. Weighted round robin and hybrid load balancing to manage resource allocation

3.3 The proposed Development Approach

The hybrid method is presented to load balancing, which is dependent on the weighted round robin and least connection algorithms, serve as the development system. Both the efficiency in which resources are deployed and the system performance as a whole are strongly impacted by the load-balancing technique that decides which target server processes a request. The suggested load-balancing module would first gather the most recent server loads in order to achieve a more accurate load distribution among several servers, the capabilities of which may or may not be identical. After that, it makes use of this information to choose the server that is most suited to handle the processing of a new request. The processing overhead is required in order to collect the most recent server loads. As a result of the fact that servers are required to regularly collect their load current status and transmit it to the Fog scheduler, and as a result of the fact that the Fog is required to keep received load status for all servers, this has an influence on the packet processing traffic.

3.3.1 The used Weighted Round Robin (WRR) algorithm

It is designed in the proposed system to better handle servers with different processing capacities. Each server can be assigned a weight, an integer value that indicates the processing capacity. Servers with higher weights receive new connections first than those with less weights, and servers with higher weights get more connections than those with less weights and servers with equal weights get equal connections.

In addition to this, the pseudocode for the Weighted Round Robin that was implemented in Algorithm (3.2). W = (W0, . . . , WM −1) is the weight matrix in the method, where I is just the servers that was chosen during the previous installment, Wi is the weighting for servers I + 1 with I = 0,..., M 1, and cw is the present weight. The acronym gcd stands for the "greatest common divisor".

The main steps of the used Weighted Round Robin algorithm presented as follows :

1. Initializing the main class Function to find the waiting time for all processes
- Make a copy of burst times to store remaining burst times.
2. Keep traversing processes in WRR manner until all of them are not done.
- Traverse all processes one by one repeatedly.
- If burst time of a process is greater than 0 then only need to process further.
- There is a pending process.
- Increase the value of time of process has been processed.

- Decrease the burst time of current process quantum.

3. Checking the burst time

- If burst time is smaller than or equal to quantum. Last cycle for this process.
- Increase the value of time process has been processed.
- Waiting time is current time minus time used by this process.
- As the process gets fully executed make its remaining burst time = 0.
- If all processes are done

4. Building the function to calculate turn around time by:

- Calculating turnaround time by adding burst time and waiting time.

5. Building the function to calculate average time.
6. Building the function to find waiting time of all processes.
7. Building the function to find turn around time for all processes.

- Display processes along with all details.
- Calculating total waiting time and total turn around time

3.3.2 The Least connection (LC) algorithm

In the system that is being proposed, it will route connectivity to the server that already has the fewest number of connections established. In order to have an accurate picture of the load being put on each server, it must dynamically count the amount of connection being made to each one. The connection count of the each server is recorded by the Fog load balancer.

The connection value of a service is increased whenever a novel connection is redirected to that server, and the number of connections that may be maintained by a server is reduced in case that one of those connections fails or times out. It is essential to take into consideration that a server having a weight of zero implies that it is not accessible, and that server should be taken off the list of servers that are available. The pseudocode for the least connection method can be found in Algorithm (3.3), the main parameters are servers sets are S = {S0, S1, ..., Sn-1}, W(Si) is the weight of server Si, C(Si) is the current connection number of server Si.

The process of the proposed Least Connection algorithm presented as follows :

1. Initializing the main class function to visit the nodes of a graph
- If current node is already visited return the node.
- If current node is not visited return the IP node as idle node.
2. Utility function to check if it is possible to connect all computers or not
- Stores whether a node is visited or not
3. Build the adjacency servers list
4. Storing the fog count of network connections
- Stores count of components
- Count redundant edges
- Check if components can be rearranged using redundant edges
5. Function to check if it is possible to connect all the devices or not
- Stores count of minimum operations required
- return the minimum number of operations required
6. Body of algorithm processes
- Given number of computers
- Given set of connections.

3.3.3 The proposed hybrid load balancer algorithm

The hybrid approach of the used load balancing based on both algorithms implemented at the same time and the proposed value weight generated from both algorithms based on their characteristics and then each Fog node in distributed environment evaluated the servers periodically to decide which optimal server to process the incoming request. The steps of the hybrid approach for explanation in example :

- IoT sensors make request to access to specific website as HTTP request, or file upload request.
- Request passed to fog nodes to redirect it to specific server.
- Fog nodes periodically send ICMP echo request packets to servers and it creates weighed value (depending on the response time and number of connections) for each active servers, in addition, Fog nodes still sending check packet periodically to be updated with the last servers state and evaluate servers to be optimal for processing the next request.

The pseudocode for the proposed hybrid weighting in fog computing environment is shown in Algorithm (3.1).

Algorithm (3.1): the pseudocode for the proposed hybrid least weight

```
Input: Number of active request based on LC, required time based on WRR.
Output: Weight value as the amount of required load.
Begin
Initial main class WScheduler(), and parameters.
 cw = 0 as counter of weight
 i = -1 counter of processes
 max_s = None
 gcd_s = None
 len_s = None
 counter = 0
repostryevents(self, s): Record number of connection in repository
   self.reposnt = s
   self.max_s = max(s, key=lambda x: x[1])
   self.gcd_s = reduce(fractions.gcd, weight for data, weight in s)
   self.len_s = len(s)
schedule(self): Task scheduling with PSO
 while True:
     self.i = (self.i + 1) % self.len_s
   if self.i == 0:
     self.cw = self.cw - self.gcd_s
     if self.cw <= 0:
       self.cw = self.max_s
       if self.cw == 0:
  return None
  if self.data_set[self.i][1] >= self.cw:
     self._inc_counter(self.data_set[self.i)
  return self.data_set[self.i]
inc_counter(self, item): Calculating required time for each process with WRR
    self.counter[item[0]] += 1
     except KeyError:
  self.counter[item[0]] = 1
set_data(self, s):
 self.reset()
 self._init_dataset(s)
reset_counter(self) if there isn't: server found
 self.counter= 0
get_next(self, n = 1):
 if n > 1:
  return [ self.schedule() for i in range(0,n)]
 return self.schedule()
```

End of algorithm

The proposed Hybrid algorithm based on the combining algorithms Least Connection and Weighted Round Robin, the process explained as follows:

1. When a connection is made, this function will be triggered, if a new fog node or endpoint is connected to this fog node, this fog node will send a fog ready message
2. When this fog node receives data, data received will be triggered
3. Saving this connection to connection repository
4. Distributing this task to local processing, neighbour fog node or neighbour server
5. Handling the received results
6. Handling the received state information of neighbour fog nodes
7. Handling the fog ready message, respond a fog ready acknowledge message.
8. Handling the fog ready acknowledge message, calculate delay
9. When a connection is lost, this function will be triggered.
10. Analyzing the incoming task, status of the task queue and neighbour fog state table to determine where to process the task.

11. Sending the task to the best neighbour fog node in case of distributed fog nodes
12. Sending the task to server
13. Processing the task locally by the selective server
14. Using task delay to insert the task to the task queue for processing
15. Distributing the task to local processing, neighbour fog node or server based on the result of task Inspection function.
16. Return the result to the fog node that offloads the task to this fog
17. Storing the received state information of neighbour fog node in state table in weight attribute column.
18. Putting the link between a fog node and a fog neighbor in the repository for storage.
19. When a fog node's neighbour connection is complete, the connection with that fog node will be deleted. Specifying the period of state sharing in distributed fog node
20. Sending the state information to neighbour fog nodes
21. Finding the best neighbour fog node with minimal estimated waiting time, number of active connection.

Chapter Four

Results and Discussion

4.1 Overview

This chapter discusses the results for the proposed system that are described in chapter three. The proposed system implemented in OMNET ++ simulation tool. It implemented with three load balancing algorithms, as the Least Connection (LC), Weighted Round Robin(WRR), Hybrid algorithm (LC, and WRR). The approach that has been presented, which is based mostly on two primary case studies depending on the fog computing node. The 1st case study based on the one centralized fog node to distribute load among the used three servers. The 2nd case study based on three distributed fog node to distribute request among them depending on the task scheduling approach, and then redirect incoming requests to the optimal server.

4.2 The proposed System Implementation

An system with the following requirements, as shown in Table(4.1), has been used to test the proposed system. In addition, the code for the proposed system has been created in C++ using the OMNET++ simulation tool.

Table 4.1: Environment specifications for the proposed system.

Operating Systems	Windows 10 pro, 64-Bit
CPU	Core (TM) I5-4210U
RAM	8.00 GB
Implementation Tools	OMNET++ 4.6, INET 3.3.0, and FogNetSim++

The proposed system implemented with two main case studies as follow :

A- Centralized Fog Computing

The centralized fog computing approach is based on the fog node controlled on the traffic and redirects the request to the servers in load balancing state, in addition to the sixth IoT sensors, three servers, and wired connection among overall network elements. Table (4.2) presents the specifications of each network elements which effect the resource allocation process, with network configuration IP Address, network interface, and port number is 21. As mentioned the prefix /30 due to the point to point connection between IoT sensors and next device fog, and fog to the server.

Table 4.2: The Specification of the Fog environment.

Node IP (Point-to-Point)	**The behavior**	**Network interface**	**Local port**
10.0.0.2/30	IoT sensor 1	eth0-eth0	1000
10.0.0.6/30	IoT sensor 2	eth0-eth1	1005
10.0.0.22/30	IoT sensor 3	eth0-eth5	1006
10.0.0.30/30	IoT sensor 4	eth0-eth7	1007
10.0.0.14/30	IoT sensor 5	eth0-eth3	1008
10.0.0.18/30	IoT sensor 6	eth0-eth4	1009
10.0.0.1/30	Fog Node	eth0-to-eth8	1010
10.0.0.10/30	Server 1	eth0-eth2	1002
10.0.0.26/30	Server 2	eth0-eth6	1003
10.0.0.34/30	Server 3	eth0-eth8	1004

Initially, the network environment is initialized based on the Fog load balancer for resource allocation management that is based on the network servers with the main setting as:

Step 1: Initializing queue with Queue management.

- Buffer management

Step 2 : Configuring local port number, interfaces, message length, packet type, and time management.

Besides, the main steps of the used system are :

1- Building system topology.
2- Fog configuration with the network setting for IoT sensors and network servers.
3- Changing network packets to build route table and system paths.
4- Fog node received IoT sensor packets and calculating weight for each servers dynamically.
5- The Fog and data path is built with the flow FTP file traffic, and port numbers. Incoming packet with destination address checked with the flow table to see where to forward it, or Fog decide how deal with it.

In addition, the recommended phase for the system is as follows:

1- The servers received queries from the IoT sensors to process their request as HTTP web access or File uploading service through Fog node.
2- After receiving requests from sensors, fog nodes adjust flow rules and attempt to distribute the load evenly over all of the accessible servers.

The Internet of Things sensors are only aware of the fog transparent transmission; they are unaware of the backend servers.

3- The fog node will, at regular intervals, send an ICMP echo signal to the servers in order to calculate the overall weight (load) of the servers as well as the best server response that has the least amount of weight.

4- Check the link statuses of each of the three servers, and adjust the weight value accordingly, taking into account the connection traffic characteristics and the amount of congestion in the network.

5- IoT sensors will send a request, and servers will respond with the answer to each request.

Figure (4.1) demonstrates the proposed system's implementation in OMNET++.

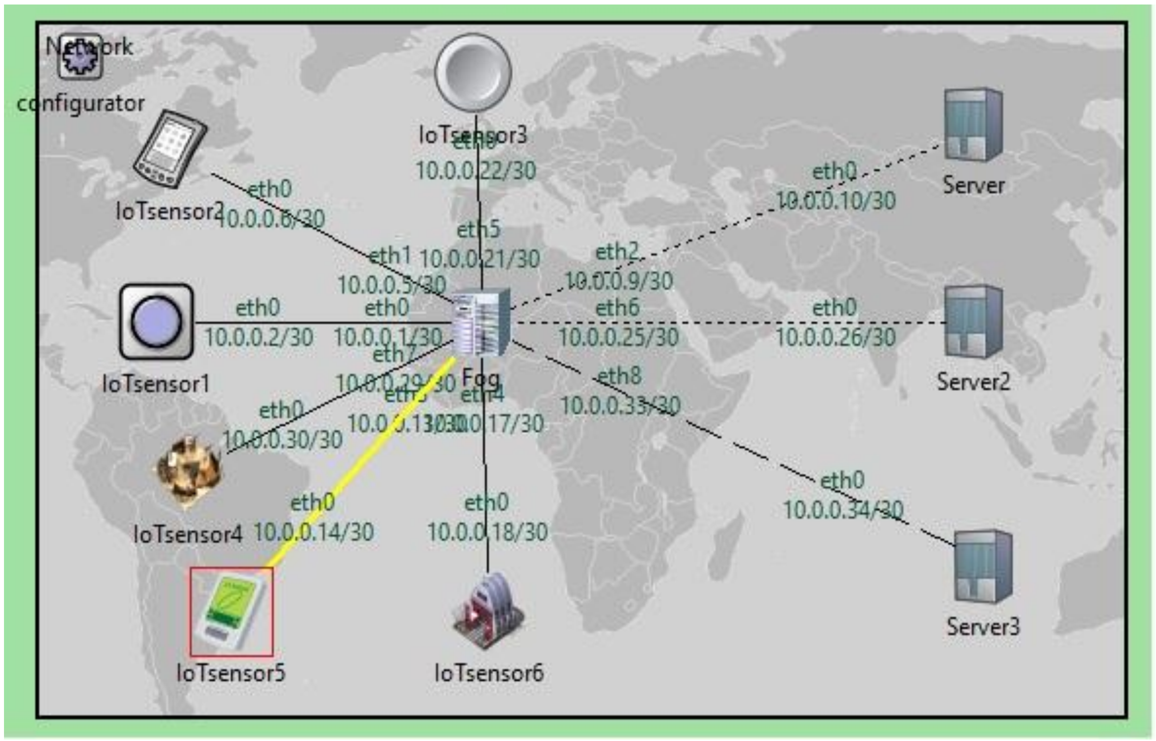


Figure 4.1: The simulated implementation of the proposed system in OMNET++.

