MAJOR PROJECT REPORT

ON

"FUZZY BASED TASK OFFLOADING IN EDGE - CLOUD ENVIRONMENT"

In Complete fulfilment for the award of the degree of

BACHELOR IN TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

(2019-23)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PARALA MAHARAJA ENGINEERING COLLEGE

BERHAMPUR,ODISHA 761003 (2022-2023)

UNDER THE GUIDANCE OF

Mr.Kodanda dhar Naik

Submitted By

Sonali Mallick(1901109184)
Sonam Jagdev(1901109185)
Bommana Mythri Sri(2021109077)
Pamini Deep(2021109081)
Swaraj Gaurav Parhi(1901109197)

YEAR: 4th (8TH SEMESTER)

PARALA MAHARAJA ENGINEERING COLLEGE, BERHAMPUR

CERTIFICATE

This is to certify that the project report entitled" Fuzzy-based Task Offloading in Edge-Cloud Environment" submitted by BOMMANA MYTHRI SRI (2021109077), SONAM JAGDEV(1901109185), SONALI MALLICK(1901109184), PAMINI DEEP(2021109081), SWARAJ GOURAV PARHI(1901109197) in complete fulfillment of the requirements for the award of the Bachelor of Technology in the Department of Computer Science and Engineering at Parala Maharaja Engineering College, Berhampur is an authentic work carried out by them under my supervision and guidance.

6/6/23

(External(s) Examiner)

(Project Guide)

Mr. Kodanda Dhar Naik

Assistant Professor

6/6/23

(Head Of the Department)

Dr. Debasis Mohapatra

Computer Science and Engineering

ABSTRACT

It is anticipated that next-generation communication networks will support a significant number of novel and resource-hungry applications that can be made available to a wide variety of end users. Task offloading is now a problem since computation-intensive jobs must be transferred to remote devices with larger resources like Edge computing and cloud computing. Although Edge Computing is a promising enabler for latency-sensitive related issues, its deployment produces new challenges. Besides, different service architectures and offloading strategies have a different impact on the service time performance of IoT applications.

We are going to present a novel approach for task offloading in an Edge-Cloud system in order to minimize the overall service time and energy consumption for latency-sensitive applications. This approach adopts fuzzy logic algorithms, considering application characteristics (e.g., CPU demand, network demand, and delay sensitivity) as well as resource utilization and resource heterogeneity and boost quality of services (QOS). By the use of fuzzy logic, we can make sure where to offload the task i.e., either in the edge layer or in the cloud. Most studies and related offloading strategies are based on the assumption that mobile users are fully stationary when they are offloading tasks to edge servers. However, this is often not realistic in a world where edge users are keeping moving, which has a great impact on the success of task offloading and further affects the response time of mobile applications. So in our project, We will consider the mobility of mobile devices. Scheduling of assignment of tasks to respective layers in their trajectories predicted by a Random waypoint Model. We will conduct simulation experiments to study the performance of our proposed work.

KEYWORDS:- Edge–cloud computing, Task offloading, mobility-aware, Energy optimization, latency sensitivity

ACKNOWLEDGEMENT

we consider it as a great privilege to express my deep gratitude to many respected personalities who guided, inspired and helped me in the successful completion of our project.

we would like to express my deepest gratitude to our guide **Mr. Kodanda Dhar Naik, Assistant Professor, Department of Computer Science and Engineering, Parala Maharaja Engineering College, Berhampur,** for his constant supervision, guidance, suggestions and invaluable encouragement during this project. He has been a constant source of inspiration and helped us in each stage.

we are grateful to **Dr. Debasis Mohapatra, Head of the Department, Computer Science and Engineering, Parala Maharaja Engineering College, Berhampur** for his mortal support to carry out this project.

we are very thankful to the Project Evaluation Committee, for their kind cooperation and support given throughout the project work. Lastly, I would extend my gratitude to my family as well as my friends who have given valuable suggestions and constant support and help in all stages of development of the project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

With the advancement of technology, services such as mobile gaming, streaming, augmented and virtual reality, and image and video processing are becoming more popular[1-6]. Applications providing these services ought to be energy efficient and delay-sensitive to reduce battery usage and to provide good quality of service. These applications require a large number of computational resources. Limitation resources of IoT devices, such as memory and battery, prevent the practical usage of those applications[5]. Most of the prevailing solutions to tackle the resource limitations of these devices indulge offloading tasks to nearby clouds which provide resources on-demand, especially data storage and computing power. To alleviate these limitations and meet the communication/processing delay requirement, complex computations can be offloaded to more resourceful devices[3].

Cloud Computing is considered viable and promising technology to support this growth by enabling on-demand access to a massive pool of computation resources for services process and data analytics. Notwithstanding this, cloud computing resources are centralized and far away from IoT devices where the enormous amount of data generated by IoT devices is required to be transferred and processed in a real-time manner. Therefore, the cloud computing paradigm is not suitable for addressing low latency, real-time interaction and high Quality of Service (QoS) applications due to network delay.

To address the cloud computing limitations, an **Edge computing** paradigm has emerged where it provides a pool of virtually operated computational and storage capabilities at the edge of the network that are in proximity to IoT devices and thereby fulfill the latency gaps. In addition, it provides the opportunity to serve better streaming services, which are both latency-sensitive and bandwidth-intensive such as Google Stadia and Netflix. Moreover, edge computing architecture avoids uploading/downloading massive files and prevents pre-processing of offloading tasks, which contributes to minimizing the overall service time.

Conventional methods for making offloading decisions usually consider that user position is static and time-invariant. Such methods may not be appropriate in a real-world scenario since edge users are with high mobility. The optimal offloading solution becomes more challenging when considering user mobility. It has a large impact on task execution. Each user can have multiple edge servers in their communication range. Assigning tasks to directly connected edge servers may not lead to an optimal offloading decision. Users may

upload tasks to an edge server and get results via another server which involves communication among the edge servers and transmission cost. To effectively address the impact of user mobility on task execution and offloading decision problem, we consider resource allocation to mobile users based on their trajectories predicted by Random Waypoint Model. Mobility aware delay constrained energy minimal task offloading scheme is proposed by making use of cooperation between edge servers at base stations along the users' trajectories.

Random Waypoint (RWP) model is a commonly used synthetic model for mobility, e.g., in Ad Hoc Networks. In random-based mobility simulation models, the mobile nodes move randomly and freely without restrictions. To be more specific, the destination, speed, and direction are all chosen randomly and independently of other nodes.[7] It is an elementary model which describes the movement pattern of independent nodes by simple terms. This kind of model has been used in many simulation studies.

The movement of nodes is governed in the following manner: Each node begins by pausing for a fixed number of seconds. The node then selects a random destination in the simulation area and a random speed between 0 (excluded) and some maximum speed. The node moves to this destination and again pauses for a fixed period before another random location and speed. This behavior is repeated for the length of the simulation.

Briefly, in the RWP model:

- Each node moves along a zigzag line from one waypoint Pi to the next Pi+1.
- The waypoints are uniformly distributed over the given convex area, e.g., unit disk.
- At the start of each leg, a random velocity is drawn from the velocity distribution.
- (In the basic case the velocity is constant 1)
- Optionally, the nodes may have so-called "thinking times" when they reach each waypoint before continuing on the next leg, where durations are independent and identically distributed random variables.

