e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:12/December-2024 Impact Factor- 8.187 www.irjmets.com

USING MICROSERVICES PLANNING FOR ADDITIONAL CREATED HELP PARTAKING IN IOT EDGE CONDITIONS

P. Chandra Sekhar[*1], G. Sravani[*2], Ch. Deekshitha[*3], G. Nandini[*4]

[*1]Assistant Professor, Department Of Computer Science & Engineering, Guru Nanak Institute Of Technology, Ibrahimpatnam, RR District, Telangana, India.

[*2,3,4]Student, Department Of Computer Science & Engineering, Guru Nanak Institute Of Technology, Ibrahimpatnam, RR District, Telangana, India.

DOI: https://www.doi.org/10.56726/IRJMETS64859

ABSTRACT

Mobile edge computing has shown its potential in serving emerging latency-sensitive mobile applications in ultra-dense 5G networks via offloading computation workloads from the remote cloud data center to the nearby network edge. However, current computation offloading studies in the heterogeneous edge environment face multifaceted challenges: Dependencies among computational tasks, resource competition among multiple users, and diverse long-term objectives. Mobile applications typically consist of several functionalities, and one huge category of the applications can be viewed as a series of sequential tasks. In this study, we first proposed a novel multiuser computation offloading framework for long-term sequential tasks. Then, we presented a comprehensive analysis of the task offloading process in the framework and formally defined the multiuser sequential task offloading problem. Moreover, we decoupled the long-term offloading problem into multiple single time slot offloading problems and proposed a novel adaptive method to solve them. We further showed the substantial performance advantage of our proposed method on the basis of extensive experiments.

I. INTRODUCTION

The rapid growth of mobile devices (MDs) has brought us to the era of network interconnections of all things, and the demand for low-latency mobile applications is soaring. However, the computational capabilities of MDs are insufficient to support the real-time requirements of these applications because of MDs' inherent limitations, such as limited battery capacity, limited context awareness, and system architecture. For example, to ensure the quality of service and prevent the users from feeling dizzy and sick, the virtual reality system not only needs to transmit high- resolution images with a frame rate of more than 120 frames per second but also needs to complete computation-intensive tasks, such as target recognition and virtual holographic projection modeling. As a promising trend, mobile edge computing (MEC) is capable of handling the challenging requirements of these emerging applications. To provide low-latency services for users, MEC allows users to offload their tasks to nearby MDs through a wireless access network. By making full use of idle resources around users, MEC builds a carrier-grade service environment that enables users to enjoy an elastic, high-quality network experience. However, compared with centralized data centers, edge clouds usually cannot provide powerful computation capacity because MDs are lightweight, and the wireless resources are limited and can be jammed when many users try to offload their tasks simultaneously. Therefore, the limited resources must be jointly managed to exploit the potential of MEC. As a key component in MEC, computational offloading allocates computational tasks from MDs to edge servers. Computation offloading has a significant impact on system performance and can alleviate MDs' deficiencies in energy efficiency, resource storage, and computational performance. By looking for a nearby MEC server for task offloading, mobile users with limited resources can save energy and optimize their experience.

Numerous efforts have been exerted in developing efficient computation offloading strategies, aiming to improve the system performance of MEC. They have reduced energy consumption and processing delay effectively or made a trade-off between them by selecting task offloading decisions flexibly. For instance, by adopting a graph to model the dependencies among tasks, the task offloading strategy proposed in optimizes the application delay. The task scheduling strategy is put forward for local and edge offloading in Ref. by utilizing the Markov decision process method and achieves a short average execution delay. To utilize computation offloading efficiently in MEC, many issues remain in real mobile application scenarios:

1. Heterogeneity: MEC servers are typically lightweight and heterogeneous, which means they are along with different data formats, latency requirements, computation resources, and management. These factors explain why MEC servers are massively deployed on edge networks and often come from multiple service providers. Thus, they have different performance in accomplishing data storage, computation, and transmission. The heterogeneity of MEC servers leads to high uncertainties in task execution cost and task execution efficiency and becomes a bottleneck for computational tasks.
2. Multiuser: Most existing studies focus on single-user task offloading and ignore the resource competition problem in multiuser scenarios. Specifically, the competition among MDs leads to low wireless rates in the network and long queuing times on edge servers. Therefore, developing an effective offloading strategy is necessary to meet the user demand and improve the utilization of the computational resources of MEC servers.
3. Dependency: Many mobile applications, such as online games, consist of a set of consecutive dependent tasks. However, existing works pay attention to the computational offloading of independent tasks rather than dependent tasks and fail to address the task dependency challenge. Moreover, multiuser scenarios make the offloading problem more challenging than usual. For example, Google Project Glass is equipped with augmented reality technology that includes the following functional modules: Video capture, video parsing, target recognition, content mapping, video synthesis, and real-time display. These functions must be executed in strictly sequential order. Among them, video parsing and content mapping put high demand on computational resources, and offloading them to edge servers can effectively reduce the processing latency of the whole application.
4. Long-term execution: Most existing studies are about the static offloading scenario where the optimized offloading strategy is determined before the task execution and ignores the existence of the long-term execution of sequentially dependent tasks. Dynamic events, such as task departure, subsequent task arrival, and computing resource changing, make it even difficult to deal with long-term execution.

II. METHODOLOGY

2.1 Methodologies

2.1.1 Module Overview

This project having the following five modules:

- User Interface Design
- User
- Edge Server
- Server

1. User Interface Design

In this module we design the windows for the project. These windows are used for secure login for all users. To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password and Email id, into the server. Server will create the account for the entire user to maintain upload and download rate. Name will be set as user id. Logging in is usually used to enter a specific page.

2. User

This is the first module Data User can register and Login. After login Data User have an option of searching the files as a file name. Data user can also have a download file it will show an encrypted data. Data user can also send a trapdoor request to the server. Server can accept the request and then data user can takes permissions from the owner then the file it will downloaded in plain text.

3. Edge Server

This is the Second module of this project. In this module Data Owner should register and Login. Data Owner will Uploads the files into the database. Data owner can also send request to the data user.

4. Server

This is the third module of this project. In this module Cloud Server can login. After login it will see all data owners' information. Cloud server can see all users' information. Cloud server can see an all stored data files. Cloud server can give keys request to the user. Cloud server can also see an attacker information of file.