In addition, Figure (4.2) shows the proposed Fog node select optimal server to pass request from IoT sensors.

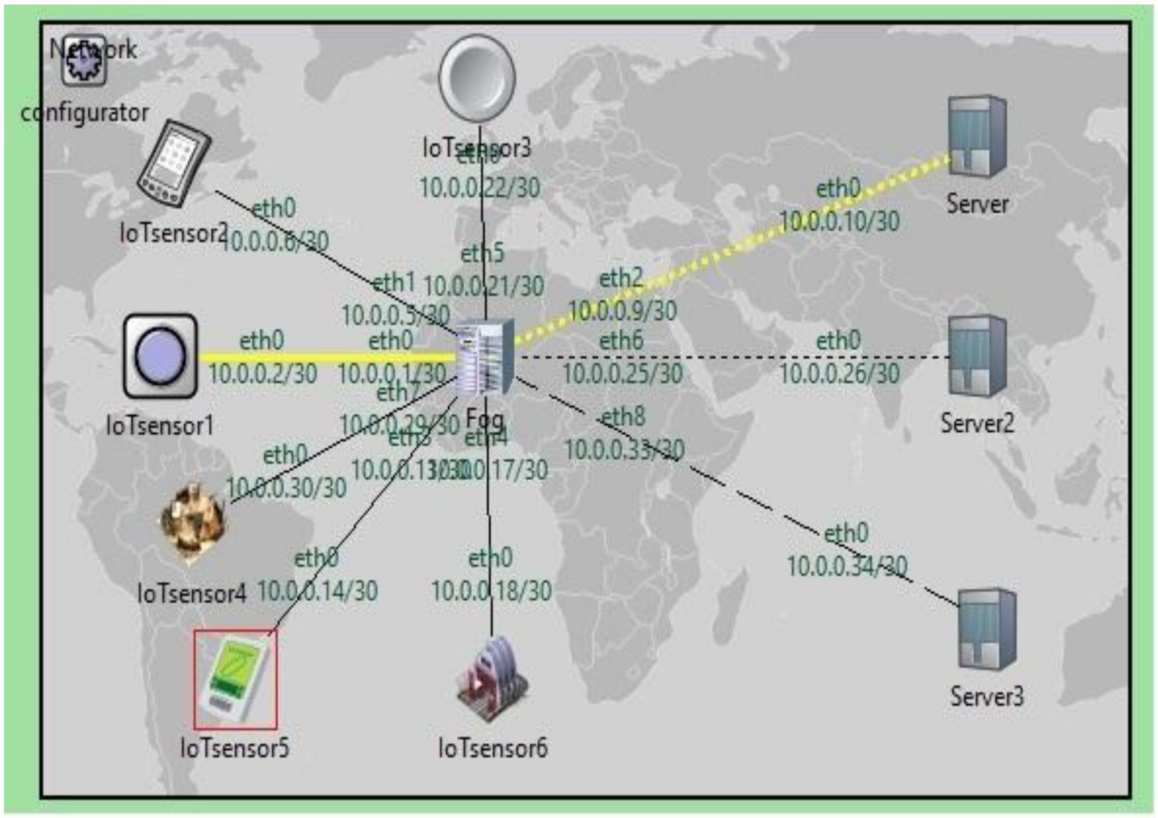


Figure 4.2: The proposed Fog node redirect request to the optimal server.

B- Distributed Fog Computing

The distributed fog computing system is based on three fog nodes to control traffic in the network and redirect the incoming request to the selective server taking into consideration the load balancing approach, in addition, sixth IoT sensors wireless connected with access point, gateway router, base broker, three fog nodes, and three servers.

Table (4.3) presents the specifications of each network elements which affected the resource allocation process, with network configuration IP Address, network interface, and port number.

Table 4.3: The Specification of the distributed Fog environment.

Node IP (Point-to-Point)	**The behavior**	**Network interface**	**Local port**
10.0.0.2/30	Wireless-IoT-sensor 1	Wireless connection with Mobility service	1000
10.0.0.6/30	Wireless-IoT-sensor 2		1005
10.0.0.22/30	Wireless-IoT-sensor 3		1006
10.0.0.30/30	Wireless-IoT-sensor 4		1007
10.0.0.14/30	Wireless-IoT-sensor 5		1008
10.0.0.18/30	Wireless-IoT-sensor 6		1009
10.0.0.46/30	Fog Node 1	Eth1-to-eth3	1010
10.0.0.30/30	Fog Node 2	eth0-to-eth3	1010
10.0.0.62/30	Fog Node 3	eth0-to-eth3	1010
10.0.0.45/30	Server 1	eth0-to-eth2	1002
10.0.0.29/30	Server 2	eth0-to-eth2	1003
10.0.0.53/30	Server 3	eth0-to-eth2	1004
/	Access point	/	/
10.0.0.17/30	Gateway Router	eth0-to-eth1	1014
10.0.0.1/30	BaseBroker(virtual)	Eth0-to-eth3	1015

Initially, the network environment is initialized based on the distributed Fog load balancer for resource allocation management that is based on the network servers with the main setting as:

Step 1: Initializing queue with Queue management for each network elements.

- Queue management (Buffer management)

Step 2 : Configuring local port number, interfaces, message length, packet type, and time management.

Besides, the main steps of the used system are :

6- Building system topology.
7- Distributed Fog configuration with the network setting for IoT sensors and network servers.
8- Exchanging setting messages from wireless IoT sensors with access point and gateway router to build network table and update sensor locations.
9- Router received packets from wireless sensors through access point and pass the packet to virtualize layer as basebroker device which passed the incoming messages to the distributed Fog nodes (three Fog nodes).
10- Fog nodes received Wireless-IoT-sensor packets and calculating weight for each servers dynamically.
11- Incoming packet with destination address is checked with the traffic table to see where to forward it, or Fog decides how deal with it.

In addition, the suggested system phase is as follows:

6- IoT sensors sent queries to servers to process their request as HTTP web access or File uploading service through Fog nodes.
7- Network elements such as access point, gateway router, and basebroker configured to pass data messages to the Fog nodes layer.
8- Fog nodes accept requests from sensors, adjust flow rules, and distribute load over all accessible servers. IoT sensors are unaware of backend servers; only Fog transparent transmission is known to them.

9- Fog nodes deliver frequent ICMP echo messages to servers in order to detect the most recent load (load) of servers as well as the optimal server response with the least weight.

10- Fog nodes are responsible for monitoring the link conditions of all three servers and providing an updated weight value based on the network traffic characteristics and network overloading.

11- Fog nodes depend on the incoming request weight to distribute the request in some cases of high overloaded data to multiple servers to reply with the response to each request by the IoT sensors.

In addition, the proposed distributed Fog system implementation with wireless IoT sensors and network elements in OMNET++ are shown in Figure (4.3).

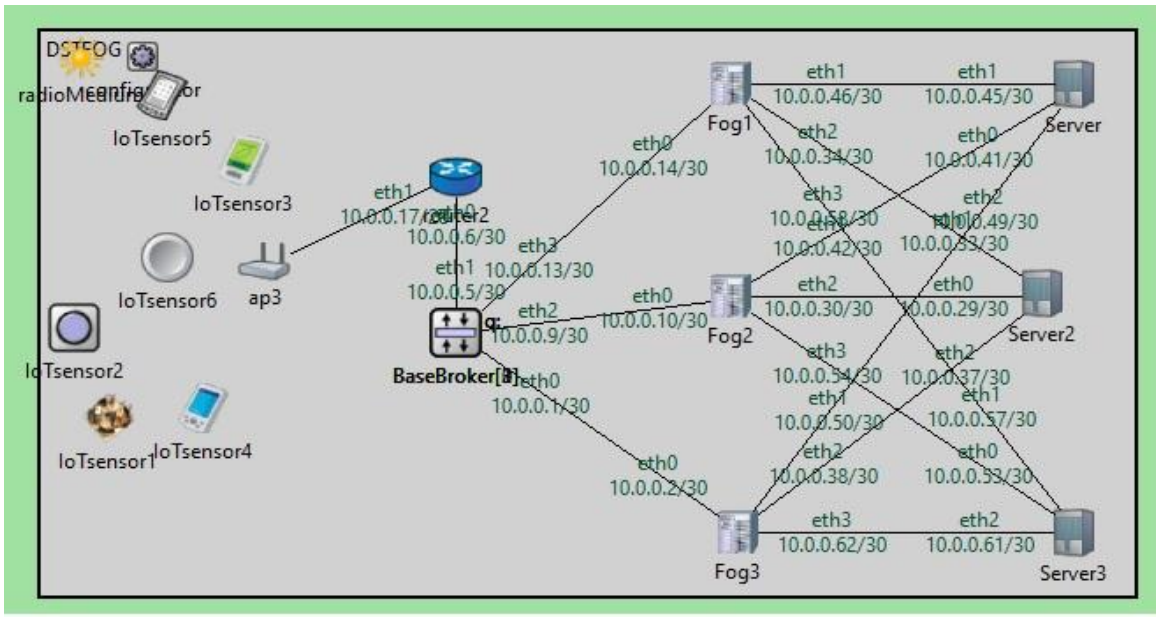


Figure 4.3: The proposed system simulated in OMNET++.

In addition, Figure (4.4) shows the proposed Fog node select optimal server to pass request from IoT sensors, in case of requested passed from different Fog nodes at the same time, to redirect to different servers.

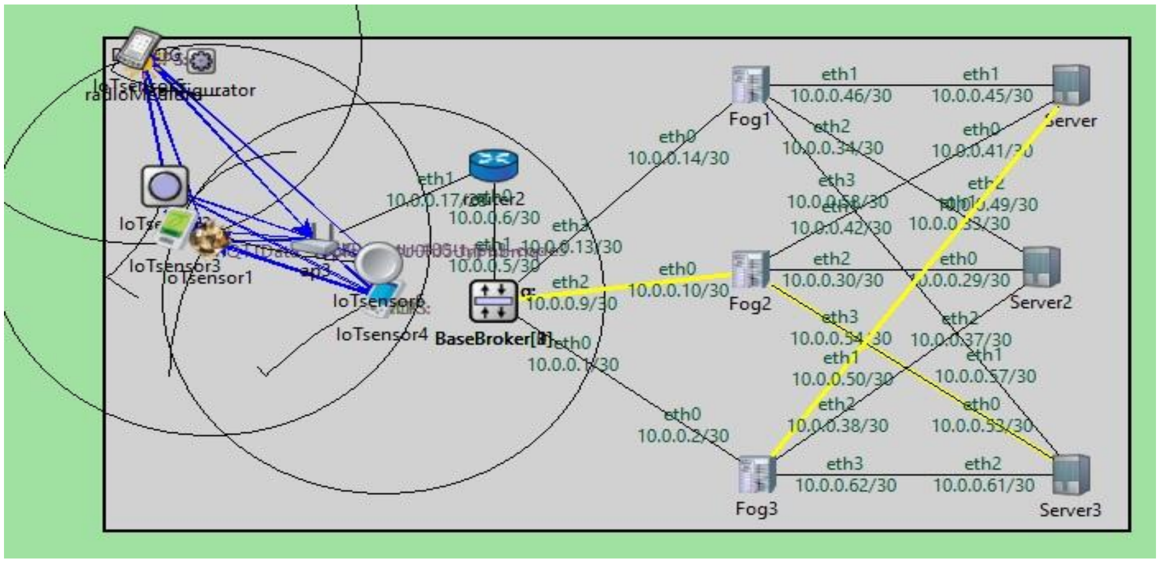


Figure 4.4: The proposed Fog node redirect request to the optimal server.