Fuzzy Logic (FL) is a method of reasoning that seems closer to the way our brains work. The concept of fuzzy logic is to abstract the problem complexity to a level that can be understood. It helps to model the imprecision and uncertainty of the system, where it can define the imprecise information in a more logical and meaningful fashion. It can also handle system uncertainty by dealing with many input and output variables and can represent the problem with simple if-then rules. Many researchers in the field of distributed systems use fuzzy logic to deal with the challenges caused by vagueness, uncertainty, and the dynamicity of the environment. Therefore, Fuzzy Logic is considered to be among the most feasible solutions for a multi-objective optimization problem when the activity of multiple parameters is significant. It can be easily adapted to the dynamicity of computational

resources and application parameters as well as providing scalability within the context of the system. It also averts computational complexity and can provide decisions very quickly. As a consequence, fuzzy logic has been adopted in this research to determine where to offload the tasks based on application and system parameters.

The aim of our research is to develop an approach for offloading tasks to handle the requirements of latency-sensitive IoT applications and efficiently utilize the resources also minimizing the energy consumption in Edge-Cloud environments.we formally model the problem as an optimization and constraint satisfaction problem. We then proposed an approximate approach to obtain a near-optimal offloading decision satisfying the delay constraint. We conduct extensive simulation experiments and the results show that the proposed work can significantly reduce the energy consumption of tasks in Edge-Cloud Environment.

Chapter 2

RELATED WORK

Computation offloading is not a new paradigm; it is widely used in the area of Cloud Computing. Offloading transfers computations from the resource-limited mobile device to resource-rich Cloud nodes in order to improve the execution performance of mobile applications and holistic power efficiency. Users' devices are evenly located at the edge of the network. They could offload computation to Edge and Cloud nodes via Wireless Local Area Network (WLAN) or 4G/5G networks[3]. Generally, if a single edge node is insufficient to deal with the surging workloads, other edge nodes or cloud nodes are ready for assisting such an application. This is a practical solution to support IoT applications by transferring heavy computation tasks to powerful servers in the Edge-Cloud system. Also, it is used to overcome the limitations of IoT devices in terms of computation power (e.g., CPU and memory) and insufficient battery. It is one of the most important enabling techniques of IoT because it allows performing sophisticated computational tasks more than their capacity. Thus, the decisions of computational offloading in the context of IoT can be summarized as follows:

- First, whether the IoT device decides to offload a computational task or not. In this case, several factors could be considered, such as the required computational power and transferred data.
- Second, if there is a need for offloading, do partial offloading or full offloading[4]. Partial offloading refers to the part of the tasks that will be processed locally at the IoT device and other parts in the Edge-Cloud servers. Also, factors such as task dependency and task priority can be considered in this case. Full offloading means, the whole application will be processed remotely in the Edge-Cloud servers.

However, our paper aims at Mobility aware delay constrained energy minimization problem by optimizing the task offloading decision. The major contributions of this paper are:

- Introduce an Edge-Cloud system architecture that includes the required components to support scheduling offloading tasks of IoT applications.
- Propose a novel approach that adopts the fuzzy logic technique, which considers the application characteristics (e.g., CPU demand, network demand and delay sensitivity) as well as resource utilization and resource heterogeneity in order to minimize the overall time of latency-sensitive applications.

- Mobility aware delay constrained energy minimal task offloading scheme is proposed by making use of cooperation between edge nodes at base stations along the users' trajectories.
- Mobility of users in this paper refers to the random waypoint model which is widely used in many other research works related to Edge computing
- The performance of the proposed scheme was evaluated by comparing energy consumption and task completion percentage with certain conventional schemes.

Chapter 3

PROPOSED SYSTEM ARCHITECTURE

As illustrated in Figure 3.1, the edge-cloud system from bottom to the top consists of three layers/tiers: IoT devices (end-user devices), multiple Edge Computing nodes and the Cloud (service provider). The IoT level is composed of a group of connected devices

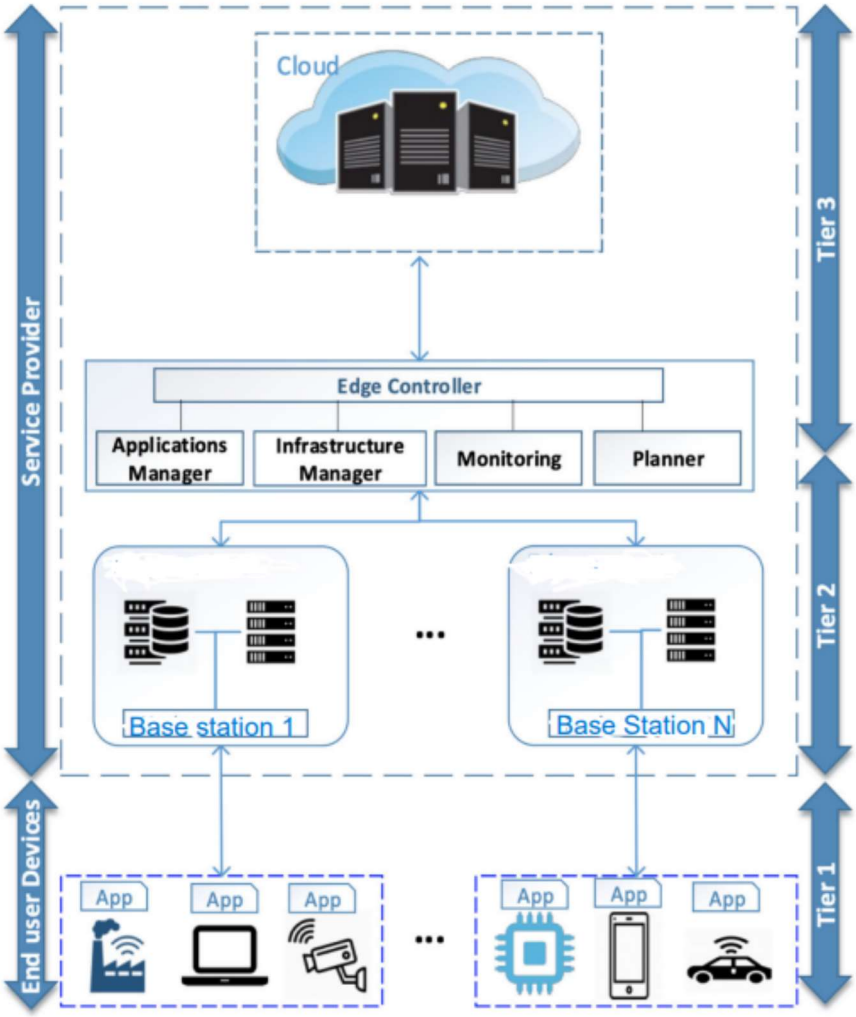

Figure 3.1: An overview of Edge-Cloud system

(e.g., smartphones, self-driving cars, smart CCTV); these devices have different applications where each application has several tasks (e.g., smart CCTV application consists of movement dedication and face recognition). These services can be deployed and executed

in different computing resources (connected Edge node, other Edge nodes or Cloud), where the infrastructure manager and service providers have to decide where to run these services.

