APPENDIX 1

LOW LATENCY SCHEDULING FOR DELAY SENSITIVE APPLICATIONS

A PROJECT REPORT

Submitted by

ARAVINTH S - 2018105010

GNANAVEL K M - 2018105528

GOWTHAM P - 2018105531

YOKESH N - 2018105623

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2022

APPENDIX 2

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"Low latency scheduling for delay sensitive applications"** is the bonafide work of **"Aravinth S, Gnanavel K M, Gowtham P, Yokesh N"** who carried out the project work under my supervision.

SIGNATURE

Dr. M. Meenakshi

HEAD OF THE DEPARTMENT

Professor

Department of ECE

College of Engineering, Guindy

Anna University,

Chennai-600025

SIGNATURE

Dr. M. A. Bhagyaveni

SUPERVISOR

Professor

Department of ECE

College of Engineering, Guindy

Anna University,

Chennai-600025

ACKNOWLEDGEMENT

First and foremost, we thank God for an abundance of grace and countless blessings in making this work a great success.

We also express our sincere gratitude to our Head of the Department Dr. M. MEENAKSHI, Professor, Department of Electronics & Communication Engineering for her enthusiastic encouragement and support throughout the project.

We present our sincere thanks and gratitude to our project supervisor, Dr. M.A. BHAGYAVENI, Professor, Department of Electronics and Communication Engineering for her whole hearted support, patience, valuable guidance, technical expertise and encouragement in our project.

We thank all the teaching and non-teaching staff of Department of Electronics and Communication Engineering, for their kind help and co-operation during the course of our project.

Last but not the least, we would not have made this project without the support from our beloved family and friends.

ABSTRACT

In today's world, Internet of Things (IoT) is developing rapidly. Wireless sensor network (WSN) as an infrastructure of IoT has limitations in the processing power, storage, and delay for data transfer to cloud. The large volume of generated data and their transmission between WSNs and cloud are serious challenges.

Fog computing (FC) as an extension of cloud to the edge of the network reduces latency and traffic; thus, it is very useful in IoT applications such as healthcare applications, wearables, intelligent transportation systems, and smart cities. Resource allocation and task scheduling are the NP-hard issues in FC. Each application includes several modules that require resources to run. Fog devices (FDs) have the ability to run resource management algorithms because of their proximity to sensors and cloud as well as the proper processing power.

Certain delay sensitive applications require low latency, to address this we introduce a number of scheduling algorithms. Our proposed method was simulated in iFogsim(simulator) as a standard simulator for FC. We compare the benefits of different scheduling algorithms First-Come-First-Serve (FCFS), Priority Scheduling, Priority Scheduling based on threshold. We review the scheduling strategies and parameters as well as providing a knapsack-based scheduling algorithm for allocating resources appropriately to modules in fog network. The results show that the energy consumption, execution cost, and sensor lifetime in Knapsack algorithm are better than those of the FCFS, Priority Scheduling algorithm, Priority Scheduling based on threshold. Application that uses knapsack algorithm performs 3.6 times better in terms of latency and decrease in network usage by 92% than that of FCFS algorithm.

திட்டப்பணிச்சுருக்கம்

இன்றைய உலகில், விஷயங்களின் இணையம் (IoT) வேகமாக வளர்ந்து வருகிறது. IoT இன் உள்கட்டமைப்பாக வயர்லெஸ் சென்சார் நெட்வொர்க் (WSN) செயலாக்க சக்தி, சேமிப்பு மற்றும் மேகக்கணிக்கு தரவு பரிமாற்றத்திற்கான தாமதம் ஆகியவற்றில் வரம்புகளைக் கொண்டுள்ளது. உருவாக்கப்பட்ட தரவுகளின் பெரிய அளவு மற்றும் WSNகள் மற்றும் கிளவுட் இடையே அவற்றின் பரிமாற்றம் ஆகியவை கடுமையான சவால்களாகும். ஃபோக் கம்ப்யூட்டிங் (எஃப்சி) நெட்வொர்க்கின் விளிம்பிற்கு மேகக்கணியின் விரிவாக்கம் தாமதம் மற்றும் போக்குவரத்தை குறைக்கிறது; எனவே, சுகாதாரப் பயன்பாடுகள், அணியக்கூடிய பொருட்கள், அறிவார்ந்த போக்குவரத்து அமைப்புகள் மற்றும் ஸ்மார்ட் நகரங்கள் போன்ற IoT பயன்பாடுகளில் இது மிகவும் பயனுள்ளதாக இருக்கும். வள ஒதுக்கீடு மற்றும் பணி திட்டமிடல் ஆகியவை FC இல் உள்ள NP-கடினமான சிக்கல்களாகும். ஒவ்வொரு பயன்பாட்டிலும் இயங்குவதற்கு ஆதாரங்கள் தேவைப்படும் பல தொகுதிகள் உள்ளன. மூடுபனி சாதனங்கள் (FDs) சென்சார்கள் மற்றும் கிளவுட் மற்றும் சரியான செயலாக்க சக்தி ஆகியவற்றுக்கு அருகாமையில் இருப்பதால் வள மேலாண்மை அல்காரிதம்களை இயக்கும் திறனைக் கொண்டுள்ளன.

சில தாமத உணர்திறன் பயன்பாடுகளுக்கு குறைந்த தாமதம் தேவைப்படுகிறது, இதை நிவர்த்தி செய்ய நாங்கள் பல திட்டமிடல் அல்காரிதங்களை அறிமுகப்படுத்துகிறோம்.

எங்களின் முன்மொழியப்பட்ட முறை FCக்கான நிலையான சிமுலேட்டராக iFogsim(சிமுலேட்டர்) இல் உருவகப்படுத்தப்பட்டது. வெவ்வேறு திட்டமிடல் அல்காரிதம்களின் பலன்களை நாங்கள் ஒப்பிடுகிறோம் (முதலில் வருபவர்கள் முதல் சேவை (FCFS), முன்னுரிமை திட்டமிடல், வரம்பு அடிப்படையில் முன்னுரிமை திட்டமிடல்). நாங்கள் திட்டமிடல் உத்திகள் மற்றும் அளவுருக்களை மதிப்பாய்வு செய்வோம், அத்துடன் மூடுபனி நெட்வொர்க்கில் உள்ள தொகுதிகளுக்கு சரியான முறையில் வளங்களை ஒதுக்குவதற்கு நாப்சாக் அடிப்படையிலான திட்டமிடல் அல்காரிதத்தை வழங்குகிறோம். நாப்சாக் அல்காரிதத்தில் உள்ள ஆற்றல் நுகர்வு, செயல்படுத்தும் செலவு மற்றும் சென்சார் ஆயுட்காலம் ஆகியவை ஃபர்ஸ்ட் கம்-ஃபர்ஸ்ட்-சர்வ் (FCFS), முன்னுரிமை திட்டமிடல் அல்காரிதம், த்ரெஷோல்ட் அடிப்படையிலான முன்னுரிமைத் திட்டமிடல் ஆகியவற்றை விட சிறந்தவை என்று முடிவுகள் காட்டுகின்றன.

APPENDIX 3

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IoT	INTERNET OF THINGS
WSN	WIRELESS SENSOR NETWORK
FC	FOG COMPUTING
FD	FOG DEVICE
FCFS	FIRST COME FIRST SERVE
FN	FOG NODE
QOS	QUALITY OF SERVICE
MDCS	MICRODATACENTERS
PES	PROCESSING ELEMENTS

EEG	ELECTROENCEPHALOGRAPHY
VM	VIRTUAL MACHINE
EEGTBG	EEG TRACTOR BEAM GAME
MCT	MINIMUM COMPLETE TIME
VR	VIRTUAL REALITY
MEC	MOBILE EDGE COMPUTING
CMAS	COST MAKESPAN AWARE SCHEDULING
GUI	GRAPHICAL USER INTERFACE
MIPS	MILLION INSTRUCTION PER SECOND
RAM	RANDOM ACCESS MEMORY
HB	HOST'S BANDWIDTH
FB	FOG'S BANDWIDTH
TAM	TOTAL ALLOCATED MIPS
LU	LAST UTILIZATION
CEC	CURRENT ENERGY CONSUMPTION
HP	HOST POWER
LUUT	LAST UTILIZATION UPDATE TIME
CC	CLOUDSIM CLOCK

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In the modern world, many communication devices have built-in wireless sensors. Wireless sensor networks (WSNs) collect data in applications including healthcare, transportation, smart cities, smart energy grid, urgent computing. These networks as the infrastructure of IoT require real-time processing and decision-making. In a cloud-based system, data collected by WSNs should be transmitted to cloud over a period of time and processed there. In fact, processing in cloud has a lot of delays and bottlenecks for a large amount of collected data by end devices and sensors.

Data transmission from end-sensor nodes to cloud by passing several mid sensor nodes, routers, and gateways has high total network power consumption and long delays. In many sensitive cases such as medical care and transportation systems, high number of delays in IoT applications can lead to a patient's death or cause an accident. Edge computing has been the next generation of cloud computing, which performs computations in the vicinity of sensors and does not send the data to cloud. In many applications, computing or storage is required in the absence of sufficient resources at the edge of the network, so fog computing (FC) is appropriate for this work.

FC, as a new paradigm, is located at the middle level of cloud and sensor nodes so that data collection, processing, and storage can be done locally and data are sent to cloud only if needed. Processing in the fog device(FD), due to the proximity of sensors to the network edge, increases the speed and also reduces network traffic. FC with a large number of nodes has lower energy consumption than centralized cloud computing systems. Many challenges of the

big data processing can be solved by FC. Also, FC has many challenges as large-scale and distributed sensors and fog nodes (FNs), dynamically changing due to the on-off switching of IoT applications and the mobility of FNs, and quality of service (QoS) requirements.

1.2 LITERATURE SURVEY

In [1], the author proposed the scheduling in distributed environments is generally divided into three categories: resource scheduling, workflow scheduling, and task scheduling. Scheduling is defined as follows: finding an optimal solution for allocation of a set of resources R = {R1, R2, ..., Rm} to a set of tasks T = {T1, T2, ..., Tn} or workflow. Scheduling can be done by the deployment of a set of predefined constraints and objective functions. The task is a small part of the work that must be performed within a specified time. One of the goals of task scheduling is to maximize the use of existing resources and minimize the waiting time of jobs. In scheduling problems, the service user interest corresponds to optimal make span, budget, deadline, security, and cost. Also, the service provider's objective is load balancing, resource utilization, and energy efficiency.