III. MODELING AND ANALYSIS

System Architecture

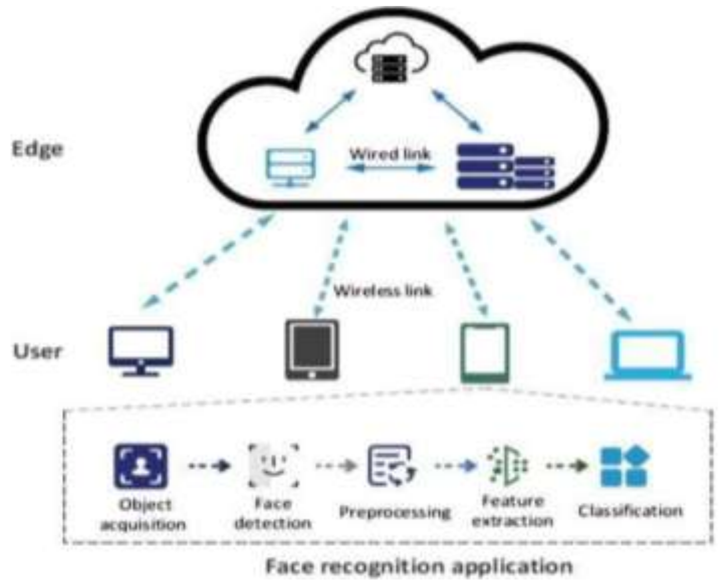


Fig 3.1: System Architecture

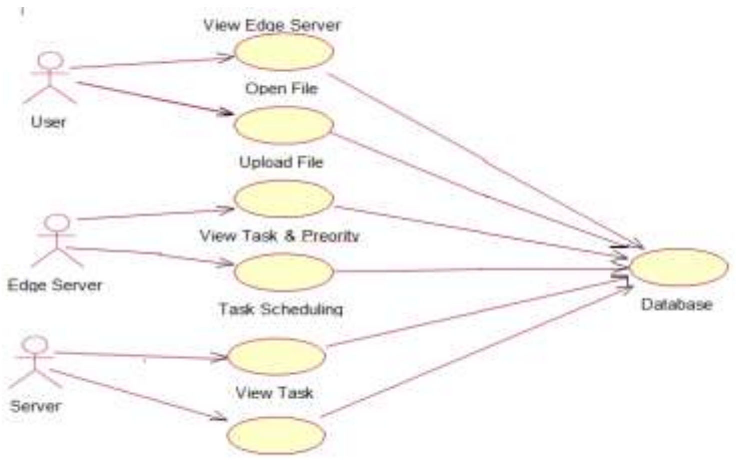


Fig 3.2: Use Case Diagram

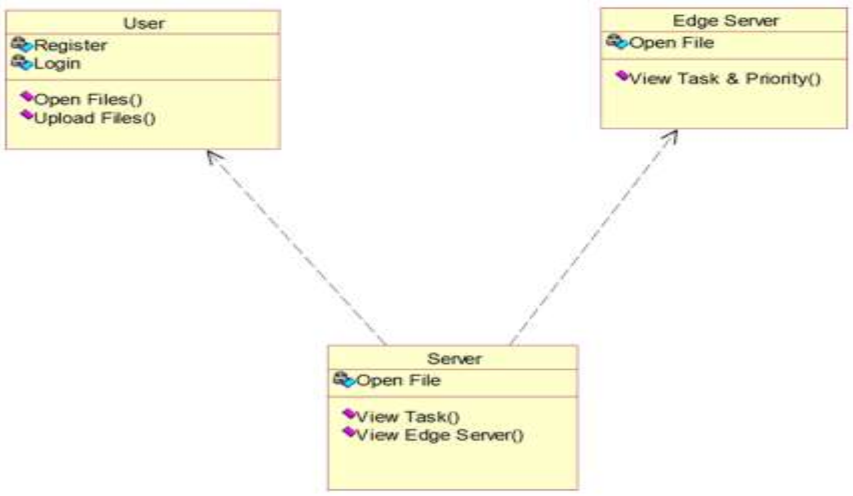


Fig 3.3: Class Diagram

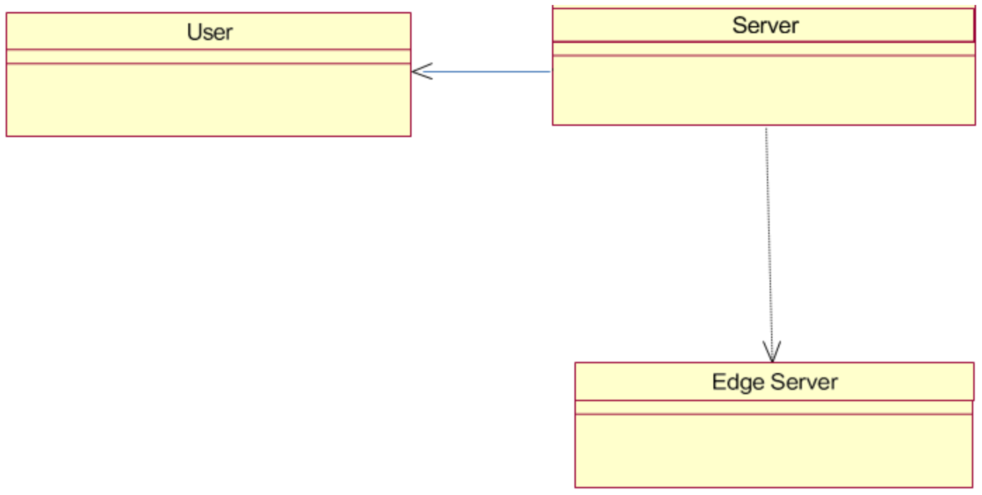


Fig 3.4: Object Diagram

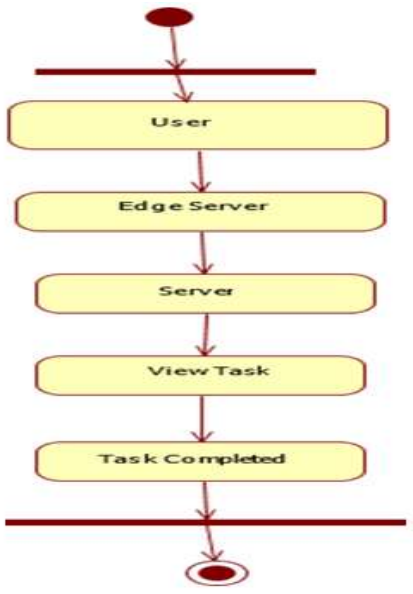


Fig 3.5: Activity Diagram

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering.