4.3 The results of Centralized Fog Computing (Case 1)

The proposed structure is predicated in four case studies: the first case study is represented without a load balancer; the second case study is based just on load balancer with the Least Connection (LC) algorithm; the third case study is based on the algorithms to Weighted Round Robin (WRR); and the fourth case study is based on the hybrid approach merged LC and WRR. The first case study is represented without a load balancer; the second case study is based just on load balance.

4.3.1 The 1st case study without Load Balance

The system that has been suggested, which is based on the 4 main findings. The first case study discusses the state of overburdening and high traffic load, which required a load balancing method to balance the load and reduce response time. This state is represented by the second case study all the way up to the fourth case study of performing multiple strategy with a combination of dynamic and static load balancing . The HTTP web request results made by IoT sensors is shown in Table (4.4), and it was sent to link servers through Fog Node. The Response Time is calculated from IoT sensors to the Fog node as point to point connection so the servers are not showed the response time but it calculated between sensors and servers are far away from them, instead the servers calculated the network latency of total sent and received packets.

Throughput is calculated in double values due to the frames are fragmented when it transmit from source node to the destination nodes through the required time for each packets to arrive. Packet loss rate showed the value of lost packets when it transmitted from the source to the destination, it calculated as (number of sent frames – number of received frames) to find the ratio of lost packets.

Table 4.4: The Main Evaluation Parameters Of The 1st Case Study with HTTP Requests of without load balancer.

HTTP WEB request command	**Throughput / Bps**		**Latency / ms**	**Response Time/ms**	**Packet Loss Rate (%)**
	Frames/sec Sent	Frames/sec Received			
IoT sensor 1	18.35915	17.679	40602 ms	55.6188 ms	0.959 %
IoT sensor 2	2.209	0.960	982.8 ms	1.179 ms	1.677 %
IoT sensor 3	20.24	18.96	20961.6 ms	50.674 ms	2 %
IoT sensor 4	21.608	20.8	14871.6 ms	44.793 ms	1.262 %
IoT sensor 5	17.871	16.882	13761.6 ms	57.339 ms	1.545 %
IoT sensor 6	2.817	2.089	1059.6 ms	0.529 ms	1.137 %
Server 1	1.274	1.200	1059.81 ms	/	0.115 %
Server 2	49.918	44.958	30534.820 ms	/	7.75 %
Server 3	31.495	29.696	19490.545 ms	/	2.811%
Fog	82.608	75.855	51085.176 ms	/	10.804 %
HTTP Packet Size	1024 Byte				

Table (4.5) shows the channel resources allocation for IoT sensors and how sensors get benefits from the channel through channel utilization, and the amount of time required for each packets in queue before started to process by the servers. Channel utilization is represented as the amount of used channel by each sensor as channel allocation, the high value is the better and channel idle is represent as the channel is optimal to use by nodes.

Table 4.5: Channel resources allocation for 1st Case Study of without load balancer.

Requests	**Channel Idle (%)**	**Channel Utilization (%)**	**Waiting Time in sec**
IoT sensor 1	68.744	0.097	27.3
IoT sensor 2	68.122	6.428	2.6
IoT sensor 3	69.277	0.73	1.3
IoT sensor 4	60.763	7.097	35.1
IoT sensor 5	69.339	0.0081	36.4
IoT sensor 6	69.349	7.194	15.6
Server 1	69.300	6.7970	26
Server 2	69.282	0.0417	70.2
Server 3	66.568	6.726	2.6
Fog (avg all interfaces)	69.341	3.439	19.337

Table (4.6), Table (4.7), and Table (4.8) showed the upload and download files to the servers (server 1, server 2, and server 3)with different file size to measure speed of data transfer and required time to complete the process. The used protocol for this purpose is the FTP protocol with 21 port number.

Table 4.6: FTP file upload/download by server 1 of the 1st case study of without load balancer.

Server 1: 10.0.0.10/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	91.2	1.771
2 MB	186.675	5.644
15 MB	187.387	88.315

Table 4.7: FTP file upload/download by server 2 of the 1st case study of without load balancer.

Server 2: 10.0.0.26/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	109.725	1.799
2 MB	194.512	5.661
15 MB	193.087	88.526

Table 4.8: FTP file upload/download by server 3 of the 1st case study of without load balancer

Server 3: 10.0.0.34/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	101.887	1.845
2 MB	190.95	5.649
15 MB	192.375	22.626

4.3.2 The 2nd case study of Least connection load balance algorithm

In the second case study, a suggested system was used to balance the load inside the fog computing environment. This system was based on an algorithm that required the fewest connections possible by distributing the IoT sensors to the idle server taking into consideration the amount of connection overloading. Table (4.9) showed the HTTP request distributed among three servers.

Table 4.9: The Main Evaluation Parameters Of The 2nd Case Study with HTTP Requests of Least connection load balancer algorithm.

HTTP WEB request command	**Throughput / Bps**		**Latency / ms**	**Response Time/ms**	**Packet Loss Rate (%)**
	Frames/sec Sent	Frames/sec Received			
IoT sensor 1	21.599	20.7994	33835 ms	46.349 ms	0.7996%
IoT sensor 2	2.599	1.201	819 ms	0.983 ms	1.398 %
IoT sensor 3	25.3	23.7	17468 ms	42.229 ms	1.6 %
IoT sensor 4	27.01	26.00	12393 ms	37.328 ms	1.01 %
IoT sensor 5	22.339	21.103	11468 ms	47.783 ms	1.236 %
IoT sensor 6	3.522	2.612	883 ms	0.441 ms	0.91 %
Server 1	1.593	1.501	883.175 ms	/	0.092 %
Server 2	62.398	56.198	25445.684 ms	/	6.2 %
Server 3	39.369	37.120	16242.121 ms	/	2.249 %
Fog	103.26	94.819	42570.98 ms	/	8.441 %
HTTP Packet Size	1024 Byte				

Table (4.10) showed the channel resources allocation for IoT sensors and it was enhanced compared with the 1st case study through channel availability. Utilization are increased while Waiting Time is decreased.

Table 10: Channel resources allocation for 2nd Case Study of Least connection load balancer algorithm.

Requests	**Channel Idle (%)**	**Channel Utilization (%)**	**Waiting Time in sec**
IoT sensor 1	94.171	0.133	21
IoT sensor 2	93.318	8.806	2
IoT sensor 3	94.901	1	1
IoT sensor 4	83.238	9.723	27
IoT sensor 5	94.985	0.0112	28
IoT sensor 6	94.999	9.855	12
Server 1	94.932	9.311	20
Server 2	94.908	0.0572	54
Server 3	91.190	9.215	2
Fog (avg all interfaces)	94.988	4.712	14.875

Table (4.11 to 4.13) showed the upload and download files to the idle servers with different file size and it showed the least connection useful for file uploading.

Table 4.11: FTP file upload/download by server 1 of the 2nd case study of Least connection load balancer algorithm.

Server 1: 10.0.0.10/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	128	1.250
2 MB	262	3.982
15 MB	263	62.304

Table 4.12: FTP file upload/download by server 2 of the 2nd case study of Least connection load balancer algorithm.

Server 2: 10.0.0.26/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	154	1.270
2 MB	273	3.995
15 MB	271	62.453

Table 4.13: FTP file upload/download by server 3 of the 2nd case study of Least connection load balancer algorithm..

Server 3: 10.0.0.34/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	143	1.302
2 MB	268	3.986
15 MB	270	62.412

4.3.3 The 3rd case study of Weighted Round Robin load balance algorithm

In this instance, the suggested system distributes work using a technique called WRR balancing load among the connected server depending on the weight value assigned to each server to decide which optimal server for the incoming requests. The primary criteria for judging the third case study were laid forth in Table 4.14.This algorithm better than from least connection in case of speed or required time(response time), and it used for requests to servers in a rotational manner, even if some servers have more active connections than others. It adding the ability to spread the incoming client requests across the shared servers.

Table 4.14: The Most Important Criteria for Evaluating the 3rd Case Study with HTTP Requests of Weighted Round Robin load balancer.

HTTP WEB request command	**Throughput / Bps**		**Latency / ms**	**Response Time/ms**	**Packet Loss Rate (%)**
	Frames/sec Sent	Frames/sec Received			
IoT sensor 1	22.678	21.839	32143.25 ms	44.031 ms	0.759 %
IoT sensor 2	2.728	1.261	778.05 ms	0.933 ms	1.328 %
IoT sensor 3	26.565	24.885	16594.6 ms	40.117 ms	1.52 %
IoT sensor 4	28.360	27.3	11773.35 ms	35.461 ms	0.959 %
IoT sensor 5	23.455	22.158	10894.6 ms	45.393 ms	1.174 %
IoT sensor 6	3.698	2.742	838.85 ms	0.418 ms	0.864 %
Server 1	1.672	1.576	839.016 ms	/	0.087 %
Server 2	65.517	59.007	24173.399 ms	/	5.89 %
Server 3	41.337	38.976	15430.014 ms	/	2.136 %
Fog	108.423	99.559	40442.431 ms	/	8.018 %
HTTP Packet Size	1024 Byte				

Table (4.15) showed the channel allocation for fog computing environment

Table 4.15: Channel utilization for 3rd case study of Weighted Round Robin load balancer.

Ping	**Channel Idle (%)**	**Channel Utilization (%)**	**Waiting Time in sec**
IoT sensor 1	99.128	0.139	19.952
IoT sensor 2	98.230	9.246	1.9
IoT sensor 3	100	1.05	0.951
IoT sensor 4	92.231	10.209	25.65

IoT sensor 5	99.985	0.0117	26.6
IoT sensor 6	99.999	10.347	11.4
Server 1	97.929	9.776	18
Server 2	98.904	0.0600	51.3
Server 3	94.99	9.675	1.9
Fog (avg all interfaces)	97.988	4.947	14.131

Table (4.16 to 4.18) showed the upload and download files to the idle servers with different file size in the Fog computing environment. It showed how file uploaded with different data size into 3 servers in some cases frames packets are fragmented into piece of frames to pass through channel from IoT sensors to the servers.

Table 4.16: FTP file upload/download by server 1 of the 3rd case study of Weighted Round Robin load balancer.

Server 1: 10.0.0.10/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	121.6	1.312
2 MB	248.9	4.181
15 MB	249.85	65.419

Table 4.17: FTP file upload/download by server 2 of the 3rd case study of Weighted Round Robin load balancer.

Server 2: 10.0.0.26/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	146.3	1.333
2 MB	259.35	4.194
15 MB	257.45	65.575

Table 4.18: FTP file upload/download by server 3 of the 3rd case study of Weighted Round Robin load balancer.

Server 3: 10.0.0.34/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	135.85	1.367
2 MB	254.6	4.185
15 MB	256.5	65.532

4.3.4 The 4th case study of Hybrid load balance approach

It used the hybrid methodology to distribute the load among servers through the weighted summation approach, which assigned weight value for each server and redirected traffic to the least weight value depending on the least connection and weighted round robin algorithms. Table (4.19) showed the used evaluation parameters for the 4th case study.

Table 4.19: The Main Evaluation Parameters Of The 4th Case Study with HTTP Requests of Hybrid load balancer.

HTTP WEB request command	**Throughput / Bps**		**Latency / ms**	**Response Time/ms**	**Packet Loss Rate (%)**
	Frames/sec Sent	Frames/sec Received			
IoT sensor 1	24.265	23.367	29250.357 ms	40.068 ms	0.5589 %
IoT sensor 2	2.918	1.349	708.025 ms	0.849 ms	0.92412 %
IoT sensor 3	28.424	26.626	15101.08 ms	36.506 ms	1.0579 %
IoT sensor 4	30.345	29.211	10713.74 ms	32.269 ms	0.6664 %
IoT sensor 5	25.096	23.709	9914.08 ms	41.307 ms	0.8170 %
IoT sensor 6	3.956	2.933	763.35 ms	0.380 ms	0.6012 %
Server 1	1.789	1.686	763.50 ms	/	0.0604 %
Server 2	70.103	63.137	21,997.79 ms	/	4.0996 %

Server 3	44.230	41.704	14041.31 ms	/	1.4863 %
Fog	116.012	106.528	36802.61 ms	/	5.2531 %
HTTP Packet Size	1024 Byte				

Table (4.20) showed the channel utilization with queue time of the 4th case study.

Table 4.20: Channel utilization with Waiting Time of the 4th case study of Hybrid load balancer.