In [2], the optimization strategies include heuristics, meta heuristics, and other methods. The resource models include virtual machine (VM) leasing model, single or multiple VM types, single or multiple providers, intermediate data sharing model, data transfer and storage cost awareness, static, dynamic, subscription, time unit VM pricing model, and VM provisioning delay. Scheduling strategies have different parameters and algorithms. We categorized these methods as follows: Traditional methods are easy to use for scheduling. In the scheduling of multilevel deadline-constrained scientific workflows, each provider offers a few heterogeneous VMs and global storage service for data

sharing. One of the scheduling methods is the earliest finish time, in which cost and make span objectives are optimized.We analyzed commercial infrastructure as a cloud service. They used the Pareto front as a decision support tool for the trade-off between appropriate solutions. They reduced the scheduling cost by half, but the make span increased by 5%.

In [3], the researchers studied the QoS parameters on three scheduling methods, i.e. concurrent, FCFS, and delay-priority in fog network. In the concurrent method, the arrival tasks are allocated to a cloudlet regardless of usage capacity. In the FCFS method, the tasks run in the order of entry, also if the processing power of the data center is less than the task request, then that program is placed in the scheduler queue. In the delay-priority method, tasks are scheduled based on lower delay. The researcher used iFogsim library with two games as video surveillance/object tracking application and EEG tractor beam game (EEGTBG). The results show that the concurrent method has a greater delay than FCFS and delay-priority methods. In EEGTBG, the number of modules in each device for the concurrent method is greater than those of the other two, also this parameter in FCFS method is greater than those of others in cloud. In the final comparison, the number of modules per device for delay priority method is approximately equal in all three methods.

In [4], author have studied knapsack algorithm for task scheduling for parallel video transferring in cloud by minimum complete time (MCT) objective. They executed the max-min algorithm on the powerful computers for task's mapping to a number of segments and scheduling segments by MCT algorithm. The results show that the max-min algorithm is better than MCT in the execution time and the number of segments. Researchers in solved the offloading in mobile edge computing (MEC). They minimized the energy consumption under the time-sharing constraint. Their method decomposed computations onto the mobile devices and the MEC servers and make them

autonomously collaborate to iteratively reach the optimum. In another research, task offloading of DAG-based applications is done in [30]. In this work, a cost-make span aware scheduling (CMaS) algorithm was proposed. The results satisfy the user-defined deadline constraints and improve the QoS.

The authors in [5], found the optimal orders of running tasks based on deadline and minimum cost using knapsack with dynamic programming. The different scheduling problem solved by knapsack based ACO to find the best solution as a mapping between multiple knapsacks and load scheduling. In a resource scheduling in cloud solved by the combination of the knapsack and GA with the fitness function include utilization of CPU, network throughput, and input/output rate of the disk. They decreased the energy consumption of the physical machine and the number of migration compared to the standard methods. In resources were allocated to tasks in FC. Machine learning is another method for solving the scheduling problem in IoT. Researchers combined reinforcement learning with the quality of experience (QoE) from the users to build prestored cost mapping table for optimal resource allocation.

1.3 OBJECTIVE

The data generated by the IoT devices sent to the cloud for processing the high amount of data. There is a huge delay due to this, Our idea is to reduce the delay between the IoT devices and cloud services. So it is very useful in IoT Applications such as healthcare applications, wearables, intelligent transportation systems and smart cities. Certain delay sensitive applications require low latency , to address this issue we introduce a number of scheduling algorithms. We compare the benefits of different scheduling algorithms using iFogsim simulator.

1.4 PROJECT SPECIFIC INTRODUCTION

An application in the fog network includes some modules. The modules in FDs can do a variety of computations, filter data to remove unnecessary items, and also assign a label to the data. The FDs, as micro datacenters (MDCs), provide the resources for application modules. An optimal allocation policy of processing elements (PEs) to application modules will increase the productivity of resources inside the MDC. In FC, scheduling is assigning resources to modules in a specified order. As there are many resources and modules in a fog architecture, scheduling is a serious challenge. The resource allocation is based on a number of QoS parameters. Knapsack is an optimization problem in two ways, which includes the arbitrary and 0-1. Knapsack 0-1 is more important. It offers many practical applications such as capital budgeting, project selection, resource allocation, cutting stock, investment decision-making. The knapsack algorithm uses a simple sort with high execution speed. Thus, we used it to reduce the time delay in optimal allocation of PEs to modules in the fog network.

Our scheduling strategy was simulated by greedy knapsack algorithms in the fog network. It is based on the collection of data related to VR game response, data transmission to the edge of the network, processing, and sending them to cloud. Our key contributions in this paper are as follows. We formulate the resource scheduling problem in FC using a knapsack-based method. This method causes fast processing of the proposed algorithm. As knapsack-based scheduling after module placement in FDs then the best FDs are allocated to the sensor requests. VR gaming application using EEG (Electroencephalography) beam tractor was analyzed by some parameters as the number of executed modules in FDs, transmission time, and resource management interval.

The proposed method is compared with other algorithms based on different configurations by areas, departments, cameras, mobiles, and the number of users. The simulation results show that the quality of GKS as the energy consumption, delay, network usage, and total execution cost is better than first-come-first-served (FCFS), concurrent, and delay-priority algorithms.

1.5 CONTRIBUTION TO THE PROJECT

Fog computing is becoming more and more practical nowadays. Our contribution is creating various algorithms in the existing fog computing infrastructure. We defined a specific application based on real time applications and divided its processing into seperate modules and mapped them to fog modules. Then we implemented several algorithms. We compared each algorithm to find the optimal algorithm that produces low latency for the same fog structure and parameters.

1.6 OVERVIEW OF THE REPORT

i. Chapter 2 deals with the Fog Computing and its architecture
ii. Chapter 3 deals with the Low latency scheduler
iii. Chapter 4 deals with the Fog parameters
iv. Chapter 5 deals with the First Come First Server Algorithm
v. Chapter 6 deals with the Priority Scheduling Algorithm
vi. Chapter 7 deals with the Priority Scheduling based on Threshold
vii. Chapter 8 deals with the Knapsack Algorithm
viii. Results and Discussion are given in Chapter 9
ix. Conclusion in given in Chapter 10

CHAPTER 2

FOG COMPUTING ARCHITECTURE

2.1 INTRODUCTION

To gain maximum economic profit, proper resource allocation is necessary. Resource allocation improves the overall efficiency of the system with the customer satisfaction. As mentioned before, server virtualization is the core part of resource allocation and it improves the overall response time and the cost of the system. The architecture is based on Cloud computing environment with Fog computing technology as can be seen. It is good to note here that Fog layer is not used as replacement of the Cloud environment. But instead Fog layer helps in reducing the draw backs of the Cloud layer, by providing with low latency and geographical distribution. After analyzing the different existing algorithms for resource allocation, we have come up with the proposed framework of the solution.

2.2 ARCHITECTURE OF FOG COMPUTING

Fog architecture involves the distribution of functions at different layers, the types and the number of protocols used, and the constraints imposed at various layers.

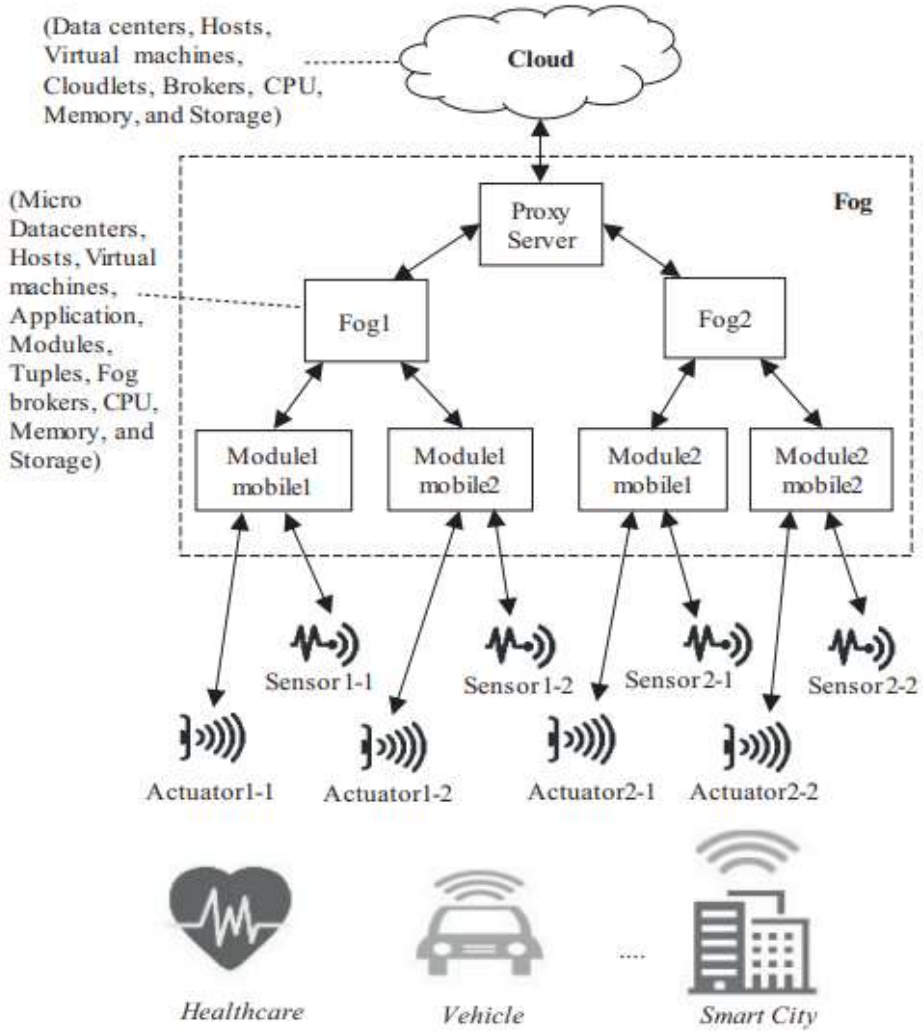

Figure 2.1: Fog network architecture

The hierarchical fog architecture comprises of following three layers:

2.2.1 TERMINAL LAYER

The terminal layer is the basic layer in fog architecture, this layer includes devices like mobile phones, sensors, smart vehicles, readers, smart cards, etc. The devices which can sense and capture data are present in this layer. Devices are distributed across a number of locations separated far apart from each other.

The layer mostly deals with data sensing and capturing. Devices from different platforms and different architectures are mainly found in this layer. The devices have the property of working in a heterogeneous environment, with other devices from separate technologies and separate modes of communication.

