MSc Research Project

Temporal Logic Algorithms for Multiple Users and Services in Mobile Edge Computing

Programme Name

Abdul'Rauf Ijaoba
Student ID: x19232292

School of Computing
National College of Ireland

Supervisor: Jitendra Kumar Sharma

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Abdul'Rauf Ijaoba
Student ID:	x19232292
Programme:	Programme Name
Year:	2021
Module:	
Supervisor:	Jitendra Kumar Sharma
Submission Due Date:	16/08/2021
Project Title:	Temporal Logic Algorithms for Multiple Users and Services in Mobile Edge Computing
Word Count:	Jitendra Kumar Sharma
Page Count:	26

Signature:	
Date:	16th August 2021

Temporal Logic Algorithms for Multiple Users and Services in Mobile Edge Computing

Abdul'Rauf Ijaoba
x19232292

Abstract

Edge networks in mobile computing have alternated a new dimension that enables end users avoid delay in server response and delivery and delivery by bringing data and processing closer to user. The possibility of network distribution has also created leverages for multi-users in cloud environments, thus processing users request withing their variable geolocation. This multi-usage provision in mobile computing fits into its recognition as a 'multi-edge computing'. However, the rapid increase in user traffic, user conglomeration in a network area and mobility of devices are beginning to create a gap in mobile computing. The gap is noted in the time it takes to input data and get responses from the server. Delay, and throughput in cloud-based services are determined by edge server limitations, plural utilization in real time and the area deployment of server for end users. This study thus adopts temporal logic algorithms to enable low latency in multiple user cloud environment and create a functional service for quality user experience, processing simultaneous request prior to time, instance selection and considering mobility path for each end user, considering the computational resources available in the mobile edge server.

1 Introduction

Mobile edge computing otherwise known as Multi-edge computing has become a rising force in today's digital world. The rising is occasioned by a rapid adoption of tech-based communication from transportation to agriculture to education and other phases of human endeavors. In education, the COVID-19 pandemic disrupted physical learning activities and educational institutes and participants shifted unprecedentedly to the online learning mode. The online interactions in classes are growing and this growth is beyond education to other facets of life. Importantly, the consistent use of online services and engagement depicts the precision of technology in this modern time. Traffic in cloud services by end users interrupt information and data flow and the overall user experience in cloud functions.

Mobile edge computing is a fast-expanding aspect of cloud computing. Cloud computing is expressed as a production of low-cost computing capacities and services through combined computational and distributed grids Prasad et al. (2013). The use of the internet for relational processes and interactivity rests solely on the speed of connection and the time mark for responses. In online gaming, classes, transaction and purchases, the real-time presence is remarkable and determines how consistent the user becomes in using the application or platform. The storage and computing of digital activities now banks on the cloud and that has made computation and data storing feasible on the cloud. Cloud computing is essential because it reduces processing burden on end users of an application or a software Ahmed and Rehmani (2017). The computing and storage on the cloud are now higher in wide use because they are easily scalable, available and reduce cost with flexibility.

The resourcefulness of instant response on online platforms and applications is attached to the condition of latency, connectivity, interactivity, processing and storage. The latency and response rate from the IT infrastructure (compute, storage and networks) influence the end-user experience whether on locally hosted or cloud-based server system. The effect is that, users become dissatisfied with the system when responses, displays and information from a time-based application takes longer than usual in delivery creating delay or discrepancy in received information. Hence questions the authenticity of the received information. Cloud computing creates a leverage for information and data processing on applications and platforms through system segmentation into, applications, platforms and infrastructure Prasad et al. (2013). The three segments run around data, runtime, middleware, operating system, virtualization, servers, storage and networking Prasad et al. (2013). The platform segment involves more of the applications and data while the infrastructure segment revolves around applications, data, runtime, hypervisor, and operating system. Cloud computing is essential in online processing and activities because it creates a variety of workloads control and customization, speedy processing on virtual machines, measurable and provides instant system evaluating resources Prasad et al. (2013).

The challenges in the online utilization of server-backed platforms and applications are increasing than ever. The increase is traceable to the catalyzing growth of end-users who adopt mobile computing resources and cloud activities on a rapid basis. The depletion of the physical learning mode enables a greater use of online learning. This results in a rise of the information and data available on schools' platforms. It thus leads to a need to host teaching-learning activities on a platform that works well in connectivity, input and output delivery. As a result, it is one of the elements influencing the expansion of cloud computing. The availability of multi-player option of cloud games has increased the potential participants and the interestingness of the games and thus creates a sporadic increase in gamer and game development. Notably, the increase in the users of applications and cloud activities results in traffic which generates an overall end-user dissatisfying effect in multi-access of cloud services occur when data transmission to end user is slow, and data delivery is taking more time due to a bottle neck in the underlie deployed infrastructure. In server-based activities, slow data transmission due to network reduces content quality and transmission speed. Fast and instant responses in gaming makes quality gaming experience possible. However, cloud computing has made it possible to monitor traffic while creating functional effects to reduce latency. Despite this possibility, cloud service usage is overwhelmed with information processing difficulty from end-user to server, privacy issues and data insecurity.

The research-intensive response and digital development creation for transmission and delivery in cloud service have been identified as cloud services to mitigate above challenges. Mobile edge computing is the development that brings the storage and computing, as well as networks and applications closer to the end user. The adoption of mobile edge computing is occasioned by the insufficiency of cloud computing for low latency, mobility support and location awareness Ahmed and Rehmani (2017). The use of mobile edge computing is described as the deployment of edge networks to bridge the gap between the end user and the cloud services Ahmed and Rehmani (2017). The working system of the mobile edge computing is premised on the integration of processes, responses and storage on cloud activities Ahmed and Rehmani (2017). The target is to lessen latency in order to quicken access and delivery whether on live transmission of digital services or video gaming from the cloud—otherwise regarded as gaming on demand.

Mobile edge computing has created new fora of possibilities for cloud services developers and end users as users can now have network system for applications and platforms within their boundary or immediate environment. This is targeted at increasing the speed of cloud processing and graphical and informational responses to keep the information real time and maintain its authenticity. Mobile edge computing does not delete the existence and utilization of cloud computing resources but extends it closer to end users and provided a leverage which is faster, easily accessible and can generate real-time transmission and delivery. Connected servers can generate instant responses without processing delay. This possibility is available through network and server proximity to end users, mobile support and geography-based use of edge networks for computing Ahmed and Rehmani (2017). Mobile devices, laptops and personal computers can thus be devised from any location to access cloud services in an integrated format. Mobile edge computing depends on the fastness and the accessibility platforms' servers. The speedy connection and delivery have been identified with the evolving 5g technology and thus creates a new spectrum that addresses connectivity, throughput and delay in cloud computing Ahmed and Rehmani (2017).

Mobile edge computing functions for delay sensitivity and context-based applications Ahmed and Rehmani (2017). The sensitivity and context in this usage is that, edge network utilization reduces delay in cloud services and makes available optimal resources that are context sensitive depending on the qualities and functions of a server. Mobile edge computing is identified as capable of addressing the usage and management challenges in cloud computing because it is an open system that centralizes the end users while making the network mobile for their use. The usefulness of mobile edge computing is placed in its use for advanced storage as opposed to primary data storage on cloud, edge analytics as opposed to using cloud data centers, real time transmission and delivery as opposed to using back-based networks and effective cloud service management as opposed to primary gateways for control Ahmed and Rehmani (2017). Mobile edge computing saves cost in infrastructure hosting and this requires less floor availability. Hyperconverged infrastructure are also adopted in data-rooms instead of full-fledged data center.

1.1 Background

The distributed pattern of information use in cloud computing is identified as disruptive and time-consuming given the present cloud-based services Sri[1] and Vemuru (2019). In a multi-client cloud environment, it is simple to process and store data. However, it is more difficult to access the data due to user and engagement traffic. The difficulties with using the cloud service in a multi-user environment have been identified to include network problems, control issues, information erasure difficulties and respectability in reviewing Sri[1] and Vemuru (2019). These user environment difficulties couple with mobile devices and geographical difficulties and all these affect the real-time use and transmission on edge networks. Temporal logic algorithms have been identified as alternative solutions for a speedy connectivity and transmission on cloud services. Strategies including accessibility, respectability and confidentiality have been identified for improving edge network services and functionalities Sri[1] and Vemuru (2019). Confidentiality in edge computing is seen in data safety for developers and end users of edge computing services. Accessibility in edge computing means that users are able to explore mobile-based services without restrictions on content and data. Network latency is reduced when the algorithms are combined to improve connection for users. Authentication and access control have also been included as areas for improving cloud services and utilization Sri[1] and Vemuru (2019). Hence, user experience is improved when users can access a cloud service severally without space, storage and usage restriction. The simplified connection, access and usability of cloud services encourage users to continuously use the services. It is vice-versa when the cloud service functionalities are subpar to the multi-user environment.

Virtual machine introspection has been suggested as capable of limiting user difficulties on the cloud service utilization but has also been found to occupy space as more hosting area is required of the cloud methodology Sri[1] and Vemuru (2019). However, temporal logic algorithms work to enhance time reduction and improves instant data input and transmission for users in edge networks and computation capacity. User mobility and service invocation are combined with structured algorithms to enable the unrestricted connection and utilization of cloud services on edge networks. This is done by hosting the server close to the end user. Proximity of server to end users reduces network latency in edge computing.

