School of Computing
FACULTY OF ENGINEERING AND PHYSICAL SCIENCE

UNIVERSITY OF LEEDS

Final Report

Edge Architecture Simulation: An Empirical Investigation

Yuvraj Mahida

Submitted in accordance with the requirements for the degree of
BSc Computer Science with Artificial Intelligence

2020/2021
40 credits

The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s)**
Deliverables 1	*Final Report*	*Supervisor, Assessor, SCSSO*

Type of Project: Empirical Investigation

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) YM

Summary

Increasingly, IoT devices use Artificial Intelligence and Machine Learning to provide intelligence and autonomy to systems. The unprecedented scale and complexity of data generated by IoT devices uncover the limitations of cloud computing. As a result, edge computing, a distributed computing paradigm that brings data storage and computation resources to the network edge, is proposed to fulfil IoT requirements such as latency, location awareness, and mobility support.

However, edge computing involves more complex architecture design; hence both computational and networking resources are analysed. In this investigation, EdgeCloudSim is utilised to investigate the effect of scalability parameters in an edge environment. A technical evaluation for each significant parameter identifies if the architecture remains effective after increasing the number of users (load condition) and resources.

Crucially, this project investigates the scalability of three edge architectures: single-tier, two-tier, and two-tier with Edge Orchestrator (EO) for a face recognition / augmented reality application that demands low latency and percentage of failed tasks.

The main accomplishments include successfully designing experiments using a sound methodology, generating valid simulation data using EdgeCloudSim, and profoundly evaluating results to provide valuable recommendations for future IoT applications.

Acknowledgements

I express appreciation to my project supervisor Dr Karim Djemame who has provided support and expert knowledge. Their guidance has been why this project has been completed successfully, especially during the most challenging year I have faced. Additionally, during progress meetings, my supervisor's true passion for the subject area has motivated me throughout the year.

Table of Contents

Chapter 1
Introduction

1.1 Introduction and Project Background

The Internet of Things (IoT) aims to bring connected devices such as wearables devices, smartphones, environmental sensors, and vehicles online [1]. Increasingly, IoT devices use Artificial Intelligence and Machine Learning to provide intelligence and autonomy to systems and processes. [2]. The vast growth of IoT devices has resulted in unprecedented volumes of data. Studies estimate that by 2025, 38.6 billion connected IoT devices will generate over 90 zettabytes of data [3]. Furthermore, volumes of data will continue to grow with the introduction of 5G networks, increasing the number of connected devices, notably mobile devices. However, IoT devices have limited computational and energy resources; therefore, the processing and storing of IoT data in these devices is usually inefficient [4].

The promise of cloud computing with characteristics such as on-demand services, rapid elastics, and scalability has previously been an effective way for computing, storage, and network management functions to work in a centralised manner [5]. However, with the vast expansion of smart sensors, smart actuators, networking technology, and low-power consuming chips, an immense surge of data transmission over the internetwork back-haul exists [6]. Moreover, the emerging IoT has created challenges in existing centralised cloud computing architecture, such as latency to serve users' requests, heavy load on the internetwork backbone, capacity constraints, uninterrupted services with intermittent connectivity, and security [6] [7].

An advanced cloud computing paradigm that can overcome the centralised architecture and address current latency and capacity constraints is needed [7]. As a result, edge computing envisions positioning network end-devices to the close proximity of user/application, allowing computation and storage at the network edge, minimising network latency and enhancing response times. Additionally, edge computing enables devices to request services from the cloud and handle computation tasks such as processing, caching, storing, and load balancing of data [9]. Edge architecture has many use cases involving real-time and data-intensive applications such as Driverless Vehicles, Manufacturing/Industrial IoT, Oil and Gas Remote Monitoring, and Edge Video Orchestration. In these cases, sending all data to the cloud requires prohibitively high network bandwidth. An efficient scheduling and adaptive

offloading scheme can minimise computation complexity and demand on the centralised data centre. Such capabilities of edge computing can lead to the edge-IoT ecosystem outperforming other existing architectures and paradigms. Hence, it is clear-cut that edge architecture is predominant for future IoT infrastructure [10]

1.2 Problem Formulation

Edge Computing is a distributed computing framework that moves computing capabilities and resources closer to the edge of the network meaning data can be analysed and processed closer to data sources such as IoT devices or local edge servers [4]. As a result, edge Computing can fulfil many IoT requirements such as low latency, location awareness, and mobility support by deploying cloud-computing-like capabilities at the network's edge [4]. However, the scalability of edge computing applications is affected by computational and network system parameters. Therefore, this empirical investigation aims to investigate the effect of these parameters in an edge environment using EdgeCloudSim [11], facilitating a technical evaluation on the scalability of three different edge architectures: single-tier, two-tier, and two-tier with edge orchestrator (EO).

1.3 Possible Solution

The possible solution entails evaluating the performance of different edge computing architectures by considering both network and computational resources [11]. EdgeCloudSim provides a modular architecture to support various crucial functionality, including WLAN and WAN network modelling, edge server modelling, device mobility, and load generation [11]. The simulator provisions the design of controlled experiments where parameter values can be changed. Furthermore, the simulator allows monitoring of simulation parameters by modifying scripts and key values – and testing values to determine the most important parameters for the investigation. An initial experiment will help develop hypotheses based on the most important parameters in relation to scalability. Following the collection, analysis, and interpretation of results, the design and implementation of follow up experiments will enable comparison of results for different edge architectures and determine reliable and conclusive findings. The possible solution delivers a collection of graphical representations of results for each experiment. The goal is to use these results and findings to propose further recommendations for edge computing applications in terms of performance, scalability, and future design.

1.4 Demonstrating the Quality of the Solution

The overall quality of the solution is demonstrated by setting clear objectives and determining a clear rationale for why the investigation is essential. There must be fundamental reasoning for undertaking the project and some use or benefit for this research. The project investigates the effect of computational and network system parameters on the scalability of three different edge architectures by simulating an environment that represents the use of an augmented reality application. The investigation is essential because edge architecture's capabilities and performance, especially with the future deployment of 5G networks, requires comprehensive investigation to provide the most suitable and beneficial system design. Next, a clear and logical methodology supports demonstrating the quality of the solution. Thus, following a methodical process to conduct the investigation will display the thought process behind complex decisions. By closely following an agile process, iterations and improvements are encouraged at any stage if required, ensuring a complete solution to the problem statement.

The quality of the solution is also demonstrated by ensuring the simulation data and graphically presented results are accurate and valid. Conducting an initial experiment and further trial experiments to explore the simulator will ensure that the application and scenario are represented as expected and that the simulation shows the effect of the correct key parameters. Furthermore, a comparison of experimental results derived from this investigation with existing academic studies allows for convincing support of the findings. Finally, broadening the project's scope will determine the quality of the solution, therefore designing and implementing simulation experiments for hypotheses with key parameters where no study currently exists is beneficial.

1.5 Aim

Edge Computing involves a distributed architecture that aims to process client data at the network's periphery, close to the originating source. It is an exceptionally fast-growing field enabling breakthroughs in technologies such as Cloudlets, Multi-Access Edge Computing (MEC), Fog Computing, and Heterogeneous Cloud Architectures. The project aims to simulate three types of edge architectures, particularly single-tier, two-tier, and two-tier with Edge Orchestrator (EO) deployed in an Augmented Reality application use case. A primary purpose of the empirical investigation is to determine performance results in relation to scalability.

1.6 Objectives

The following project objectives are specific, measurable, achievable, relevant, and time-bound:

1. Identify the critical issues about edge architecture simulation and edge computing performance.
2. Experiment design and implementation by configuring key parameters to investigate performance-based on single-tier, two-tier, and two-tier with EO architectures.
3. Interpret experiment results based on performance and scalability by conducting an analysis of results/data from the produced graphs.
4. Based on a thorough technical evaluation of results, present findings and justifications using existing literature. Verify hypothesis using strong evidence.
5. Provide recommendations based on the architecture of edge computing applications in relation to scalability.

1.7 Deliverables

The project deliverables submitted via report submission: The report includes:

1. **Background Research** – Research carried out on relevant theory to provide an in-depth knowledge base required for completing the project.
2. **Literature review** – Critical analysis of studies and reports addressing edge architecture presented in an organised manner. Appraisal of information to identify strengths and limitations of existing experiments to formulate ideas for new investigations.
3. **Simulation Experiment Design** – Construct a clear plan consisting of experiment designs and the initial experiment.
4. **Experiment Implementation** – Generate experiment results through running simulations using EdgeCloudSim, determine hypothesis for investigation and output accurate and valid results to evaluate findings.
6. **Technical Evaluation** – Critical analysis and interpretation of simulation results to validate results and verify hypotheses.
7. **Project Evaluation** – Discuss the success of the project in terms of meeting aims and objectives. Conduct a personal reflection and summarise the empirical investigation with mention of potential future work.

Chapter 2
Project Schedule and Project Management

2.1 Methodology

Before designing the experiments, a clear and transparent methodology supports achieving the objectives and aims of the empirical investigation. An agile approach is employed, allowing several iterations of each stage of the experimental process [6]. The iterations of each step in the process provide an opportunity to redesign and refocus experiments through regular visualisation and interpretation of results to guarantee the experiments fulfil the purpose of the investigation. In addition, this process facilitates easy identification of errors arising from experiments and inconsistencies in results through comparison with existing literature and expert findings. Simulation is recognised as the method to investigate edge architecture performance and capabilities. The simulator allows the representation of specific applications and generates results for scenarios through repeated and controlled simulation by changing a single fundamental parameter to show the effect clearly. As a result, a key strength of simulation is the ability to further improve experiments in the context of edge computing as parameters of interest become known from generating the data [8]. The project methodology encompasses simulation through five predefined phases:

1. **Design Experiments**

Design of experiments considering key parameters in relation to scalability and performance. Design and implement an initial experiment. Analyse and interpret data to identify the key parameters for the specific application. Then, decide on performance metrics for evaluation.