2.2.2 FOG LAYER

Fog layer includes devices like routers, gateways, access points, base stations, specific fog servers, etc., called as Fog nodes. Fog nodes are located at the edge of a network. An edge can be a hop distance from the end device. The Fog nodes are situated in-between end devices and cloud data centers.

Fog nodes can be static, e.g., located in a bus terminal or coffee shop, or they can be moving, e.g., fitted inside in a moving vehicle. Fog nodes ensure services to the end devices. Fog nodes can compute, transfer and store the data temporarily. Fog nodes and cloud data center connections are enabled by the IP core networks, providing interaction and cooperation with the cloud for enhancing processing and storage capabilities.

2.2.3 CLOUD LAYER

This layer consists of devices that can provide large storage and machines (servers) with high performance. This layer performs computation analysis and stores data permanently, for back-up and permanent access to the users. This layer has high storage and powerful computing capabilities. Enormous data centers with high computing abilities form a cloud layer.

The data centers provide all the basic characteristics of cloud computing to the users. The data centers are both scalable and provide compute resources on-demand basis. The cloud layer lies at the extreme end of the overall fog

architecture. It acts as a back-up as well as provides permanent storage for data in a fog architecture. Usually, data that isn't required at the user proximity is stored in a cloud layer.

2.3 ARCHITECTURE DESIGNED USING GUI

The topology created in iFogSim to evaluate the fog scenario results. We created three fog nodes in this topology and three cameras were attached to each fog node. This topology was created to evaluate the latency and network usage in iFogSim.This can be achieved either by using the GUI or programmatically using iFogSim classes.

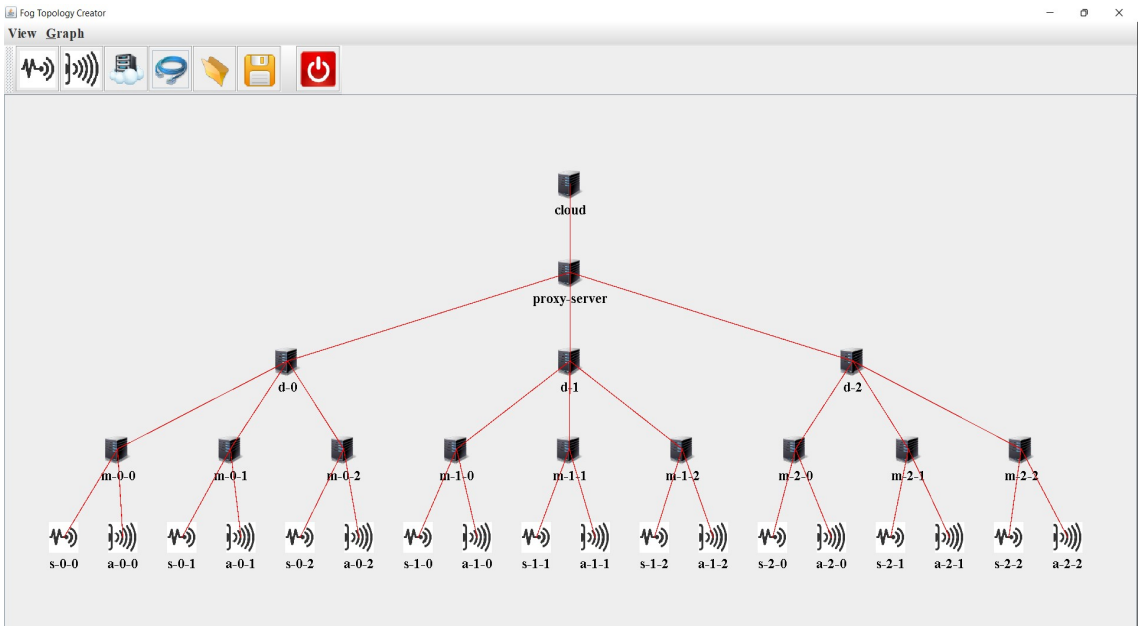

Figure 2.2: Topology of 9 sensors using 3 fog nodes

CHAPTER 3

LOW LATENCY SCHEDULER

3.1 SYSTEM MODEL

FC architecture has a hierarchical arrangement of sensor nodes, edge devices, and cloud. Sensors are located in the lower geographic location at the bottom of the architecture and send the collected data to the up-level by gateways. The actuators at the bottom of the fog architecture control the environment or change it. The fog network process includes sensing and sending data, processing in the FDs, dividing an application into several modules, and allocating resources to them for execution. Applications are used to collect and store data in micro datacenters as well as future analyses and processes. The received data are processed in FDs or send to cloud. The FD and application properties are explained as follows. An FD is an MDC that analyzes, filters, and stores the received data from the sensors.

The FD's properties include million instruction per second (MIPS), RAM, up bandwidth, down bandwidth, the level number in the topology, rate per MIPS, power in the busy state, and idle power. Each FD includes hosts as {Host1, Host2, ..., Hostn}. A host's properties include RAM, bandwidth, storage, and processing elements. In a host, $FB_{Lower} \leq HBi \leq FB_{Upper}$, where FB is fog's bandwidth, HB is host's bandwidth before FBLower is the lower bandwidth and FB_{Upper} is the upper bandwidth of each FD. HB_i is the bandwidth of ith host. N is the number of hosts. The total bandwidth of all hosts in each FD is between FB_{Lower} and FB_{Upper}. In FD, processing elements of hosts are allocated to application modules and execute them. The most important feature of a PE is MIPS. These values are set at the start of simulation for all FDs.

After allocation of a PE to an application module, the total allocated MIPS of all PEs is updated. Thus, we have

$$TAM = FB_{Lower} \leq \sum_{i=1}^{N} HBi \leq FB_{Upper}$$

where TAM is the total allocated MIPS of a FD that is less than or equal to MIPS of that FD ($TAM \leq FD\ MIPS$) and PEM is the PE's MIPS. N is the number of hosts in the FD, M is the number of PEs in a host, and PEMij is the MIPS of jth PE in ith host. The application module is a type of VM. The module's properties include MIPS, size, bandwidth, and the number of PEs. The number of modules in each FD is more than the number of PEs. In fact,

$$\sum_{i=1}^{C} Module_i > \sum_{j=1}^{K} FD_j$$

where C is the total number of modules and K is the total number of FDs.

3.2 APPLICATION MODEL

User applications that access the public cloud do so through an access point that allows data exchange through the core network to reach the cloud data center. With the introduction of computing capacity at the edge of the network, these access points can be extended to provide computing and storage services: the cloudlets. The cloudlets concept within the hierarchical infrastructure of the fog. This fog computing architecture presents a hierarchical, bi-directional computing infrastructure edge devices communicate with cloudlets and cloudlets communicate with clouds.

Cloudlets can also communicate with each other to perform data and process management in order to support application requirements, and to exchange fog control/management data (such as user device and application state). In fog computing, processing and storage capacity is one hop away from

the data production/consumption, which can benefit different types of applications.

Applications with low latency requirements, such as pedestrian and traffic security, surveillance, applications for vision, hearing, or mobility impaired users, online gaming, augmented reality, and tactile computing can benefit from lower latencies because of a single hop connection to a cloudlet. Applications that currently rely on the cloud can also benefit from lower delays and response times when adopting a fog-based deployment if their data and processing is carried out by a nearby cloudlet. This can also reduce data traffic to the cloud. Raw data collected by many devices often does not need to be transferred to the cloud for long term storage: data can be processed, filtered, or aggregated to extract knowledge and produce reduced data sets, which in turn are to be stored; or it can be processed and utilized right-away to other edge devices in the so-called sensor/actuator loop. In both cases, the fog computing paradigm can reduce network traffic from the edge to data centers.

Cloudlets can provide reduced latencies and help in avoiding/reducing traffic congestion in the network core. However, this comes at a price: more complex and sophisticated resource management and scheduling mechanisms are needed. This raises new challenges to be overcome, e.g., dynamically deciding what, when, and where (device/fog/cloud) to carry out processing of requests to meet their quality of service requirements. Furthermore, with smart and wearable devices, such mechanisms must incorporate mobility of data sources and sinks in the fog. Traditional resource management and scheduling models for distributed systems do not consider mobility and timeliness of data production and consumption in the resource management and allocation

process. Fog computing scheduling must bring users location to the resource allocation policies to uphold the benefits of fog computing proximity to the user.

CHAPTER 4

FOG PARAMETERS

4.1 LATENCY(ms)

Latency is the necessary factor to be reduced in environments demanding high performance in real-time. A key benefits of the fog computing is that it evades frequent accesses to the cloud and performs computations at the edge of network to offer quick response back to the client device thereby minimizing the latency. The latency is calculated by Cloudsim clock and the tuple's end time.

$$\text{TupleEndTime} = \{CC - T1 \text{ if } A = \text{False}, T1 * C1 + (CC - T1)C1 + 1 \text{ if } A = \text{True}$$

The tuple's end time is when the tuple is completed. Here, T1 is the tuple start time, T2 is the tuple type to average CPU time, CC is the Cloudsim clock, (CC − T1) is the execution time, and C1 is the number of executed tuple type. A is the condition, so if T2 is calculated then A is true else A is false.

The application loop delay Delay = CC − ET where CC is the Cloudsim clock and ET is the emitting time of a tuple. ET is calculated by sending time of a module to another module.

$$\text{TupleReceiptTime} = T1 * C2 + \text{Delay}\, C1 + 1$$

4.2 ENERGY COSUMPTION(MJ)

The energy consumption is calculated for the full topology of the network with

$$\text{Energy} = \text{CEC} + (\text{NT} - \text{LUUT}) * \text{HP}$$

The energy consumption of FD is calculated by the power of all hosts in a certain time frame of execution,where CEC is the current energy consumption, NT is the current time, LUUT is the last utilization update time, and HP is the host power in LU.

A great deal of attention has been paid to the energy consumption of Cloud services and data centers in an endeavor to reduce the energy consumption and carbon footprint of the ICT industry. Since the data in Cloud services is processed and stored in data centers, an obvious focus for studying energy consumption of Cloud services is the data centers. However, the energy consumption of a Cloud service is not just due to data centers, it also includes energy consumption of the transport network that connects end-users to the Cloud and the energy consumption of end-user devices when accessing the Cloud. In most of previous studies on energy consumption of Cloud computing services, the energy consumed in the transport network and end-user devices has not taken into account. To show the importance of energy consumption of these ignored parts, the total energy consumed by three well-known Cloud applications, Facebook, Google Drive and Microsoft OneDrive, is studied using measurements and modeling. The results show that achieving an energy-efficient Cloud service requires improving the energy efficiency of the transport network and the end-user devices along with the related data centers.