1.2 Problem Description

Cloud computing has reshuffled the processing and storage patterns in online interactivities and usages. From online teaching to gaming and social instant communication, processes and storage have been shifted from the manual or primary segment to the fast-paced server system. Digital users be it gamers or online learners appreciate instant data input and responses. The user experiences of a cloud service are formed by the functions of the server especially, the time it takes for their mobile devices to display the information received the server. The more time it takes to load information in graphics, avatar and textual basis, the lesser the user experience becomes. This user experience formation is attached to network latency which has become a rising point of concern in cloud technology. The increase in user traffic is another dimension of difficulty that is being experienced in cloud computing. This difficulty is prevalent in multi-user cloud environment because user traffic influences the accessibility of resources and services available in the server. Mobile edge computing is challenged by mobile constraint which is resource-poor and this affects the information delivery Satyanarayanan (1996). Distributed systems are affected by the independence of mobility in cloud services Satyanarayanan (1996). The speed of connectivity for multiple users of cloud services has also become a point of concern in the cloud technology field. Asides connectivity, security challenge has increased with the proliferation of cloud services users. The result of this proliferation is skyrocket security problems which have been classified into confidentiality, an unauthorized access to peculiar information; integrity, an unauthorized disruption of information; availability, threats to information availability to authorized users; legitimate, ensuring information access only to permitted users and accountability which is linking users and their activities on the cloud services and utilization Balasubramania (2015). Although, mobile edge computing has made it possible to lessen the difficulties faced by developers and end users in cloud computing, there exists other difficulties which are threats to impressive user experiences, functional cloud service utilization, speedy connection, access and utilization and edge network performances. This study therefore seeks to create improved edge network utilization in mobile computing through the adoption of temporal logic algorithms. The target of the study is to improve overall user experiences and enable effective cloud service functions and utilizations.

1.3 Objectives of the Study

This study aims to devise solutions to latency problems in mobile computing given the proliferated adoption and use of cloud-enabled services. The study is targeted at using a novel approach to reduce latency in cloud compute and storage and improve real time transmission and delivery towards a functional and satisfying user experience.
The study specifically seeks to:

- Improve network operations for cost efficiencies using temporal logic algorithms.
- Improve distributed computing capacity using temporal logic algorithms in edge networks.
- Create mobile computing solutions for functional capacity and effective user experience.

1.4 Research Questions

User experience in cloud services is formed from the response time, delay and throughput. The trio of these factors are attached to latency in cloud-oriented activities, low latency improves user access to information directly from the cloud. Cloud computing enables seamless input and accessibility to data on server but increased use of cloud services has occasioned traffic. Traffic in cloud computing is determined by the latency rate of the platform—a low latency speeds input and data delivery for users. Cloud service utilization is influenced by a high number of participant usage. This has been identified as leading to slow server responses, low graphical display and information delivery especially in a multi-user cloud resourced environment. The development and rapid usage of cloud services have been covered by growing concerns for how user experience can be improved. The study aims to answer the question:

- How can temporal logic algorithms be used to improve multiple users and services in mobile edge computing?

2 Related Work

2.1 Addressing Latency in Mobile Edge Computing

Due to high latency constraints and limited resources, effective service scheduling in the edge computing scenario has been quite resourceful, the idea of score-based algorithm was implemented for the evaluation of computational requirement and networking on available VM's then proceed to deploying services to the most suitable one. Researchers have tested the use of the novel idea for solving latency and the results show that, simulation of experiments in live video comes with high possibilities and improvements Aral et al.

(2019). The comparison of cloud computing and content delivery in latency-conscious and data-conscious environment proves that algorithm edge-based scheduling is an effective solution for latency in mobile computing Aral et al. (2019).
Smart technology devices are revolutionizing the way of doings things universally. Hence there is a rapid global demand for Internet of Things (IoT). However, latency has become a challenge in IoT and mobile computing and requires immediate and effective solutions Saranya et al. (2019). Although, different scholars have experimented and proposed methodologies and techniques to achieve low latency in cloud computing. Further latency-solving methods are itemized. The methods include cloud-fogging which is the use of genetic algorithm to achieve load balancing and lessen service latency; review of edge paradigms which is the presentation of communication latency that depends on 5g network services; latency offloading algorithm is used to reduce traffic and improve delivery rates on cloud-based activities; and hierarchical naming method, an efficient information-based method that enables fast content cloud delivery and security and measurability Saranya et al. (2019). The researcher added that latency can be reduced in mobile edge computing by proximity-based computing, coded computing, proactive computing, edge machine learning, and the use of matrix and generic operators Saranya et al. (2019). Using multi-layer latency aware workload assignment strategy (MLAWAS) for allocating mobile users' workloads can be used for optimal cloudlets. The experiment of the proposed method shows that using MLAWAS yields a faster response compared to other strategies. Ensuring a faster network access is premised on bridging the gap between mobile internet users and access to the server. The distance between mobile users and server creates high end-user latency but the geographical distribution of cloudlet networks can reduce the latency Sajnani et al. (2018).
Mobile edge computing is described as a promising and fast-rising in the technology space. It is contended that latency has become the main challenge reducing user quality experience and the effectiveness of mobile computing and cloud-based activities. It is proposed that the use of mobility-aware hierarchy as a framework for mobile computing. The researchers applied the game-based theory to reduce energy consumption for users, and improve mobile computing processes for users. The experiment shows that mobility aware hierarchy in mobile edge computing brings effective results through functional activities. The experiment applicability covers Internet of Things and is targeted at reducing content delivery time Zhang et al. (2018).
The challenges of mobile computing include mobile devices restriction Qi and Gani (2012). This restriction connotes that mobile devices are short of processing capacity required for certain levels of mobile computing Qi and Gani (2012). It also includes that network access to the server from phones, laptops and personal computers is restricted because of capacity. It is recommended that upgrading bandwidth for wireless connection, deploying application processes on edge, duplicating mobile devices to cloud using virtualization, and using mobile terminal optimization for application push will improve mobile access to cloud services and overall user quality experience. Also, it is suggested that the distribution of applications for optimality in data intensive and compute-intensive cloud services Qi and Gani (2012).
Mobile edge computing although emerging has gained a wide acceptability. The acceptability results in high use of servers and cloud activities. However, the challenge to effective use of cloud services has been traffic. Traffic affects the time it takes for information to be relayed to the end users in cloud activities. Mobile edge computing is the solution to traffic challenges in edge computing and mobile delivery of cloud contents to end users.

Mobile edge computing addresses storage and computation difficulties in cloud activities and application developers leverage on edge opportunities to improve real-time information relay from server to end user and vice versa Khan et al. (2019).
Researchers have developed a novel architecture to ensuring time responsive cloud processes involving mobile devices. The developed architecture is designed to enable the distribution of loads in computing and delivery to mobile devices users Lee et al. (2020). The distribution of computing loads in edge computing using the architecture reduces delay in data conceptualization, processing and information delivery from cloud system Lee et al. (2020). The main idea of the proposed mobile edge computing architecture is to enable rapid response which gives impressive user experience in cloud services Lee et al. (2020). The architectural design includes a service requestor that performs request functions by request, access in mobile computing which aids the fragmentation of data for simplified processing and delivery to end users, and mobile devices which are used as tools for the information fragmentation processes Lee et al. (2020). To improve system response in edge computing, it is proposed that, the service requester sends a metadata and analyses the request and generates fragments of information Lee et al. (2020). The segmented information is processed and task offloading is performed and notably with a reduced time for processing and delivery.

[1]

2.2 Toward Improved User Mobility in Mobile Edge Computing

Mobile edge computing bridges the gap between end users and servers Lee et al. (2019). The possibilities of faster cloud service access and functionality are a boost to addressing rising challenges in cloud computing. However, the present system designed to relay information from the server to the end users is affected by distance difficulties Lee et al. (2019). The availability of distance between cloud users and server results in delay in response delivery to end users Lee et al. (2019). Mobile edge computing has provided increased leverages to reduce network latency in cloud computing. However, the delay difficulty remains and lessens the quality of experience for end users. The novelty of mobile edge computing is bringing the cloud infrastructure close to the users and reduce the distance in processed data delivery Lee et al. (2019). To improve access to a faster cloud service, zone-based multi-access has been introduced to cloud computing for a mobile management. Zone-based multi-access has been introduced to arrange mobile edge computing server to allow users access effective server and content delivery Lee et al. (2019). Zone-based mobile edged cloud services are addressing network latency in cloud computing while also creating functional access to users even when they are mobile; an additional feature which mobile edge computing does not offer Lee et al. (2019). Zone-based multi-access is improving service and cloud optimality for result-giving user experience.
Wireless collisions among mobile users can be optimized to reduce delay in mobile computing Wang et al. (2018). In a multi-user access zone, assigning different function to the nodes has been found helpful in improving response speed. This possibility is enabled by the identification of the function assignment as a constraint that is accounted for by lightweight algorithms Wang et al. (2018). Task information is introduced with the

[1]Like this one: `http://www.ncirl.ie`

addition of information station to enable mobile users access cloud services. The use of task information is practiced by a simulation experiment and the result shows that, using small base information station is efficient in reducing delay in mobile edge computing Wang et al. (2018).

In Mehrabi et al. (2019) caching strategies can be used to reduce delay in cloud transmission in end-to-end. Cache architecture is put into three layers to tackle network difficulties. Tasks are shared into paths and directed to specific functions with real-time transmission focusing on industrial domains Mehrabi et al. (2019). To solve data fetching problem, Hungarian algorithm is introduced in Mehrabi et al. (2019) and this strategy is used to achieve improved transmission process, and advance the processing strength of nodes and edge cloud servers Mehrabi et al. (2019). Smart vehicles have been introduced to engage the sharing of caching agents in a 5g network-based method. This method is put into sharing information using base stations and the compression of mobile edge computing servers Mehrabi et al. (2019). Efficiency in mobile edge computing can be ensured by optimizing, computing and caching smart vehicles and base stations Mehrabi et al. (2019).