In this proposed system, at the Edge level, each Edge computing node is a micro datacenter with a virtualized environment. It has been placed close to the connected IoT devices at the base station or Wi-Fi access point. These edge nodes have been distributed geographically and could be owned by the same Cloud provider or other brokers. Note that, it has limited computational resources compared to the resources in the cloud. Each edge node has a node manager that can manage computational resources and application services that run on. All the edge nodes are connected to the Edge Controller.

The offloading tasks can be achieved when the IoT devices decide to process the task remotely in Edge-Cloud environments. Applications running on IoT devices can send their off-loadable tasks to be processed by the Edge-Cloud system through their associated Edge node. We assume that each IoT application is deployed in a server in the edge node and the cloud. IoT devices offload tasks that belong to a predefined set of applications, these tasks are varied in terms of the computational requirement (task length) and communication demand (amount of transferred data). It is assumed that tasks are already offloaded from the IoT devices, and each task is independent; thus, the dependency between the tasks is not addressed in this paper.The locations of IoT devices are important for the service time performance because it is assumed that each location is covered by a dedicated base station with an Edge node and the IoT devices connect to the related WLAN when they move to the covered location.

The proposed architecture is just a possible implementation of other architectures in the literature . The main difference in the proposed architecture is the introduced layer between the edge nodes and the cloud. This layer is responsible for managing and assigning offloading tasks to the edge nodes. In practice, the edge computing nodes are connected through an intermediate layer (for example, backbone router) that serves as a central control manager to monitor them. In addition, software-defined network (SDN) technology can be utilized at this layer to monitor and manage the application services between the edge computing nodes depending on the data gathered, in which where SDN has a global view of the network and is capable of making more efficient and precise decisions. More details about the required components and their interactions within the proposed architecture are as follows.

EDGE CONTROLLER

Edge Controller (EC) is designed similarly to some studies called Edge Orchestrator, which is a centralized component that is responsible for planning, deploying and managing application services in the Edge-Cloud system. EC communicates with other

components in the architecture to know the status of resources in the system (e.g., available and used), the number of IoT devices, their applications' tasks, and where IoT tasks have been allocated (e.g., Edge or Cloud). EC consists of the following components: Application Manager, Infrastructure Manager, Monitoring and Planner. The location of the Edge Controller can be deployed in any layer between Edge and Cloud. The EC is synchronizing its data with the centralized Cloud because if there is any failure, other edge nodes can take EC responsibility from the cloud.

Application Manager

The application manager is responsible for managing applications running in the Edge-Cloud system. This includes requirements of application tasks, such as the amount of data to be transferred, the amount of computational requirement (e.g., required CPU), and the latency constraints. Besides, the number of application users for each edge node.

Infrastructure Manager

The role of the infrastructure manager is to be in charge of the physical resources in the entire Edge-Cloud system. [3]For instance, processors, networking, and the connected IoT devices for all edge nodes. As mentioned earlier, Edge-Cloud is a virtualized environment; thus, this component is responsible for the servers as well. In the context of this research, this component provides the EC with the utilization level of the servers.

Monitoring

The main responsibility of this component is to monitor application tasks (e.g., computational delay and communication delay) and computational resources (e.g., CPU utilization) during the execution of application tasks in the Edge-Cloud system[3]. Furthermore, detecting the tasks' failures due to network issues or the shortage of computational resources.

Planner

The main role of this component is to propose the scheduling policy of the offloading tasks in the Edge-Cloud system and the location where they will be placed (e.g., local edge, other edges, or the cloud). In the context of this research, the proposed approach for offloading tasks work on this component and passes its results to EC for execution.

3.1 Tasks scheduling approach for minimum Energy and latency

In the Edge-Cloud environment, IoT devices produce a stream of incoming offloading tasks that differ in terms of their computation and network demand. This would require an efficient task-scheduling technique that considers these differences in order to enhance the overall service performance and minimize the energy and delay in processing offloaded tasks.

Therefore, the proposed approach supports the resource manager in the Edge-Cloud system regarding scheduling the offloading tasks in order to minimize the overall service time and improve the efficiency of Edge-Cloud resources. As shown in Figure 3.2, the approach can be described using the MAPE method (Monitoring, Analyzing, Planning, and Executing) to assign the tasks to appropriate resources and monitoring the system performance periodically. The proposed approach works in the Edge Controller (EC) is as follows.

First, the edge controller receives the IoT devices information summary from edge computing nodes including the number of connected IoT devices, task length, the required number of cycles for the task, and the deadline requirement to complete the task. In addition, the status of computation resources at each edge node is monitored periodically. Subsequently, the edge controller computer and decides the optimal strategy for scheduling and assigning the computation tasks to the best server (i.e. one of the edge computing server nodes or the cloud server) for execution based on the gathered information through Fuzzy logic and DCEMTO(delay constrained energy minimal task offloading) algorithms. Later, we present the proposed algorithms in detail.

Fuzzy logic system

In this stage, the proposed approach will get the information on the offloading tasks and server utilization in order to determine the appropriate location of the offloading tasks, as depicted in Figure 3.3 The following is a brief description of the process of a fuzzy logic system.

1. **Fuzzy Input Variables:** In this step, the necessary inputs are specified for the fuzzy system. The required inputs are server utilization at the edge, task length, the amount of data to be transferred for each task and delay sensitivity. All these variables are represented as a linguistic variable: Low, Medium and High, as depicted in Fig. 3. These categories represent the dynamic changing over Edge-Cloud infrastructure and the Characteristics of applications offloaded tasks.
   (a) **Server Utilization:** this parameter indicates the current utilization level of the

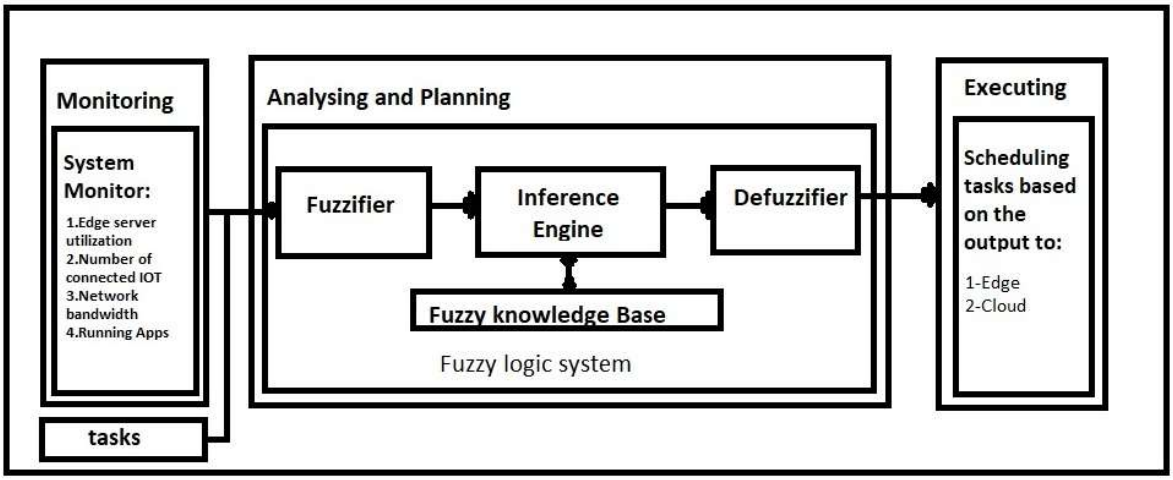

Figure 3.2: The proposed approach of scheduling offloading tasks

server hosted by the local edge server. Thus, we can know-how much resource capacity is available on that server. If it is highly utilized, then offloading to other edge servers or the cloud could be the solution, depending on the task characteristics in terms of computational, communication and latency sensitivity.