2. **Implementation**

Implement the designed experiments using EdgeCloudSim. Represent the specific application accurately utilising the simulator through modification of scripts and changing of parameters. Iterate the implementation stage as necessary until the simulator accurately represents all experiments.

3. **Collect the results**

Execute the experiments to collect the results. Obtain the required data output and produce graphical representation to visualise the data of each specific experiment for each edge architecture. Iterate collection of the results stage if anomalous data exists, ensuring accuracy and integrity of results. Adjust parameter values or refocus experiment designs if necessary.

4. Analysis

Analysis of data collected and create a set of graphical representations. Identify correlations in the data and produce statements showing trends of each graph. Next, test the quality of results by analytically comparing to existing findings from the literature review.

5. Interpretation

Interpretation of data to apply unambiguous meaning in the specific context of the empirical investigation. Determine and validate clear findings and evaluate the significance of results whilst providing recommendations for future applications and further study.

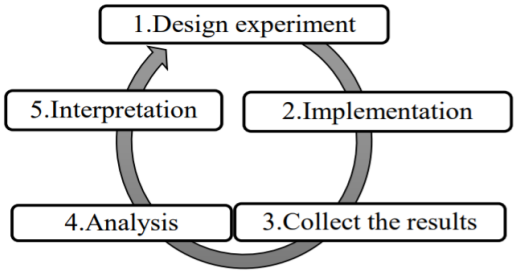


***Figure 01**. The agile process for the project [6].*

2.2 Tasks, Milestones and Timeline

The project Gantt chart is designed to track progress and task completion time frames. A complete list of project milestones is outlined in Appendix B.

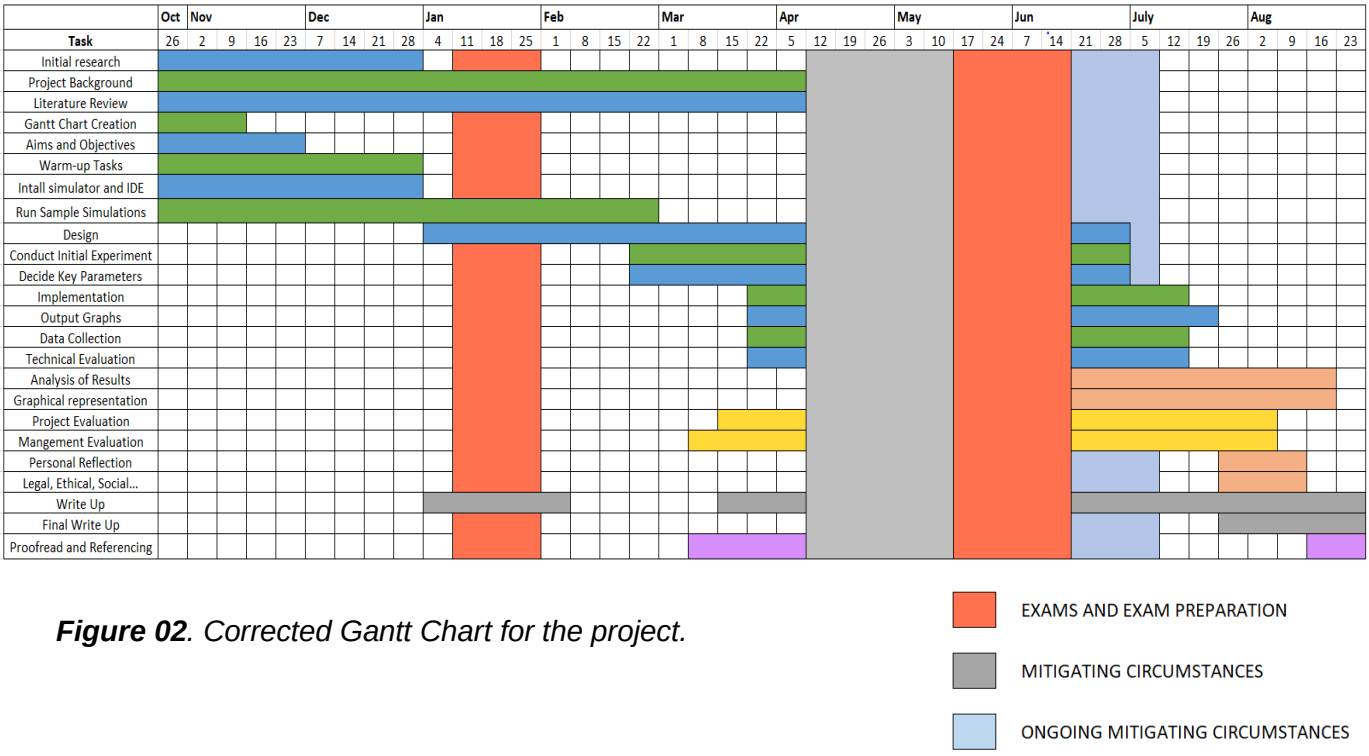


***Figure 02**. Corrected Gantt Chart for the project.*

Chapter 3
Background Research

3.1 Cloud Services

Cloud Computing enables pervasive, accessible, and on-demand network access to a shared group of configurable computing resources, including networks, servers, storage, services, and applications [12]. Shifting computing tasks to the cloud is an efficient method to process data due to increased computational power and resources in the cloud than in edge devices. Although data-processing speeds have improved, the bandwidth of networks that send data to and from the cloud is limited. Thus, with a significant increase in end device-generated data, the network is cloud computing's bottleneck [9]. Applications such as IoT devices utilise cloud computing capabilities. However, these applications necessitate low latency, mobility support, and location awareness to execute their functions with suitable performance [13]. Computing paradigms such as Edge Computing, Fog Computing, Mobile Edge Computing (MEC), and Cloudlet are viable solutions to support Cloud Computing to deliver the necessities of edge applications [13].

Mobile cloud computing integrates cloud computing into the mobile environment. The cloud server provides superior computational power and storage than mobile devices, enabling computation offloading. Offloading tasks improves application performance and energy consumption of end devices, suitably addressing the processing, storage, and battery life constraints of end devices [14].

However, for resource-intensive and latency-sensitive applications, like augmented reality with real-time constraints, offloading to the remote cloud is inadequate due to the high latency of Wide Area Networks (WAN) [15]. Essentially, applications such as driverless cars must process data in real-time to determine critical driving decisions such as lane changing or braking. Sending large amounts of local data to the cloud for processing would yield an inadequate response time, and a large number of autonomous vehicles in one location would further strain network bandwidth and availability. Processing data at the network edge yields shorter response times and efficient processing. Undeniably, micro-datacentres and cloudlets provide powerful computing capabilities. Providing a resource-rich server within the vicinity of the mobile users achieves a real-time interactive response via one-hop, high bandwidth access to the cloudlet. Furthermore, based on mobile users' requests, more

custom virtual machines can be instantiated immediately for remote execution of applications in a thin client manner.

3.2 Internet of Things

3.2.1 IoT

The Internet of Things (IoT) is a system of interrelated, internet-connected objects such as devices and sensors containing embedded technology, which constantly produce data and exchange messages via modern network infrastructures to provide intelligent insights. Intriguingly, IoT devices use AI and machine learning to build intelligence to processes in smart manufacturing, medical advancements, autonomous vehicles, and home automation.

3.2.2 Why does IoT Require New Architecture?

Traditionally, shifting computing, storage, control, and network management functions to a centralised data centre has previously been an effective solution. But, unquestionably, the rising number of powerful end-user, network edge, and access devices: smartphones, smart home appliances, edge routers, industrial control systems, and small cellular base stations, provides new challenges the cloud and host computation model cannot solve alone [16]:

Strict latency requirements – IoT applications such as autonomous vehicle communications, virtual Reality, gaming, and real-time financial trading applications require strict latency and near to real-time response, which cannot be attained by centralised cloud computing alone [16].

Network bandwidth constraints – The exponential rate of data creation means relaying all data to the cloud requires excessively high bandwidth, limiting infrastructure and network performance.

Resource-constrained devices – Most devices constituting IoT are constrained devices such as sensors, actuators, and controllers with limited CPU processing, memory, and power resources. Consequently, the growing computation consumption and memory demand introduces challenges for resource constraint devices [17]. As a result, it is impractical for these devices to perform resource-intensive tasks centred on cloud interaction.

Cyber-physical systems – Integrating smart cities and industrial control systems means interruptions and downtime must be avoided due to the primacy of continuous safe operation. Therefore, due to bandwidth and delay limitations, time and mission-critical applications that require updating cannot use the cloud for processing.

Uninterrupted s**ervices with intermittent connectivity to the Cloud** – Applications such as autonomous cars, drones, and oil rigs must continue to provide services during a connection interruption or loss of service to the internet or network connection with the cloud [2].

Emerging security issues – Preceding cloud architecture provisions perimeter-based protection against threats and processing resource-intensive security tasks in the cloud. The limitations of the cloud paradigm are evident due to the impracticality of protecting a large number of resource-constrained devices and requiring devices to connect to the cloud for security updates. Further, a conventional cloud faces difficulty meeting the scalability requirements to determine a trustworthy appraisal of the distributed system's security [17].

3.2.3 IoT Related to Edge Computing

Motivated to solve the challenges of IoT, research into the Edge Computing paradigm has thrived. Edge computing attempts to execute application computation on downstream data for Cloud services and upstream data for IoT services at the network edge [13]. By deploying cloudlets and micro data centres near end devices, data processing can ensue at the edge, improving response times. Furthermore, data analysis in a cloudlet reduces bandwidth demands as only small metadata is sent to the cloud. The potential of Edge computing is evident due to the high suitability for integration with IoT to address edge application requirements.