Ping	**Channel Idle (%)**	**Channel Utilization (%)**	**Waiting Time in sec**
IoT sensor 1	99.623	0.145	18.954
IoT sensor 2	98.721	9.708	1.805
IoT sensor 3	100	1.102	0.903
IoT sensor 4	94.075	10.719	24.367
IoT sensor 5	99.991	0.012	25.27
IoT sensor 6	100	10.864	10.83
Server 1	99.929	10.264	17.1
Server 2	99.904	0.063	48.735
Server 3	96.99	10.158	1.805
Fog (avg all interfaces)	99.988	5.194	13.424

Table (4.21 to 4.23) showed server speed and required time to the upload and download files by idle servers with different file size streaming.

Table 4.21: FTP file upload/download by server 1 of the 4th case study of Hybrid load balancer.

Server 1: 10.0.0.10/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	134.4	1.187

2 MB	275.1	3.782
15 MB	276.15	59.188

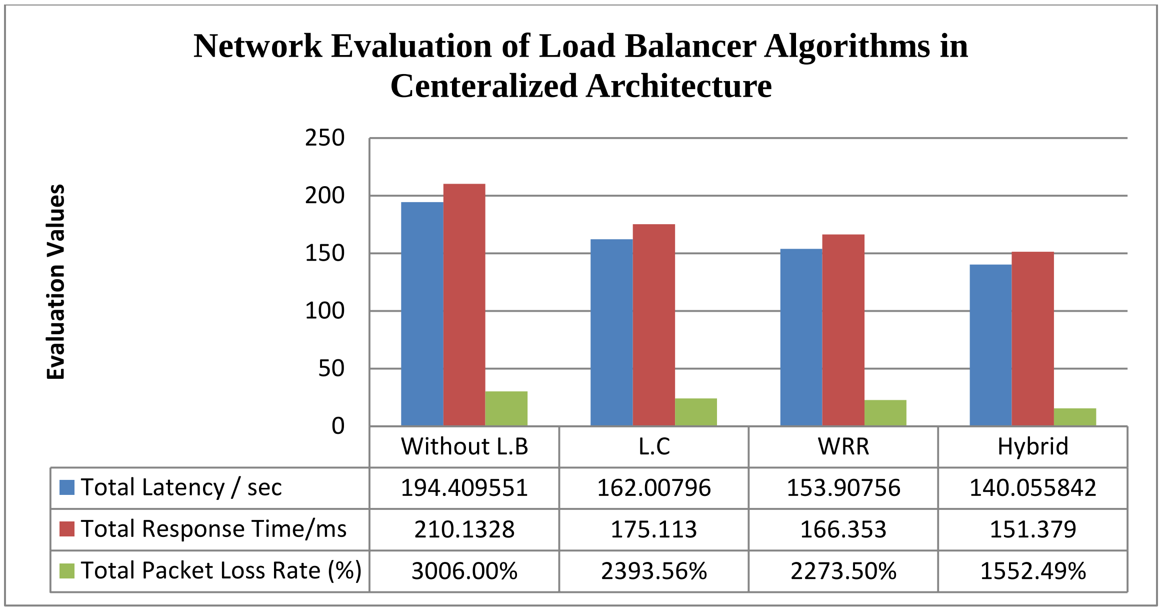
Table 4.22: FTP file upload/download by server 2 of the 4th case study of Hybrid load balancer.

Server 2: 10.0.0.26/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	161.7	1.206
2 MB	286.65	3.795
15 MB	284.55	59.330

Table 4.23: FTP file upload/download by server 3 of the 4th case study of Hybrid load balancer.

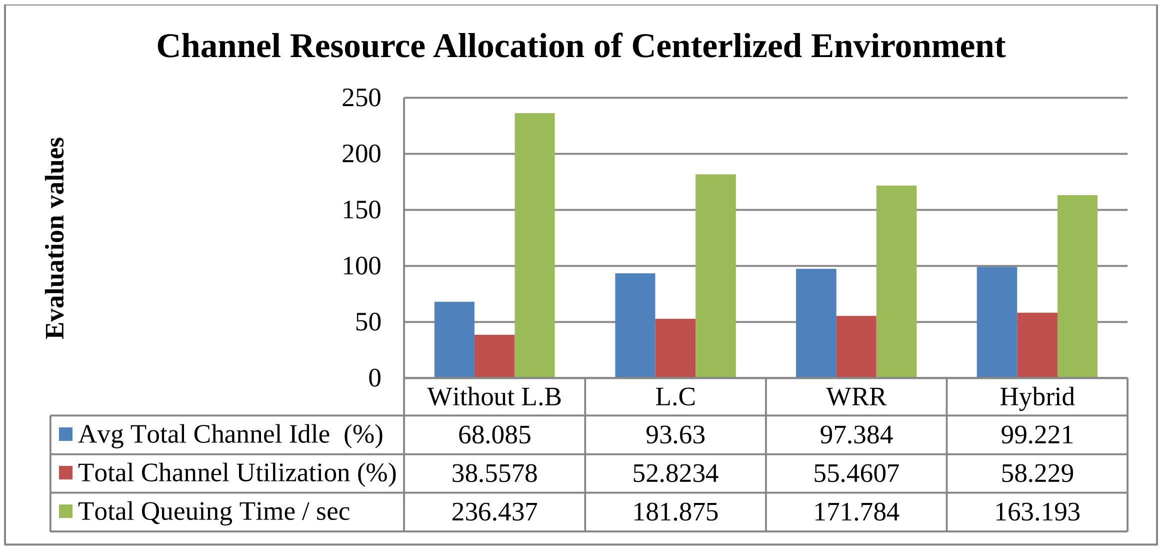
Server 3: 10.0.0.34/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	150.15	1.236
2 MB	281.4	3.786
15 MB	283.5	59.291

The proposed system showed that the hybrid approach is better in case HTTP and FTP file transfer with different file size with the least connection algorithm and WRR algorithms. The least connection is better from weighed round robin in case of FTP file transfer, while the weighted round robin is better from the least connection in case of HTTP requests. In addition, the hybrid approach is better in channel utilization from both standalone algorithms. Figure (4.5) and Figure (4.6) showed the system comparison.



	Without L.B	L.C	WRR	Hybrid
🟦 Total Latency / sec	194.409551	162.00796	153.90756	140.055842
🟥 Total Response Time/ms	210.1328	175.113	166.353	151.379
🟩 Total Packet Loss Rate (%)	3006.00%	2393.56%	2273.50%	1552.49%

Figure 4.5: The total latency ,response time and packet loss rate of the proposed system.

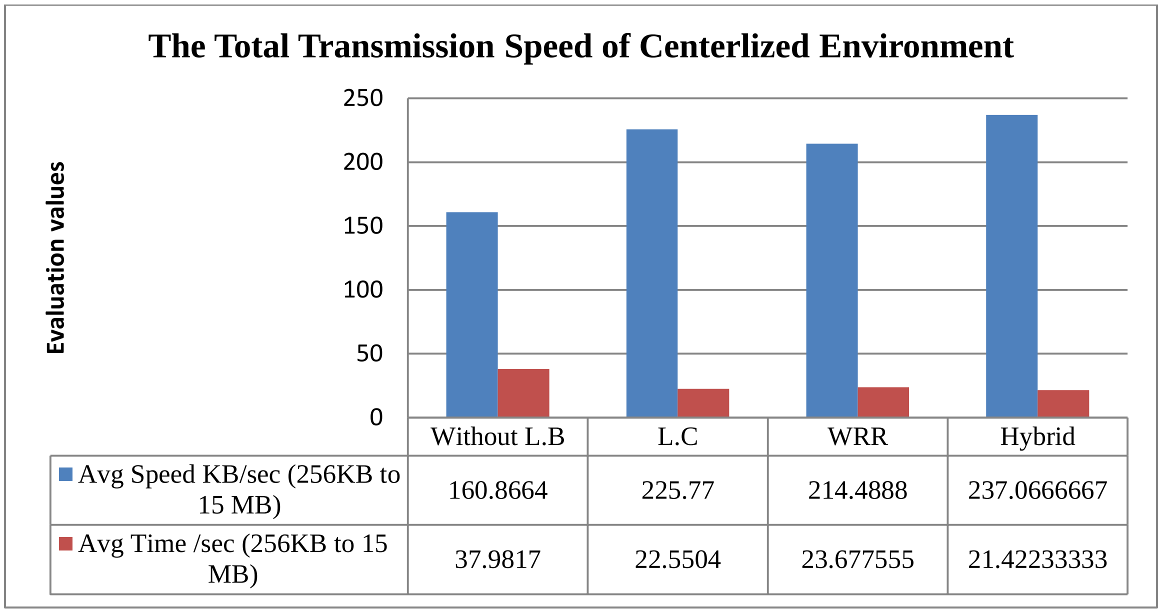


	Without L.B	L.C	WRR	Hybrid
🟦 Avg Total Channel Idle (%)	68.085	93.63	97.384	99.221
🟥 Total Channel Utilization (%)	38.5578	52.8234	55.4607	58.229
🟩 Total Queuing Time / sec	236.437	181.875	171.784	163.193

Figure 4.6: Channel resource allocation for the proposed Hybrid system case studies.

Figure (4.7) showed the transfer speed and time elapsed to execute the FTP file transmission job for the used case studies, It was shown that the hybrid strategy that was presented performed better when compared to the other cases of load balancing as it represented the average for all three used servers speed in KB/sec (FTP File size 256KB to 15 MB) as 237.0666

KB/sec, and average time /sec (256KB to 15 MB) as 21.42233 sec. While the least connection algorithm showed the second level enhancement in data transmission traffic with speed is 225.77 KB/sec, and time is 22.5504 sec.



	Without L.B	L.C	WRR	Hybrid
🟦 Avg Speed KB/sec (256KB to 15 MB)	160.8664	225.77	214.4888	237.0666667
🟥 Avg Time /sec (256KB to 15 MB)	37.9817	22.5504	23.677555	21.42233333

Figure 4.7: The total transmission speed and required time of the proposed centralized system.

The proposed system compared with other related works and it showed better evaluation results as it showed in Table (4.24).

Table 4.24 : A comparison of the proposed system with the other relevant works.

Load Balancing comparisons					
Ref.Year	**Arch, Tool**	**Algorithm**	**Avg Speed KB/sec**	**Packet Loss Rate (%)**	**Time Taken for Resource Allocation**
[10], 2020	Fog-4 servers, Ns-3	Weighted Round Robin	110 KB/sec	180 %	/
[24], 2022	Fog-IoT, CloudAnalyst, CloudSim	Queuing Theory based Cuckoo Search (QTCS) model	/	/	8 sec

The proposed system	Fog-3 Servers, Fognetsim++, OMNET++	**Weighted Round Robin, Least Connection**	**237.066 KB/sec**	**15.5249%**	**3.786c**

4.4 The results of Distributed Fog Computing (Case 2)

The proposed system is based on four case studies, the 1st is represented without load balancer, while the second case is based on the middleware with the Least Connection (LC) method, while the third case study just on algorithms with Weighted Round Robin (WRR), and the fourth case is based upon the hybrid model combining LC and WRR that was implemented for each fog node. Particle Swarm Optimization (PSO) task scheduling has been used in distributed fog computing to optimize and scheduling incoming request to the specific fog node. PSO is used to manage tasks in processing (active requests) executed on the Fog, also have different direction to specific server. Therefore, in order to acquire an optimum choice of fog, it is required to discover an ideal way to put the various responsibilities evenly on the various fogs, taking into consideration the various properties of both the fog as well as the application.

4.4.1 The 1st case study of without Load Balance

The system that has been suggested, which is based just on four major findings. The first case study discusses the state of swamping and high traffic load, which required a load balancing method to balance the load and reduce response time. This state is represented by the second case study all the way up to the fourth study case of load balancing strategy with a combination of dynamic and static load balancing. Table (4.25) showed the HTTP web request by Iot sensors to the connected servers through fog node.

Table 4.25: The Main Evaluation Parameters Of The 1st Case Study with HTTP Requests of without Load Balance in distributed environment.

HTTP WEB request command	**Throughput / Bps**		**Latency / ms**	**Response Time/ms**	**Packet Loss Rate (%)**
	Frames/sec Sent	Frames/sec Received			
Wireless-IoT-sensor 1	18.5427	17.855	39789.96 ms	54.506 ms	0.6877 %
Wireless-IoT-sensor 2	2.231	0.969	963.144 ms	1.155 ms	1.263 %
Wireless-IoT-sensor 3	20.442	19.149	20542.368 ms	49.660 ms	1.523 %
Wireless-IoT-sensor 4	21.824	21.008	14574.168 ms	43.897 ms	0.816 %
Wireless-IoT-sensor 5	18.049	17.050	13486.368 ms	56.192 ms	1.011 %
Wireless-IoT-sensor 6	2.845	2.109	1038.408 ms	0.518 ms	0.789 %
Server 1	1.286	1.212	1038.613ms	/	0.067 %
Server 2	50.417	45.407	29924.123 ms	/	5.128 %
Server 3	31.809	29.992	19100.73 ms	/	1.796%
Fog 1	27.118	25.102	17008.123 ms	/	2.018 %
Fog 2	29.302	27.233	16010.109 ms	/	2.906 %
Fog 3	28.100	27.206	15905.056 ms	/	0.898 %
HTTP Packet Size	1024 Byte				

Table (4.26) shows the channel resources allocation for IoT sensors and how sensors gets benefits from the channel through channel utilization, and the amount of time required for each packets in queue before started to process by the servers.