(b) **Task Length:** this parameter represents the computational demand of the task; it measures by Million Instruction Per Second (MIPS). As the edge has a limited computational resource, heavy tasks might be appropriate to offloaded to the cloud and vice versa. However, we cannot take this parameter without considering other parameters such as server utilization, communication demand and delay sensitivity.

(c) **Network Demand:** this parameter represents the required communication of the tasks for both uploading and downloading. It is an important measure for the offloading decision to consider where to offload the task (local edge, other edge, or cloud). For example, tasks of Augmented Reality (AR) applications that require video streaming must upload the request, then do some processing (e.g., 3D rendering, image processing, etc.), and then receive the results as a video stream. This requires transferring a high amount of data for uploading and downloading, which takes a significant amount of the total service time.

(d) **Delay Sensitivity of the Task:** this parameter refers to the sensitivity of the tasks to accept the level of latency (computation or communication delay). For example, some application has urgent tasks that require ultra-low latency; whereas, some tasks may accept some higher level of latency. This parameter could help the task scheduler to assign the tasks to an appropriate server within the Edge-Cloud system.

2. **Fuzzification:** In the fuzzification stage, the fuzzifier will take all the required values as numerical input from system infrastructure monitoring and incoming tasks.

Then, assign each value to its predefined linguistic variables in the membership functions (e.g., Low, Medium, and High). After that, fuzzy variables are combined and evaluates in the fuzzy rules-base to take the decision and produce the output in the defuzzification stage.

(a) **Fuzzy Membership Functions:** the fuzzy membership function is used to quantify the linguistic term for each fuzzy variable. In this research, four functions have been used (average server utilization, task length, network bandwidth and delay sensitivity) and each function has three variables (Low, Medium, High). The values of each fuzzy variable are determined empirically based on a number of experiments. Figure 3.4 shows the four membership functions.

(b) **Fuzzy Rules-Base:** : a fuzzy rules base is composed of a set of fuzzy rules that similar to the reasoning process of humans. It is a simple if-then rule that covers all the possible situations of the application characteristics and system conditions. These rules play critical directions to define the overall system performance. An example of the rules, if the task length is high AND Network demand is low AND server utilization is high AND the delay sensitivity is high THEN offloaded the task to the cloud. The output will be used in the defuzzification stage. Table 1 gives the results examples of the system's fuzzy rules. The main aim is to provide low latency for the IoT applications by reducing the data movement from IoT devices to the cloud and avoid the overloaded node, which will affect the end to end service time.

3. **Defuzzification:** Defuzzification is the process to convert the fuzzy rules output to a specific value based on the output membership function. There are a range of ways to produce the output membership function in the fuzzy logic system and these examples of the often-used method (e.g., maximum, mean of maximum and centroid). This work adopts the maximum approach because our membership function has one maximum at a time. Figure 3.5 represents the output membership function of the fuzzy logic system.

After deciding by fuzzy logic that the task is where going to offload either in edge or cloud, If the task is going to offload in Edge then we are going further consider the mobility, and according to that Which task is going to offload in which base station that will be decided by DCEMTO algorithm. So, Our Edge network consists of several Base Stations (BS), each equipped with an Edge server that is capable of receiving executing and transmitting computation tasks offloaded by users within the signal range of BS. Also, BSs communicate with each other using Edge Controller(EC). IoT Device users are allowed to offload their tasks to Edge servers to improve the task latency and reduce energy consumption. Mobile users can move out of the signal range of a BS at any time because of mobility. For Example in some cases, the user moved out of BS1 after offloading the task T1. In this case, the Edge server at BS1 can transmit the offloaded task to Edge Server at BS2 where the user currently is. This way, all Edge Servers, in the user's trajectory, are capable of executing the offloaded task.