Algorithm application to solving latency problems in cloud computing has been functional. In Wei et al. (2020) a 13.21 percent of service response was realized through the selection of algorithm back-propagation based—a result entirely different from other algorithm-centric interventions. The strategy which yielded an improved service response and delivery is backed by processes including the idealization of a geometric model to focus on user mobility, and making of predictions from trajectory mining data on the users. Cache services and result prediction for users are patterned using base station with algorithm-based allocation to the patterns Wei et al. (2020). A cache selection based on neural network is the support basis for the novel strategy and the result shows that response service is improved with users on movement Wei et al. (2020). In a case of no mobility prediction for users, a 15.19 percent of result is identified; this shows the importance of caching and prediction in improving accesses, usages and functions in mobile edge computing. In Xia et al. (2020) the importance of caching to solving latency problems in cloud computing is validated. Cache-free approach to reducing latency has been tested effective compared to cache-focused approach to lessening network latency Xia et al. (2020). While caching is significantly helpful in reducing latency problems in mobile computing, cache-free approach reduces computational risks because only a half of the transmission time is used for the network. This results into speedy processing and response delivery for users. Cache-based strategies are delayed in execution because full signals are not explored for direct signals and relay links Xia et al. (2020). In signal-to-noise ratio, cache free network is relatable and resulting Xia et al. (2020). Destinations are provided for cache-aided and cache-free approaches; cache-aided approach is based on the chain gain of relay link and the channel gain of relay link and the gain of direct link Xia et al. (2020).

Ugwuanyi et al. (2019) agree that mobile edge computing is highly important and having increased use and participation. Direct delivery of content is enabled through the edge without having to get data from the remote server Ugwuanyi et al. (2019). The benefit of multi-access edge computing is that end—users are close to functional services. Polynomial fit is used in Lagrange interpolation to increase caching efficiency and data access for mobile edge computing [18]. It is shown that, central processing utilization are speedy when involved in algorithm-based replacement Ugwuanyi et al. (2019).

Peng et al. (2018)The essence of mobile computing is identified as related to end-to-end users. Although, mobile edge computing saves the life battery of mobile users through cloud integration for mobile networks, execution delay problems exist for users Peng et al. (2018). To solve delay difficulties in mobile computing, computation offloading and data offloading have been presented as functional taxonomies which are also effective for mobile users Peng et al. (2018). A solution that addresses execution delay has been formulated as integer programming and developed into a two-phase optimization Peng et al. (2018). The optimization is targeted at satisfying all the computing requirements that are needed to reduce delay in computing resources across the edge nodes. The edge nodes are based on local cloudlets and global cloudlets that have proven effective to achieving the targeted results Peng et al. (2018). The integration of wireless power transfer into access points has been found functional in improving sustainable energy consumption for power-restricted end users. This functionality comes with the direct application of the unmanned aerial vehicle which are compatible for flexible computing resources for end-to-end usages Hu (2020). The inclusion of the unmanned aerial vehicle is notable because it improves system performance while also enabling multiple input and output capacity in cloud services Hu (2020). Small base stations and macro base stations are also used to manage data offloading toward reducing cloud computation difficulties Hu (2020). Algorithm-backed strategies are becoming progressively significant in solving high energy consumption complexities in mobile edge computing Hu (2020).

2.3 Service in Instance in Mobile Edge Computing

Experiments in Zou et al. (2021) show that service instance in mobile edge computing can be improved. In Zou et al. (2021) service invocation was adopted in improving user experience with the experimental conception of limitation on edge server based on computation capacity, geographical variations in user mobility paths and user cluster leading to service delay. The three factors were input into a model for addressing delay in data transmission in mobile edge computing Zou et al. (2021). Although, mobile edge computing has been significant for end users' time of service invocation, multiple access to the server has become a challenge; mostly resulting to transmission delay and low quality of experience for end users Zou et al. (2021). Improvement by service invocation is premised on algorithm-centric approach that is time aware and regarded as significant over the existing baseline in mobile edge computing Zou et al. (2021).
Sharing mobile service through edge nodes is identified as a smart approach to improving data transmission in mobile computing Roy et al. (2020). However, continuous deployment of service on edge nodes increases the overall cost of network in the mobile service Roy et al. (2020). Cloud infrastructure has been proposed as an alternative solution to reducing network delay without adding extra infrastructure cost on the services **?**. The cloud infrastructure is patterned on Multi-objective Integer Linear Programming (MILP) by integrating prediction model for users' paths Roy et al. (2020). An artificial intelligence-based solution is designed to cater for service instance without additional firewall cost. The AI design is termed Binary Particle Swarm Optimization (BPSO) algorithm and it achieves optimized performances "within a polynomial time" Roy et al. (2020). The design improves quality of experience for end users and most effective when backed with a 5g network Roy et al. (2020).
In *mobile* (n.d.) service improvement is positioned on using capacity-limited based stations to improve data movement in mobile edge computing. Multiple deployment of

service instance on base stations may result in lags for end users. To avoid lagging, a service request forwarding is used to address problems including correlations in base stations, joint placement and allocation of resources designated for mobile computing *mobile* (n.d.). A decentralized algorithm complemented with matching is functional in meeting the service needs in cloud computing especially for high-size networks *mobile* (n.d.).

Year	Algorithm	Core Idea	Advantage	Limitations	Ref
2019	score-based Algorithm	Scoring VM's based on service network, computing requirements, and reliability. Hecnce assigning VM based on its appropriateness to host the scheduled service.	Speedy data transmission for end-users mainly on output relays	Infrastructure not detailed	[6]
2020	Genetic algorithm	Cloud fogging for solving latency problems	Reduces network latency with load balancing	Bandwidth/distance not capitalized	[7]
2018	Multi-layer Latency Aware Workload Assignment	Optimal cloudlets allocation of server strategies	Faster response among competing strategies	Geographical distribution based on cloudlet network	[8]
2018	Game-based theory	Mobile aware hierarchy for functional results	Reduces network latency	Energy consumption does not cover all mobile devices	[9]
2012	Data intensive strategy	Mobile terminal optimization	Improves overall end-user quality of experience	Generalized data utilization for improvement	[10]
2018	Cloud content method	Solving computation and storage complexities	Real-time information processing and delivery	Focuses on end-users	[11]
2020	Data fragmentation	Distribution of competing loads using architectural nodes	Improves system response in mobile computing	Over information segmentation leading to higher service cost	[12]
2019	Zone based multi-access	Cloud optimality and improvement	Gives multiple access to end users; cluster access to mobile server	Mobility delay and multiple access	[13]
2018	Lightweight algorithm	Wireless collisions	Stability of cloud network with improved content delivery	Base-by-base server allocation	[14]
2019	Hungarian algorithm	Caching architecture/ caching strategies	Real-time transmission on industrial domains	Based mainly on 5g network	[15]
2020	Geometric model	Cache-free approach to latency reduction	Predictive access and reduced network latency for end-to-end data movement	Caching strategies in focus	[16]
2020	Cache aid/cache free comparison	Caching approach in cloud computing	Direct improvement of cloud services and communication	Attention on relay links	[17]
2020	Content optimization approach	Local and global cloudlets for functional mobile cloud services	Lessens power consumption for end-user mobile devices	Unmanned aerial vehicle concentration	[18]
2018	Flexible computation	Wireless collisions	Towards energy saving for mobile users in edge computing	Wireless connection and server processes	[19]
2020	Unmanned aerial vehicle	Enable multiple input and output capacity	Reduce energy consumption difficulties in mobile edge computing	Aerial access as focus	[20]
2021	Invocation-based algorithm	Node-centric server transmission	Time-aware and speedy delivery on mobile computing	Experimental approach limited to computation and geographical problem solving	[21]
2020	Multi-objective Integer Linear Programming (MILP)	Multiple server deployment on edge nodes	Improves network access and delivery with less firewall cost	Requires 5g network	[22]
2018	Decentralized algorithm	Multiple server deployment	Capacity for high-size networks in mobile edge computing	Sever allocation requires high cost	[23]

Table 1. Summary of review literature

3 Methodology

3.1 MobFogSim

This study adopts MobFogSim for migration and mobility in fog computation. MobFogSim advances iFogSim as a model for mobility in devices for fog computing. MobFogSim works by:

- Adhering to processing instructions.
- Providing user mobility input directory from dataset.
- Initializing CloudSim before entity creation.
- Creation of broker.
- Use of one virtual machine.
- Application creation.
- Network configuration.
- Starting simulation.
- Realizing results after simulation *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.).

In Puliafito et al. (2020) MobFogSim is identified as a network edge cloud extension that solves mobility difficulties in cloud computing. Although, mobile computing is advantageous and has multiple access for users, network mobility and latency have become challenges to functional and effective real-time cloud computing. Fog service migration is thus the means of addressing delay in network transmission and reception as users move from one end to another Puliafito et al. (2020). MobFogSim tops iFogSim because it enables the evaluation of service migration in computing environment—the reason for its involvement in this study Puliafito et al. (2020). Network slicing is workable in reducing latency and ensuring optimality use of cloud service. Gonçalves et al. (2020)combine network deployment with MobFogSim to enhance network performance in term of speed and service quality. MobFogSim is functional in planning and managing cloud computing processes and services for an overall user quality experience Gonçalves et al. (2020).