Table 4.26: Channel resources allocation for 1st Case Study of without Load Balance in distributed environment.

Requests	**Channel Idle (%)**	**Channel Utilization (%)**	**Waiting Time in sec**
Wireless-IoT-sensor 1	69.431	0.098	27.027

Wireless-IoT-sensor 2	68.803	6.492	2.574
Wireless-IoT-sensor 3	69.969	0.737	1.287
Wireless-IoT-sensor 4	61.370	7.167	34.749
Wireless-IoT-sensor 5	70.032	0.0081	36.036
Wireless-IoT-sensor 6	70.042	7.265	15.444
Server 1	69.993	6.864	25.74
Server 2	69.974	0.0421	69.498
Server 3	67.233	6.793	2.574
Fog 1(avg all interfaces)	70.134	3.473	9.928
Fog 2(avg all interfaces)	72.236	5.919	10.919
Fog 3(avg all interfaces)	71.458	4.790	12.980

Table (4.27 to 4.29) shows the upload and download files to the servers with different file size to measure speed of data transfer and required time to complete the process.

Table 4.27: FTP file upload/download by server 1 of the 1st case study of without Load Balance in distributed environment.

Server 1: 10.0.0.10/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	93.024	1.735
2 MB	190.408	5.531
15 MB	191.134	86.548

Table 4.28: FTP file upload/download by server 2 of the 1st case study of without Load Balance in distributed environment.

Server 2: 10.0.0.26/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec

256 KB	111.919	1.763
2 MB	198.402	5.547
15 MB	196.948	86.755

Table 4.29: FTP file upload/download by server 3 of the 1st case study of without Load Balance in distributed environment.

Server 3: 10.0.0.34/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	103.924	1.808
2 MB	194.769	5.536
15 MB	196.222	22.173

4.4.2 The 2nd case study of Least connection load balance algorithm

The proposed system in the second case study was based on an algorithm called least connection load balancing. This algorithm was designed to balance the load that was being carried by each of the three fog nodes. This was accomplished by distributing the IoT sensors from the active server to the idle server pool through take into consideration the amount of connection overloading. Table (4.30) showed the HTTP request distributed among three servers.

Table 4.30: The Main Evaluation Parameters Of The 2nd Case Study with HTTP Requests of Least connection Load Balance in distributed environment.

HTTP WEB request command	**Throughput / Bps**		**Latency / ms**	**Response Time/ms**	**Packet Loss Rate (%)**
	Frames/sec Sent	Frames/sec Received			
Wireless-IoT-sensor 1	22.678	21.839	32143.25 ms	44.031 ms	0.783 %
Wireless-IoT-sensor 2	2.728	1.261	778.05 ms	0.933 ms	1.370 %
Wireless-IoT-sensor 3	26.565	24.885	16594.6 ms	40.117 ms	1.56 %

Wireless-IoT-sensor 4	28.360	27.3	11773.35 ms	35.461 ms	0.989 %
Wireless-IoT-sensor 5	23.455	22.158	10894.6 ms	45.393 ms	1.211 %
Wireless-IoT-sensor 6	3.698	2.742	838.85 ms	0.418 ms	0.891 %
Server 1	1.672	1.576	839.016 ms	/	0.090 %
Server 2	65.517	59.007	24173.39 ms	/	6.07 %
Server 3	41.337	38.976	15430.01 ms	/	2.204 %
Fog 1	36.124	33.590	13480.117 ms	/	2.534 %
Fog 2	35.236	34.172	12480.287 ms	/	1.101 %
Fog 3	38.780	37.120	15160.097 ms	/	1.661 %
HTTP Packet Size	1024 Byte				

Table (4.31) showed the channel resources allocation for IoT sensors and it enhanced compared with the 1st case study through channel. Availability and utilization are increased while Waiting Time is decreased.

Table 4.31: Channel resources allocation for 2nd Case Study of Least connection Load Balance in distributed environment.

Requests	**Channel Idle (%)**	**Channel Utilization (%)**	**Waiting Time in sec**
Wireless-IoT-sensor 1	95.112	0.134	20.58
Wireless-IoT-sensor 2	94.251	8.894	1.96
Wireless-IoT-sensor 3	95.850	1.01	0.97
Wireless-IoT-sensor 4	84.070	9.820	25.92
Wireless-IoT-sensor 5	95.934	0.0113	26.88
Wireless-IoT-sensor 6	95.948	9.953	11.52
Server 1	95.881	9.404	19.2
Server 2	95.857	0.057	51.84
Server 3	92.101	9.307	1.92

Fog 1(avg all interfaces)	95.937	4.996	4.68
Fog 2(avg all interfaces)	96.896	5.187	5.127
Fog 3(avg all interfaces)	97.855	5.615	5.068

Table (4.32 to 4.34) showed the upload and download files to the idle servers with different file size and it showed the least connection useful for file uploading.

Table 4.32: FTP file upload/download by server 1 of the 2nd case study of Least connection Load Balance in distributed environment.

Server 1: 10.0.0.10/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	134.4	1.187
2 MB	275.1	3.782
15 MB	276.15	59.188

Table 4.33: FTP file upload/download by server 2 of the 2nd case study of Least connection Load Balance in distributed environment.

Server 2: 10.0.0.26/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	161.7	1.206
2 MB	286.65	3.795
15 MB	284.55	59.330

Table 4.34: FTP file upload/download by server 3 of the 2nd case study of Least connection Load Balance in distributed environment.

Server 3: 10.0.0.34/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	298.777	1.236
2 MB	281.4	3.786
15 MB	283.5	59.291

4.4.3 The 3rd case study of WRR L.B algorithm

The weighted rounds robin load balancing method has been proposed to distribute load among the connected distributed servers with distributed fog nodes depending on the weight value assigned to each server to decide which the optimal server for the incoming requests. The most important assessment criteria for the third case study showed in table (4.35).

Table 4.35: The 3rd Case Study with HTTP Requests of WRR Load Balance in distributed environment.

HTTP WEB request command	**Throughput / Bps**		**Latency / ms**	**Response Time/ms**	**Packet Loss Rate (%)**
	Frames/sec Sent	Frames/sec Received			
Wireless-IoT-sensor 1	23.811	22.930	30536.087 ms	34.431 ms	0.743 %
Wireless-IoT-sensor 2	2.864	1.324	739.147 ms	0.793 ms	1.301 %
Wireless-IoT-sensor 3	27.893	26.129	15764.87 ms	34.099 ms	1.489 %
Wireless-IoT-sensor 4	29.778	28.665	11184.682 ms	30.141 ms	0.939 %
Wireless-IoT-sensor 5	24.627	23.265	10349.87 ms	38.584 ms	1.150 %
Wireless-IoT-sensor 6	3.882	2.879	796.907 ms	0.355 ms	0.846 %
Server 1	1.755	1.654	797.065 ms	/	0.085 %
Server 2	68.792	61.957	22964.729 ms	/	5.772 %
Server 3	43.4038	40.924	14658.513 ms	/	2.093 %
Fog 1	43.102	42.357	12132.105 ms	/	0.734 %
Fog 2	41.689	40.631	11232.258 ms	/	1.058 %
Fog 3	40.147	39.859	13644.087 ms	/	0.288 %
HTTP Packet Size	1024 Byte				

Table (4.36) showed the channel allocation for distributed fog computing environment with distributed three servers.

Table 4.36: Channel utilization for 3rd case study of WRR Load Balance in distributed environment.

Ping	Channel Idle (%)	Channel Utilization (%)	Waiting Time in sec
Wireless-IoT-sensor 1	100	0.140	19.153
Wireless-IoT-sensor 2	99.212	9.338	1.824
Wireless-IoT-sensor 3	100	1.060	0.912
Wireless-IoT-sensor 4	93.153	10.311	24.624
Wireless-IoT-sensor 5	100	0.0118	25.536
Wireless-IoT-sensor 6	100	10.450	10.944
Server 1	98.908	9.873	17.28
Server 2	99.893	0.060	49.248
Server 3	95.939	9.771	1.824
Fog 1(avg all interfaces)	98.967	5.194	4.586
Fog 2(avg all interfaces)	98.125	5.243	5.024
Fog 3(avg all interfaces)	98.829	5.293	4.561

Table (4.37 to 4.39) showed the upload and download files to the idle servers with different file size in the Fog computing environment.

Table 4.37: FTP file upload/download by server 1 of the 3rd case study of WRR Load Balance in distributed environment.

Server 1: 10.0.0.10/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	125.248	1.272
2 MB	256.367	4.055
15 MB	257.345	63.456

Table 4.38: FTP file upload/download by server 2 of the 3rd case study of WRR Load Balance in distributed environment.

Server 2: 10.0.0.26/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	150.689	1.293
1 MB	267.130	4.068
16 MB	265.173	63.607

Table 4.39: FTP file upload/download by server 3 of the 3rd case study of WRR Load Balance in distributed environment.

Server 3: 10.0.0.34/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	139.925	1.325
2 MB	262.238	4.059
15 MB	264.195	63.566

4.4.4 The 4th case study of Hybrid load balance approach

The fourth case study of evaluating the proposed system to distribute the load among servers through the weighted summation approach which assign weight value for each active servers and redirect traffic to the least weight value depending on the least connection and weighted round robin algorithms depending on the distributed fog nodes. Table (4.40) showed the used evaluation parameters for the 4th case study.

Table 4.40: The 4th Case Study with HTTP Requests of Hybrid Load Balance in distributed environment.

HTTP WEB request command	**Throughput / Bps**		**Latency / ms**	**Response Time/ms**	**Packet Loss Rate (%)**
	Frames/sec Sent	Frames/sec Received			
Wireless-IoT-sensor 1	25.001	24.076	29009.282ms	32.709 ms	0.705 %

Wireless-IoT-sensor 2	3.007	1.390	702.189 ms	0.753 ms	1.235 %
Wireless-IoT-sensor 3	29.287	27.435	14976.626 ms	32.394 ms	1.414 %
Wireless-IoT-sensor 4	31.266	30.098	10625.447ms	28.633 ms	0.892 %
Wireless-IoT-sensor 5	25.858	24.428	9,832.376 ms	36.654 ms	1.092 %
Wireless-IoT-sensor 6	4.076	3.022	757.061 ms	0.337 ms	0.803 %
Server 1	1.842	1.736	757.211 ms	/	0.080 %
Server 2	72.231	65.054	21816.492 ms	/	5.483 %
Server 3	45.573	42.970	13925.587 ms	/	1.988 %
Fog 1	45.257	44.474	10918.894 ms	/	0.697 %
Fog 2	43.773	42.662	10109.032 ms	/	1.005 %
Fog 3	42.154	41.851	12279.678 ms	/	0.273 %
HTTP Packet Size	1024 Byte				

Table (4.41) showed the channel utilization with queue time of the 4th case study.

Table 4.41: Channel utilization with Waiting Time of the 4th case study of Hybrid Load Balance in distributed environment.

Ping	**Channel Idle (%)**	**Channel Utilization (%)**	**Waiting Time in sec**
Wireless-IoT-sensor 1	100	0.159	17.058
Wireless-IoT-sensor 2	99.708	10.678	1.624
Wireless-IoT-sensor 3	100	1.212	0.812
Wireless-IoT-sensor 4	96.015	11.790	21.930
Wireless-IoT-sensor 5	100	0.013	22.743
Wireless-IoT-sensor 6	100	11.950	9.747
Server 1	100	11.290	15.39
Server 2	100	0.069	43.861

Server 3	98.929	11.173	1.444
Fog 1(avg all interfaces)	100	5.609	3.668
Fog 2(avg all interfaces)	100	6.232	4.421
Fog 3(avg all interfaces)	99.99	7.271	4.104

Table 4.42, Table 4.43, and Table 4.44 showed the distributed servers speed and required time to the upload and download files by idle servers with different file size streaming.

Table 4.42: FTP file upload/download by server 1 of the 4th case study of Hybrid Load Balance in distributed environment.