The main purpose of a USE CASE DIAGRAM is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

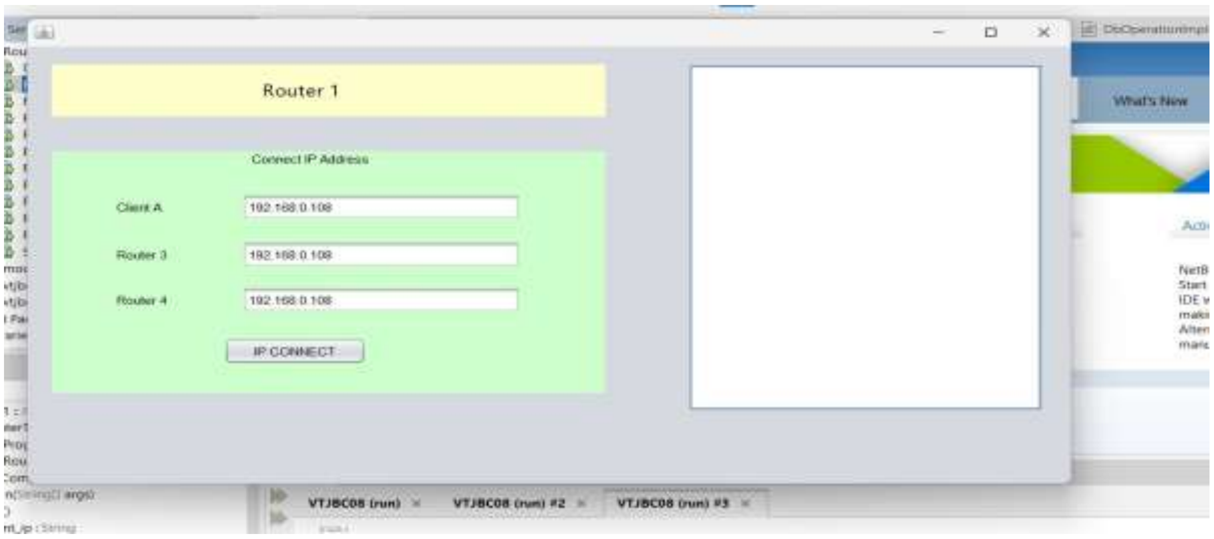
In this CLASS DIAGRAM represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

In the above ACTIVITY DIAGRAM tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security

IV. RESULTS AND DISCUSSSION

This project is implements like web application using COREJAVA and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

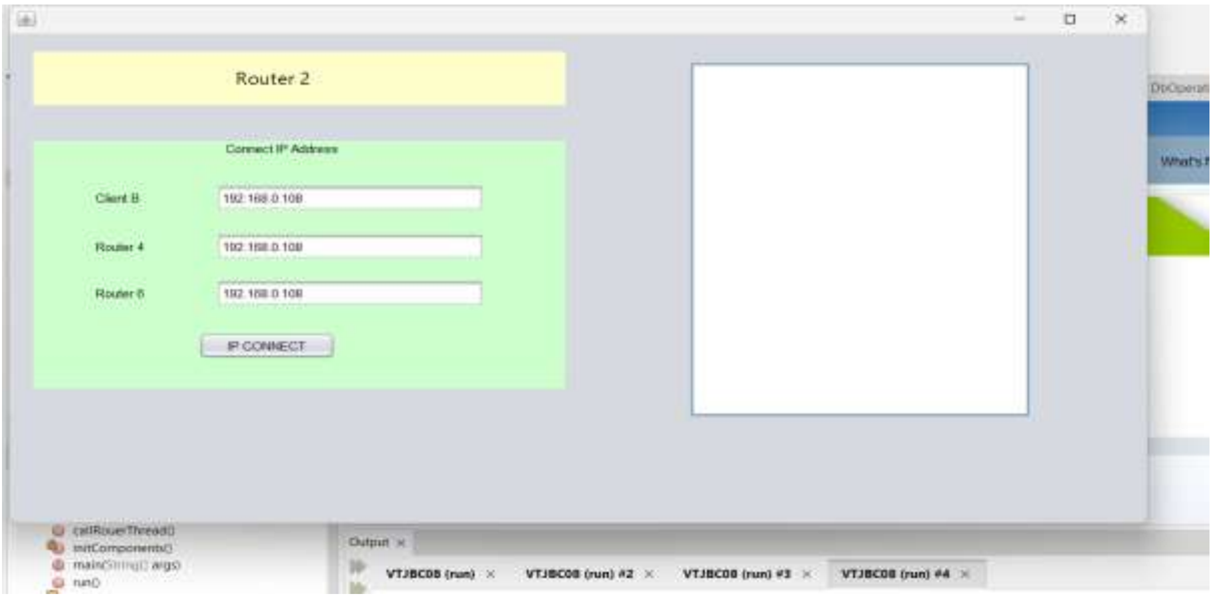
e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:12/December-2024 Impact Factor- 8.187 www.irjmets.com