3.2 Java Programming Languages

Java is a programming language that works for multiple user access for software. The language is designed by James Gosling in 1991 with the aim of coding once and running everywhere *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). Using Java enables functional running of application on mobile devices, medical devices, and personal computers*MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). Java runs on the rules and syntax and C and C++ languages *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). Java is portable and thus a suitable programming language for the diagnostic application which is the focus of this study. Java's portability means that codes are easily moved to mobile devices and easy to run for end users *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). Java is preferred for this design because it can be run anywhere without restrictions and can also be compelled; whereas, JavaScript does not need to be compiled and only runs web browsers *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). Java has had a reputable use for over 20 years and is competitively functional to designing a widely accepted diagnostic public-centered application. Java codes are Windows, Linux and macOS and mobile phones compatible when used for web application designs *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). Codes on Java are turned into bytecodes that are deployed in Java environment through Java Virtual Machine *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). The use of Java Virtual Machines enables an anywhere run of Java-based mobile application—a suitable design for multi-access public diagnostic mobile application. For Java software, the development environment works with Java API and Java Virtual Machine for functional running *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). Java bytecodes are interpreted by the virtual machine for making a user-enabled mobile application. The decision for Java in this application design is because Java performs optimally for android devices which host the largest group of persons using smartphones. The software design combines inter-platform connectivity, simplicity and functional security for end users' devices especially *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.). Java is optimized on visibility and easy to manage; making it a choice for a public-enabled diagnostic application. Java is scalable, adaptable and operable especially in multiple user environment *MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing* (n.d.).

3.3 Temporal Logic Algorithm in Mobile Edge Computing

The temporal logic algorithm is deployed for multi-access edge computing to reduce the possible time it takes to get responses on requests made on the server from end users. The algorithm in use was first introduced by Pneuli. The algorithm was later developed into temporal logic action which is a combination of logic and action Pnueli (1977). The algorithm for the implementation of this model is synthesized from the Temporal logic formula.
Temporal logic relies on recursion such that the loop varies from number of users n up to the first user. The time taken to complete a request is taken as variable t.
Input: A query q

Statement	**... is true just when**
U$\models$ a[u]	V(a,u)=true
U $\models \neg\ \phi[u]$	not U $\models \phi[u]$
U $\models (\phi\lambda\psi)[u]$	U $\models \phi[u] andU \models \psi[u]$
U $\models (\phi\nu\psi)[u]$	U $\models \phi[u] orU \models \psi[u]$
U $\models (\phi \to \psi)[u]$	U $\models \psi[u] ifU \models \phi[u]$
U $\models$ G$\phi[u]$	U $\models \phi[v] forall\ v\ with\ u < v$
U $\models$ H$\phi[u]$	U $\models \phi[v] forall\ v\ with\ v$¡$u$

Table 2. Temporal Logic operators In this research, Linear Temporal Logic will be used to implement Service Instance Selection Problem (SISP). SISP is properly explained in details in definition 6 in the definition section.
section 2.

4 Design Specification

In this section, we introduce the concept of mobile edge computing (MEC) system architecture and briefly discuss all the components that are integrated to deploy a MEC system, then briefly discuss and work through some examples to illustrate the problem of MEC server instance selection when user initiate a request and finally define concepts and formulae that will be used as a guide in the implementation.

subsectionMobile Edge Computing System Design Mobile Edge Computing (MEC) is European Telecommunication Standard Institute (ESTI) defined network architecture that enables cloud computing capabilities and IT service infrastructure at the edge of a network. The primary goal of the architecture is to run applications and process user requests close to the user making the request via a base station (BS) to reduce network congestion, reduce request response time thereby reducing latency and improving the application performance and the overall user experience with respect to the application. Mobile Edge Computing system has four major components but it is not limited to these namely Cloud servers, a designated base station (BS) with running instances of edge servers, a collection of services running on the cloud servers and on the edge servers and the mobile user that subscribe to the services deployed on the running server instances. In a MEC infrastructure, the BS contains edge servers and the servers are running the MEC applications that handle mobile users' requests. The BS are generally mapped to a physical location operating within a defined radius. MEC system is an aggregation of many BSs which are network access points for the mobile user and these aggregate

BSs within marked geographical space will be designated as a region in MEC system. The BSs in a regional MEC system are logically interconnected to each other and they communicate with each other. The interconnectivity of BSs will suggest that there will be transmission of data and services or requests between BSs as a BS has physical limitation or a boundary or radius where it operates and responds to mobile user request, mobile user request can be transferred from one BS to another one directly or adjacent to it. The time taken to process the mobile user request will be increased due to data transmission between the BSs. The BS time for executing a service is dependent on the number of requests it is processing at a time. As the workload of a BS increases with many mobile users simultaneously requesting services, the execution time or response time increases accordingly, especially when the resources of the server instances are stretched to a certain threshold.

Figure 1 below shows the architecture of the mobile edge computing (MEC) system.

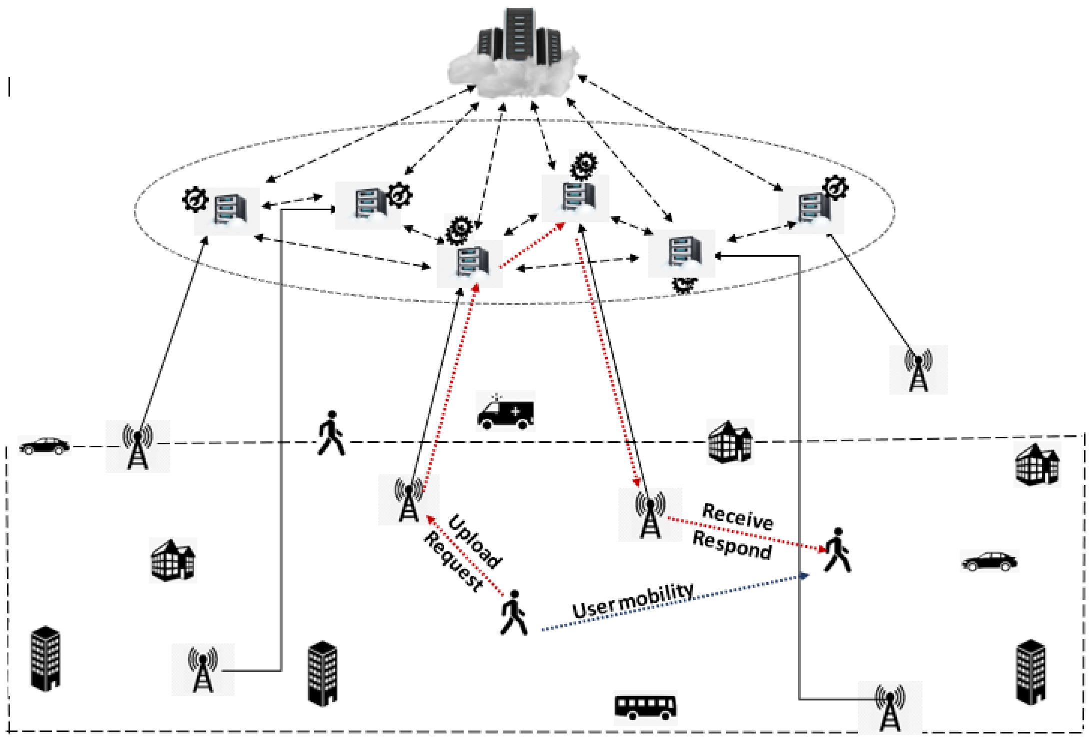


Figure 1: Architecture of a Mobile Edge Computing System or Infrastructure

4.1 Mobile Edge Computing Use-Case and Application

As a study use case for this research, Beaumont Hospital Ambulance (BHA), in a rescue mission is moving a patient named Jerry Tom to the hospital. Jerry Tom is on a life support device that monitors his vitals and sends them to a cloud service that helps to stabilize Jerry Tom. The sensor device on BHA needs to update the vitals of Jerry Tom to the cloud from the pickup point to the hospital.
We assume that BS1 and BS2 are base stations that the ambulance passes on its way to the hospital. Assuming that it takes 35s to move from BS1 to BS2, the data to be transmitted is 10MB and the response data is about 30MB. The data transmission rate between BSs is 2MB/s. The data transmission rate between BS1 and the ambulance is 10MB/s and data transmission rate between BS2 and the ambulance is 20MB/s.
Using the scenario described above, if BS1 requires 5s to process BHA requests and BS2 10s to process the same request. In any case we will have two different outcomes if the request was sent to the different BSs. If the BHA request is sent to BS1, the request is uploaded to a server instance at BS1 and when the processing is completed, the response is downloaded to the BHA. The execution time (t) for the request is

$\boldsymbol{t = t0 + t1 + t2}$

where

t0 is the time it takes to upload the request,

t1 is the request processing time and

t2 is the time taken to download the response to the request.

Applying execution time theory to BHA request to BS1, we have 10/10 + 5 + 30/10 = 9s as one outcome, or the other outcome is that BHA request is allocated to the BS2. BHA uploads the input data to BS2 by the path R => BS1 => BS2 and then receives the returned data in BS2, because BHA data has already arrived at BS2 after the request is executed by the instance on BS2. The response time for BHA's request is 10/2 + 10 + 30/20 = 16.5s. From the execution time of both outcomes, BS1 is the best option for BHA without consideration of the other mobile edge users.

In practice, multiple mobile users send requests to the same BS at the same time that may interfere with each. This interference can strongly impact the execution time of BS as a result of the increased workload on the BS. Applying this line of thought to multiple requests to BS1 or BS2 from multiple mobile users simultaneously, this increase in the workload of the BS will lower the efficiency of servers in those BS causing an increase in the execution time.

Using Figure 2 below, suppose the self-driving cars were also sending requests to BS1 at the same time as BHA such that the execution time for BS1 12s, and the execution time (t) will have 10/10 + 12 + 30/10 = 16s which gives us the third outcome. Comparing the three possible outcomes, the second outcome is still more expensive in terms of resource consumption; but it can be observed that the time cost for the third outcome is way higher than the time complexity for the first outcome.