We consider N = 1, 2, . . . , i, . . . , N be the mobile users moving around the

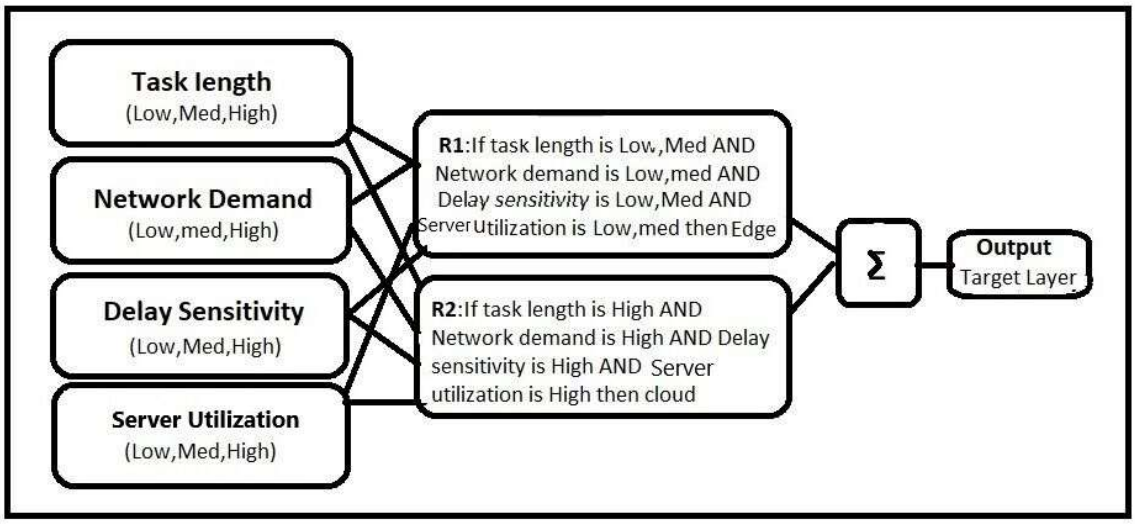

Figure 3.3: Process of the proposed fuzzy logic system

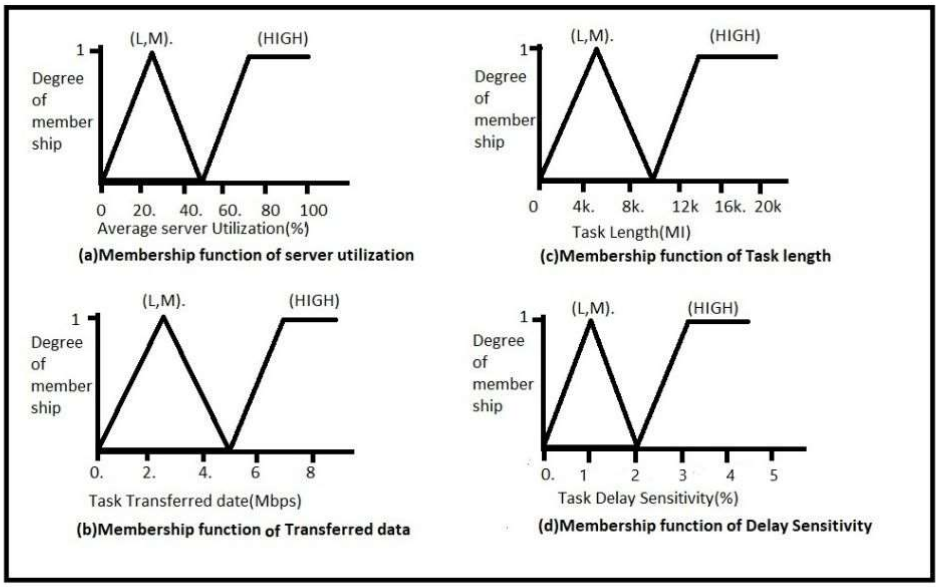

Figure 3.4: The four membership functions

Table 3.1: Fuzzy rules-base

Task Length (MIPS)	Network Demand (Mbps)	Server Utilization	Delay Sensitivity	Output Decision
L,M	L,M	L,M	L,M	Edge
L,M	L,M	L,M	High	Edge
L,M	L,M	High	L,M	Edge
L,M	L,M	High	High	Edge
L,M	High	L,M	High	Edge
L,M	High	L,M	L,M	Edge
L,M	High	High	L,M	Edge
L,M	High	High	High	Edge
L,M	L,M	L,M	High	Edge
High	L,M	High	L,M	Edge
High	L,M	High	High	Edge
High	L,M	L,M	L,M	Edge
High	L,M	L,M	L,M	Edge
High	High	L,M	High	Cloud
High	High	L,M	L,M	Cloud
High	High	High	L,M	Cloud
High	High	High	High	Cloud

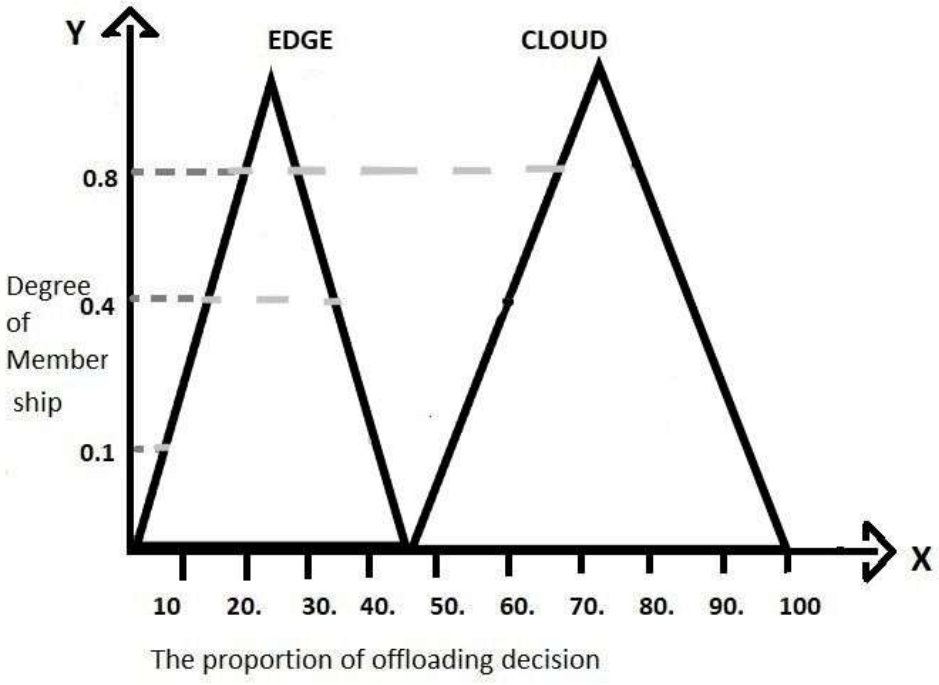

Figure 3.5: The output membership function of the fuzzy logic system

area under Edge Network. Each user is having a task T_i , denoted as (s_i ,c_i , t_i^{max}) where s_i is the size of the computational task, c_i is the required computation resource (in cycles) and t_i^{max} is the task deadline. Also, there are a total of M base stations each with signal range r_j and the associated Edge server having computation capacity a_j.

Energy Consumption and Task Delay

1. **Task Uploading:** IOT devices first, offload the task and the communication rate between user i and server j can be expressed as

   $r^u_{i,j} = Blog_2(1 + \frac{p^t_i d^{-\alpha}_{i,j}}{N_0 B})$ where B indicates the channel bandwidth, $d_{i,j}$ is the distance between the user i and Edge server j, N_0 is the noise power spectral density and is the channel fading parameter. The uploading time depends on the communication rate and task size and can be denoted as

   $t^u_{i,j} = \frac{s_i}{r^u_{i,j}}$

   and the uploading energy can be calculated by

   $e^u_{i,j} = t^u_{i,j} p^i_t$

   where p^i_t is the transmission power of mobile i

2. **Task Transmission:** As Edge servers can communicate with each other by making use of the Edge controller, the transmission rate between the server j1 and j2 can be expressed as

   $r^{tr}_{j1,j2} = Blog_2(1 + \frac{p^t_{j1} d^{-\alpha}_{j1,j2}}{N_0 B})$

   where $d_{j1,j2}$ is the distance between the server j1 and j2. The transmission time and transmission energy can be formulated as

   $t^{tr}_{j1,j2} = \frac{s_i}{r^u_{j1,j2}}$

   $e^{tr}_{j1,j2} = t^u_{j1,j2} p^{j1}_t$

   where p^{j1}_t is the transmission power of the Edge server j1.

3. **Task Execution:** The computation time of task i in server j can be denoted as

   $t^c_{i,j} = \frac{c_i}{a^u_j}$

   and the computation energy can be calculated as

   $c^c_{i,j} = t^c_{i,j} p^c_j$

   where p^c_j is the computation power of Edge server j.

   The download transmission delay and energy are not considered because the size of the result of the task after processing is very less and the energy consumption and delay are negligible.