Server 1: 10.0.0.10/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	147.84	1.068
2 MB	302.61	3.403
15 MB	303.765	53.269

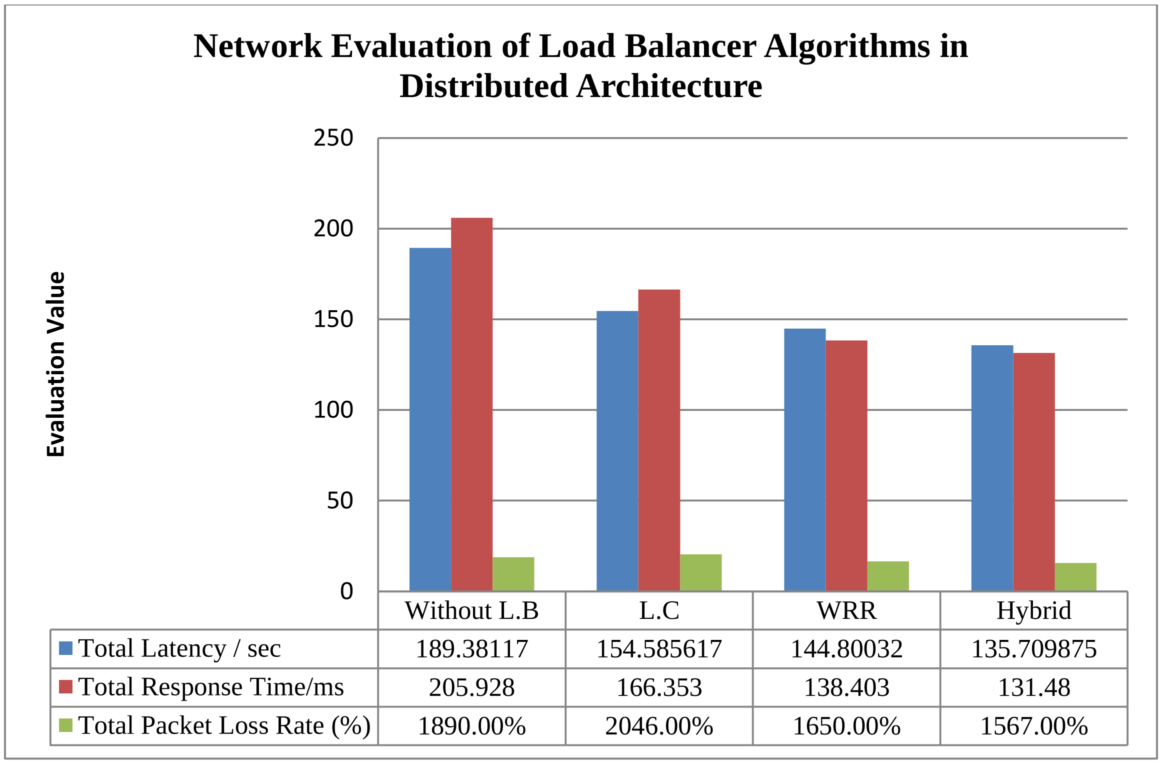
Table 4.43: FTP file upload/download by server 2 of the 4th case study of Hybrid Load Balance in distributed environment.

Server 2: 10.0.0.26/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	177.87	1.085
2 MB	315.315	3.415
15 MB	313.005	53.397

Table 4.44: FTP file upload/download by server 3 of the 4th case study of Hybrid Load Balance in distributed environment.

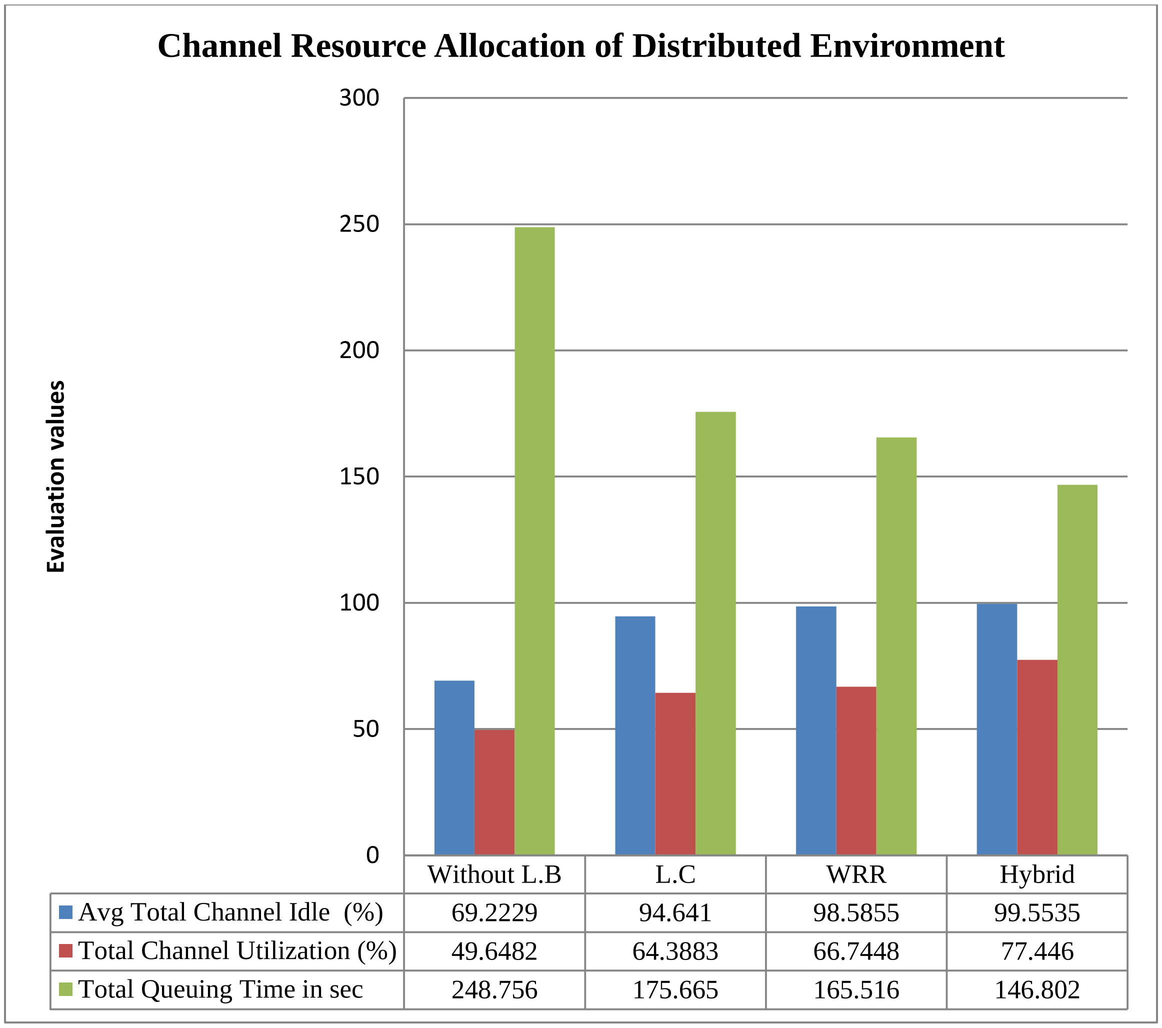
Server 3: 10.0.0.34/30		
FTP File Size Upload/Download	Speed in KB/sec	Time in Sec
256 KB	165.165	1.112
2 MB	309.54	3.407
15 MB	311.85	53.361

The proposed system showed the distributed fog hybrid load balancer approach better in case HTTP and FTP file transfer with different file size and the least connection algorithm and WRR method. The least connection is better than weighed round robin in case of FTP file transfer, while the weighted round robin is better than least connection in case of HTTP requests. In addition, the hybrid approach was better in channel utilization than both standalone algorithms. Figure (4.8) and Figure (4.9) showed the system comparison. The best latency from the Hybrid approach due to the channel allocation in better way compared with others and response time is decreased due to the decreased required time to pass request from IoT sensors to the servers.



	Without L.B	L.C	WRR	Hybrid
🟦 Total Latency / sec	189.38117	154.585617	144.80032	135.709875
🟥 Total Response Time/ms	205.928	166.353	138.403	131.48
🟩 Total Packet Loss Rate (%)	1890.00%	2046.00%	1650.00%	1567.00%

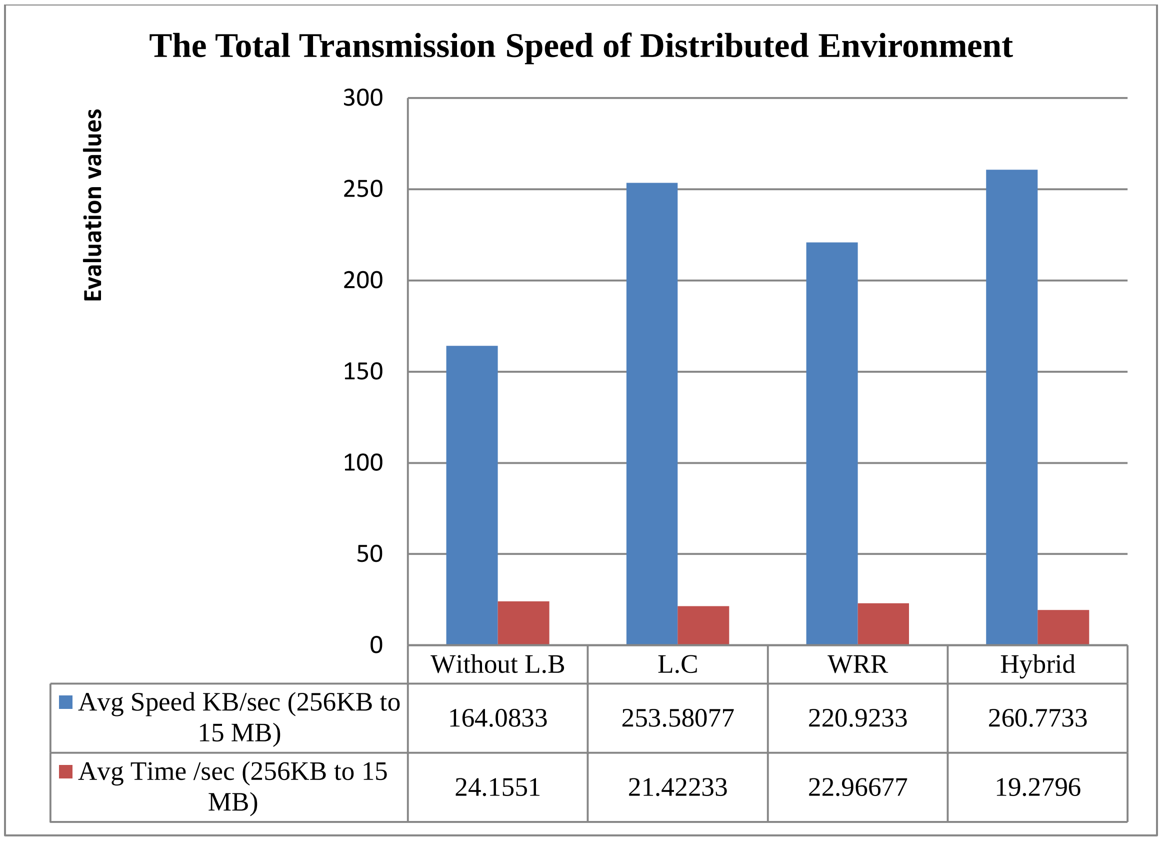
Figure 4.8: The total latency ,response time and packet loss rate of the proposed system.



	Without L.B	L.C	WRR	Hybrid
🟦 Avg Total Channel Idle (%)	69.2229	94.641	98.5855	99.5535
🟥 Total Channel Utilization (%)	49.6482	64.3883	66.7448	77.446
🟩 Total Queuing Time in sec	248.756	175.665	165.516	146.802

Figure 4.9: Channel resource allocation for the proposed system case studies.

Figure (4.10) showed the transfer speed and time elapsed to execute the FTP file transmission job for the used case studies, it showed that the proposed distributed hybrid approach better compared with the other cases of load balances as it represented the average for all three used distributed servers speed in KB/sec (FTP File size 256KB to 15 MB) as 260.7733 KB/sec, and average time /sec (256KB to 15 MB) as 19.2796 sec. While the least connection algorithm showed the second level enhancement in data transmission traffic with speed is 253.58077 KB/sec, and time is 21.42233 sec.



	Without L.B	L.C	WRR	Hybrid
🟦 Avg Speed KB/sec (256KB to 15 MB)	164.0833	253.58077	220.9233	260.7733
🟥 Avg Time /sec (256KB to 15 MB)	24.1551	21.42233	22.96677	19.2796

Figure 4.10: The total transmission speed and required time of the proposed system.

The proposed system compared with other related works showed better evaluation results as shown in Table (4.45).

Table 4.45 : A comparison of the suggested system with the other relevant works.