Router 1

Connect IP Address

Client A	192.168.0.108
Router 3	192.168.0.108
Router 4	192.168.0.108

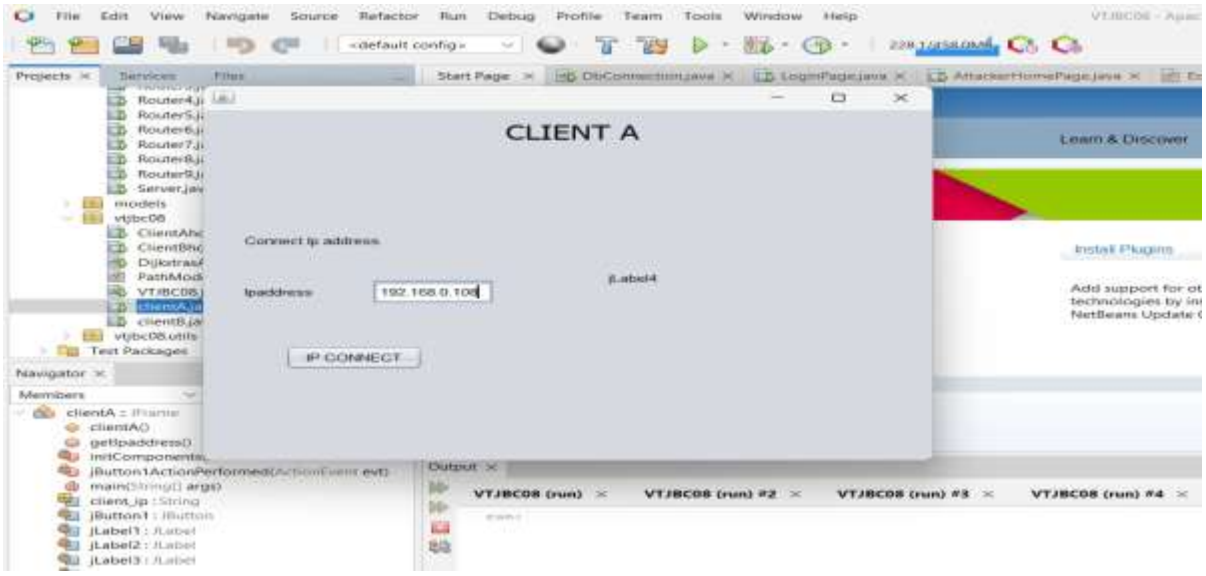
Fig 4.1: Index Page


Router 2

Connect IP Address

Client B	192.168.0.108
Router 4	192.168.0.108