4.3 TOTAL EXECUTION COST(dollars if i/p cost = 0.01 dollars)

$$\text{Cost} = \sum_{i=1}^{F} [\text{T C} + (\text{CC} - \text{LUUT}) * \text{RPM} * \text{LU} * \text{TM}]$$

To calculate the execution cost, the total MIPS of hosts is calculated by time frame. The time frame is different between the current time of simulation and the last utilization time. In the equation, F is the number of FDs, TC is the execution cost, CC is the CloudSim clock or current time of simulation, LUUT is the last utilization update time, RPM is the Rate Per MIPS that is different for each inter module edges, and TM is the total MIPS of the host. Also, LU is the last utilization (LU) that is calculated as LU = M in(1, TMA/TM), where TMA is the total allocated MIPS of the host.

4.4 NETWORK USAGE(kb)

When the traffic increase on cloud server then, only the cloud resources are used. Increase of traffic on the cloud server results in increased network usage. Consequently, the data rates on the network decrease due to the increased traffic. For geographically distributed servers, one fog node is dedicated for one geographical area to deal with the request of that area. Consequently, the network usage in that case decreases and the transmission rate for the rest of the traffic increases.

$$\textit{Network usage} = \textit{Latency} * \textit{tupleNWSize}$$

CHAPTER 5

FIRST COME FIRST SERVE ALGORITHM

5.1 INTRODUCTION

First come first serve (FCFS) scheduling algorithm simply schedules the jobs according to their arrival time. The job which comes first in the ready queue will get the resource first for execution. The lesser the arrival time of the job, the sooner will the job get the resource. FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs. In our fog architecture we try to process the tasks one by one in the fog, when to find that the resource in fog is not sufficient enough to process a task that task is sent to cloud for processing.

5.2 FLOW CHART

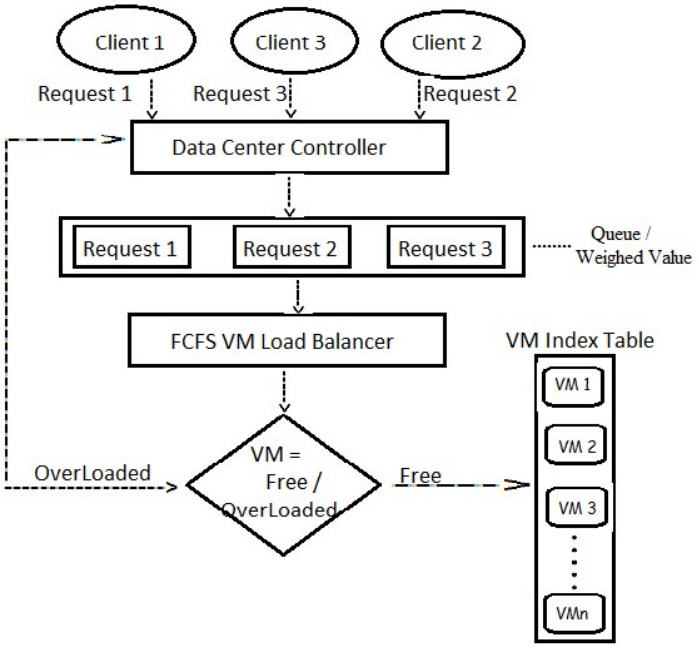

Figure 5.1: FCFS flowchart

CHAPTER 6

PRIORITY SCHEDULING ALGORITHM

6.1 INTRODUCTION

The time and order of the resources being assigned is crucial to get the maximum benefits of using a virtual server as highest throughput of the system can be obtained and the clients will not be charged exorbitantly. The availability of resources should be such that the high priority tasks are not kept waiting till the very end of the task queue. This could lead to sub optimal use of the virtual servers and possible loss of business.

Hence, assigning resources in a prioritized fashion to gain maximum profit is of great importance and a good area of research. This paper proposes a system where Fog layer is used in-between client and Cloud layer to cater to the applications which have low tolerance for latency. A task is said to have higher priority when the deadline of the task to be processed is closest than any other task. The higher priority tasks are processed first in the Fog layer. If all the data centers in the Fog layer are busy, then the high priority task is propagated to the Cloud layer for timely execution. The data centers in the Fog layer in a region can communicate with each other to check availability of servers and for load balancing.

Let's take a real world multiplayer gaming application as an example here. In gaming there will be two states of a player, one is idle state where the player waits in the game lobby idly and the other will be active player state where the player makes moves against their opponent.

In such cases priority to use the resources of the fog device is higher for the player who is in the active state and gets low latency services while the idle player doesn't care about how much responsive the game is.

6.2 FLOW CHART

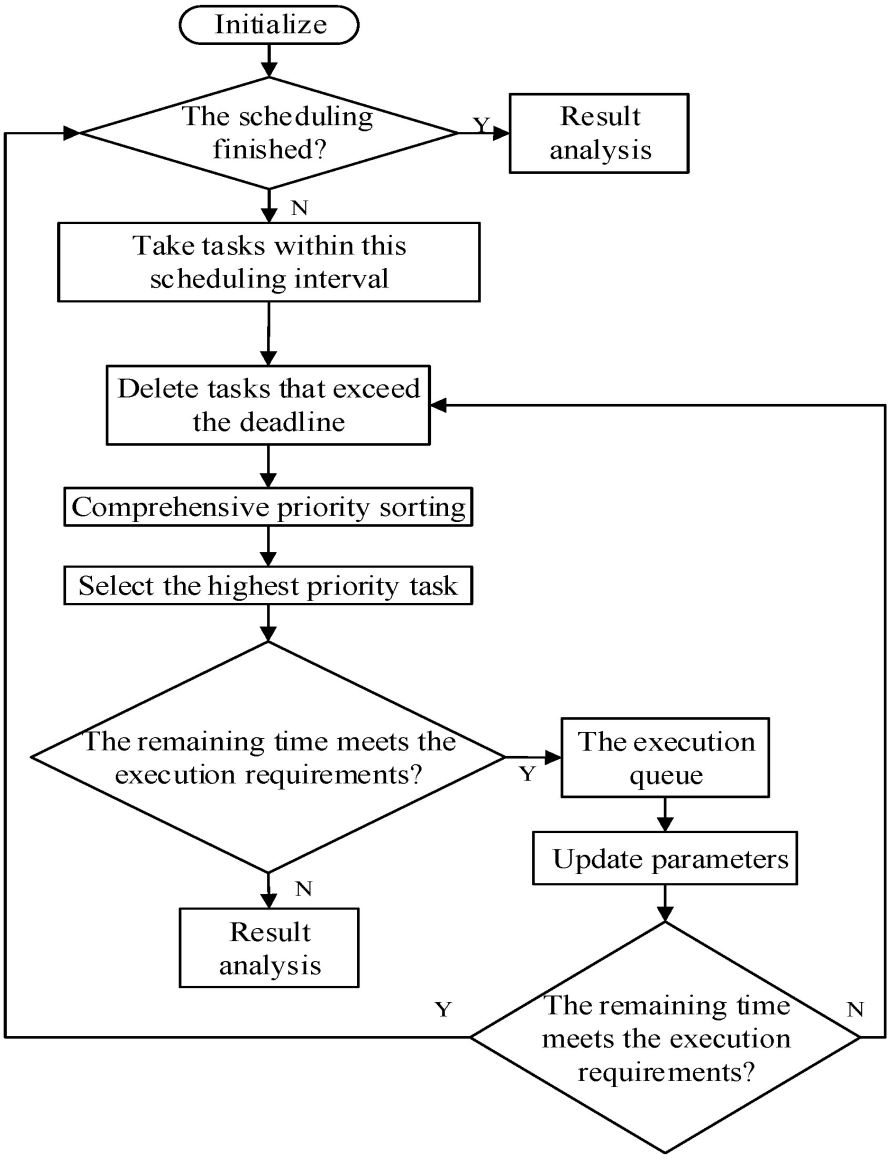

Figure 6.1: Priority scheduling flowchart

CHAPTER 7

PRIORITY SCHEDULING BASED ON THRESHOLD

7.1 INTRODUCTION

The algorithm proposed takes in threshold value as a parameter and based on the condition provided, the task gets scheduled in a prioritized manner. Let us say, we have taken the values below the threshold value as high priority task, then those task gets executed as top priority. The rest of the tasks which have their values above the threshold value gets executed as low priority task. Sorting of task is done based on the threshold priority.

This algorithm can be applied in VR games wherein location of a player can be taken as a parameter for threshold value, players in the lobby can be given low priority and the players in the battlefield can be given high priority. Hence there will be a seamless flow of data between the user and server.