Chapter 4

PROBLEM FORMULATION

We assume that each user i is having a task T_i which can be offloaded[12]. As the user is moving, the edge server to which user i can offload the task varies with time and we denote S_t^i as the set of available Edge serverss at time t and let mt_i be the total time user i is moving. We aim to find a Edge server j from the set of available servers in user $i's$ i.e., $S_i = (U_{t=0}^{mt} S_i^t)$,satisfying task i's deadline to minimize the total energy consumption. The task offloading problem can then be modeled as

$$min \Sigma_{i \in N} \Sigma_{j \in S_i} x_{i,j} E_{i,j}$$

s.t. C1: $t_{i,j} <= t_i^{max}, \forall i \in N$

P1: C2: $\Sigma_{j \in S_i} j \in x_{i,j} = 1, \forall i \in N$

C3: $x_{i,j} \in \{0, 1\}$

It's very clear from the conditions C2, C3 that the task T_i should be computed only once at some server j Si and can't be divided. The energy $E_{i,j}$ and time ti, j in the above formulation can be expressed as

$$E_{i,j} = e_{i,j1}^u + \sum_{t=t1}^{t2} e_{j \in S_i^t, j \in S_i^{t+1}}^{tr} + e_{i,j}^c$$

$$t_{i,j} = t_{i,j1}^u + \sum_{t=t1}^{t2} t_{j \in S_i^t, j \in S_i^{t+1}}^{tr} + t_{i,j}^c$$

Where $j1 \in S_i^0$

$0 <= t_1 < t_2 < mt_i$

$j \in S_i^{t2+1}$

The above equation indicates that total energy and total time includes energy and time spent in uploading, transmission, and computation of a task.

Table 4.1: Symbols Description

Symbol	**Description**
s_i	size of the computational task Ti
c_i	computation resource (in cycles) needed for task Ti
t_i^{max}	deadline for task T_i
r_j	signal range of the edge server j
a_j	computation capacity of the server j
$r_{i,j}^u$	communication rate between user i and server j
$r_{j1,j2}^{tr}$	transmission rate between servers j1and j2
B	channel bandwidth
N_0	noise power spectral density
α	channel fading parameter
$t_{i,j}^u, e_{i,j}^u$	delay and energy consumption for task uploading from user i to server j
$t_{j1,j2}^{tr}, e_{j1,j2}^{tr}$	delay and energy consumption for task transmission between servers j1and j2
$t_{i,j}^c, e_{i,j}^c$	delay and energy consumption for execution of a task from user i in server j
$d_{i,j}$	distance between user i and server j
p_i^t	transmission power of iot devices i
p_j^t	transmission power of server j
p_j^c	computation power of server j
S_i^t	the set of available edge servers at time t
mt_i	the total time user i is moving.
$x_{i,j}$	Boolean indicator of whether task I is allowed to server Sj at time t
$E_{i,j}, t_{i,j}$	Total energy and total time
T_{length}	length of the task
$T_{Network}$	Communication required for tasks downloading and uploading
N	Number of Iot devices
M	Number of Base stations
x_i	The energy minimal server satisfying delay constraint of task T_i
td_i	maximum time delay required for executing the task i
k	user i's trajectory at t+1

Chapter 5

PROPOSED ALGORITHM

The problem **P1** is a constraint-satisfaction problem and it is NP-complete[1][12]. We propose a heuristic to solve the problem and it is as follows:

Each user i offloads task Ti with the required information: size of the computational task, required computation resource (in cycles) and the task deadline. The Edge Controller then assigns each task to a edge server satisfying task deadline and least energy consumption. The delay constrained energy minimal task offloading algorithm shortly referred to as DCEMTO algorithm is shown in Algorithm 1.

For each user $i \in N$, at each time step t $[0, mt_i]$, available edge servers S_i^t is found out. Among the servers in S_i^t , our goal is to find out two servers, one for execution to which task can be assigned and one for transmission from which task can be offloaded to a server ($\in S_i^{t+1}$) situated in the next time step locality. As the task assignment has to satisfy delay constraint, the time needed for uploading, transmission, and execution is also calculated and checked for delay constraint in each timestep.

For each available server ($\in S_i^t$), the energy consumed due to transmission (up to timestep t-1), and execution is calculated. This process is repeated for all time steps and the server with the least energy is assigned to the task Ti for execution[1].

The energy consumed due to transmission at timestep t can be found out by adding transmission energy up to timestep t-1 and energy needed to offload task from server's location to user's locality in next time step t + 1.

In the end, the algorithm outputs $x_i \forall_i \in N$, the energyoptimal server satisfying delay constraint of task T_i.

5.1 Algorithm 1

INPUT: Applications' tasks T_i with their parameters T_{Length}, $T_{Network}$ and T_{Delay} and IOT devices N, base stations M

OUTPUT: Select computational resource at the target layer: R_{Edge}, R_{cloud} and $x_i \forall_i \in N$

1.**for each** device $i \in N$ do

2.**for all** tasks in T_i do

3.Read all the average utilization of servers S_i

4.F = Fuzzy logic system (T_i^{Length}, $T_i^{Network}$,T_i^{Delay} ,Servers(S_i))

5.**if** F <= F(low,med) **then**

6.Allocate T_i on R_{Edge}

7.**for each** time step $t \in [0, mt_i]$ do

8.$S_i^t \leftarrow$ get available Edge servers at timestep t

9.**for each** server $j \in S_i^t$ do

10.**if** t == 0 **then**

11.$t_j^c = t_{i,j}^u + t_{i,j}^c$

12.**if** $t_j^c <= td_i$**then**

13.$e_j^c = e_{i,j}^u + e_{i,j}^c$

14.**else**

15.$e_j^c \leftarrow \infty$

16.**end if**

17.**else**

18.$t_j^c = t_{j0----t-1}^{tr} + t_{jt-1,j}^j + t_{i,j}^c$

19.**if**$t_j^c <= td_i$ **then**

20.$e_j^c = e_{j0----t-1}^{tr} + e_{jt-1,j}^{j} + e_{i,j}^{c}$

21.**else**

22.$e_j^c \leftarrow \infty$

23.**end if**

24.**end if**

25.**end for**

26.$j_*^t = argmin(e_j^c)$

27.Let k be the user i's trajectory at t+1

28.**for each** server $j \in S_i^t$ do

29.$e_{j,k}^t = p_j d_{j,k}/r_{j,k}$

30.**end for**

31.$j_t = argmin_j(e_{j,k}^t)$

32.$e_{j0---t-1}^{tr} = e_{j0---t-1}^{tr} + e_{jt-1,jt}^{tr}$

33.$t_{j0---t-1}^{tr} = t_{j0---t-1}^{tr} + t_{jt-1,jt}^{tr}$

34.**end for**

35.$x_i = argmin_{j*}(e_{jt*})$

36.return $X_i, \forall_i \in M$

37.**else**

38.allocate T_i onR_cloud

39.**end if**

40.**end for**

41.**end for**

Table 5.1: Simulation Parameters

Parameters	**Value/Range**
Coverage radius of BS	70 to 100 meters
Edge server's CPU capacity	[7, 20] GHz
Edge server's computation power	[3,5] watt
Edge server's transmission power	[0.1,1] watt
Channel bandwidth	1MHz
User's mobility speed	13 MPH
Task CPU requirement	[1,10] GHz
Input data size	[1,3] MB
Task deadline constraint	[0.1,1] sec
Simulation Time	30 minutes
Warm-up Period	3 minutes
Number of IoT devices	200-2000

Chapter 6

RESULT AND EVALUATION

In this section, we evaluated the performance of our proposed algorithm to achieve energy-minimal task offloading with users' mobility and delay constraints into consideration[1][12]. we used data from base stations within the Melbourne central business district area in Australia, which has a total area of 6.2 km^2 provided by the Australian Communications and Media Authority . There are a total of 126 base stations situated inside the Melbourne CBD area. We follow the Random waypoint mobility model for the users moving in the area covered by the base stations provided in the dataset. The random waypoint model is a random model for the movement of mobile users, and how their location, velocity and acceleration change over time. Without loss of generality, we refer to the parameter settings in the existing works ,

. The coverage radius of BS is randomly taken varying from 70 to 100 meters and the associated Edge Server's CPU capacity is in [7, 20] GHz, computation power is in [3,5] watts and transmission power ranges from 0.1 to 1 watt. The bandwidth of the communication channel is set up as 1MHz. Users are moving with the speed of 1 3 MPH and generate tasks with input data size varies from 1 MB to 3 MB having CPU requirement is in [1,10] GHz and deadline constraint within [0.1,1] seconds. Users' mobiles are having CPU capacity ranges from 1 to 10 GHz, transmission power is randomly taken from [7,15] watts and computation power from [7,10] watts.