From the above use case of multiple users, there is no way to define a constant state for the BSs as mobile users are constantly in motion and the number of requests per BS vary with time. These two factors can impact execution time. In reality, it is difficult to estimate where the response data will be received after the process has been executed. From this analysis, it is safe to conclude that there is no universal optimal solution for the selection of service instances to mobile users' requests. However, this challenging issue can be mitigated by modelling the problem's relevant construct and designing an

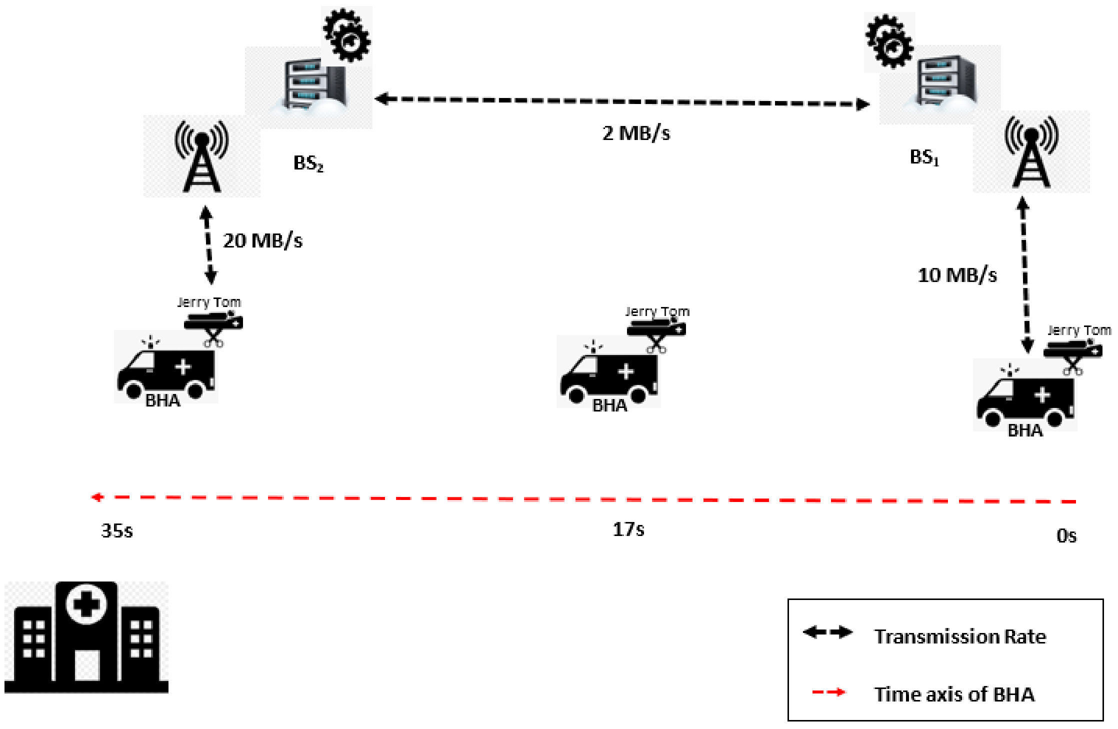


Figure 2: Adopted user-case for service instance selection in MEC system

efficient, yet simple technique to distribute service requests to the appropriate service instance deployed at the BSs in the MEC system.
In selecting a service instance in a MEC system according to the mobile users' requirements, the workload status of the BSs and the location information of the mobile users are variables as seen from the use case work through. In order to better appreciate the service instance selection in MEC systems, we will abstract some of the components and operations in application scenarios into services and deploy several instances of them in a distributed edge computing environment, which will decrease execution time or latency by selecting appropriate, nearby service instances.

4.2 MEC System Modelling

We will attempt to clearly define the key concepts from a component on a service instance selection in a MEC system.

Definition 1 (***Service Request***): A service request initiated by a mobile user can be represented by a four-tuple ***(I, O, C, QoS***), where:

I - is the input parameter

O - is the output parameter

C - the resources needed by the service to execute, where Ci represents all the resources. C1, C2, ..., Cn are the type of resources consisting of memory, processing, storage and network components.

QoS - <Q1, Q2, ..., Qn> is a n-tuple where each Qi is the workload property of a service instance which comprises execution expense, response period, network throughput, reputation, etc.

In this paper, we focus primarily on the response time's QoS property. While users' mobility affects the variation of bandwidth needed for data transmission between users and the edge server which affects data transmission time, the response time is a part of the total period needed to execute a service. Because of the limited coverage provided by the edge servers, if the mobile user leaves the coverage area of one edge server and goes into another area serviced by another edge server, in order for the mobile user to get resources, the mobile user must connect to the edge server. The rate at which data is transmitted between the ambulance and BS1 may be different from the data rate transmission between the ambulance and BS2.

Definition 2 (***Edge Server***). An edge server is depicted by ((***x, y axes***), ***radius***, ***C***, ***r***), where

(x and y axes) - are the longitude and latitude of the edge servers,

radius - is the radius covered by the edge server,

C - is a n-tuple <C1, C2, ..., Cn> that denotes the capacity of an edge server where Ci is the type of resource used by an edge server which includes memory, processing, storage and network resources,

r - represents the average rate of data transmission between mobile users and edge servers.

The arrangement of edge servers is done in a distributed manner, typically close to a base station, and because only a specific geolocation is covered by each edge server. The proximity limitations should be factored. Whereas, each edge server has limited computational capability represented by C. As a result, the total workload spawned by services on a server should not surpass the remainder of that server's capacity. We also assume that the remote cloud servers have ample computing resources and the mobile user can connect to them.

Definition 3 (***Requested Service***). A requested service is expressed as a 4-tuple (***e, t^{cc}, l^{cu}, l^{max}***), where

e - is the server whereby an instance is hosted.

t^{cc} - denotes the computing capacity of an instance. A higher t^{cc} means lower execution of the instance. It is the period quotient used by an instance coupled with the instance having the highest execution and the same workload percentage to perform the same request.

l^{cu} - is the present workload deployed by an instance. Considering a service selection approach to fulfil a user request set R, represented as 0 = {0r1, 0r2...0rn} where every component illustrates a service instance chosen for corresponding user request, $l^{cu}{}_{s}$ is the aggregate of

$l^{cu}{}_{s} = l^{0}{}_{s} + \sum r \epsilon R : \theta r = s \ l^{w}{}_{r}$

where

$l^{0}{}_{s}$ - is the implicit workload of an instance s before any user requests are allocated, as well as the total summation of the workload produced from the user requests carried out on instance s.

l^{max} - is the highest amount of workload an instance can handle where the value of l^{cu} cannot exceed $l^{\ max}$.

Definition 4 (Execution Time). We take a service ws = (I, O, C, QoS) and a chosen server, si. Assume that ws is called at time t1. The time cost of calling the service is given by:

$tc = t_{du} + Q_{ws} + t_{dd} + t_{rt}l_{1} + t_{dt}l_{2}$

where:

t_{du} is the time latency of uploading input data which is denoted by:

t_{du} = D(I)/r_{si}
where
D(I) is the data size of I and r_{si} is the rate at which data is transmitted between the user u and server si.
$Q_{ws}$is the time taken by service ws to respond
t_{dd} is the time latency of downloading output data which is represented as:
t_{dd} = D(O)/r_{sj}
where D(O) is the data size of output data and r_{sj} is the data transmission rate between the user u and the edge server sj.
$t_{rt}I_1$ represents the round-trip latency and I1 is an indicator function which is expressed by:
I_1 = {1 is set if the user connects to the remote cloud, else, 0 is set
t_{dt} represents the downtime as the outcome of service migration, and I_2 is an indicator function demonstrated by
I_s= {1, is set if the service is migrated, else, 0 is set
The computational method for deducing the time cost for invoking a service is stated above in definition 4. Nonetheless, mobile users in the real world always invoke a stream of services while in motion. Thus, the time taken by a user to invoke the entire service composition can be calculated as:
Utc = $\psi_{ws}\ \epsilon S_{ws}$$tc_{ws}$,
where
S_{ws} - is the set of composite services
$\psi - is an operator that integrates the values of time consumption of invoking composite services. Th inte$
To simplify the calculation, we assume that the combined services are in a successive execution path, we only apply the summation (?) integration rule. Therefore, we can get the multi-user mobility-aware time latency computation as follows:
SUtc = $^U \sum_1$ Utc
Definition 5 (Mobility Path). Considering a mobile user, mobility path constitutes a collection of discrete points in a mobile user's location which are ordered sequentially. Mobility path comprises of lines between two adjacent points which can be expressed as
{(T_i,$p^l{}_i$)}q_i= 1 , where
i - is the order number of the discrete time segment
T_i = (ti, ti+1) is the time segment of the ith location point
$p^l{}_i$ - is the coordinate of the ith location point
Definition 6 (Service Instance Selection Problem). Assuming that a group of users simultaneously submit their requests, the SISP is required to choose an ideal instance for each user's request; taking into account [21].

1. the transition of multiple users
2. the varying workload of each instance
3. transmission rate between edge servers.