7.2 FLOW CHART

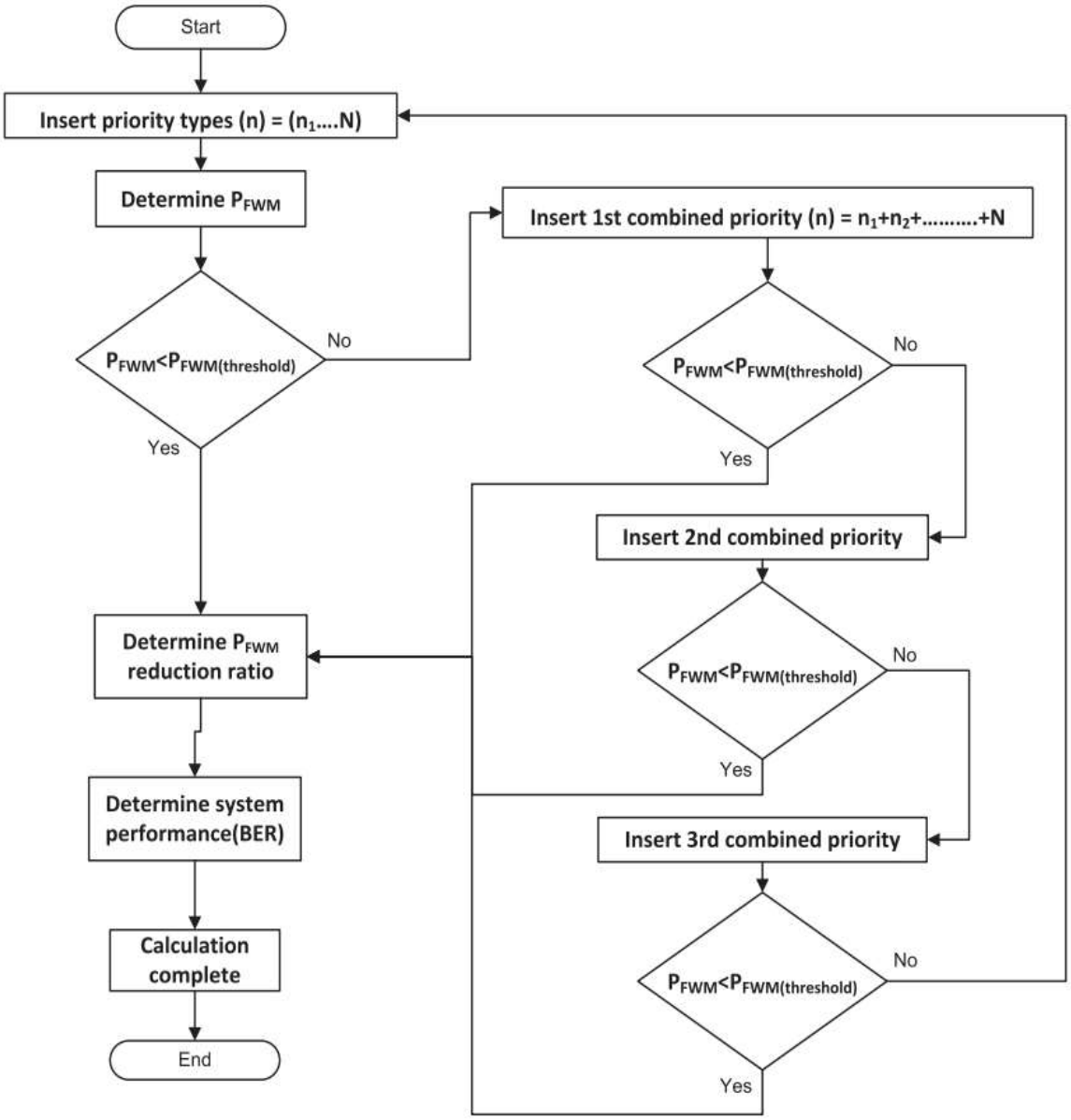

Figure 7.1: Flowchart of priority scheduling based oh threshold

CHAPTER 8

KNAPSACK ALGORITHM

8.1 INTRODUCTION

The knapsack problem is a problem in combinatorial optimization. Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. The problem often arises in resource allocation where the decision-makers have to choose from a set of non-divisible projects or tasks under a fixed budget or time constraint, respectively.

Our proposed algorithm allocates processing elements in the hosts to best modules. We updated the original application scheduling policy by changing the update Allocated MIPS method in Fog Device class in iFogsim with the Knapsack algorithm. The profit of our problem is as follows. Profiti = k1 □ TUCi + k2 □ BWi.., where TUCi is the total utilization of CPU for allocated PEs to application modules and BWi is the bandwidth of the application module. We complete this equation for knapsack problem The modules are entered into the system. Then the modules will be placed in the knapsack. FD include many processing elements. These processing elements are allocated to modules based on the knapsack algorithm. According to iFogsim, if a module fails to fit into the array of modules in the knapsack then it will be transmitted to another FD. Thus, the wait time of the modules for processing elements is optimized.

8.2 FLOW CHART

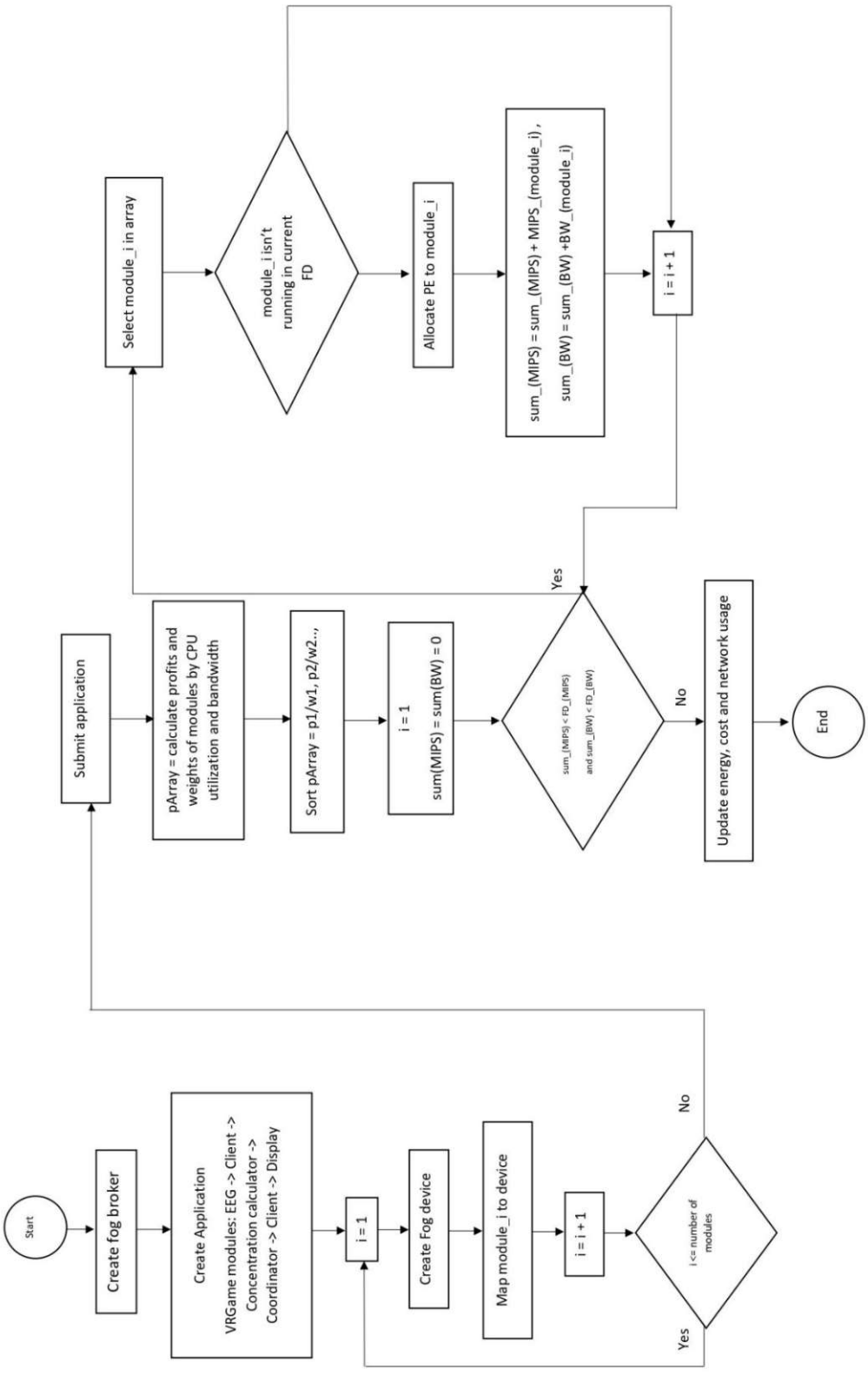

Figure 8.1: Flowchart for Knapsack algorithm

CHAPTER 9

RESULTS AND DISCUSSION

9.1 RESULTS

9.1.1 FCFS

S.No	No. of sensors	Latency (ms)	Cost of exe.(dollars if i/p = 0.01dollars)	Network usage (kb)
1	1	17.726	6912.8	2348
2	2	19.107	6912.8	4029.5
3	4	34.31	7899.2	8508.5
4	5	268.62	7048.8	13124.5
5	8	226.593	805371	152130
6	10	227.145	814198.8	193400
7	16	715.673	812475.8	303034
8	32	994.668	821847.8	584451.5
9	64	1062.97	826497.2	1154480

Table 9.1: Output values of FCFS executed using edgewards placement

S.No	No. of sensors	Latency (ms)	Cost of exe.(dollars if i/p = 0.01dollars)	Network usage (kb)
1	1	224.782	38782	18630
2	2	222.203	463468	37105
3	4	226.422	690655	73342
4	5	226.54	808896.1	95440.5
5	8	226.598	808452	153452
6	10	227.118	812138	195531
7	16	715.184	810706.2	303849.5
8	32	995.018	820895	586906
9	64	1062.565	826497.2	1154481