Router 6	192.168.0.108

Fig 4.2: Registration Page


CLIENT A

Connect Ip address

Ipaddress 192.168.0.108

Fig 4.3: Data Owner Homepage

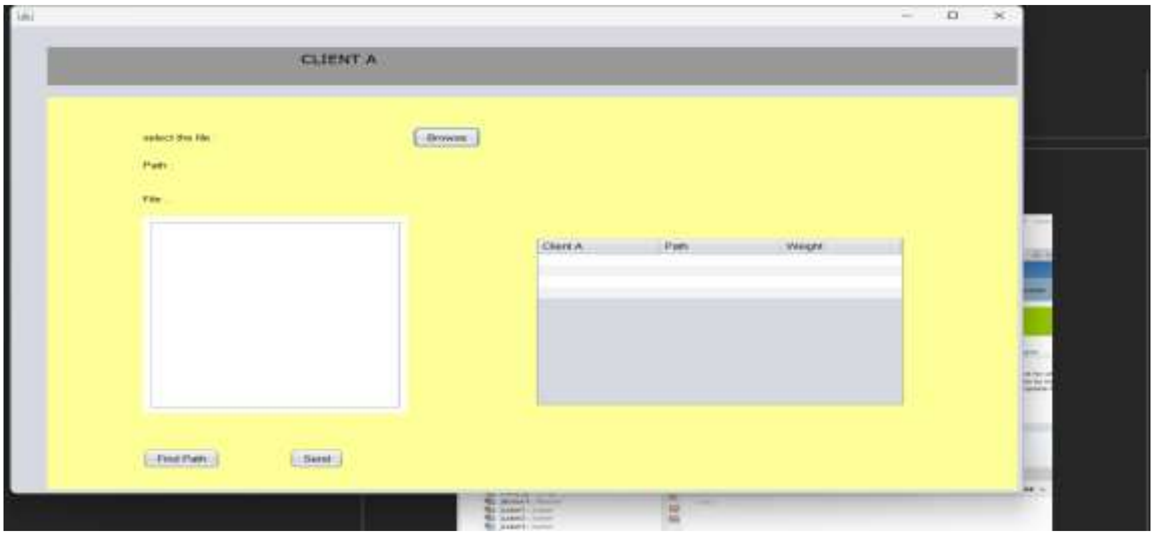
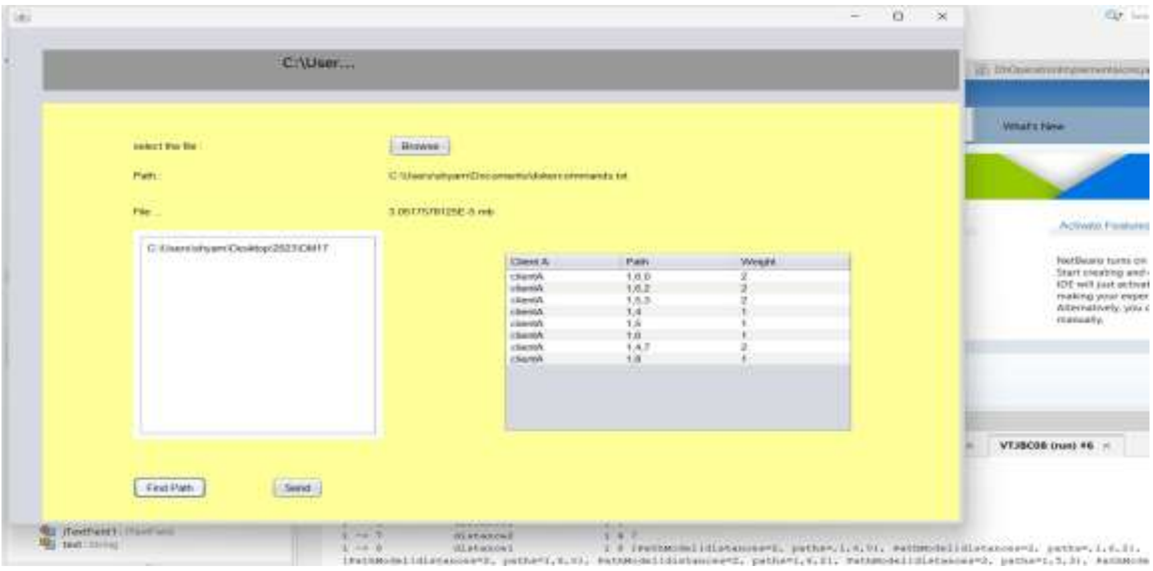