3.2.4 Industrial IoT Related to Edge Computing

Industry 4.0 drives the optimisation of existing industrial applications such as manufacturing and energy management. Industrial IoT (IIoT) refers to interconnected devices where data collection into the cloud facilitates data analysis for operational efficiency [18]. The adoption of IoT in industrial applications generates massive amounts of data to be processed. Edge computing allows autonomous decisions at the edge using locally accessible data without intercession from the cloud, enabling real-time analytics for crucial decision making [19].

3.3 Edge Computing

Incontestably, personal devices include smartphones, are generating data at unprecedented volume, velocity, and variety. Further, the emergence of Virtual Reality, Augmented Reality, Facial Recognition and Object Detection applications have increased demand on network bandwidth requirements due to the increased number of different tasks executed on computing resources [20]. Current cloud architectures using cloud computing service models (SaaS, PaaS, and IaaS) can be adapted for deployment at the network edge to support computationally expensive applications.

Edge computing is a distributed computing paradigm that brings data storage and computation resources to the network edge, providing increased bandwidth, reduced latency, and improved quality of service [21]. Edge infrastructure consists of edge servers near end devices, resulting in faster access to resources via the LAN. In addition, it reduces the load on the core network by computational offloading and enabling end device data to be processed or filtered before it is sent to the cloud, which is vital for 5G [22]. However, envisioned Edge scenarios constitute numerous devices, the highly mobile nature of end-users, intermittent traffic, and heterogeneous applications, where scalability and efficient orchestration are issues that require investigation. Ultimately, Cloud computing resources are used in conjunction to assist in cases where edge resources are lacking.

3.3.1 Edge Computing Architecture

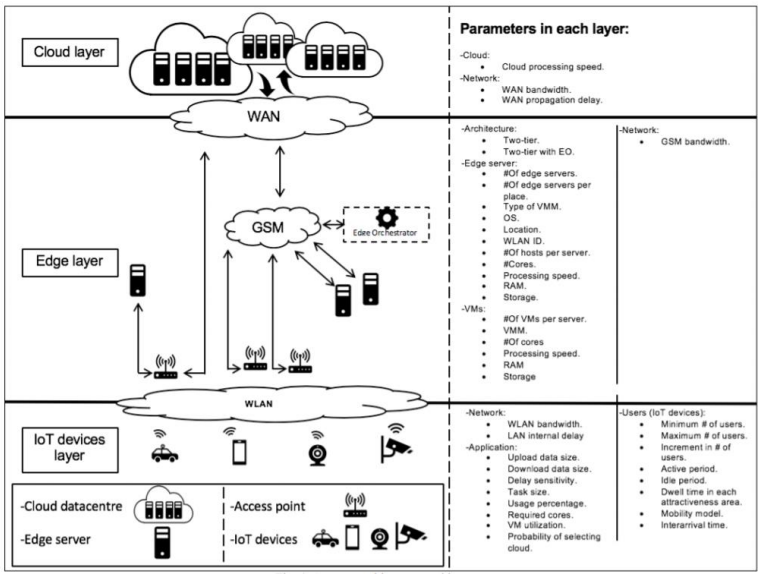


Figure 03. *System Architecture consisting of Cloud layer, Edge Layer, and IoT devices layer and Key Parameters [13]*

3.3.2 Key Characteristics of Edge Architecture

1. Energy-Efficient Edge

In Mobile Edge Computing (MEC), several offloading and resource allocation schemes provide improved energy efficiency. For instance, Mao [23] proposes an offloading scheme based on CPU cycle data and transmission power for optimal offloading decisions, whereas You et al. [24] harness the computing delay in MEC offloading to minimise weighted sum energy consumption.

However, in Vehicular Edge Computing (VEC), the dynamic vehicular network requires solving frequency handover due to the high mobility of vehicles. Hence, Zhou et al. [25] validate the relationship between energy consumption and key parameters such as transmission power utilising Queuing theory to derive traffic models. Remarkably, it was realised that edge allows opportunistic energy-saving for Internet of Vehicle (IoV) devices with limited battery life by offloading energy-intensive workloads to a VEC node [25] with higher computing capability and constant energy supply [26]. In addition, workload processing locally rapidly shortens battery endurance, negatively affecting miles per charge of vehicles. Thus, offloading saves energy expense of local processing but at the cost of increased consumption on data transmission, which is suitable for long-range electrically powered performance vehicles [27].

2. Low Latency and Delay

Edge services are deployed at a location closest to user devices, isolating network data movement from the core network, providing ultralow latency [28]. As a result, hierarchical distributed edge nodes can satisfy the demands of time and latency-sensitive applications. Furthermore, edge computing can formulate an ideal trade-off between computing latency and transmission latency via an offloading scheme to determine whether local processing, edge server offloading, or remote Cloud server offloading is optimum. For example, Tianze et al. [29] devised an offloading scheme where IoT devices utilise virtual machines to process tasks quickly. On the other hand, Taleb et al. [59] found that short migration latency does not always guarantee the quality of service –a trade-off between short migration latency at the cost of data loss is noted.

3. Reduced Bandwidth Demand

IoT data generated from sensors is vast; transmitting this data directly to Cloud Servers use immense bandwidth leading to transmission delay and network congestion, further increasing the chance of packet loss and failed tasks. Edge helps control traffic flow by

optimally migrating pre-processing and aggregation tasks to reduce bandwidth demand [30]. Sajjad et al. [31] formulated SpanEdge, which reduces latency incurred by WAN links by distributing stream processing applications across both edge and central data centres, lessening bandwidth consumption.

4. Effective Storage Balancing and Overhead Management

Edge Computing storage leverages load balancing and failure recovery to satisfy the quality of service and availability requirements. Load balancing allows offloading storage demands to edge nodes, reducing traffic at network connection links [30]. Due to IoT devices limited storage resources, data is transmitted from many devices to Cloud-based storage simultaneously, resulting in network obstruction. Edge storage enables data transmission to edge storage nodes, lessening long-distance communication. In addition, by offloading storage to the nearest Edge storage node, balancing improves storage times further using algorithmic weighting schemes or storage processing ratings to determine optimal offloading. Utilising edge servers allows pre-processing and aggregation of data packets minimising overhead.

5. Reliability and Context Awareness

Edge reduces service failure probability, offering enhanced reliability by executing processing near the source and prioritising traffic. Additionally, valuable information about the network condition is collected and used by network mechanisms to improve network operation. For instance, real-time data estimation of congestion and network bandwidth assists in future decision making for better reliability and quality of service [32].

6. Heterogeneity and Scalability

Edge handles interoperability concerns by integrating platforms, infrastructure, and communication technologies, supporting diverse technologies that influence performance. Edge computing scalability refers to the capacity to expand computing capabilities in response to changing conditions and demands - for instance, the capability of an Edge Server to execute a vast number of offloading requests simultaneously. In addition, an edge DS is scalable in size if it remains effective after increasing the number of users and resources. However, scalability involves overcoming overloading, back-haul links, resource misallocation, conflicting administrative policies, and performance degradation [33]. Furthermore, in an instance of cloud service unavailability, a scalable framework allows offloading of delay-sensitive tasks and the masking of failure/delay during high workload. Another aspect is maintaining the availability of resources through fault tolerance – which is

attributable to the multiple autonomous components and points of control/failure in a decentralised system [34]

7. **Mobility and Geographical Distribution**

Arising mobility of users and devices in the wireless network requires low-latency processing of tasks. Edge computing supports the dynamic migration of virtualised resources among multiple edges to maintain service performance [35]. For instance, Ouyang et al.[36] achieved a preferable balance between user-perceived latency and migration cost by optimising two mobility-aware service placement algorithms for MEC using Lyapunov optimisation. Geographic distribution of resources is a crucial factor in achieving low latency communication. Wang et al. [37] studied parameters relating to edge server placement and identified that strategic placement of servers via K-means balances workload among servers and minimises access delay between the user and edge server significantly.

8. **Security**

Distribution of storage and processing occurs across edge nodes and end-users, providing an assured level of security. Wireless data movement may be secured with encryption but affects performance. The edge ecosystem incorporates actors such as service providers where edge nodes must be authenticated. As a result, identification of nodes to cloud servers in a multi-management domain is challenging due to man in the middle attacks. In a less isolated environment, correlative attacks such as DOS on simultaneous nodes can compromise overall performance. Thus, virtualisation security must be enforced due to sharing of network instances. A single compromisation may affect entire virtualised infrastructure performance, particularly with the assimilation of APIs that are often less protected [38].

3.4 Edge Computing Simulator

Due to the inherent nature of the IoT environment promoting the integration of the physical and virtual world, research is a significant challenge. Furthermore, devising real-world deployment of edge architecture requires certifying proof of concept. So, given the enormous scale and heterogeneity of IoT systems, prototyping is not practical for the exploratory design of architecture as it requires explicit domain knowledge and vast operational costs [39]. As such, reliable and reproducible experiments via simulation enable theoretical analysis and testing of concepts. Undeniably, different simulators provide benefits in terms of capabilities and features for edge architecture simulation. Consideration of various simulators substantiates EdgeCloudSim as the simulator of choice:

CloudSim is an extensible simulation framework for modelling and experimentation with emerging cloud computing infrastructures and applications [40]. However, CloudSim lacks an edge-type load generator model and extensive mobility support in edge architecture representation. CloudSim also uses a basic network model and lacks support for running parallel experiments [41]. Although CloudSim allows the utilisation of cloudlet technology, it is not viable for IoT application scenarios in this project's scope [42].