Ref.Year	**Arch, Tool**	**Algorithm**	**Total Response Time in ms**	**Total Time Taken**
[11], 2020	Cloud-Fog load balancing algorithms, iFogSim	Proximity Algorithm	/	90 sec
		Cluster Algorithm	/	100 sec
[25], 2022	Hybrid load-balancing- Fog, iFogSim	Optimizing Processing Time (OPT)	309.5 ms	/
		First Come First Serve (FCFS)	326.7 ms	/

		Priority algorithm	323.7 ms	/
[26], 2022	cloud-fog, Java Netbeans and cloud analyst	Round Robin (RR)	270.14 ms	/
		Whale Optimization algorithm with bat algorithm (WOA-BAT)	256.59 ms	/
The proposed system	**3 Fog-3 Servers, Fognetsim++, OMNET++**	**Hybrid : Weighted Round Robin, Least Connection**	**131.48 ms**	**57.838 sec**

4.5 **Summary**

The proposed architecture is implemented practically over four different case studies in each centralized and distributed fog node. Fog load balancer layer is suggested in this architecture to improve performance such as reduce delay, response time, loss rate and increase throughput. To confirm the efficiency and efficacy of the proposed fog hybrids load balancer system technique, comparison analyses are being carried out.

The findings indicate that the proposed support resource allocation in each servers especially in distributed environment . In addition network performance is improved because it guarantees the effective servicing of Internet of Things (IoT) requests emanating from the Internet of things layer by making use of the services provided by centralized and decentralized cloud computing with hybrid task scheduling algorithm.

Chapter Five

Conclusions and Suggestions for Future Works

5.1 Conclusions

This chapter presents an explanation of the proposed system's conclusions and the main suggestions for future works. They can be summarized as :

1- The implemented system is based on two main case studies of load balancing as the 1st centralized fog computing, and the 2nd case study is distributed fog computing.

  A- The centralized fog computing approach is based on the fog node controlled on the traffic and redirect the request to the servers in load balancing state.

  B- The Distributed Fog Computing approach is based on distributed fog nodes controlled on the traffic with Particle Swarm Optimization (PSO) task scheduling to optimize and scheduling incoming request to the specific fog node.

2- The results show the proposed system improved network evaluation parameters in :

  A- Centralized hybrid approach better compared with the other cases of load balances as it represents the average for all three used servers speed in KB/sec (FTP File size 256KB to 15 MB) as 237.0666 KB/sec, and average time /sec (256KB to 15 MB) as 21.42233 sec, average total channel idle is 99.221 %, total channel utilization is 58.229 %, Total Latency is 140.055 sec, Total Response Time is 151.379 ms, and Total Packet Loss Rate is 1552.49 %.

  B- Distributed hybrid approach better compared with the other cases of load balances as it represented the average for all three used distributed servers speed in KB/sec (FTP File size 256KB to 15 MB) as 260.7733 KB/sec, average time /sec (256KB to 15 MB) as 19.2796 sec, average total channel idle is 99.55 %, total channel utilization is 77.44 %, Total Latency is 135.709 sec, Total Response Time is 131.48 ms, and Total Packet Loss Rate is 1567.00%.

3. The proposed system enhanced the network performance and resource allocation with load balancing algorithms, it increased channel idle, channel allocation, throughput, and decreased waiting time and response time due to the channel allocation and task scheduling processes applied in basebroker, and fog nodes.

5.2 Suggestions for Future Works

For the further development of current research, the following suggestions are important to implement in future:

1- Investigating the interplay of the cloud center and fog node, where the fog node can further offload the data to the cloud center if it does not have enough computational resources.

2- Implementing the real time applications which requires high bandwidth with reduced latency to ensure Quality of Service (QoS) of embedded devices deployed with the Internet of Things (IoT) technology.

3- Employing energy consumption model in the fog layer to help in proper utilization of resources, which helps to reduce latency and enhance the quality of service in in state of huge amount of IoT sensors of tasks need to be assigned equally to all the nodes in the fog layer.

REFERENCES

[1] Kaur, M., & Aron, R. (2021). A systematic study of load balancing approaches in the fog computing environment. *The Journal of Supercomputing*, *77*(8), 9202-9247.

[2] Sulimani, H., Alghamdi, W. Y., Jan, T., Bharathy, G., & Prasad, M. (2021). Sustainability of Load Balancing Techniques in Fog Computing Environment. *Procedia Computer Science, 191*, 93-101.

[3] Asghar, A., Abbas, A., Khattak, H. A., & Khan, S. U. (2021). Fog based architecture and load balancing methodology for health monitoring systems. *IEEE Access*, *9*, 96189-96200.

[4] Vilela, P. H., Rodrigues, J. J., Righi, R. D. R., Kozlov, S., & Rodrigues, V. F. (2020). Looking at fog computing for e-health through the lens of deployment challenges and applications. *Sensors*, *20*(9), 2553.

[5] Mani, N., Singh, A., & Nimmagadda, S. L. (2020). An IoT guided healthcare monitoring system for managing real-time notifications by fog computing services. *Procedia Computer Science, 167*, 850-859.

[6] de Moura Costa, H. J., da Costa, C. A., da Rosa Righi, R., & Antunes, R. S. (2020). Fog computing in health: A systematic literature review. *Health and Technology*, *10*(5), 1025-1044.

[7] Chandak, A., & Ray, N. K. (2019, December). A review of load balancing in fog computing. In *2019 International Conference on Information Technology (ICIT)*. IEEE.

[8] Muniswamaiah, M., Agerwala, T., & Tappert, C. C. (2021, June). Fog computing and the internet of things (IoT): a review. In *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE.

[9] Puthal, D., Mohanty, S. P., Bhavake, S. A., Morgan, G., & Ranjan, R. (2019). Fog computing security challenges and future directions [energy and security]. *IEEE Consumer Electronics Magazine*, *8*(3), 92-96.

[10] Al-Joboury, I. M., & Al-Hemiary, E. H. (2020). Virtualized fog network with load balancing for IoT based fog-to-cloud. *JOIV: International Journal on Informatics Visualization*, *4*(3), 123-126.

[11] Bala, M. I., & Chishti, M. A. (2020, January). Offloading in cloud and fog hybrid infrastructure using iFogSim. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence-)*. IEEE.

[12] Huang, X., Fan, W., Chen, Q., & Zhang, J. (2020). Energy-efficient resource allocation in fog computing networks with the candidate mechanism. *IEEE Internet of Things Journal*, *7*(9), 8502-8512.

[13] Mani, S. K., & Meenakshisundaram, I. (2020). Improving quality-of-service in fog computing through efficient resource allocation. *Computational Intelligence*, *36*(4), 1527-1547.

[14] Lee, S. S., & Lee, S. (2020). Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information. *IEEE Internet of Things Journal*, *7*(10), 10450-10464.

[15] Chang, Z., Liu, L., Guo, X., & Sheng, Q. (2020). Dynamic resource allocation and computation offloading for IoT fog computing system. *IEEE Transactions on Industrial Informatics*, *17*(5), 3348-3357.

[16] Naha, R. K., & Garg, S. (2021). Multi-criteria--based Dynamic User Behaviour--aware Resource Allocation in Fog Computing. *ACM Transactions on Internet of Things*, *2*(1), 1-31.

[17] Beraldi, R., Canali, C., Lancellotti, R., & Mattia, G. P. (2020). Distributed load balancing for heterogeneous fog computing infrastructures in smart cities. *Pervasive and Mobile Computing*, *67*, 101221.

[18] Noman, H. M., & Jasim, M. N. (2021, February). A Comparative Performance Analysis for Static and Dynamic Load Balancing Techniques in Software Defined Network Environment. In *Journal of Physics: Conference Series*. IOP Publishing.

[19] Mujthaba, G. M., Kolhar, M., & AlAmeen, A. Load Balancing at Fog Nodes using Scheduling Algorithms.

[20] Alqahtani, F., Amoon, M., & Nasr, A. A. (2021). Reliable scheduling and load balancing for requests in cloud-fog computing. *Peer-to-Peer Networking and Applications*, *14*(4), 1905-1916.

[21] Divya, V., & Sri, L. R. (2021). Fault tolerant resource allocation in fog environment using game theory-based reinforcement learning. *concurrency and Computation-practice & experience*, *33*(16).

[22] Wadhwa, H., & Aron, R. (2022). TRAM: Technique for resource allocation and management in fog computing environment. *The Journal of Supercomputing*, *78*(1), 667-690.

[23] Tripathy, S. S., Roy, D. S., & Barik, R. K. (2021). M2FBalancer: A mist-assisted fog computing-based load balancing strategy for smart cities. *Journal of Ambient Intelligence and Smart Environments, 13*(3), 219-233.

[24] Iyapparaja, M., Alshammari, N. K., Kumar, M. S., Krishnan, S., Rama, S., & Chowdhary, C. L. (2022). Efficient resource allocation in fog computing using QTCS model. *CMC-COMPUTERS MATERIALS & CONTINUA*, *70*(2), 2225-2239.

[25] Abuhamdah, A., & Al-Shabi, M. (2022). Hybrid Load Balancing Algorithm For Fog Computing Environment. *International Journal of Software Engineering and Computer Systems*, *8*(1), 11-21.

[26] Saoud, A., & Recioui, A. (2022). Hybrid algorithm for cloud-fog system based load balancing in smart grids. *Bulletin of Electrical Engineering and Informatics*, *11*(1), 477-487.

[27] Jamil, B., Ijaz, H., Shojafar, M., Munir, K., & Buyya, R. (2022). Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions. *ACM Computing Surveys (CSUR)*.

[28] Singh, P., Kaur, R., Rashid, J., Juneja, S., Dhiman, G., Kim, J., & Ouaissa, M. (2022). A Fog-Cluster Based Load-Balancing Technique. *Sustainability*, *14*(13), 7961.

[29] Sofla, M. S., Kashani, M. H., Mahdipour, E., & Mirzaee, R. F. (2022). Towards effective offloading mechanisms in fog computing. *Multimedia Tools and Applications*, 1.

[30] Ndubuaku, M., & Okereafor, D. (2015). Internet of things for Africa: challenges and opportunities. *vol*, *9*, 23-31.

[31] Vermesan, O., Friess, P., Guillemin, P., Sundmaeker, H., Eisenhauer, M., Moessner, K., ... & Cousin, P. (2013). Internet of things strategic research and innovation agenda. *Internet of things: Converging technologies for smart environments and integrated ecosystems*, 7-152.

[32] Karagiannis, V., & Schulte, S. (2020, May). Comparison of alternative architectures in fog computing. In *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*. IEEE.

[33] Losada, M., Cortés, A., Irizar, A., Cejudo, J., & Pérez, A. (2020). A flexible fog computing design for low-power consumption and low latency applications. *Electronics*, *10*(1), 57.

[34] Martinez, I., Hafid, A. S., & Jarray, A. (2020). Design, resource management, and evaluation of fog computing systems: a survey. *IEEE Internet of Things Journal*, *8*(4), 2494-2516.

[35] Quy, V. K., Hau, N. V., Anh, D. V., & Ngoc, L. A. (2021). Smart healthcare IoT applications based on fog computing: architecture, applications and challenges. *Complex & Intelligent Systems*, 1-11.

[36] Lai, K. L., Chen, J. I. Z., & Zong, J. I. (2021). Development of smart cities with fog computing and internet of things. *Journal of Ubiquitous Computing and Communication Technologies (UCCT), 3*(01), 52-60.

[37] Deokar, S., Mangla, M., & Akhare, R. (2021). A secure fog computing architecture for continuous health monitoring. In *Fog Computing for Healthcare 4.0 Environments*. Springer, Cham.

[38] Abdali, T. A. N., Hassan, R., Aman, A. H. M., & Nguyen, Q. N. (2021). Fog computing advancement: Concept, architecture, applications, advantages, and open issues. *IEEE Access, 9*, 75961-75980.

[39] Neware, R., & Shrawankar, U. (2020). Fog computing architecture, applications and security issues. *International Journal of Fog Computing (IJFC), 3*(1), 75-105.

[40] Habibi, P., Farhoudi, M., Kazemian, S., Khorsandi, S., & Leon-Garcia, A. (2020). Fog computing: a comprehensive architectural survey. *IEEE Access, 8*, 69105-69133.

[41] Awaisi, K. S., Hussain, S., Ahmed, M., Khan, A. A., & Ahmed, G. (2020). Leveraging IoT and fog computing in healthcare systems. *IEEE Internet of Things Magazine*, *3*(2), 52-56.

[42] Varghese, B., Wang, N., Nikolopoulos, D. S., & Buyya, R. (2020). Feasibility of fog computing. In *Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things*. Springer, Cham.