Fig 4.4: Data Owner Upload



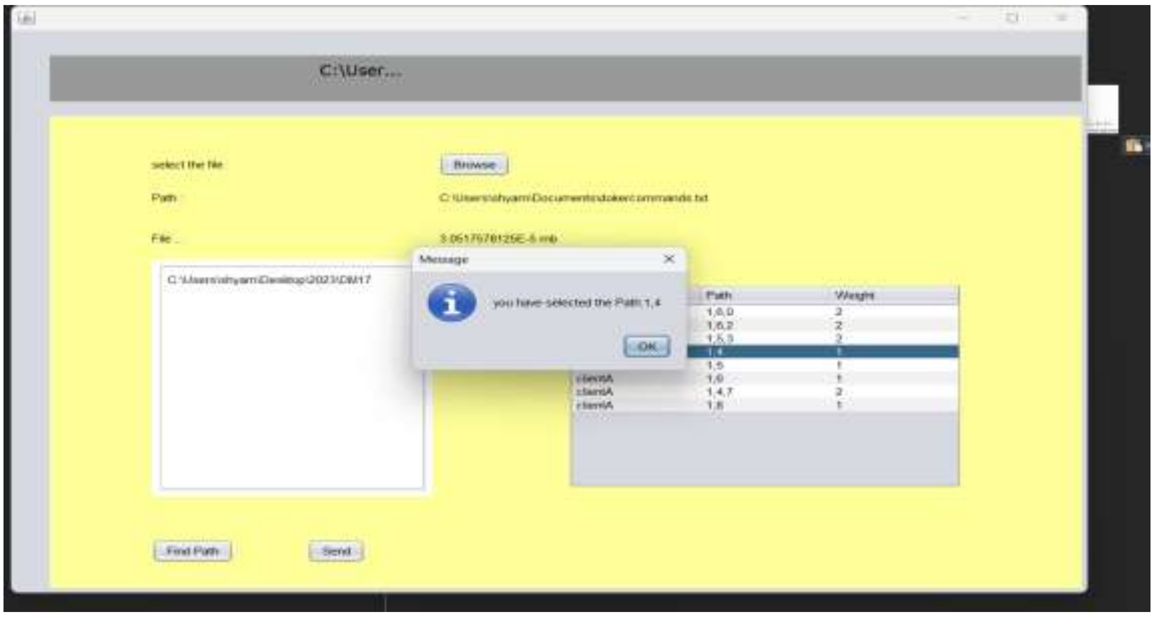
Path : C:\Users\shyam\Documents\docker commands.txt

File .. 3.0517578125E-5 mb

C:\Users\shyam\Desktop\2023\DM17

Client A	Path	Weight
clientA	1,6,0	2
clientA	1,6,2	2
clientA	1,5,3	2
clientA	1,4	1
clientA	1,5	1
clientA	1,6	1
clientA	1,4,7	2
clientA	1,8	1