XfogSim simulation framework supports latency-sensitive applications at the fog layer to evaluate cost, availability, and performance among the fog federation [43]. Interestingly, the distributed node management support enables nodes to acquire resources from near fog locations to provide efficient service. However, a simulator designed towards edge computing is more suited to this investigation.

iFogSim, based on CloudSim, simulates edge and fog domains using resource management and scheduling policies. Although it provides energy consumption and network congestion measurements for edge devices [44], some required network parameters are not supported. Further, the location of devices utilising the fog server is static because no mobility module is updated. Subsequently, iFogSim is limited to discrete event simulation (DES), meaning scalability is limited, which is unsuitable as scalability is a factor in this investigation [44]. Also, handling network interference of congested devices requires modifying latency and bandwidth attributes, adding further complexity when designing experiments.

FogNetSim++ provides simulation of fog networks with fog node scheduling algorithms and mobility modules. Thus, the ideal utilisation of heterogenous Fog nodes is represented [45]. However, capabilities such as VM migration, resource sharing between fog devices, and network properties such as congestion are not aptly supported [46].

IoTSim extends CloudSim and offers a suitable platform for investigating the performance of large IoT applications that produce big data, using the MapReduce model in a distributed computing environment [44]. It enables simulating large-scale multiple IoT applications in shared cloud datacentres but inherits DES and scalability limitations similar to iFogSim. Furthermore, metrics for network support such as response time, network delay, and energy consumption are not supported.

EdgeCloudSim: extends CloudSim and can be used to assess the computational and networking demands of edge computing. Conveniently, it integrates network and computation modelling to support mobility, orchestration, and offloading [47]. Due to its modular architecture, it can represent topologies such as WLAN and WAN. EdgeCloudSim is suitable for managing Edge server locations and the availability of edge VMs. Although EdgeCloudSim uses DES, it enables the evaluation of the edge orchestration of tasks, which is fundamental for this investigation. Therefore, this simulator provides a modular architecture comprising of five modules and is appropriate for evaluating the performance and behaviour of IoT and Edge applications:

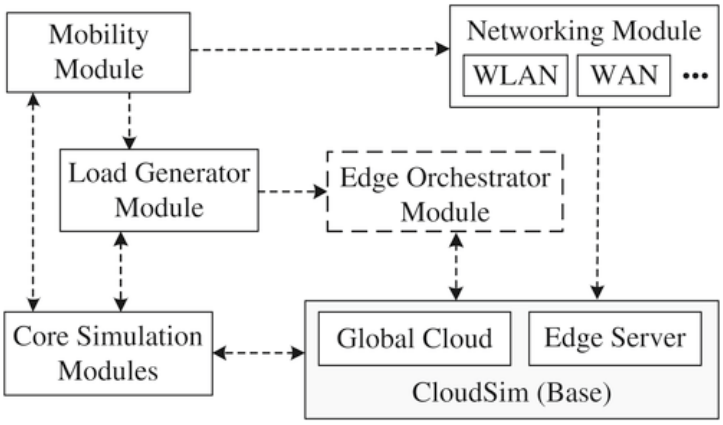


Figure 04: *EdgeCloudSim module relationship diagram [11].*

Core Simulation Module: Handles loading and execution of Edge scenarios using the configuration files and logging and saving simulation results in CSV format.
Edge Orchestrator Module: This module handles decision making when handling client requests. Extending the Edge Orchestrator class enables the utilisation of a realistic EO.
Mobility Module: The mobility module updates the location of end devices such as mobile devices. Each mobile device possesses coordinates updated using a dynamic hash table.
Load Generator Module: The load generator module interprets the configuration settings and generates tasks. Thus, acquiring results for average failed because of mobility, LAN delay, and the number of clients in a precise location.
Networking Module: This module allows the incorporation of network behaviour models and determines transmission delay in the WLAN and WAN concerning upload and download data.

Chapter 4
Literature Review

The following literature review consolidates understanding of related research and experimentation of edge architectures. Significant works can be classified clearly into three defining architectures.

4.1 Single-Tier Architecture

Single-tier architecture entails using the edge layer alone to execute the application's tasks [13]. Typically, small to medium-sized infrastructure adopt single-tier by allowing devices to request services from the edge server via Wireless Local Area Network (WLAN) [48]. This single-tier configuration enables edge nodes to be located at the network edge where short separation distances between nodes yields decreased link delay, favouring some delay-sensitive applications [49].

Sonmez et al. [11] introduced EdgeCloudSim for a performance evaluation of a face recognition application as part of an augmented reality application. Experiments representing single-tier architecture found that the service time and average failed task due to mobility increases when the number of connected mobile devices increases. This trend was attributed to single-tier architecture encountering increasing LAN delay due to network congestion. In addition, the study observed the average task size parameter and identified that for a small task size of 250 Million Instructions (MI), single-tier maintained service time due to no WAN delay as offloading tasks to the global cloud does not occur. However, it provides inferior performance for a large task size of 4000 MI as service time increases due to congestion.

A separate experiment by Sonmez et al. [48] analysed service time but for cloudlet technology, particularly processing and communication delay in single-tier architecture. The study revealed that increasing the number of end devices increases service time and failed tasks. Furthermore, as only WLAN delay is observed in the single-tier, processing delay dictates total service time due to the high computational nature of face recognition.

Suryavansh et al. [50] investigated single-tier architecture and found that when increasing the number of mobile devices, the service time for the exclusive Edge model increases from 2 to 4 seconds. Indeed, the increase in constrained devices resulted in the exhaustion of resources and heightened the probability of bottlenecks. As such, the edge server cannot

admit new requests when an excessive number of mobile users offload their applications [15]. The study also considers the impact of varying VM capacity on edge performance. A reduction in the capacity of the edge server increases service time – as high VM utilisation means tasks cannot be accepted [48]. Yet, VM capacity can diminish lower than mobile capacity, where service time performance of the edge server can be worse than mobile.

Jha et al. [51] support the aforesaid findings through a simulation of a self-driving car application that communicates with RSU (remote service unit) edge devices to attain run-time decisions. IoTSim-Edge experiments incorporate hard and soft handoff due to cars' high velocity and mobility. It found that increasing the number of cars with only a constant number of edge devices results in shared-time processing. Hence, queuing of tasks for processing occurred, increasing service time. Tan et al. [52] further support comparable claims by studying robust computation offloading of fog radio access networks. The findings for single-tier outlined that applications compete for all resources at the remote cloud rather than a subgroup of tasks. At the fog node, the limited computation resources resulted in longer minimum transmission latency compared to other architecture.

In terms of failed tasks, Suryavansh et al.'s [50] experiment results for exclusively edge or cloud architecture inferred that between 25% and 30% failed tasks occur for an augmented reality application. Application performance in failed tasks is significantly worse in single-tier due to mobile devices' insufficient computational capacity to handle simultaneous task requests. Incidentally, the availability of edge servers can vary since edge devices are less maintained and prone to failure. An experiment on the reduction of edge servers due to failure found increased overload, where an increase in failures generated a higher percentage of failed tasks when increasing the number of connected devices. Sonmez et al. [11] offered related findings, asserting that increasing mobile devices increases the average failed task due to VM capacity. However, boosting the WAN bandwidth from 4 Mbps (Megabits per second) to 40 Mbps did not affect average task failure in single-tier as tasks are not offloaded to the cloud in this architecture.

Liu et al. [15] investigated multi-resource allocation of cloudlet-based mobile cloud. The study found that in low traffic, more bandwidth resource is assigned to users, whereas in high traffic, bandwidth is reserved for future requests by assigning less bandwidth initially – due to high arrival probability. As a result, frequent and intensive mobile requests such as long processing can cause wireless bandwidth shortage, causing network congestion impacting mobile application performance, imputing unstable status and longer waiting times.

Gupta et al. [1] conducted a systematic investigation on network usage, latency, and energy use of a latency-sensitive online gaming application and intelligence surveillance over distributed camera networks. Experiments designed using iFogSim deduced that an increase in connected devices increases network load, resulting in network congestion. The growth of network usage hindered application performance. An analysis of simulation data showed average processing latency increases for cloud-only, whereas edge-ward placement maintained low latency by enabling data-intensive communication via low-latency links [1]. Pushing operators and IoT tasks to Fog devices saw a decrease in energy use in cloud data centres.

Premsankar et al. [35] studied mobile gaming performance and response delay by analysing various server deployments using the GamingAnywhere platform. Testbed results incorporating Wi-Fi and LTE access technologies emphasised the higher latency of public cloud deployment with cloud delay of 50ms at best, whereas the special-purpose cloud deployment obtained consistently low latency under 25ms. Although Wi-Fi is preferred for the shorter delay on average, LTE was deemed suitable for video streaming due to the lower variance of results and content, resulting in reduced jitter. Furthermore, in terms of HD resolution, mobile network edge configuration offered processing times of 70ms, which confirmed that increasing the availability of computational power in the cloud, instead of the edge does not counteract the increased latency. Thus, edge deployment meets the quality of service expectations for this use case.

4.2 Two-Tier Architecture

Two-tier architecture administers both edge and cloud layers to perform the application's tasks [13]. A probability-based mechanism can determine instances for selecting the cloud based on the availability of resources and application properties – for instance, if an edge node reaches the maximum capacity to manage and process incoming tasks. In addition, two-tier enables offloading of tasks to the cloud via Wide Area Network (WAN) whilst also provisioning cloud services such as Microsoft Azure.