[43] Karthika, P., Ganesh Babu, R., & Karthik, P. A. (2020). Fog computing using interoperability and IoT security issues in health care. In *Micro-Electronics and Telecommunication Engineering*. Springer, Singapore.

[44] Moura, J., & Hutchison, D. (2020). Fog computing systems: State of the art, research issues and future trends, with a focus on resilience. *Journal of Network and Computer Applications*, *169*, 102784.

[45] Ali, A., Ahmed, M., Imran, M., & Khattak, H. A. (2020). Security and privacy issues in fog computing. *Fog Computing: Theory and Practice*, 105-137.

[46] Prabhu, C. S. R. (2017). Overview-fog computing and internet-of-things (IOT). *EAI endorsed transactions on cloud systems*, *3*(10).

[47] Kaur, A., Singh, P., & Nayyar, A. (2020). Fog computing: building a road to IoT with fog analytics. In *Fog data analytics for IoT applications*. Springer, Singapore.

[48] Saeed, W., Ahmad, Z., Jehangiri, A. I., Mohamed, N., Umar, A. I., & Ahmad, J. (2021). A fault tolerant data management scheme for healthcare Internet of Things in fog computing. *KSII Transactions on Internet and Information Systems (TIIS)*, *15*(1), 35-57.

[49] Muneeb, M., Ko, K. M., & Park, Y. H. (2021). A fog computing architecture with multi-layer for computing-intensive IoT applications. *Applied Sciences*, *11*(24), 11585.

[50] Singh, J., Singh, P., & Gill, S. S. (2021). Fog computing: A taxonomy, systematic review, current trends and research challenges. *Journal of Parallel and Distributed Computing*, *157*, 56-85.

[51] Malik, U. M., Javed, M. A., Zeadally, S., & ul Islam, S. (2021). Energy efficient fog computing for 6G enabled massive IoT: Recent trends and future opportunities. *IEEE Internet of Things Journal*.

[52] Ren, J., Zhang, D., He, S., Zhang, Y., & Li, T. (2019). A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys (CSUR)*, *52*(6), 1-36.

[53] Elmagzoub, M. A., Syed, D., Shaikh, A., Islam, N., Alghamdi, A., & Rizwan, S. (2021). A survey of swarm intelligence based load balancing techniques in cloud computing environment. *Electronics*, *10*(21), 2718.

[54] Naik, K. J., & Naik, D. H. (2020). Minimizing deadline misses and total run-time with load balancing for a connected car systems in fog computing. *Scalable Computing: Practice and Experience*, *21*(1), 73-84.

[55] Shahid, M. H., Hameed, A. R., ul Islam, S., Khattak, H. A., Din, I. U., & Rodrigues, J. J. (2020). Energy and delay efficient fog computing using caching mechanism. *Computer Communications*, *154*, 534-541.

[56] Rajguru, A. A., & Apte, S. S. (2012). A comparative performance analysis of load balancing algorithms in distributed system using qualitative parameters. *International Journal of Recent Technology and Engineering*, *1*(3), 175-179.

[57] Dos Anjos, J. C., Gross, J. L., Matteussi, K. J., González, G. V., Leithardt, V. R., & Geyer, C. F. (2021). An algorithm to minimize energy consumption and elapsed time for IoT workloads in a hybrid architecture. *Sensors*, *21*(9), 2914.

[58] Khattak, H. A., Arshad, H., Ahmed, G., Jabbar, S., Sharif, A. M., & Khalid, S. (2019). Utilization and load balancing in fog servers for health applications. *EURASIP Journal on Wireless Communications and Networking*, *2019*(1), 1-12.

[59] Sharif, A., Nickray, M., & Shahidinejad, A. (2020). Fault-tolerant with load balancing scheduling in a fog-based IoT application. *IET Communications*, *14*(16), 2646-2657.

[60] Beshley, M., Kryvinska, N., Yaremko, O., & Beshley, H. (2021). A self-optimizing technique based on vertical handover for load balancing in heterogeneous wireless networks using big data analytics. *Applied Sciences*, *11*(11), 4737.

[61] Malik, N., Sardaraz, M., Tahir, M., Shah, B., Ali, G., & Moreira, F. (2021). Energy-efficient load balancing algorithm for workflow scheduling in cloud data centers using queuing and thresholds. *Applied Sciences*, *11*(13), 5849.

[62] Mubeen, A., Ibrahim, M., Bibi, N., Baz, M., Hamam, H., & Cheikhrouhou, O. (2021). Alts: An Adaptive Load Balanced Task Scheduling Approach for Cloud Computing. *Processes*, *9*(9), 1514.

[63] Rafid, M., Hossain, J. T., Ali, M. N., Abonti, N. T., & Masud, A. (2021). *Resource optimization in Cloud Computing using dynamic load balancing technique* (Doctoral dissertation, Brac University).

[64] Wang, W., & Casale, G. (2014, September). Evaluating weighted round robin load balancing for cloud web services. In *2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. IEEE.

[65] Xu, M., Tian, W., & Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurrency and Computation: Practice and Experience*, *29*(12), e4123.

[66] Kaur, N., Singh, J., Goyal, S., & Duhan, B. (2020). Load balancing in cloud computing: The online traffic management. *Journal of Natural Remedies*, *21*(2), 202-209.

[67] Pujangga, M. D. C., Hidayat, N., & Bakhtiar, F. A. (2021). Implementasi Load Balancing pada Cloud Computing dengan Algoritme Improved Weighted Least Connection. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN*, *2548*, 964X.

[68] Dagdeviren, Z. A. (2021). Weighted connected vertex cover based energy-efficient link monitoring for wireless sensor networks towards secure internet of things. *IEEE Access*, *9*, 10107-10119.

[69] Alhaidari, F., & Balharith, T. Z. (2021). Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling. *Computers*, *10*(5), 63.

[70] Chetouani, E., Errami, Y., Obbadi, A., & Sahnoun, S. (2021). Design of optimal backstepping control for a wind power plant system using the adaptive weighted

particle swarm optimization. *International Journal of Intelligent Engineering and Systems (IJIES)*, *14*(6).

[71] Kodli, S., & Terdal, S. (2021). Hybrid max-min genetic algorithm for load balancing and task scheduling in cloud environment. *Int J Intell Eng Syst.*, *14*(1), 63-71.

[72] Ebadifard, F., & Babamir, S. M. (2021). Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. *Cluster Computing*, *24*(2), 1075-1101.

[73] Miao, Z., Yong, P., Mei, Y., Quanjun, Y., & Xu, X. (2021). A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment. *Future Generation Computer Systems*, *115*, 497-516.

[74] Kashani, M. H., & Mahdipour, E. (2022). Load Balancing Algorithms in Fog Computing: A Systematic Review. *IEEE Transactions on Services Computing*.

[75] Tripathy, S. S., Roy, D. S., & Barik, R. K. (2021). M2FBalancer: A mist-assisted fog computing-based load balancing strategy for smart cities. *Journal of Ambient Intelligence and Smart Environments*, *13*(3), 219-233.

[76] Tran-Dang, H., & Kim, D. S. (2021). FRATO: fog resource based adaptive task offloading for delay-minimizing IoT service provisioning. *IEEE Transactions on Parallel and Distributed Systems*, *32*(10), 2491-2508.

[77] Wang, T., Liang, Y., Jia, W., Arif, M., Liu, A., & Xie, M. (2019). Coupling resource management based on fog computing in smart city systems. *Journal of Network and Computer Applications*, *135*, 11-19.

[78] Phi, N. X., Tin, C. T., Thu, L. N. K., & Hung, T. C. (2018). Proposed load balancing algorithm to reduce response time and processing time on cloud computing. *Int. J. Comput. Netw. Commun*, *10*(3), 87-98.

[79] Campanile, L., Gribaudo, M., Iacono, M., & Mastroianni, M. (2020). Performance evaluation of a fog WSN infrastructure for emergency management. *Simulation Modelling Practice and Theory*, *104*, 102120.

[80] Alizadeh, M. R., Khajehvand, V., Rahmani, A. M., & Akbari, E. (2020). Task scheduling approaches in fog computing: A systematic review. *International Journal of Communication Systems*, *33*(16), e4583.

[81] Abbasi, M., Mohammadi-Pasand, E., & Khosravi, M. R. (2021). Intelligent workload allocation in IoT–Fog–cloud architecture towards mobile edge computing. *Computer Communications*, *169*, 71-80.

[82] de Matos, E., Tiburski, R. T., Moratelli, C. R., Johann Filho, S., Amaral, L. A., Ramachandran, G., ... & Hessel, F. (2020). Context information sharing for the Internet of Things: A survey. *Computer Networks*, *166*, 106988.

[83] Mukherjee, A., De, D., & Ghosh, S. K. (2020). FogIoHT: A weighted majority game theory based energy-efficient delay-sensitive fog network for internet of health things. *Internet of Things*, *11*, 100181.

[84] Maswood, M. M. S., Rahman, M. R., Alharbi, A. G., & Medhi, D. (2020). A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment. *IEEE Access, 8*, 113737-113750.

[85] Van Le, T., Ioini, N. E., Pahl, C., & Barzegar, H. R. (2020, September). Edge computing simulation platforms: A technology survey. In *European Conference on Service-Oriented and Cloud Computing*. Springer, Cham.

[86] Markus, A., & Kertesz, A. (2020). A survey and taxonomy of simulation environments modelling fog computing. *Simulation Modelling Practice and Theory*, *101*, 102042.

[87] Ghenai, A., Kabouche, Y., & Dahmani, W. (2018, December). Multi-user dynamic scheduling-based resource management for Internet of Things applications. In *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*. IEEE.

[88] Kong, L., Mapetu, J. P. B., & Chen, Z. (2020). Heuristic load balancing based zero imbalance mechanism in cloud computing. *Journal of Grid Computing*, *18*(1), 123-148.

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

الخلاصة

تختار تطبيقات إنترنت الأشياء (IoTs) عقد الحوسبة الضبابية للاستجابة لمتطلبات الموارد ، وتعد موازنة الأحمال أحد العوامل الرئيسية لتحقيق كفاءة الموارد وتجنب الاختناقات والحمل الزائد والتأخير والحمل المنخفض. ومع ذلك ، لا يزال من الصعب تحقيق توازن الحمل لعقد الحوسبة في بيئة الضباب أثناء تنفيذ تطبيقات إنترنت الأشياء. تتضمن موازنة الحمل بعض الأهداف في بيئة الضباب ، بما في ذلك زيادة الإنتاجية وتقليل وقت الاستجابة وتحسين حركة المرور. يعد تحسين الاستهلاك في الموارد من جانب الخادم ، وطلب تقليل وقت المعالجة ، وتحسين قابلية التوسع في البيئة الموزعة بعض الأغراض الأخرى لتقنية موازنة حمل الحوسبة الضبابية.

لتقليل حمل تأثير عقد المرور وفشل عقدة الضباب على بنية حوسبة الضباب ، يجب أن تستخدم كل عقدة بنية موزعة للحصول على نتائج أفضل. بالنسبة لجانب حمل العقدة. تستخدم عقدة الخدمة التكنولوجيا الموزعة ، والتي تدمج موارد الحوسبة للشبكة بأكملها ، وتحسن استخدام موارد الحوسبة. تعتمد المنهجية المقترحة على موازنة الحمل الهجين مع خوارزميات(Least Connection (LC و Weighted Round Robin (WRR) مجتمعة معًا في عُقد ضبابية لمراعاة الحمل المروري ، ووقت الاستجابة لتوزيع الطلبات على الخوادم المتاحة في الشبكة.

أشارت النتائج إلى أن أداء الشبكة مُحسّن ، والذي بدوره يضمن خدمة نقل بيانات إنترنت الأشياء بشكل فعال ، كما هو الحال في النهج الهجين الموزع ضمن البيئة الموزعة ، يكون وقت الاستجابة ١٣١.٤٨ مللي ثانية ، ومعدل فقدان الحزمة الإجمالي ١٥.٦٧٠٪ ، ومتوسط إجمالي الخمول للقناة ٩٩.٥٥٪ ، وإجمالي القناة تبلغ نسبة الاستخدام ٧٧.٤٤٪ ، ومتوسط سرعة نقل الملفات ٢٦٠.٧٧ كيلوبايت / ثانية ، ومتوسط وقت نقل الملفات ١٩.٢٧ ثانية.

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

اقتراح نهج موازنة حمل هجين لتحسين أداء الحوسبة المضببة

رسالة مقدمة

إلى مجلس كلية تكنولوجيا المعلومات في جامعة بابل والتي هي جزء من متطلبات

الحصول على درجة الماجستير في تكنولوجيا المعلومات / شبكات المعلومات

من قبل الطالب

ابرار سعد كاظم حسن

باشراف

أ.م.د مهدي عبادي مانع مهدي

٢٠٢٢ م ١٤٤٤ هـ