Table 9.2: Output values of FCFS executed using cloud

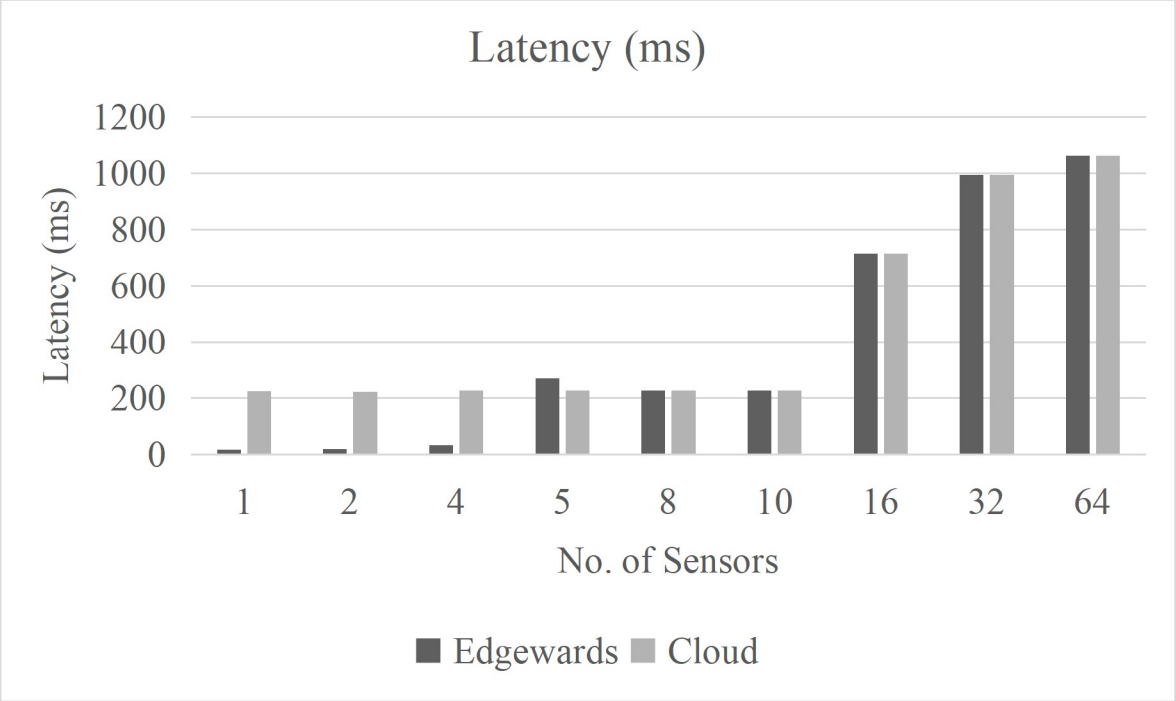

Figure 9.1: Latency output of FCFS algorithm

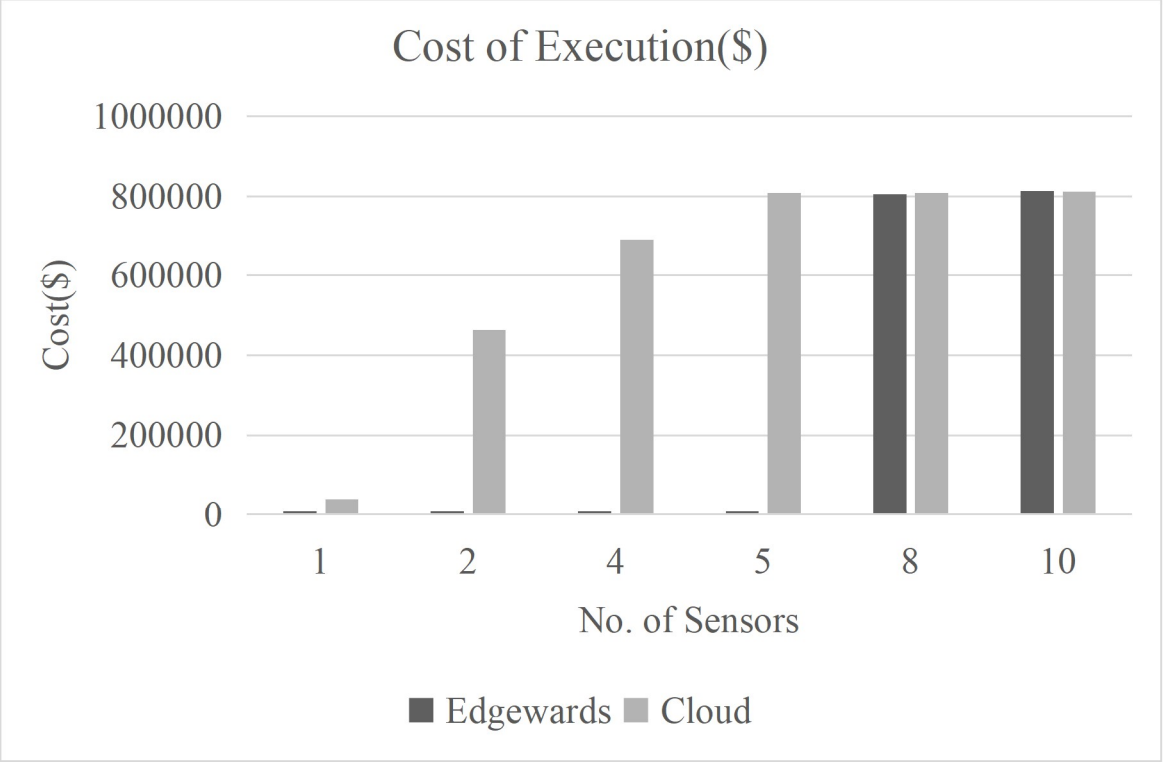

Figure 9.2: Cost of execution output of FCFS algorithm

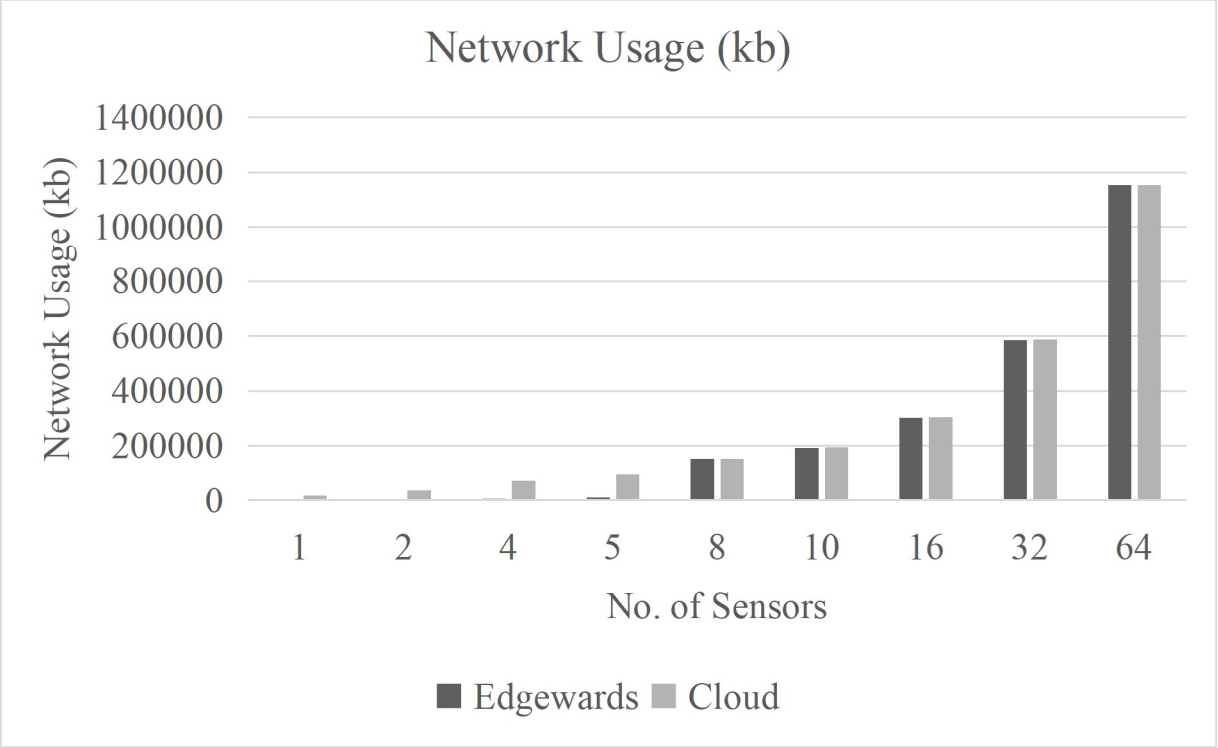

Figure 9.3: Network usage output of FCFS algorithm

9.1.2 PRIORITY SCHEDULING ALGORITHM

SENSORS	BEFORE PRIORITIZATION LATENCY	AFTER PRIORITIZATION LATENCY
m-0-0	17.7	17.7
m-0-1	35.4	176.9
m-0-2	53.09	106.2
m-0-3	70.8	53.09
m-0-4	88.8	141.6
m-1-0	106.2	35.4