Sonmez et al. [48] investigated the performance of two-tier architecture of a cloudlet assisted face recognition application in terms of service time, failed tasks, and VM utilisation. The research implies that two-tier architecture provides moderately better performance than single tier due to offloading tasks to the cloud. Indeed, in experiments with a low number of devices and limited congestion, two-tier architecture provides better LAN delay than single-tier. However, an increase in the number of devices also increases service time as the

architecture only provisions the use of cloudlets connected to the Wi-Fi access point. As such, bottlenecks are still probable at densely populated locations.

Furthermore, face recognition applications requires high CPU computation, small data download, and extensive data upload - simulated using a task transmission pattern. This computational demand means processing delay dominates service time. Djemame and Ajulayfi [13] realised a significant improvement in the average processing time for two-tier architecture when increasing VM processing speed and WLAN bandwidth in combination. In support of these claims, Sonmez et al. [11] analysed the impact of tasks size on service time. In the case of large average task size, two-tier returns similar performance to single-tier for all load conditions due to computational resource congestion on VMs; hence service time increases as the number of devices grows. Subsequently, in the case of small average task size, two-tier exhibits longer service time than single-tier due to the WAN delay properties and almost no task failure, respectively.

A paper by Jha et al. [51] investigated edge infrastructure for a driverless car application in which the computation process is distributed across edge, cloud, and the IoT device to satisfy the application’s requirements. The communication layer enables the driverless car to connect with an edge server. Once in range, it can connect with the next edge server while sending data to the initial edge server. The initial server transmits processed data to the next edge node, facilitating efficient handoff. Indeed, the edge device can still perform local processing but can send data to the cloud for complex analytics. However, cooperative edge communication between servers increases processing and execution time, thus increases service time. The study showed simulation time increases with an increase in IoT devices. Yet, two-tier architecture enables simulation of 50 IoT sensors to be completed in a satisfactory time of 4552.4 ms. Energy consumption data was equitable and did not increase drastically as IoT devices increased.

In terms of failed tasks, Sonmez et al. [11] inform that the percentage of failed tasks increases significantly in low WAN bandwidth at 4 Mbps. For two-tier architecture that requires offloading tasks to the cloud via WAN, task failure increases by 25% because of WAN congestion. Additionally, task failure due to mobility arises when a mobile device offloads a task and changes location, thus losing connection to the WLAN coverage area before receiving a response. Findings show that the average failed task due to mobility increases with the number of mobile devices – providing similar performance to single-tier architecture.

Djemame and Ajulayfi [13] investigated the performance of computational and network system parameters of object recognition in an augmented reality application for driverless cars. Two-tier architecture enables end devices (IoT and mobile users) to send processing requests to the edge servers or forward requests to the cloud. Simulation data reveals that an increase in the number of edge servers lowers the percentage of failed tasks as more requests are accepted. Hence, the workload is distributed across edge servers more efficiently, reducing the probability of request overload on any edge node. Expectedly, the percentage of failed tasks in the cloud lessens as adequate computation resources exist at the edge layer, reducing the need for offloading tasks to the cloud.

The same study [13] also investigated the effect of WLAN bandwidth. Increasing WLAN bandwidth gains no significant improvement on the percentage of failed tasks but masks and delays the presence of the bottleneck. Yet, it negatively affects VM utilisation as a faster arrival rate of tasks leads to inadequate VM resources, reflecting increased task failure due to VM capacity. Therefore, consideration for an optimal trade-off between WLAN bandwidth and desired task failure is a requisite.

Further experiment results reveal that increasing the VM processing speed reduces the percentage of failed tasks due to VM capacity in certain load conditions. However, an increase in VM processing speed also leads to WLAN congestion, increasing network delay. Sonmez et al. [48] validate that high VM utilisation occurs when tasks run on the VM, resulting in longer execution time or task requests not being accepted. It was also found that with increasing VM processing speed, failed tasks due to WLAN failure is not affected, insinuating maximum WLAN capacity. Accordingly, Djemame and Aljulayfi substantiate that scaling by increasing WLAN bandwidth and VM processing speed improves the overall performance of the architecture. Indeed, increasing both key parameters further improved processing and WLAN delay because of no WLAN congestion and no VM overload.

Gupta et al. [1] conducted a simulation experiment of an Intelligent Surveillance application to understand the average process latency of the sensing-actuation control loop. Latency was observed under varying workloads by altering configuration tuple size and transmit rate. Two-tier architecture maintained low latency due to positioning key modules at the network edge. Furthermore, data-intensive communication ensues via low latency links in two-tier architecture. As such, the edge-ward placement provides significantly lower delay at 8.47 milliseconds for all physical topology configurations than cloud-only.

Regarding scalability, data showed that increasing workload and size of the physical topology did not significantly increase RAM consumption. The architecture handled an increase in mobile devices and gateways with minimal memory overhead. Gupta et al. [1] propose that the execution time of simulation is linearly correlated to the size of the physical topology configuration. An increase in gateways enables an agreeable execution time of 25 seconds. Hence, increased availability of resources permits lower network delay since edge nodes are not overloaded.

Sonmez et al. [48] reinforced that end devices access cloudlets served in the connected WLAN in two-tier architecture. Furthermore, due to the computational intensity of face recognition, the processing delay expends most of the total service time.

4.3 Two Tier with Edge Orchestrator (EO) Architecture

Two-tier with Edge orchestrator (EO) architecture facilitates offloading an application's tasks to other edge servers or offloading tasks to the cloud. Orchestration manages the interconnections and interactions of distributed edge infrastructure workloads [21]. Shared infrastructure is leveraged fully when the edge facility mandates the same administrative organisation – the edge servers and edge orchestrator are connected to the same network with efficient coordination with the cloud datacentre [53].

Mobile client resources are often insufficient for task computation. The EO handles task requests through load balancing and redistribution among edge servers, allowing dynamic allocation [54] predicated on the current status of available resources [55]. Although minor transmission delay between the EO and edge nodes exists, orchestration helps eliminate errors introduced by scaling and diversity of resources. Interestingly, advanced orchestration to support big data and large-scale infrastructure is fulfilled via containers to cluster Cloud-to-Edge – examples include Docker container and Kubernetes architecture [55]. However, topology and orchestration schemes for edge computing still require extensive research.

Djemame and Aljulayfi [13] conclude that consuming other edge servers makes two-tier with EO architecture more scalable than other architecture as the load is balanced amongst edge servers. Evidently, for a sizable load condition, approximately 200 mobile devices or higher, increasing the VM processing speed improves the processing time than in single-tier and two-tier architecture – notably in the cases of 2000 and 3000 MIPS VM processing speed. Sonmez et al. [11] verify this claim and assert that for large task size of 4000 MI, two-tier with EO provides the best service time due to abundant computational resources. Whereas

in the case of small task size of 250 MI, single-tier offers the most favourable service time since two-tier and two-tier with EO incur WAN delay. Further, for increasing load condition, LAN delay increases faster in other architecture.

The study also evaluated VM utilisation. Two-tier with EO exhibit high utilisation due to consumption of other edge servers. Yet, this is counterbalanced as the edge orchestrator supports fewer failed tasks due to VM capacity than two-tier.

In terms of failed tasks, increasing VM processing speed slightly reduces the percentage of failed tasks [13]. Sonmez et al. [11] [48] deduce that as all tasks are distributed to all edge servers and executed quickly, average task failure due to mobility does not increase and is very low compared to other architectures. The reason is that other architectures face congestion in attractive locations, which increases service time and task failure due to mobility.

Suryavansh et al. [50] accentuate these findings through simulation of a mobile-edge hybrid model under constrained conditions. The model illustrates balancing the computational load between mobile devices and edge servers to improve service time and task failure. Evidently, as the number of mobile devices increase, two-tier with EO architecture holds a percentage of failed tasks near zero, contrasted to the high percentage of failures of single-tier. Moreover, tasks are executed on mobile devices until the computational limit; therefore, when computational resources of the mobile client are insufficient, tasks are orchestrated to edge servers – beneficial for resource-constrained mobiles already running demanding applications. Maximising edge capacity by balancing the load across edge servers before offloading to the cloud reduces the unsuitability of the edge of other architectures due to reduced overload and failure of edge servers. The lessening of requests sent to the remote datacentre eliminates the inconsistency of the cloud arising because of low WAN bandwidth and WAN delay, reducing service time for a number of mobile clients larger than 500. Lastly, Sonmez et al. [48] caution that if the probability to offload tasks to the distant cloud is high, the WAN delay caused by long-latency communication will detrimentally affect performance.

Zaitoun et al. [53] investigated the performance of two-tier with EO architecture using Poisson Inter-arrival time where any edge server can handle requests utilising data from other edge nodes or modules. The study found that two-tier with EO achieved the lowest average failed tasks due to load capacity utilising 87% load capacity for 10000 edge devices, compared to 92% and 95% for two-tier and single-tier after enhancement, respectively.

Furthermore, in terms of average network delay, single-tier provided the lowest delay for 1000 devices, but enhancement improved network delay of all architectures.

Xia et al. [56] and Premsanker et al. [35] agree that traffic flow control at the edge reduces service latency and supports the scalability of resources as the number of end devices increases. By managing high network utilisation, latency-sensitive flow relieves packet loss during network congestion for latency-critical applications such as mobile online gaming that require low response time. Further, the Edge orchestrator can utilise VMs or containers with more processing capabilities to meet the application's requirements. The mobility of end-user devices is handled via the live migration of edge resources such as VMs and containers. The architecture enables the edge node to orchestrate the federation dynamically, maximising executed tasks and scalability to achieve required application performance.

Finally, consumption of other edge servers reduces the percentage of tasks sent to the cloud, improving performance. Still, data sensitivity is an issue as other organisations may own certain edge servers, causing privacy policy breaches in applications such as healthcare [13]. As security is critical, Jalalpour et al. [57] propose dynamic security orchestration to protect edge servers such as application-layer threats and network layer flooding via rate-limiting.