We consider mobile-execution and random-allocation as our baseline algorithms:

• Local-Execution: Feasible Tasks satisfying delay constraint are executed within the mobiles without offloading.

• Random-Allocation: Each user's task is assigned to a server randomly chosen from the servers present along the user's trajectory.

Fig. 6.1 depicts the higher Task completion percentage compared to the other 2 approaches by varying Task deadline constraint. DCEMTO's task completion is 9.16% higher than Random-Allocation and 27.08% higher than Local-Execution on average. Also, Task completion percentage increases by increasing Deadline constraint.

Fig. 6.2 indicates the variation of average energy consumption with a change in the number of users. DCEMTO outperforms other approaches. Average energy consumption

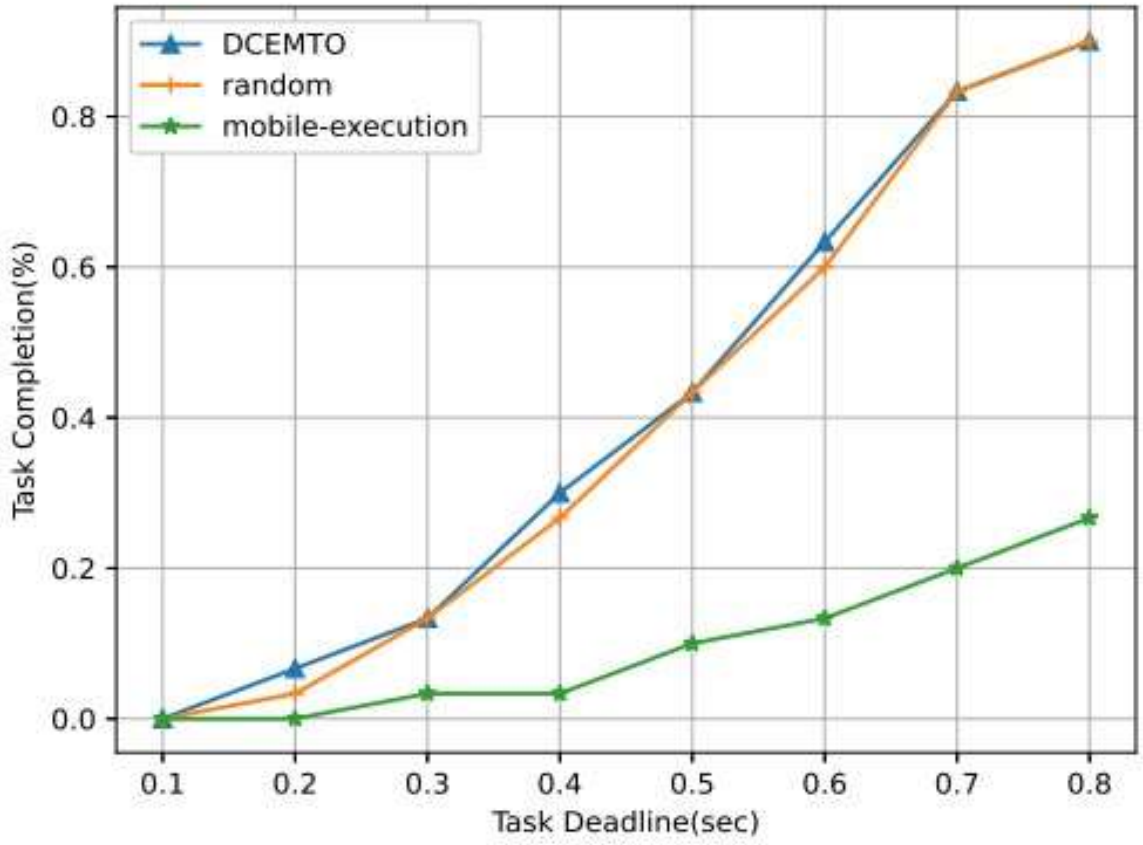

Figure 6.1: Task Completion vs Task Deadline

in DCEMTO is 10.5% energy efficient than Random-Allocation and 46.5% efficient than Local-Execution.

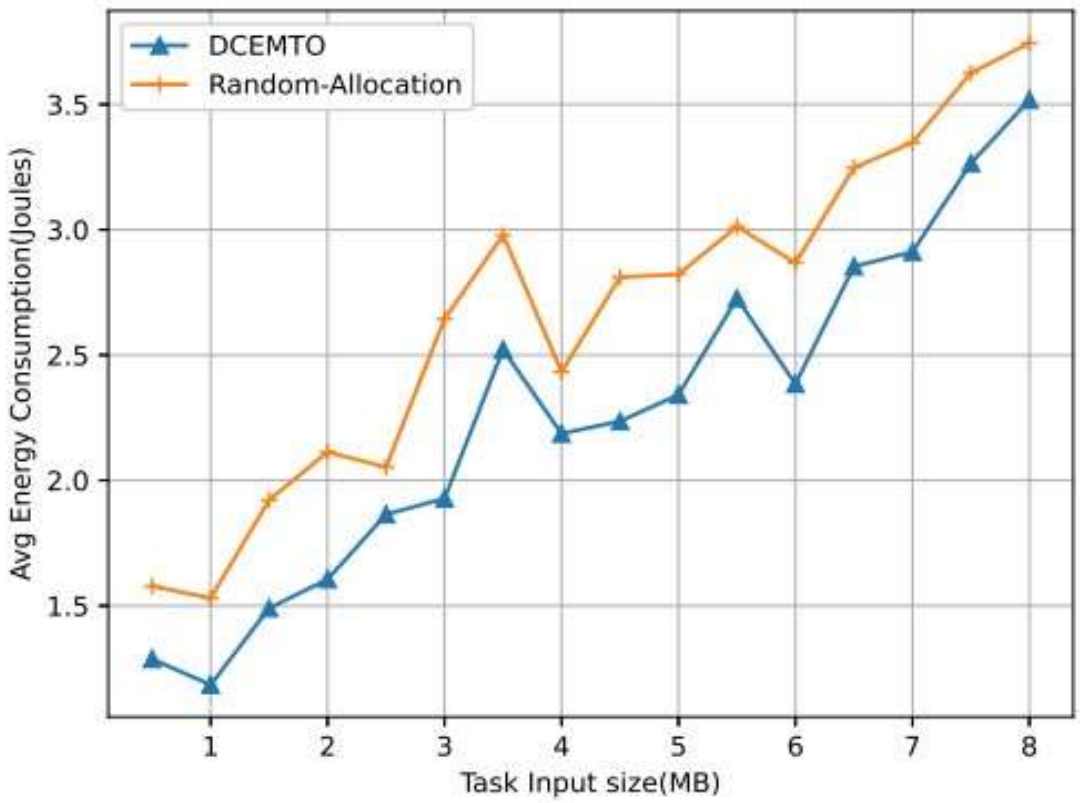

Figure 6.2: Task Input Size vs Energy Consumption

Fig 6.3 shows how energy consumption varies with varying task input size. It's clear that higher the task input size, energy consumption increases. It is due to higher transmission energy needed in task offloading. Energy consumption in DCEMTO is 9.55% efficient than Random-Allocation approach.