m-1-1	123.9	70.8
m-1-2	141.6	88.5
m-1-3	159.29	123.9
m-1-4	176.9	159.29

Table 9.3: Output values of priority scheduling algorithm

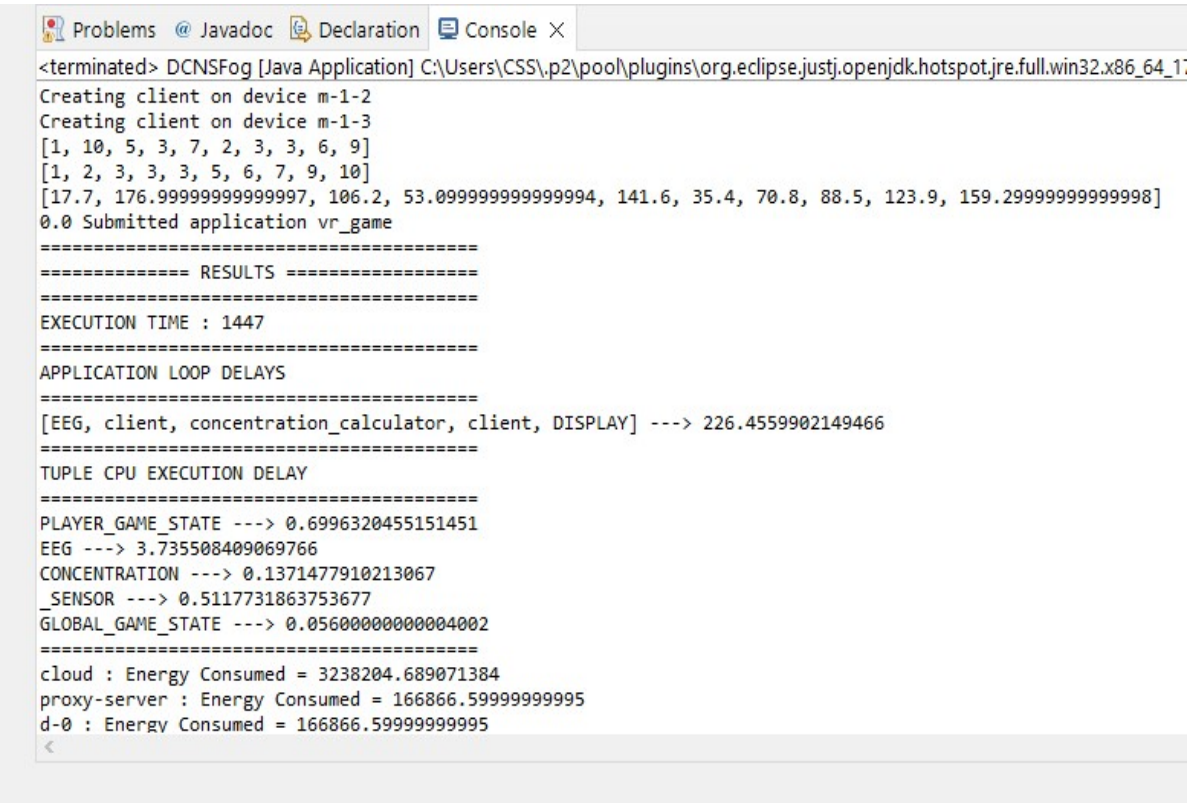

Figure 9.4: Console output of priority scheduling

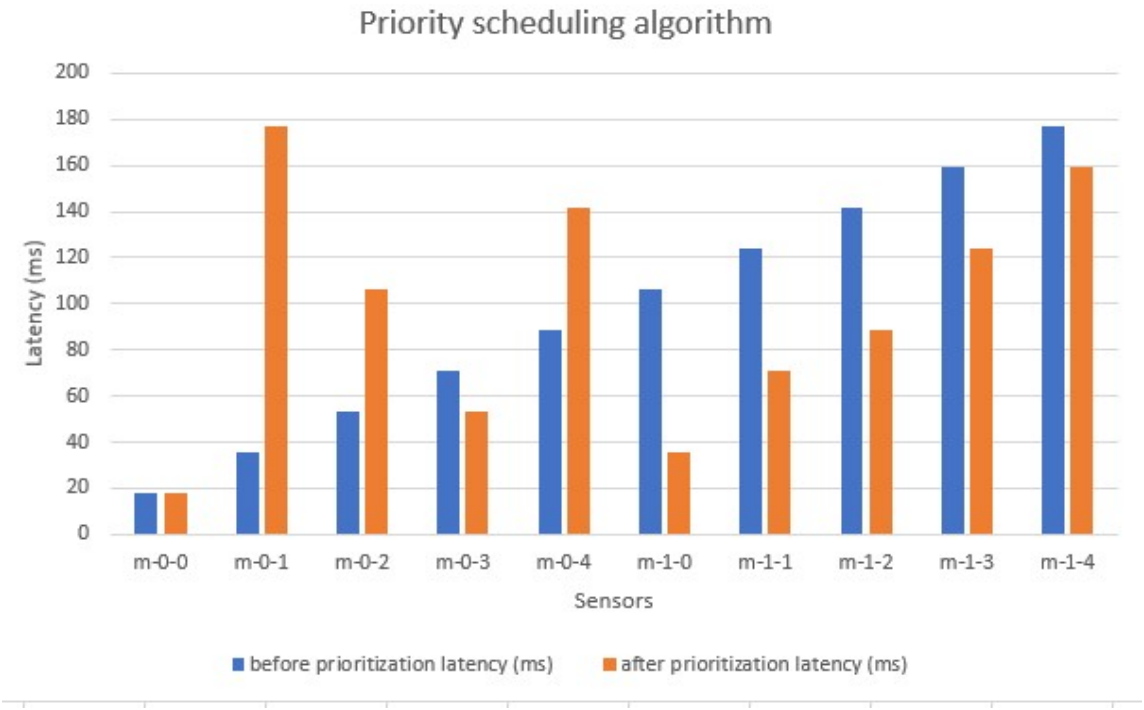

Figure 9.5: Comparison of Latency between non prioritized and prioritized scheduling

9.1.3 PRIORITY SCHEDULING BASED ON THRESHOLD

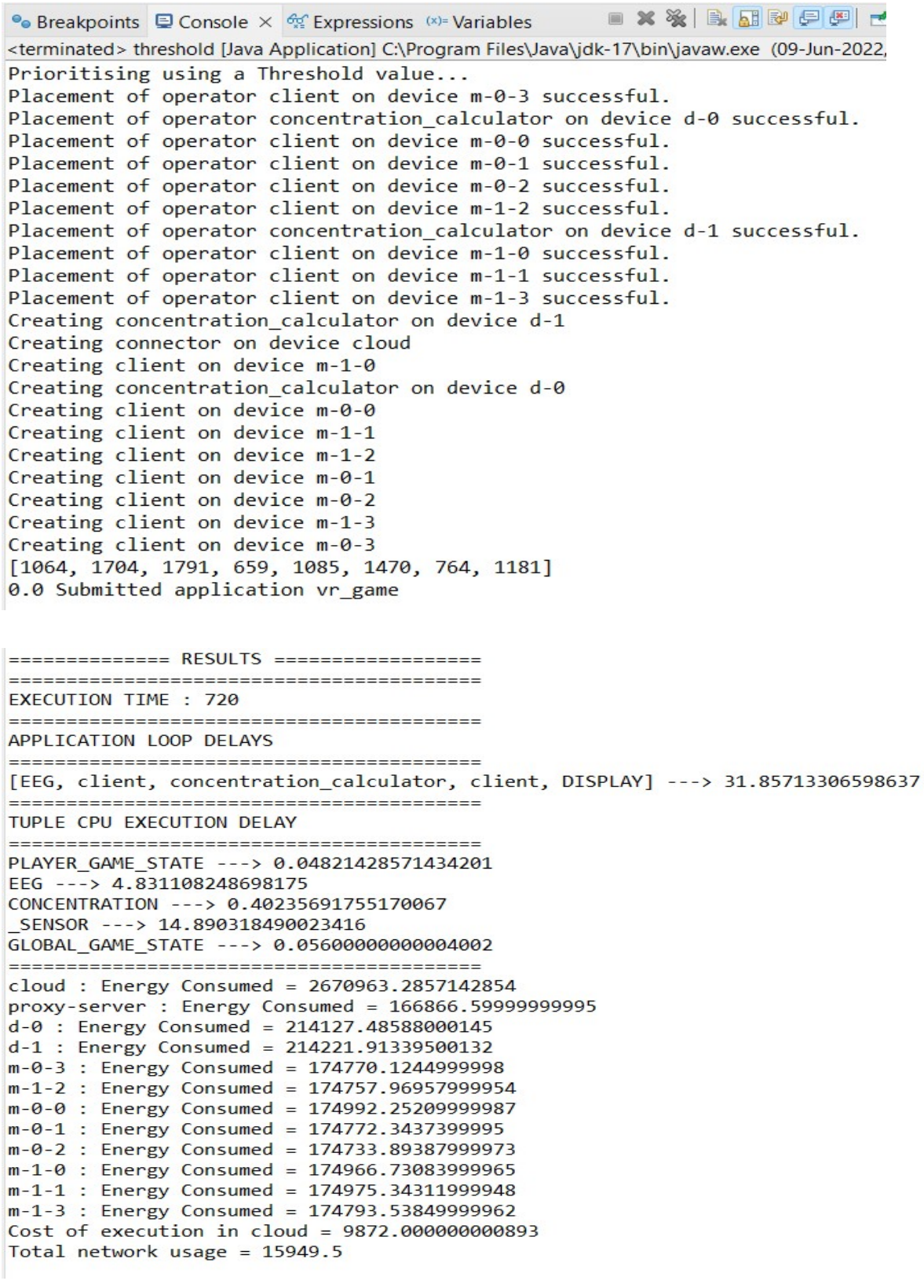

Figure 9.6: Console output of Priority scheduling algorithm based on threshold

9.1.4 KNAPSACK ALGORITHM

Figure 9.7: Console output of Knapsack algorithm

9.2 COMPARISION

9.2.1 FCFS VS KNAPSACK

S.No	No. of sensors	Latency (ms)	Network usage (kb)	Cost of exe.(dollars if i/p = 0.01dollars)
1	1	14.007	6490	3655604
2	2	16.158	13014	213296
3	3	13.606	27853	13692672
4	8	223.37	111068	2531118
5	9	221.55	125144	17113035
6	10	223.86	139241	2819097

Table 9.4: Output values of FCFS for test application

S.No	No. of sensors	Latency (ms)	Network usage (kb)	Cost of exe.(dollars if i/p = 0.01dollars)
1	1	9.53	359.52	109445.3
2	2	11.44	718.92	205428.3
3	3	21.59	1779.84	140859.1
4	8	41.16	5320.8	1454219