4.4 EdgeCloudSim Configuration Files Summary

EdgeCloudSim [11] uses the following configuration files to produce controlled and repeatable edge computing scenarios:

1. **applications.xml**: declares the characteristics of applications used. The specification of each application is defined by modification of properties such as interarrival time of tasks, active and idle period, data upload and download, task length, required cores, VM utilisation percentages, and cloud selection probability.
2. **edge_devices.xml**: stores the characteristics of the edge datacentres, enabling declaration of server hosts. Notably, each edge datacentre serves one access point, meaning mobile devices' location is established using x and y coordinates. Additionally, the utilisation of virtual machines is represented by modifying variables such as the number of cores, MIPS, and storage.
3. **default_configuration.properties**: stores the key simulation parameters, including simulation time, file logging option, number of mobile devices, networking options, cloud datacentre characteristics, orchestration policies, and simulation scenarios.

Chapter 5
Design

5.1 Hypothesis

This empirical investigation aims to evaluate edge architecture performance in terms of scalability. An edge distributed system is scalable if it remains effective after increasing the load condition. The following hypotheses are devised to determine the effect of computational and networking system parameters on performance results.

Hypothesis 1 – An increase in the number of edge servers reduces the percentage of failed tasks, service time, and VM utilisation.
Hypothesis 2 – An increase in WLAN bandwidth reduces the network delay
Hypothesis 3 – An Increase in VM processing speed reduces the percentage of failed tasks
Hypothesis 4 – An increase in VM processing speed and WLAN bandwidth reduces the percentage of failed tasks, service time, and VM utilisation

5.2 Experimental Design

5.2.1 Experiment Conventions and Assumptions

A virtual mobile edge environment is designed using EdgeCloudSim, representing a face recognition application as part of an augmented reality application. Values for parameters have been specified utilising a literature review of comparable experiments to validate accurate application representation. Each location operates one edge DC and one access point. The assigned attractiveness levels insinuate the wait time of the mobile user in a particular location. It is assumed that due to the mobility of users, users are likely to spend longer time in more attractive locations enabling a realistic assessment of scalability due to increased load. To correctly emulate the use-case, computationally intensive task distribution enables users to offload heavy computational tasks in which edge servers return small-sized responses. Upload size is assumed to be very high, with minimal data download as expected for this application. Additionally, the simulation model assumes users are active for 60 minutes and idle for 15 minutes; users send tasks for some time and wait for request completion instead of sending a surge of requests. These assumptions assist in providing realistic scenarios.

5.2.2 Initial Experiment Design

An initial experiment is performed and refined in a controlled manner to authenticate accurate scenario representation. Fine-tuning of initial parameters using the literature review found suitable real-world values for a rational base for experimentation. Tables 1 and 2 show the values for each parameter used for the initial experiment.

Table 1. *simulation configurations.*

Parameter (Unit)	Value
Simulation time (mins)	25
Warm-up period (sec)	3
Upload data size (KB)	2000
Data download size (KB.)	50
Minimum number of mobile devices	25
Maximum number of mobile devices	350
Cloud processing speed (MIPS)	20000
Number of iterations	10
Usage percentage (%)	30
Poisson interarrival of task	3
Task length	2000
Probability of cloud selection (%)	10
WAN propagation delay (sec)	0.1
WLAN internal delay (sec)	0.005
Attractiveness type 1 wait (secs)	60
Attractiveness type 2 wait (secs)	30
Attractiveness type 3 wait (secs)	15
Hosts on cloud datacentre	1
Number of VM on the cloud host	4
Storage for cloud VM	1000000
Orchestrator Policy	NEXT_FIT

Table 2. *Initial experiment key parameter values.*

Parameter (Unit)	Value
Number of edge servers in location type 1	4
Number of edge servers in location type 2	4
Number of edge servers in location type 3	4
WAN Bandwidth (Mbps)	30
WLAN bandwidth (Mbps)	300
VM Processing Speed (MIPS)	1000
VM capacity (KB)	50000

The initial experiment has been designed utilising twelve edge servers distributed across three locations of attractiveness. Each server consists of two VMs capable of executing 1000 million instructions per second (MIPS). WLAN bandwidth is initially 300 Mbps, and WAN bandwidth is 30 Mbps. Further, a VM capacity of 50,000 KB is sufficient to address experiment requirements. Mobile devices use a related WLAN per location to send tasks to the edge, and if a task is required to be offloaded to the cloud, the WAN connection is used.

5.2.1 Full Experiments Design

Table 3. *Experiment design outline.*

Hypothesis	Stage	Key parameter	Value
Initial Experiment	*1*	*The initial experiment used to obtain important parameters*	*Load condition is 25 – 350 mobile devices*
Hypothesis 1	*2*	*Edger Servers per attractive location*	*2, 4, 6, 8*
Hypothesis 2	*3*	*WLAN bandwidth*	*300, 500, 700. 900*
Hypothesis 3	*4*	*VM Processing Speed*	*1000, 2000. 300, 400*
Hypothesis 4	*5*	*WLAN bandwidth and VM Processing Speed*	*3000-1000, 500-2000, 700-3000, 900-3000*

An evaluation of the initial experiment has found several parameters that require investigation. The following stages outline the experiment design:

1. Initial experiment: This is designed as an experiment base to identify parameters that require investigation across three edge architectures.
2. Effect of edge servers: This is designed to investigate the availability of edge servers by evaluating an increase in edge servers on the percentage of failed tasks.
3. Effect of WLAN bandwidth: This is designed to investigate the effect of increasing WLAN bandwidth on the network delay.
4. Effect of VM processing speed: This is designed to investigate increasing VM processing speed on the percentage of failed tasks.
5. Effect of WLAN bandwidth and VM processing speed: This is designed to investigate the effect of increasing both WLAN bandwidth and VM processing speed on performance in terms of scalability.

5.3 Performance Metrics

Extensive performance metrics are used to evaluate simulation data, allowing strong justification for trends based on the results. Additionally, the performance metrics test the hypothesis and justify the application's trends and behaviour, enabling a profound understanding of the data in this use case. For example, a significant performance metric is the percentage of failed tasks as it evaluates how the architecture sustains increased load conditions. The next significant performance metric is service time to evaluate if task requests are completed in a suitable time for time-constrained applications. Next, VM utilisation is observed to show the effectiveness of the VMs under high load conditions. Finally, another significant performance metric is the average network delay for WAN and WLAN to visualise the cause of network congestion.

Chapter 6
Implementation

6.1 Implementation Details

The simulator advocated is EdgeCloudSim to implement the designed experiments to achieve the aims of this empirical investigation. Editing configuration files and modifying configuration files and scripts enable the representation of specific applications to evaluate performance in terms of scalability. Modification of the runner.sh script allows for parallel execution of application scenarios with command-line arguments for the number of parallel processes/cores /iterations. Running scenarios produces a log file of simulation results and data, which is used to produce graphs.

Implementation is done using IntelliJ integrated development environment in Java. Experiments are performed on a Dell Latitude 5310 system with an Intel Core i5 processor, 16 Gigabyte RAM, and 256GB SSD.

6.2 Evaluation Metrics

A full selection of performance metrics, outlined in Chapter 5, is required to investigate the effect of network and computational parameters on the architectures' scalability and performance [13]. Formulating experiments with these evaluation metrics allows identifying key parameters of interest that provide value to the investigation. These evaluation metrics are the percentage of failed tasks, average service and processing time, VM utilisation, and average network delay.

6.3 Result Visualisation

Visualisation of results is required to view, analyse, and interpret trends in the data. Successful running of experiments in the EdgeCloudSim simulator generates simulation results using log files. Then, modifying MATLAB. functions permit outputting graphical representations from the ite10 log files. These graphs require further formatting, so graphs and log data are input into various graph manipulation tools to display the visualisation clearly. This was necessary to identify anomalous results, which allowed repeating experiments for the validity of results. It also assisted in reading exact data points to analyse trends in small increments for a deeper understanding of the edge architecture performance.

Chapter 7
Technical Evaluation

7.1 Initial Experiment

By representing the application using EdgeCloudSim [11], an initial experiment is modelled to acquire reasonable simulation configuration values and identify key parameters. The results display the average results over ten iterations. Hence, the interdependence of each iteration provides more probabilistically accurate results with minor variance. Graphical figures are presented in Appendix A by order of appearance in this technical evaluation.

The initial experiment provides a foundation for investigation. Primarily, Figure 1A shows that the percentage of failed tasks increases as the load condition increases. Single-tier architecture provides the worst performance at 40% failure for the maximum load condition due to the insufficient computational capacity of mobile devices to handle simultaneous task requests. Expectedly, two-tier architecture provides 32% task failure at the same load condition. Figure 1A shows a sharp increase in failed tasks at load conditions 150, 200, and 250 for single-tier architecture and 250 for two-tier architecture. The importance of consuming other edge servers can be accentuated as the two-tier with EO architecture provides the best performance by maintaining 5% failed tasks until 300 devices where it experiences severe load and rises quickly to 25%. Although it provides slightly better performance than other architectures, performance must be improved by adjusting key parameters.

Figures 1B and 1C illustrate that the growth in percentage for the edge servers is faster than the cloud as there are limited resources at the edge. Task failure at the edge significantly increases with a high number of connected mobile devices, which this investigation must address. Task failure in the cloud fluctuates below 1% for two-tier and two-tier with EO architecture, highlighting the cloud DC's vast processing and storage power. Analysing the reason for task failure provides direction for the investigation. Figure 1D shows the failed tasks due to mobility. For load conditions up to 200, two-tier with EO provides satisfactory performance compared to other architecture, but increasing the load beyond 200, results in increased task failure to 8%. In addition, task failure due to mobility data shows an increase in failure by 5% as load conditions increase in the worst case. This increase arises when mobile devices offload a task and move location, thus losing connection to the WLAN coverage area before receiving a response.