This problem is expressed as a 5-tuple SISP = (R, P, C, E, S) where

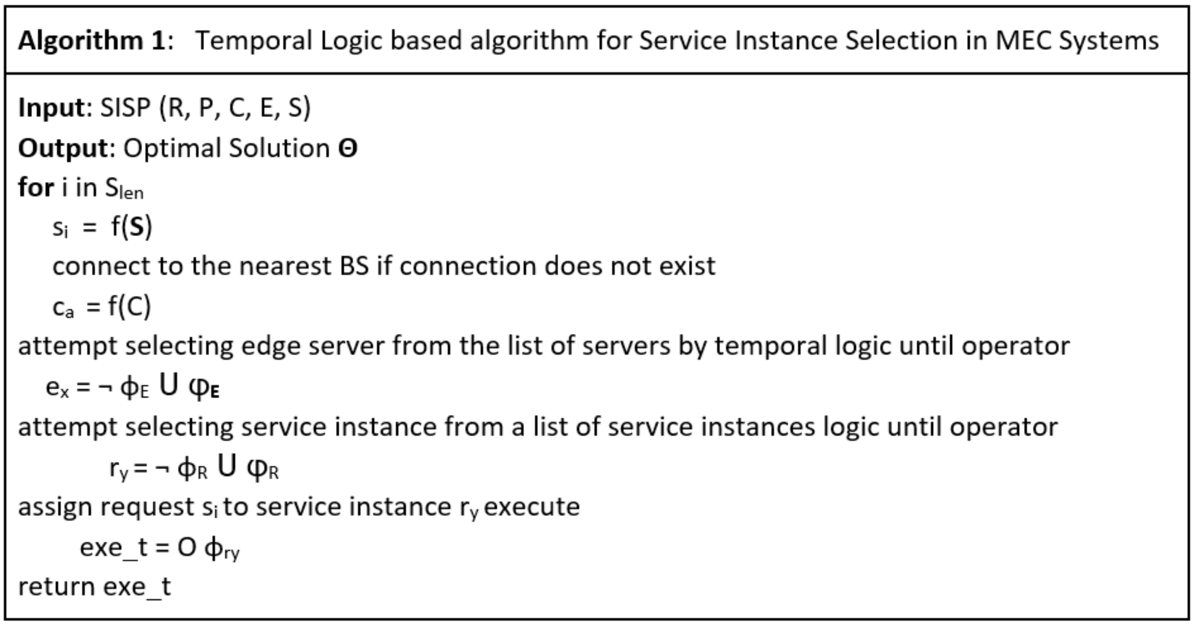
R - is a set of requested service $(r_1, r_2, \ldots, r_n)$.

P - is a set of mobility paths of n users, where each path correlates to a user's course direction between edge cells $(P_1, P_2, \ldots P_n)$.

C - denotes edge cells $C = (C_1, C_2, \ldots C_n)$ ***and the corresponding access points as well as their transmission speeds relative to the users' requests.***

E - this represents n edge servers $E = \{e_1, e_2, \ldots e_n\}$ ***and the transmission rate between them.***

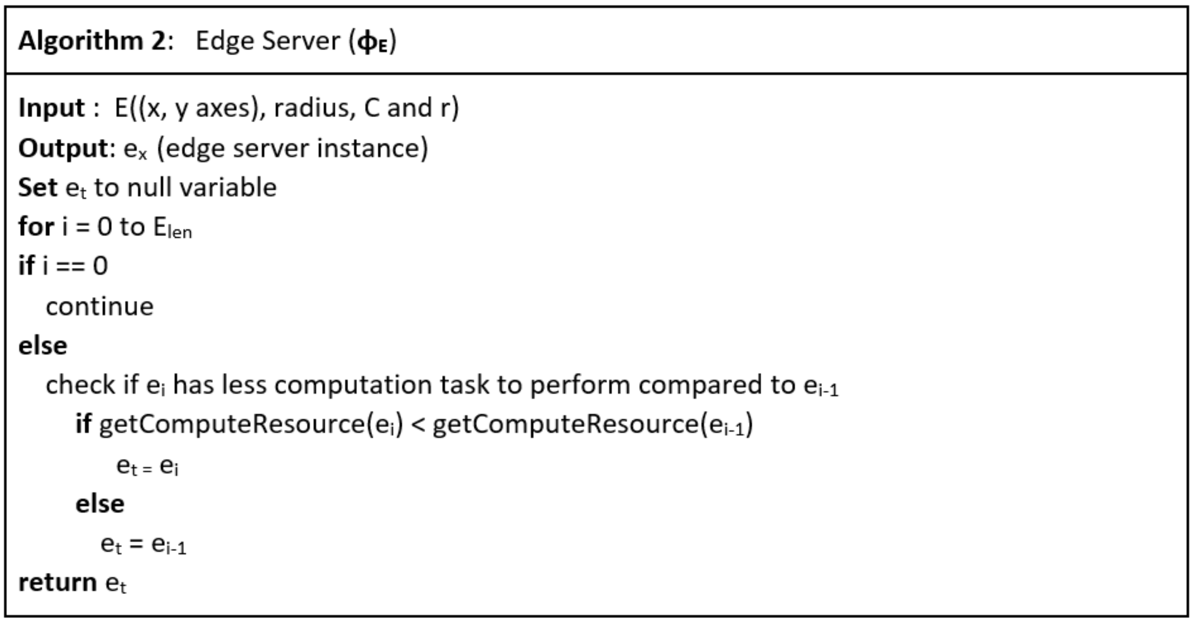
S - stands for the service request for a set $\{S_1, S_2, \ldots S_n\}$ ***where each instance is deployed on one of the n edge servers.***



Algorithm 1: Temporal Logic based algorithm for Service Instance Selection in MEC Systems

```
Input: SISP (R, P, C, E, S)
Output: Optimal Solution $\Theta$
for i in $S_{len}$
    $s_i = f(\mathbf{S})$
    connect to the nearest BS if connection does not exist
    $c_a = f(C)$
attempt selecting edge server from the list of servers by temporal logic until operator
   $e_x = \neg \phi_E \; U \; \varphi_E$
attempt selecting service instance from a list of service instances logic until operator
          $r_y = \neg \phi_R \; U \; \varphi_R$
assign request $s_i$ to service instance $r_y$ execute
        exe_t = $O \; \phi_{ry}$
return exe_t
```

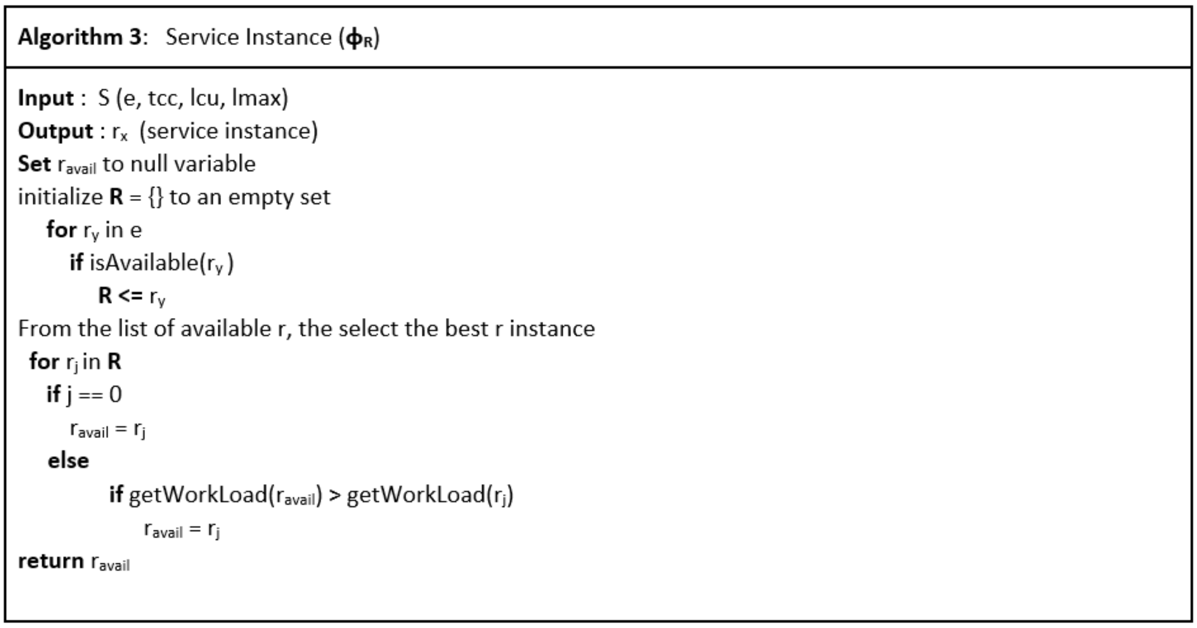
Figure 3: Table 3. Algorithm 1



```
Algorithm 2:   Edge Server ($\phi_E$)

Input :  E((x, y axes), radius, C and r)
Output: $e_x$ (edge server instance)
Set $e_t$ to null variable
for i = 0 to $E_{len}$
if i == 0
  continue
else
  check if $e_i$ has less computation task to perform compared to $e_{i-1}$
    if getComputeResource($e_i$) < getComputeResource($e_{i-1}$)
        $e_t = e_i$
    else
        $e_t = e_{i-1}$
return $e_t$
```

Figure 4: Table 4. Algorithm 2



```
Algorithm 3:   Service Instance ($\phi_R$)

Input :  S (e, tcc, lcu, lmax)
Output : $r_x$  (service instance)
Set $r_{avail}$ to null variable
initialize R = {} to an empty set
  for $r_y$ in e
    if isAvailable($r_y$)
      R <= $r_y$
From the list of available r, the select the best r instance
 for $r_j$ in R
  if j == 0
    $r_{avail} = r_j$
  else
      if getWorkLoad($r_{avail}$) > getWorkLoad($r_j$)
          $r_{avail} = r_j$
return $r_{avail}$
```

Figure 5: Table 5. Algorithm 3

4.3 Efficiency of Temporal Logic Algorithm

The efficiency of an algorithm is evaluated by taking inventory of the computational resources required execute the algorithm. These resources will usually include the memory resources (also known as the space complexity of the algorithm) and the time required to execute the algorithm (also known as the time complexity of the algorithm). The efficient of the is the measure of the largest possible usage of the computation resources while the algorithm is executed and this is also known as the worst case scenario expressed as the Big O Notation denoted as O(n), where

O - is the growth rate of the function used to measures how big the output grows

n - is the size of input data

To evaluate the time complexity of the Temporal Logic algorithm, we will take into consideration the 3 algorithms, each of the algorithm i.e., Algorithm 1, Algorithm 2 or Algorithm 3 has a for loop. A for loop has a time complexity of O(n) which is a linear time complexity. From Algorithm 1, Algorithm 2 and Algorithm 3 are nested in Algorithm 1, and thus Algorithm 1 has a Quadratic time complexity denote as $O(n^2)$*. The time complexity of the Temporal logic in the worst case scenario is* $O(n^2)$*.*

5 Implementation and Evaluation

5.1 Dataset and experiment setup

The experiments for this study are carried out on our HP workstation which runs Ubuntu 20.4 operating system, with intel core i7 8th gen @ 2.60 GHz CPU and 8GB of RAM. The implementation of the algorithm was written in Java version 1.8 while the graph was generated using python.