5	9	44.23	6026.4	1652410
6	10	47.58	7192.8	4435250

Table 9.5: Output values of Knapsack algorithm for test application

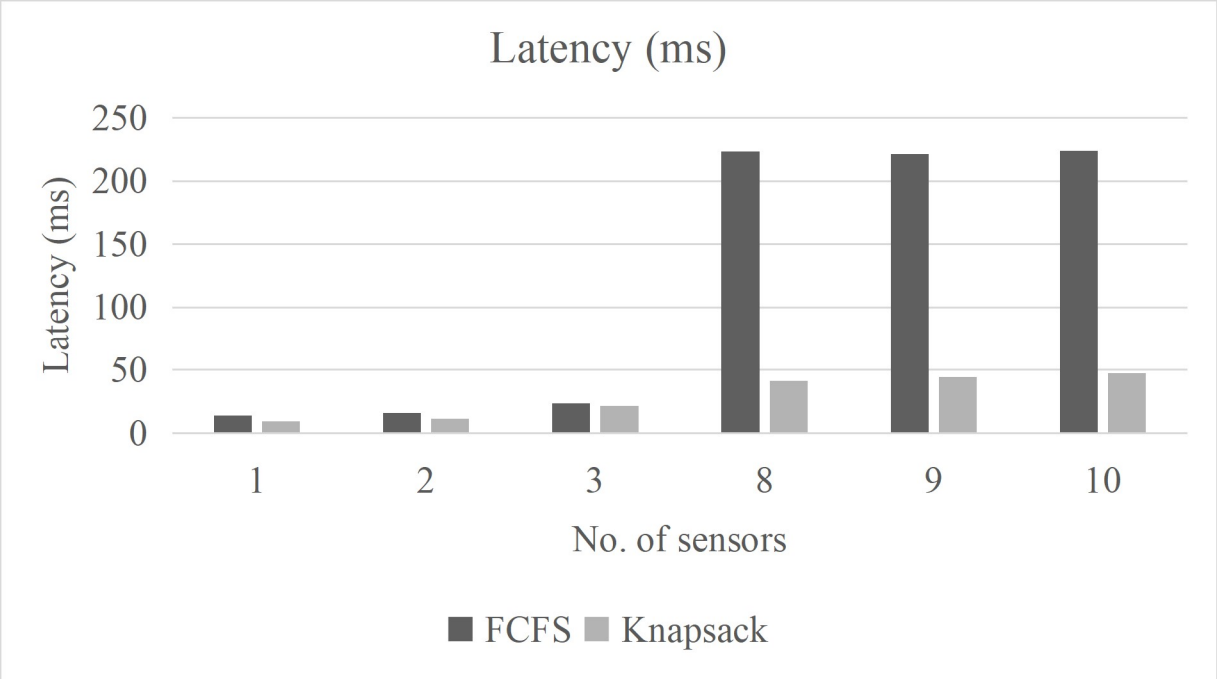

Figure 9.8: Latency Comparison between FCFS and Knapsack

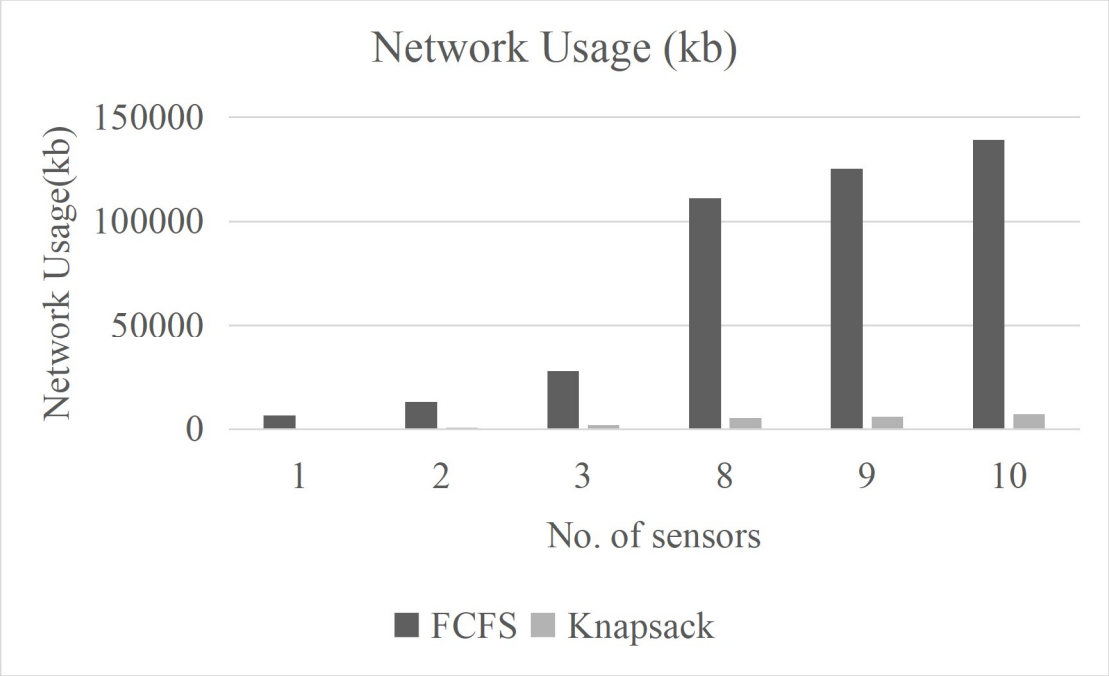

Figure 9.9: Network Usage Comparison between FCFS and Knapsack

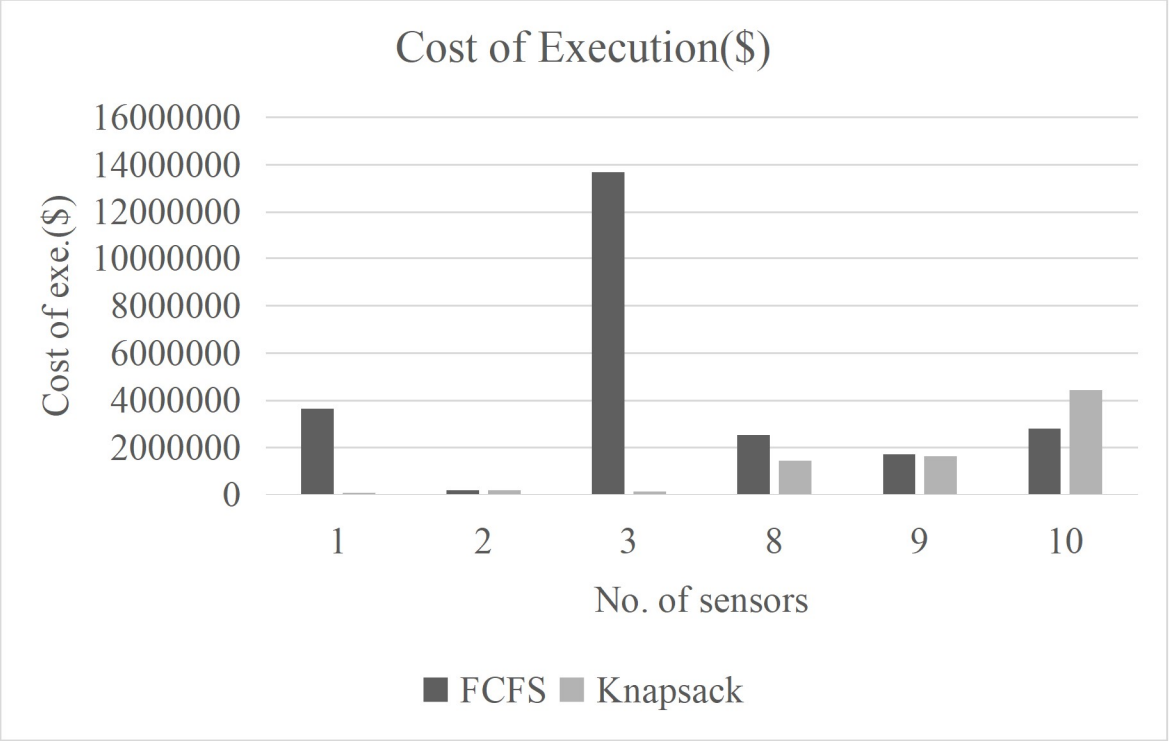

Figure 9.10: Cost of Execution Comparison between FCFS and Knapsack

9.3 DISCUSSION

From the results we obtained, we can infer that FCFS algorithm has some disadvantages like long waiting time, important tasks are not given proper priority,etc.., FCFS algorithm can be particularly troublesome for time-sharing systems. For time-sharing systems, it is essential that each task gets a share of the resource at regular intervals. Because FCFS is a non-preemptive system, this is not always feasible. Again, it could leave dependent systems idle and waiting, which is not ideal for keeping operations running smoothly and efficiently. Some of these issues got fixed in priority based scheduling algorithm. One of the main issue priority based algorithm tried to address the execution of tasks based on its importance. Still there are few issues in priority based scheduling algorithm.

Some of the disadvantages of priority based scheduling algorithm are, if high priority processes take lots of execution time, then the lower priority processes may starve and will be postponed for an indefinite time, another problem is deciding which process gets which priority level assigned to it. These problems got solved in knapsack algorithm where we take advantage of all the available resources and try to execute the tasks. As knapsack algorithm is a combinatorial optimization algorithm it is not limited to take a specific resource and execute the task, whereas it looks for all the sufficient available resources and execute the tasks there. From the output we obtained we can clearly infer that there is a drop in the latency, network usage and cost of execution. Thus the intended goal has been successfully achieved.

From the results, it was found out that some algorithms perform better for execution time and some perform better cost-wise. Depending on the requirements of the system, one can choose priority scheduling for quicker execution or one can choose Knapsack algorithm for low latency.

CHAPTER 10

CONCLUSION

10.1 CONCLUSION

Many IoT devices process lot of data and requests need to be handled with as less delay as possible. Fog layer takes care of this as it is placed between client and the cloud layer. Fog layer aids in processing the tasks which are deployed by the client by providing with the responses in less time. It brings the cloud environment physically closer to the client. The requests can have different levels of priorities and the higher priority requests should be processed before other lower priority tasks.

Fog computing provides lower communication latency and computing capacity closer to the final user. For this infrastructure to become efficient and offer actual differentiated service from the cloud computing paradigm, proper resource management mechanisms must be deployed. We show that scheduling strategies can be designed to cope with different application classes according to the demand coming from users, taking advantage of both the fog proximity to the end user and the cloud computing elastic characteristic. Thus using knapsack algorithm, we have reduced latency by 3.6 times and network usage by 92% when compared with FCFS algorithm.

10.2 FUTURE DIRECTIONS

Application classification must provide the scheduler with information about application requirements, which will allow the scheduler to prioritize the cloudlet use and optimize other objectives (e.g., reduce network use, reduce cloud costs). With that information, a fog scheduler can decide which

application should run in the cloudlet and which should run in the cloud. Moreover, application classes could also allow a system-level scheduler to prioritize applications within a cloudlet, allowing smaller granularity control over the delays observed by applications at each class.

Understanding users' behavior and mobility patterns can improve resource management by better planning the applications scheduling beforehand. This planning is crucial to avoid application delays during user movement.

Application execution costs in a fog utility model are also interesting areas to explore. Given a business model for the cloudlets (how service levels agreements are offered – how cloudlets are commercialized and charged), schedulers should take into account a trade-off between costs and application quality of service. Scheduling algorithms for hybrid clouds could be extended to consider the fog computing hierarchy. Moreover, costs and delays of both storage data transfers from/to cloud providers can also take part in the trade-off.The combination of advanced scheduling techniques, supported by applications classification and mobility prediction, with virtualization tools within an autonomic computing framework, such as CometCloud,15 would be able to handle the dynamic mobile environment of a fog infrastructure and its clients.

REFERENCES

1. Mathew T, Sekaran KC, Jose J. Study and analysis of various task scheduling algorithms in the cloud computing environment. In: International Conference on Advances in Computing, Communications and Informatics (ICACCI); New Delhi, India: IEEE; 2014. pp. 658-664.

2. Lv J, Wang X, Huang M, Cheng H, Li F. Solving 0-1 knapsack problem by greedy degree and expectation efficiency. Applied Soft Computing 2016; 41: 94-103. doi: 10.1016/j.asoc.2015.11.045.

3. Bittencourt LF, Diaz-Montes J, Buyya R, Rana OF, Parashar M.Mobility-aware application scheduling in fog computing. IEEE Cloud Computing 2017; 4(2): 26-35. doi: 10.1109/MCC.2017.27.

4. Lao F, Zhang X, Guo Z. Parallelizing video transcoding using map-reduce-based cloud computing. In: IEEE International Symposium on Circuits and Systems (ISCAS); Seoul, South Korea; 2012. pp. 2905 2908.

5. Rodriguez MA, Buyya R. A responsive knapsack-based algorithm for resource provisioning and scheduling of scientific workflows in clouds. In: IEEE 2015 44th International Conference on Parallel Processing (ICPP); Beijing, China; 2015. pp. 839-848.

6. F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, Fog Computing: *A Platform for Internet of Things and Analytics*. plus 0.5em minus 0.4em Springer International Publishing, 2014, pp. 169–186.

7. H. Shukur, S. Zeebaree, R. Zebari, D. Zeebaree, O. Ahmed, and A. Salih, "*Cloud Computing Virtualization of Resources Allocation for Distributed Systems*," Journal of Applied Science and Technology Trends, vol. 1, pp. 98-105, 2020/06/27/ 2020.

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

9. L. F. Bittencourt, O. Rana, and I. Petri, "*Cloud computing at the edges," in Cloud Computing and Services Science,* M. Helfert, V. Méndez Muñoz, and D. Ferguson, Eds. plus 0.5emminus 0.4emSpringer International Publishing, 2016, pp. 3–12.

10. B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management,*

vol. 23, no. 3, pp. 567–619, 2015.

11. S. S. Iyengar and R. R. Brooks, *Distributed sensor networks: sensor networking and applications*. plus 0.5em minus 0.4emCRC press, 2016.