Figure 1E shows that increasing the number of devices causes the percentage of failed tasks due to VM capacity to increase considerably. For a small number of devices below 150, two-tier with EO provides the best performance due to efficient orchestration of tasks while consuming resources of any edge server. However, for a large number of devices, two-tier and two-tier with EO produces similar performance at 50% task failure. Since significant task failure is evident due to the VM capacity of the edge servers and users' mobility, increasing edge servers at each location is investigated to improve scalability.

The initial experiment is designed to test the full capabilities of the edge architecture and network infrastructure in terms of scalability. Hence, applying a load condition of up to 350 devices places great stress on the system and tests if the service remains acceptable for the augmented reality application's requirements. VM utilisation simulation data in Figure 2A shows that for load conditions up to 200, two-tier with EO architecture handles the tasks with the lowest utilisation with single-tier providing the worse performance. Yet, past load condition 200, single and two-tier architecture provide similar VM utilisation whereas two-tier with EO architecture exhibits over 90% VM utilisation. Therefore, increasing VM processing speed to reduce the number of failed tasks and improve VM utilisation is investigated.

Figure 3A represents the average network delay and shows that single-tier provides the lowest delay of 0.015 seconds due to not encountering WAN delay since offloading tasks to the global cloud does not occur. For low load conditions up to 175 devices, two-tier with EO architecture offers a higher delay of 0.065 than 0.057 for two-tier architecture. However, for high load conditions above 275 devices, the two-tier architecture offers a much higher delay of 0.08 than 0.067, respectively. The trend shows that orchestration and better availability of resources help manage many devices more effectively. Inspecting Figures 3B and 3C show that two-tier and two-tier with EO architectures face a high WAN delay of 0.035 seconds. Additionally, two-tier with EO faces a much higher WLAN delay of 0.026 seconds than 0.017 of the other architectures. Since network delay for WLAN is smaller than WAN due to edge servers located in the proximity of edge devices, increasing WLAN bandwidth is investigated to reduce the network delay.

Figure 4A shows that service time increases as the load condition increases. The lengthiest service time is provided by single-tier architecture, and the shortest is provided by two-tier with EO architecture. However, increasing to 225 mobile devices, two-tier EO provides the lengthiest service time of over 3 seconds. A similar trend is seen for processing time in Figure 4B, where two-tier with EO architecture faces steep increases in processing time for high load conditions.

7.2 Experiment Results and Discussion

7.2.1 Effect of Edge Servers

The experiment is designed to vary the number of edge servers to investigate the effect on scalability. The initial experiment consists of four edge servers per location, an increase to six and eight edge servers is simulated. Further, a reduction in edge servers to two per location is also explored.

Figure 5A shows that a reduction in edge servers negatively impacts the scalability of all architectures due to increased overload, causing an increase in the percentage of failed tasks. Furthermore, VM congestion is evident with limited server availability due to the edge's insufficient processing and storage capabilities. This claim is proven by Figure 10A, where two-tier with EO architecture experiences near 100% VM utilisation for load condition 175 and above. This high utilisation occurs due to insufficient resources when consuming other edge servers. Undoubtedly, numerous tasks currently running on the VM result in lengthy execution times and the inability to accept task requests.

A reduction in edge servers causes a considerable increase in the percentage of failed tasks for two-tier with EO architecture – acquiring over 50% failed tasks, similar to two-tier architecture. Figure 7A reveals a high percentage of failed tasks occurring due to VM capacity. A series of bottlenecks at 150 and 200 load conditions are visible due to the very high utilisation of the VMs. Inspection of the log file reveals that the anticipated VM capacity was exceeded due to near maximum VM utilisation, resulting in increased task failure. Figures 8A and 9A prove that average service time and processing time significantly increases. As a result of lengthier processing time and service time, Figure 6A shows that two-tier with EO encounters a significant increase in failed tasks due to mobility between 125 and 225 load conditions before balancing out to obtain similar performance to the other architectures. Seemingly, congestion occurs after 100% VM utilisation is reached, leading to mobile devices leaving the WLAN coverage area not served. Subsequently, the VM releases failed tasks requests due to mobility to continue running at very high utilisation. Indeed, scalability requirements were not satisfied in this case as edge resources could not service the high mobility of users in an acceptable time. By increasing the number of edge servers, Figures 7C and 7D show that the recorded bottlenecks no longer exists as close to 0% failed tasks due to VM capacity occur.

Regarding VM utilisation, two-tier with EO provides the lowest utilisation for a small load of mobile devices but the highest utilisation for a large load. Figure 10A identifies that once

near 100% VM utilisation is reached, two-tier with EO architecture sustains very high VM utilisation from the load condition of 175 devices. Since the VM's are executing many tasks to cope with the increased load, requested tasks cannot be accepted, leading to a high percentage of failed tasks. Figures 9A and 8A show that processing and service times become worse than the two-tier architecture for sizeable load conditions. Figure 10D confirms that increasing edge servers resolves the high VM utilisation of two-tier with EO architecture. By load balancing across multiple VMs, two-tier with EO architecture can achieve 25% VM utilisation, with two-tier and single-tier architecture achieving 30% and 35%, respectively.

Figure 5 demonstrates that an increase in edge servers significantly reduces the average percentage of failed tasks for all architectures. Undeniably, maximising edge capacity by balancing the load across several edge servers reduces VM overload and failure of edge servers. Furthermore, the data shows that an increase in server availability enables more tasks to be accepted. Since workload distribution occurs more efficiently across additional edge servers, the probability of request overload lessens. Expectedly, the percentage of failed tasks in the cloud lessens as adequate computation resources exist at the edge layer, reducing the need for offloading tasks to the cloud. Compared to the initial experiment, Figure 5D shows a reduction in failed tasks from 40% to 20% for single-tier and a reduction from 30% to 15% for two-tier architecture. Expectedly, two-tier with EO architecture provides less than 5% failed tasks in this case. Indeed, it is evident that the EO fulfils task requests through load balancing and redistribution among edge servers, facilitating dynamic allocation [54] predicated on the current status of available resources.

Lastly, an increase in edge servers reduces the service time and processing time. In the initial experiment, single-tier architecture encounters the highest service time of 3 seconds for low load conditions, but two-tier EO architecture encounters a lengthier service time of 3.5 seconds for high load conditions. As the processing delay expends most of the service time, it is understood that two-tier with EO encounters higher network delays, as evident in Figure11, due to server cooperation and communication between edge servers increasing service time. However, as shown in Figures 8D and 9D, an increase in servers demonstrates that two-tier with EO can achieve significantly lower service and processing times than other architectures – suitably servicing time-sensitive augmented reality applications.

7.2.2 Effect of WLAN bandwidth

The experiment is designed to increase the WLAN bandwidth to investigate the effect on scalability. The initial experiment is configured using 300 Mbps bandwidth. Therefore, further experiments are simulated using 500. 700, and 900 Mbps bandwidth to show the effect on performance parameters.

This experiment finds that increasing the WLAN bandwidth gains no significant improvement in the failed tasks compared to the initial experiment, as evident in Figure 12. However, for load conditions above 300 devices, an exceptionally minimal reduction in the percentage of failed tasks is realised in Figure 12D. This trend is negligible and is attributed to the interdependency of simulation experiments.

The average network delay is represented in Figure 17. Expectedly, single-tier architecture induces the lowest network delay of 0.015 seconds. Two-tier architecture retains lower network delay for a low number of mobile devices than two-tier with EO architecture; however, the opposite is valid for a high number of mobile devices. Compared to the initial experiment, there is a slight improvement in network delay as the WLAN bandwidth increases.

Regarding WLAN delay, increasing the WLAN bandwidth reduces the average WLAN delay. Two-tier with EO architecture encounters slightly higher WLAN delay due to communication between servers when orchestrating tasks. Nevertheless, the results show that the two-tier with EO architecture WLAN delay improved from 0.023 to 0.014 seconds when bandwidth is increased to 900Mbps. Similarly, two-tier architecture improved from 0.014 seconds to below 0.01 seconds. However, the WAN delay remains the same. Therefore, it is understood that increasing the WLAN bandwidth masks and delays the presence of a bottleneck.

Regarding the percentage of failed tasks due to mobility, no significant improvement in the number of failed tasks can be noticed when the load condition exceeds 250 devices. However, increasing the WLAN bandwidth limits the appearance of bottlenecks, as shown in Figures 13C and 13D. Furthermore, further log file analysis finds that VM utilisation remains high, meaning it can be deduced that in high traffic of 300 and 350 mobile devices, bandwidth is reserved by the server for future requests due to high arrival probability. Hence, the average failed tasks remains constant as well as the service time. Figure 15 confirms this claim as service time remains at 3 seconds for two-tier with EO architecture, which is

unfitting for the time-sensitive requirements of the application. Therefore, an increase in WLAN bandwidth with another parameter can be advantageous.

7.2.3 Effect of VM processing speed

The experiment is designed to vary the VM processing speed to investigate the effect on scalability. The initial experiment is configured with 1000 MIPS processing speed, so an increase to 2000, 3000, and 4000 MIPS is studied.

The experiment results establish that increasing the VM processing speed substantially reduces the percentage of failed tasks for all architectures. For example, Figure 19 indicates 40% task failure for single-tier architecture, further reduces to 16% with 2000 MIPS processing speed and 3% with 4000 MIPS processing speed. Moreover, increasing the VM processing speed lowered task failure due to VM capacity. Two-tier with EO architecture experiences phenomenal performance utilising 2000 MIPS or higher with near-zero task failure due to VM capacity.