We conducted experiments on two datasets that are commonly used in edge computing to authenticate the efficacy and productivity of our algorithm. The datasets used in the experiment is the EUA dataset from Australia [34] *which contains data collected from real-world data sources. The datasets contain data from edge server and user locations which includes user addresses, location etcetera. We use Gaussian distribution to randomly generate and replicate different groups of users mustered together. In the experiment, we generated two datasets for the users where they are closely converged and another where they are sparsely dispersed.*

Each base station's coverage radius is randomly selected from a range of 150m to 200m. The edge users mobility path is created using the popularly recognised random waypoint mobility model (RWP) [21]*. The speed of mobile users is randomly selected within the range of 1m/s and 10 m/s while the transmission rate between two edge servers ranges between 1 to 10 MB/s.*

The rate at which data is transmitted between the edge servers and the cloud server is set at 1 MB/s. The closest base station to its centre point is the access point of each edge cell. A user's data transmission rate in an edge cell varies between 1 and 10 MB/s.

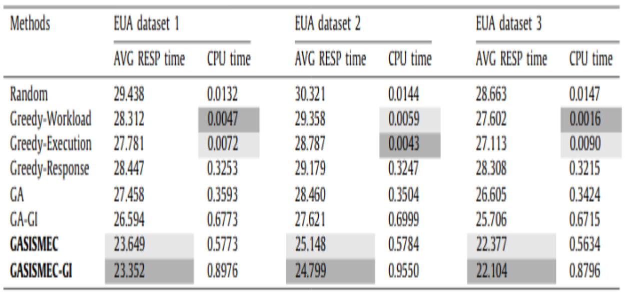
The user's request input data is randomly allocated a size between 5 and 20 MB. The time spent in executing a service request and the size of the user's request input data can be linearly associated. The minimum user's request execution time, relative to the size of the input data is set between 5s to 10s. The returned output data size of the user's request is equivalent to the input data size. 1 and 1.5 are the minimum and maximum values set for the computing capacity of the service instances respectively while the inherent workload ranges between 0 and 50 for the service instance. Given the constrained resources of the edge server and the ample resources of the cloud server, maximum workload range is set at 400, 500 for the edge servers and 5000 for the cloud.

6 Evaluation

6.1 Baseline

The model architecture we use for our baseline is GASISMEC-GI. The model adopts an implementation such that, the area covered is divided into nine edge cells. There are three users with their requests and three possible location points. The three users upload their requests from their respective location points. There are four edge servers and three service instances are deployed on them respectively.

An allocation strategy is generated randomly at the initial stage and then the average response time of each strategy is calculated with respect to data size, number of users, user's locations, workload, computing capacity, edge servers and service instances.



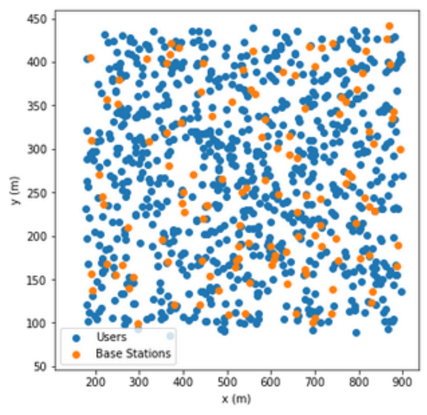
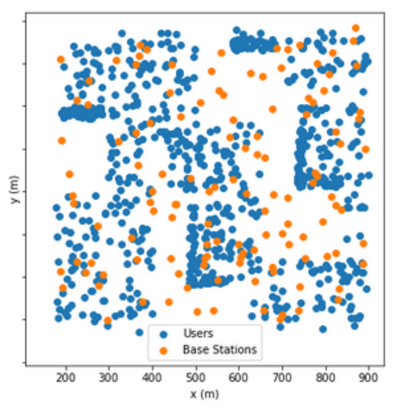
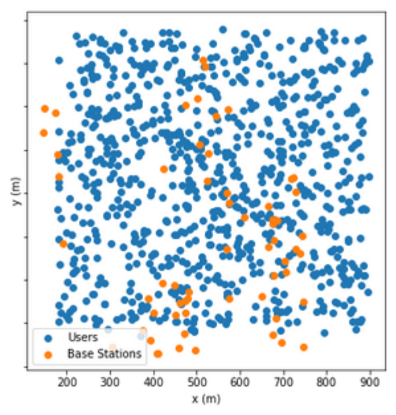
Methods	EUA dataset 1		EUA dataset 2		EUA dataset 3	
	AVG RESP time	CPU time	AVG RESP time	CPU time	AVG RESP time	CPU time
Random	29.438	0.0132	30.321	0.0144	28.663	0.0147
Greedy-Workload	28.312	0.0047	29.358	0.0059	27.602	0.0016
Greedy-Execution	27.781	0.0072	28.787	0.0043	27.113	0.0090
Greedy-Response	28.447	0.3253	29.179	0.3247	28.308	0.3215
GA	27.458	0.3593	28.460	0.3504	26.605	0.3424
GA-GI	26.594	0.6773	27.621	0.6999	25.706	0.6715
GASISMEC	23.649	0.5773	25.148	0.5784	22.377	0.5634
GASISMEC-GI	23.352	0.8976	24.799	0.9550	22.104	0.8796

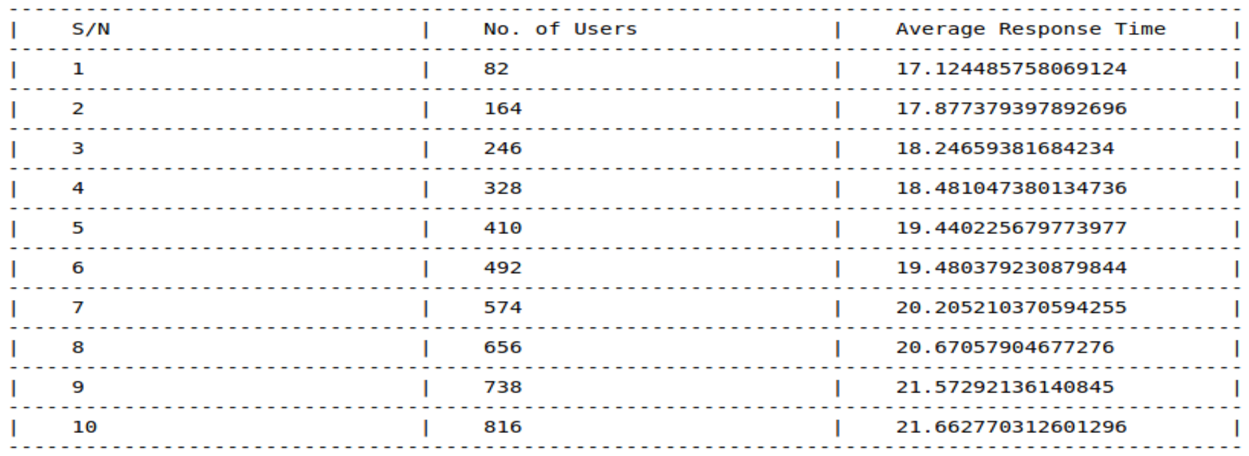
Figure 3. Result of experiment. Source adopted from [21]

6.2 Result

In the experiments, we set the linear temporal logic algorithm parameters to be uniform as described in [21]. The users are selected at random from the EUA dataset. We compare the results average response time between the baseline to our proposed algorithm which uses linear temporal logic with various distributions. The results obtained from the experiments are summarized below.

Figure 4. Different distributions of users and small base stations of EUA dataset.

		
EUA dataset 1	EUA dataset 2	(c) EUA dataset 3



S/N	No. of Users	Average Response Time
1	82	17.124485758069124
2	164	17.877379397892696
3	246	18.24659381684234
4	328	18.481047380134736
5	410	19.440225679773977
6	492	19.480379230879844
7	574	20.205210370594255
8	656	20.67057904677276
9	738	21.57292136140845
10	816	21.662770312601296

Figure 5. Experimental results on EUA datasets without linear temporal logic with varying number of users

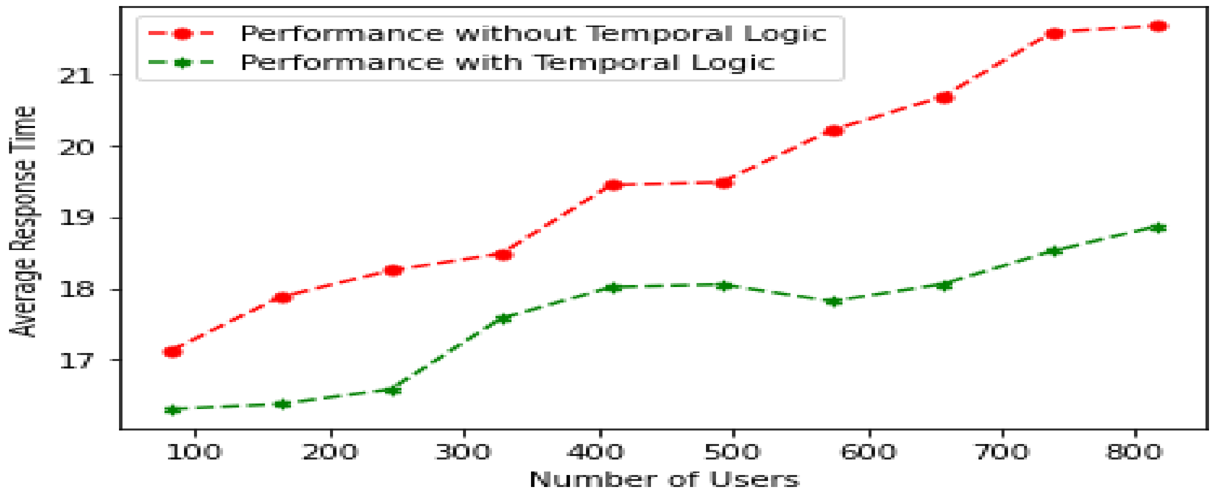


Figure 6. Experimental results on EUA dataset with linear temporal logic with varying number of users