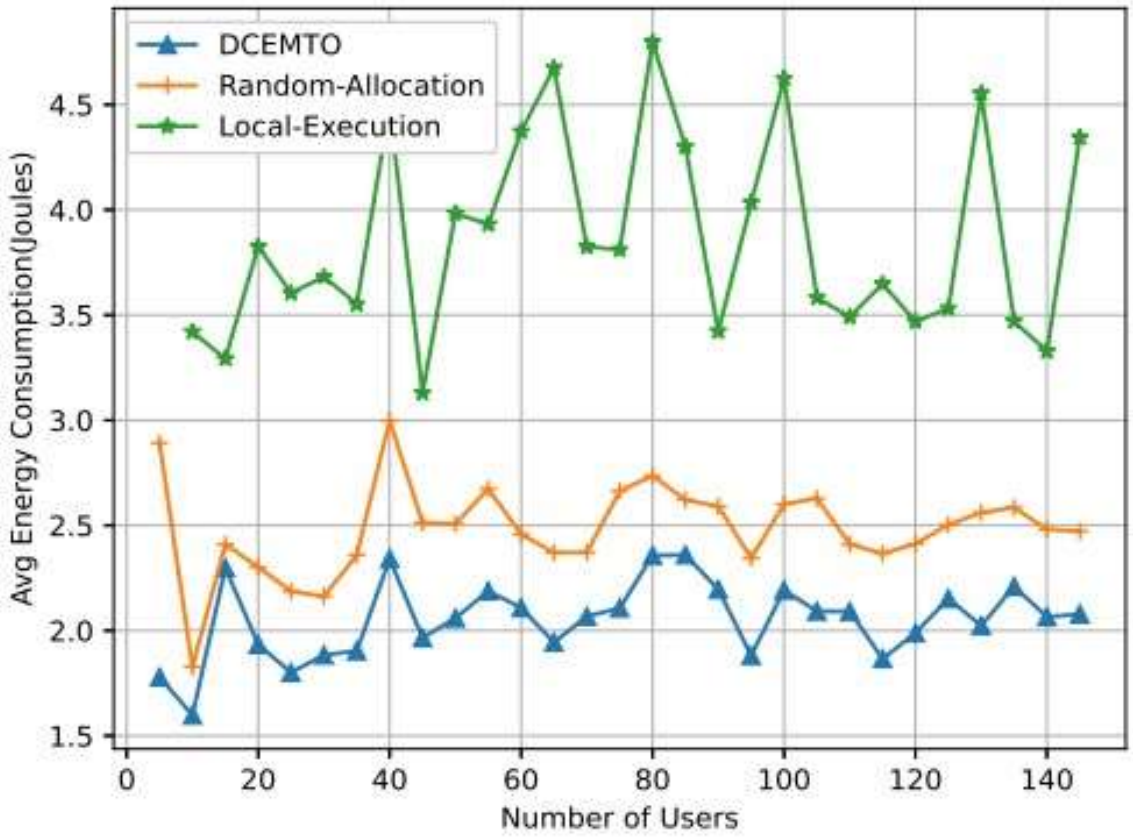

Figure 6.3: Avg Energy Consumption vs Number of Users

Chapter 7

CONCLUSION

In this paper, we explored the problem of Energy-minimal task offloading in the Edge-Cloud Computing environment by considering the mobility of users and task deadline constraint. We mathematically formulated the problem and proposed a heuristic algorithm DCEMTO, Where We first applied Fuzzy logic to decide which task will be offloaded in the edge and which will be offloaded in the cloud and then we decided in which respective server the task will be executed.

Here, also the user mobility, task deadline and base station information are utilized to optimize the overall energy consumption for task execution as our main aim. We evaluated our algorithm on the Melbourne CBD base stations dataset by comparing with baseline approaches like Random-Allocation and Local-Execution. The results convey that our DCEMTO algorithm outperforms other approaches in metrics like Task Acceptance rate, Average Energy consumption and overall Energy Consumption.

Chapter 8

REFERENCES

1. Zaman, Sardar Khaliq uz, et al. "Mobility-aware computational offloading in mobile edge networks: a survey." Cluster Computing (2021): 1-22.

2. Wang, Jin, et al. "An energy-efficient off-loading scheme for low latency in collaborative edge computing." IEEE Access 7 (2019): 149182-149190.

3. Almutairi, Jaber, and Mohammad Aldossary. "A novel approach for IoT tasks offloading in edge- cloud environments." Journal of Cloud Computing 10.1 (2021): 1-19.

4. Saeik, Firdose, et al. "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions." Computer Networks 195 (2021): 108177.

5. Shahzad, Haleh, and Ted H. Szymanski. "A dynamic programming offloading algorithm for mobile cloud computing." 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE, 2016.

6. Shekhar, Shashank, and Aniruddha Gokhale. "Dynamic resource management across cloud-edge resources for performance-sensitive applications." 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE, 2017.

7. Liu, Zhaolin, et al. "Mobility-aware task offloading and migration schemes in scns with mobile edge computing." 2019 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2019.

8. Li, Zhongjin, et al. "Energy-aware task offloading with deadline constraint in mobile edge computing." EURASIP Journal on Wireless Communications and Networking 2021 (2021): 1-24.

9. Huang, Xiaoge, et al. "Blockchain-enabled task offloading and resource allocation in fog computing networks." Wireless Communications and Mobile Computing 2021 (2021): 1-12.

10. Ren, Qian, Kui Liu, and Lianming Zhang. "Multi-objective optimization for task offloading based on network calculus in fog environments." Digital Communications and Networks 8.5 (2022): 825-833.

11. Kai, Yuan, Junyuan Wang, and Huiling Zhu. "Energy minimization for D2D-assisted mobile edge computing networks." ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, 2019.

12. Wu, Chunrong, et al. "Mobility-aware tasks offloading in mobile edge computing environment." 2019 Seventh International Symposium on Computing and Networking (CANDAR). IEEE, 2019.

13. Zhao, Rui, et al. "Deep reinforcement learning based mobile edge computing for intelligent Internet of Things." Physical Communication 43 (2020): 101184.

14. Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas and S. Khaf, "A Deep Learning Approach for Energy Efficient Computational Offloading in Mobile Edge Computing," in IEEE Access, vol. 7, pp. 149623-149633, 2019, doi: 10.1109/ACCESS.2019.2947053.