Fig 4.5: Data Owner send Keys to User



Path : C:\Users\shyam\Documents\docker commands.txt

File .. 3.0517578125E-5 mb

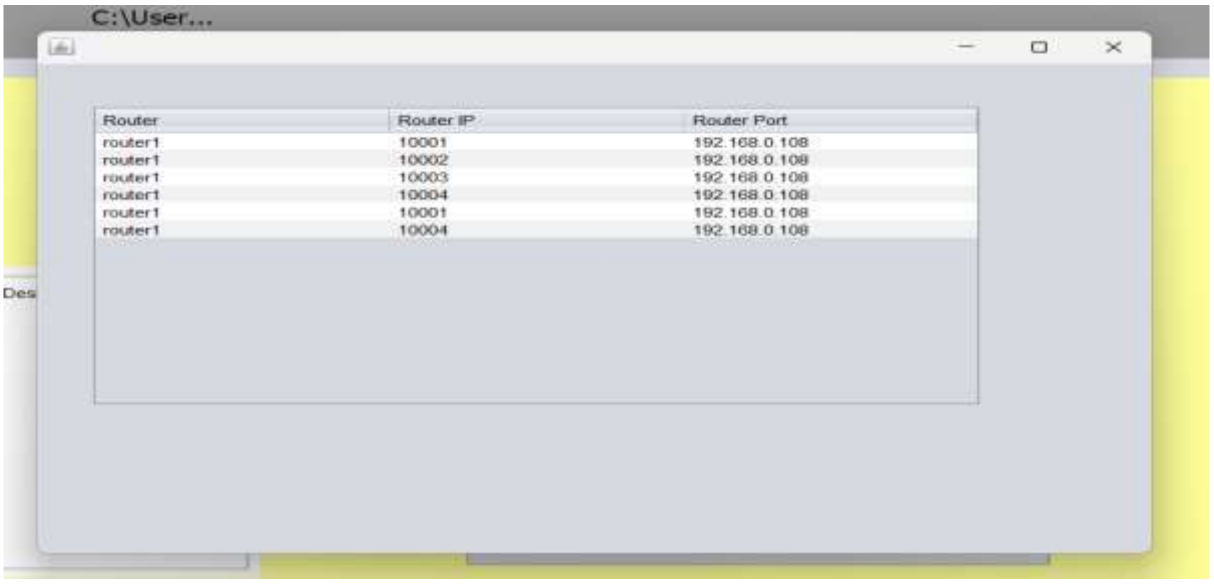
C:\Users\shyam\Desktop\2023\DM17

you have selected the Path 1,4

	Path	Weight
	1,6,0	2
	1,6,2	2
	1,5,3	2
	1,4	1
	1,5	1
clientA	1,6	1
clientA	1,4,7	2
clientA	1,8	1

Fig 4.6: Data User Registration Page

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:12/December-2024 Impact Factor- 8.187 www.irjmets.com



Router	Router IP	Router Port
router1	10001	192.168.0.108
router1	10002	192.168.0.108
router1	10003	192.168.0.108
router1	10004	192.168.0.108
router1	10001	192.168.0.108
router1	10004	192.168.0.108

Fig 4.7: Data User Search Query Page

V. CONCLUSION

This study investigated the multiuser sequential task offloading problem in heterogeneous MEC environments. Our goal was to minimize the average application delay. To this end, we proposed the MCO-LST framework to deal with offloading for long-term sequential tasks. Specifically, we formally defined the problem. Then, we proposed an effective algorithm AMUST to obtain the optimal offloading strategy. Finally, we conducted extensive experiments to demonstrate the effectiveness of AMUST.

VI. REFERENCES

[1] T. K. Babu, R. K. Kolati, P. Chandra Sekhar, N. Murali Krishna, S. Vikruthi and B. Rajeswari, “Computer-Assisted Leukemia Detection and Classification using Machine Learning,” 2024 International Conference on Expert Cloud and Applications (ICOECA), Bengaluru, India, 2024, pp. 1021-1021.

[2] E. Bastug, M. Bennis, M. Medard, and M. Debbah, toward interconnected virtual reality: Opportunities, challenges, and enablers, IEEE Commun. Mag., vol. 55, no. 6, pp. 110–117, 2017.

[3] N. X. Chen, Y. Yang, T. Zhang, M. T. Zhou, X. L. Luo, and J. K. Zao, Fog as a service technology, IEEE Commun. Mag., vol. 56, no. 11, pp. 95–101, 2018.

[4] B. Yang, X. L. Cao, X. F. Li, Q. Q. Zhang, and L. J. Qian, Mobile-edge-computing-based hierarchical machine learning tasks distribution for IIoT, IEEE Internet Things J. vol. 7, no. 3, pp. 2169–2180, 2020.

[5] M. K. Jia, J. N. Cao, and L. Yang, Heuristic offloading of concurrent tasks for computation- intensive applications in mobile cloud computing, presented at the 2014 IEEE Conf. Computer Communications Workshops (INFOCOM WKSHPS), Toronto, Canada, 2014, pp. 352–357.

[6] W. W. Zhang, Y. G. Wen, and D. O. Wu, Energy-efficient scheduling policy for collaborative execution in mobile cloud computing, presented at the 2013 Proc. IEEE INFOCOM, Turin, Italy, 2013, pp. 190–194.

[7] J. Wang, J. Hu, G. Y. Min, A. Y. Zomaya, and N. Georgalas, Fast adaptive task offloading in edge computing based on meta reinforcement learning, IEEE Trans. Parallel Distrib. Syst., vol. 32, no. 1, pp. 242–253, 2021.

[8] J. P. Champati and B. Liang, Semi-online algorithms for computational task offloading with communication delay, IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 4, pp. 1189–1201, 2017.

[9] B. D. Shang, L. J. Liu, and Z. Tian, Deep learning assisted energy-efficient task offloading in vehicular edge computing systems, IEEE Trans. Veh. Technol., vol. 70, no. 9, pp. 9619–9624, 2021

[10] J. Liu, Y. Y. Mao, J. Zhang, and K. B. Letaief, Delayoptimal computation task scheduling for mobile-edge computing systems, presented at the 2016 IEEE Int. Symp. Information Theory, Barcelona, Spain, 2016, pp. 1451– 1455.