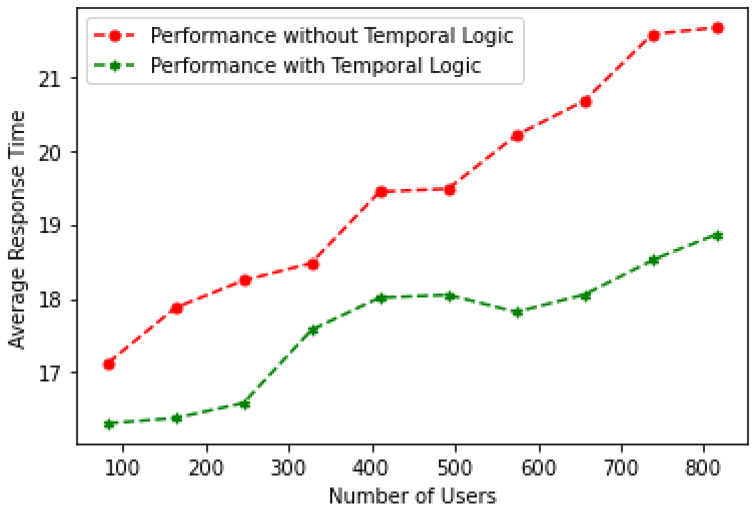


Figure 7. Performance comparisons of average response time on EUA dataset

The results illustrate that the average response time of a non-linear temporal logic is significantly higher than that obtained from the linear temporal logic approach as shown in figures 5 and 6 in all the iterations. When LTL algorithm was compared to GASISMEC [21] and non-linear temporal logic technique, significant improvement was shown. While [21] used 512 users, the average response time was about 23 seconds while LTL model fulfilled the same requests for 574 users in about 18 seconds. However, non-LTL model completed the same requests with 574 users with about 3 seconds less than GASISMEC. The lowest response time obtained by GASISMEC was 22 seconds with users n set to 512; meanwhile, with 816 users, the non-LTL model fulfilled the request with the average response time of almost 22 seconds and our model obtained an average response time of 19 seconds.

The outcome derived from the experiment clearly illustrates that our approach of using linear temporal logic surpasses that of the model proposed in Zou et al. (2021).

6.3 Limitations

In our experiment, we assumed that the edge user's mobility path is known by applying location prediction methods. In real world applications, location prediction techniques are error-prone and a user may suddenly alter their course direction from the predetermined mobility path. In addition, during high-speed mobility of mobile devices, uploading and downloading request data may be difficult to achieve. These limitations substantially increase the complexity of our system.

Another limitation to our approach stems from server connectivity. We assume there is interconnectivity between two randomly selected servers by one or more access points with different transmission rates. Fully interconnected edge servers are not completely feasible in real world usage and once the topology is set up by different telecom vendors; when applied to our system, the complexity becomes marginally higher.

7 Conclusion and Future Work

In this research we conducted, we studied service instance selection and the problems associated with it in mobile edge computing (MEC) environment. We factored in our study, the workload, the users' mobility path, user distribution and service instances. We focused primarily on the optimizing the average response time of service instance selection. In order to reduce the average response time, we proposed a model which capitalizes on linear temporal logic algorithm to optimally select the best possible service instance to users' requests. The experiment carried out on the EUA dataset proves how effective and efficient our proposed model when juxtaposed with the baseline techniques.

While significant research has been done on SISP in MEC environment in our study and others, drawbacks such as lack of real-world application has led to insufficient experimental outcomes. In further research, our aim will be to select edge servers with respect to both time and energy consumption. We will also explore more effective and efficient algorithms to determine which edge server is optimal for mobile users.

References

1 *Ahmed, E. and Rehmani, M. H. (2017). Mobile edge computing: opportunities, solutions, and challenges.*

Aral, A., Brandic, I., Uriarte, R. B., De Nicola, R. and Scoca, V. (2019). Addressing application latency requirements through edge scheduling, Journal of Grid Computing 17*(4): 677–698.*

Balasubramania, T. (2015). Mobile computing–an introduction with issues in mobile security, International Journal of Review and Research in Applied Sciences and Engineering 7*(1): 15–19.*

Gonçalves, D., Puliafito, C., Mingozzi, E., Rana, O., Bittencourt, L. and Madeira, E. (2020). Dynamic network slicing in fog computing for mobile users in mobfogsim, 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, IEEE, pp. 237–246.*

Hu, X. (2020). Mobile edge computing in wireless communication networks: design and optimization*, PhD thesis, UCL (University College London).*

Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I. and Ahmed, A. (2019). Edge computing: A survey, Future Generation Computer Systems 97*: 219–235.*

Lee, J., Kim, D. and Lee, J. (2019). Zone-based multi-access edge computing scheme for user device mobility management, Applied Sciences 9*(11): 2308.*

Lee, J., Kim, J.-W. and Lee, J. (2020). Mobile personal multi-access edge computing architecture composed of individual user devices, Applied Sciences 10*(13): 4643.*

Mehrabi, M., Salah, H. and Fitzek, F. H. (2019). A survey on mobility management for mec-enabled systems, 2019 IEEE 2nd 5G World Forum (5GWF)*, IEEE, pp. 259–263.*

MobFogSim. Simulation of mobility and Migration for fog computing MobFogSim. Simulation of mobility and Migration for fog computing *(n.d.). `https:////github.com/diogomg/MobFogSim`. Accessed: 2021-09-30.*

mobile *(n.d.). `https://wiki.openstack.org/wiki/Edge_Computing_Group/Use_Cases`. Accessed: 2021-09-30.*

Peng, K., Leung, V., Xu, X., Zheng, L., Wang, J. and Huang, Q. (2018). A survey on mobile edge computing: Focusing on service adoption and provision, Wireless Communications and Mobile Computing 2018*.*

Pnueli, A. (1977). The temporal logic of programs, 18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, IEEE, pp. 46–57.*

Prasad, M. R., Naik, R. L. and Bapuji, V. (2013). Cloud computing: Research issues and implications, International Journal of Cloud Computing and Services Science 2*(2): 134.*

Puliafito, C., Gonçalves, D. M., Lopes, M. M., Martins, L. L., Madeira, E., Mingozzi, E., Rana, O. and Bittencourt, L. F. (2020). Mobfogsim: Simulation of mobility and migration for fog computing, Simulation Modelling Practice and Theory 101*: 102062.*

Qi, H. and Gani, A. (2012). Research on mobile cloud computing: Review, trend and perspectives, 2012 second international conference on digital information and communication technology and it's applications (DICTAP)*, ieee, pp. 195–202.*

Roy, P., Sarker, S., Razzaque, M. A., Hassan, M. M., AlQahtani, S. A., Aloi, G. and Fortino, G. (2020). Ai-enabled mobile multimedia service instance placement scheme in mobile edge computing, Computer Networks 182*: 107573.*

Sajnani, D. K., Mahesar, A. R., Lakhan, A., Jamali, I. A. et al. (2018). Latency aware and service delay with task scheduling in mobile edge computing, Communications and Network 10*(04): 127.*

Saranya, N., Geetha, K., Sarumathy, C. and Rajan, C. (2019). Literature survey of data latency reduction techniques in mobile edge computing-iot, Journal of Critical Reviews 7*(6): 2020.*

Satyanarayanan, M. (1996). Fundamental challenges in mobile computing, Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, pp. 1–7.*

Sri^1, V. D. S. and Vemuru, S. (2019). Survey on data security issues related to multi-user environment in cloud computing, Journal of Critical Reviews 7*(4): 2020.*

Ugwuanyi, E. E., Ghosh, S., Iqbal, M., Dagiuklas, T., Mumtaz, S. and Al-Dulaimi, A. (2019). Co-operative and hybrid replacement caching for multi-access mobile edge computing, 2019 European Conference on Networks and Communications (EuCNC)*, IEEE, pp. 394–399.*

Wang, Z., Zhao, Z., Min, G., Huang, X., Ni, Q. and Wang, R. (2018). User mobility aware task assignment for mobile edge computing, Future Generation Computer Systems 85*: 1–8.*

Wei, H., Luo, H. and Sun, Y. (2020). Mobility-aware service caching in mobile edge computing for internet of things, Sensors 20*(3): 610.*

Xia, J., Li, C., Lai, X., Lai, S., Zhu, F., Deng, D. and Fan, L. (2020). Cache-aided mobile edge computing for b5g wireless communication networks, EURASIP Journal on Wireless Communications and Networking 2020*(1): 1–10.*

Zhang, K., Leng, S., He, Y., Maharjan, S. and Zhang, Y. (2018). Mobile edge computing and networking for green and low-latency internet of things, IEEE Communications Magazine 56*(5): 39–45.*

Zou, G., Qin, Z., Deng, S., Li, K.-C., Gan, Y. and Zhang, B. (2021). Towards the optimality of service instance selection in mobile edge computing, Knowledge-Based Systems 217*: 106831.*