Significant improvement is also realised in failed tasks due to mobility – reducing to below 2% task failure for 4000 MIPS processing speed. It is inferred that as tasks are distributed to all edge servers and executed quickly, average task failure due to mobility remains low compared to the initial experiment. It can be deduced from the initial experiment that lower processing speed introduces congestion in attractive locations, which increases service time.

Figure 25 illustrates that increasing the VM processing speed significantly reduces VM utilisation. Remarkably, with 4000 MIPS processing speed, single-tier, two-tier, and two-tier with EO architectures achieved 23%, 20%, and 15% VM utilisation, respectively – demonstrating significant improvement from the initial experiment. Furthermore, regarding average processing time, the processing time is substantially improved with increased processing capabilities. Thus, it is reasoned that the fast execution of task requests minimise the probability of VM overload or congestion.

Enhanced VM utilisation and processing time results in reduced service time, even for very sizable load conditions. The processing time expends most of the total service time due to the computational intensity of face recognition. Therefore, improving the processing time certainly impacts service time. Figure 22D shows the improvement of adopting 4000 MIPS processing speed as all architectures achieve a service time of 0.7 seconds or less. Nonetheless, Figure 24 reinforces that no improvement in network delay is realised due to

increasing VM processing speed. Sonmez et al. [11] [48] state that increasing the VM processing speed increases the network delay when WLAN congestion occurs. However, the network delay remains constant in this experiment and analysis of the log files reveal that WLAN congestion is not present, validating the reason for network delay not increasing.

Further inspection of WAN delay and average network delay data insinuates that full WLAN capacity is attained. Therefore, the edge architecture may realise further performance improvements by increasing the WLAN bandwidth. Hence, based on these interpretations, increasing WLAN bandwidth and VM processing speed simultaneously is the subsequent logical investigation to improve performance in terms of scalability of all layers.

7.2.4 Effect of WLAN bandwidth and VM processing speed

The experiment is designed to simultaneously increase the WLAN bandwidth and VM processing speed to investigate the effect on scalability. Sub experiments are designed with the following values: 300 Mbps – 1000 MIPS, 500 Mbps – 2000 MIPS, 700 Mbps – 3000 MIPS, and 900 Mbps – 4000 MIPS.

The experiment results conclude that increasing the WLAN bandwidth and VM processing speed significantly reduces the percentage of failed tasks. Figure 26A represents the 300 Mbps – 1000 MIPS sub experiment. Single-tier, two-tier, and two-tier with EO provide 45%, 32%, and 15% average failed tasks. Each successive sub experiment shows improved performance. Comparatively, Figure 26D represents the 900 Mbps – 4000 MIPS, single-tier, two-tier, and two-tier with EO provide 2.5%, 1.6%, and 1.2% average failed tasks, respectively. The results proclaim improved performance compared to Figure 19D, which shows the effect of 4000 MIPS VM processing speed alone.

Figure 28 features the importance of two-tier with EO architecture consuming the resources of other edge servers in the network infrastructure. Failed tasks due to VM capacity for this architecture is zero or close to zero due to the abundant computational resources when both values are increased. Regarding failure due to mobility, sub experiment 900 Mbps – 4000 MIPS in Figure 27D demonstrate less than 1.6% failed for load condition of 350 devices. The mobility of end-user devices is handled via the live migration of edge resources. Hence, the EO can utilise VMs with more processing capabilities to meet the application’s requirements.

Figure 30D displays average processing time; the data contends that processing can be completed within 0.6 seconds – suitably meeting the computationally intensive and time-

sensitive application requirements of augmented reality while providing a very acceptable quality of service. Furthermore, an increase in WLAN bandwidth and VM processing speed significantly reduces VM utilisation. Therefore, more efficient VM utilisation and reduced average processing time impact the service time. This improvement is evident in Figure 29D, where 0.5 second service time is achieved for a load condition of 350 devices.

Lastly, Figure 31 represents average network delay; increasing the WLAN bandwidth and VM processing speed offers minor improvement in network delay. This slight improvement is not apparent in Figure 24, in which the effect of VM processing speed is investigated alone. As a result, increasing both parameters improves the overall performance due to reduced VM overload and diminished WLAN congestion.

7.3 Findings and Recommendations

7.3.1 Findings of the investigation

Table 4. *Findings of the investigation supported by the literature review*

Key parameter	Findings of the investigation supported by the literature review
Effect of Edge Servers	A reduction in edge servers adversely impacts performance due to increased VM overload and unavailability of required computational resources, causing an increase in the average percentage of failed and failed tasks due to VM capacity – supported by [50]. High VM utilisation was apparent, leading to increased service and processing time. Hence, increasing the number of devices produces increased failure due to mobility, revealing limitations in terms of scalability as simultaneous requests of mobile users are not processed in a timely manner for an augmented reality application. Two-tier with EO provides the best performance but faces near-maximum VM utilisation and high failure due to mobility. However, this is counterbalanced as the EO provides fewer failed tasks due to VM capacity than two-tier and single-tier architecture – supported by [13]. An increase in edge serves significantly reduces the average percentage of failed tasks – supported by [13] [48]. Due to more effective workload distribution and increased availability of computational resources, VM congestion reduces. Hence, processing time and service time reduces. VM utilisation also decreases due to increasing edge servers. Single-tier architecture provides the worst performance, closely followed by two-tier architecture. Two-tier with EO provides the best performance due to load balancing and distribution of tasks among any edge server. The slight transmission delay between the

	EO and edge nodes is negated as the architecture eliminates flaws introduced by scaling the distributed system with increasing load conditions – supported by [13] [50].
Effect of WLAN bandwidth	An increase in WLAN bandwidth gains no significant improvement in the average percentage of failed tasks – confirmed by [11] [13]. However, an increase in bandwidth reduces the average WLAN delay, reducing the average network delay slightly – supported by [13]. Thus, increasing WLAN bandwidth only masks the presence of bottlenecks [13]. In the case of network delay, two-tier with EO architecture provides a slightly higher delay and worse performance than two-tier architecture for load conditions below 200 mobile devices. On the other hand, two-tier with EO provides a slightly lower delay and better performance than two-tier architecture for load conditions above 200 mobile devices. Single-tier provides the lowest network delay due to no WAN delay as tasks are not offloaded to the cloud – backed by {15].
Effect of VM processing speed	An increase in VM processing speed reduces the average percentage of failed tasks – supported by [13]. The percentage of failed tasks due to VM capacity reduces significantly for all load conditions. VM utilisation decreases significantly, hence reducing processing time and, therefore, service time – validated by [11] [48] [13] [50]. VM utilisation is significantly reduced for all architecture layers. As a result of enhanced VM utilisation, the percentage of failed tasks due to mobility reduces also. However, increasing the VM processing speed realised no improvement in network delay. Overall, single-tier architecture offers the worst performance, followed by two-tier architecture. Two-tier with EO offers the best performance in failed tasks, service time, and VM utilisation.
Effect of WLAN bandwidth and VM processing speed	The relationship between VM processing speed and WLAN bandwidth is clear. The findings of the VM processing speed experiment alone show no improvement in network delay. However, the findings of the WLAN bandwidth experiment alone recognise reduced WLAN delay and consequently reduced network delay. Hence, combining two significant parameters investigates whether a high arrival probability of tasks due to increased bandwidth and additional computational processing speed improves scalability. An increase in both parameters simultaneously significantly reduces the percentage of failed tasks – supported by [13]. Furthermore, the increase in task arrival probability due to increased bandwidth allows full utilisation of the enhanced VM processing capabilities. As a result, VM utilisation reduces significantly, and faster processing time reduces service time. In addition, minor improvement in network delay is evident, which was not present when investigating the effect of VM processing speed alone. Overall combining the parameters provides better overall performance than the previous experiments.

	Generally, single-tier architecture offers the worst performance, closely followed by two-tier architecture. Two-tier with EO architecture provides the best overall performance in terms of scalability.

7.3.2 Hypothesis Confirmation

The devised hypotheses for this empirical investigation are validated with the findings of the experiment. ✅ signifies hypothesis holds true for this investigation.

✅ **Hypothesis 1** – An increase in the number of edge servers reduces the percentage of failed tasks, service time, and VM utilisation.

✅ **Hypothesis 2** – An increase in WLAN bandwidth reduces the network delay

✅ **Hypothesis 3** – An Increase in VM processing speed reduces the percentage of failed tasks

✅ **Hypothesis 4** – An increase in VM processing speed and WLAN bandwidth reduces the percentage of failed tasks, service time, and VM utilisation

This investigation investigates scalability explicitly on the use case of a face recognition application as part of an Augmented reality application for mobile devices, where the application demands large data upload but small data download. Therefore, the percentage of failed tasks is a very important evaluation metric for the characteristics of the application. Hence a ranking can be formulated based on the percentage of failed tasks as follows:
1. VM processing speed and WLAN bandwidth 2. VM processing speed 3. Edge servers 4. WLAN bandwidth.

7.3.3 Recommendations

1. **Edge Architecture scalability recommendations –** The recommended edge architecture for a face recognition application as part of augmented reality is two-tier with EO architecture as it provides the best performance and is more scalable than the other architecture. In addition, increasing the load condition found that two-tier with EO consistently outperforms other architecture in the percentage of failed tasks and service time. VM utilisation is high due to the consumption of other edge servers but can be significantly reduced by increasing a combination of parameters.

2. **The link between VM processing Speed and WLAN bandwidth –** Edge servers, VM processing speed, WLAN bandwidth, and VM processing speed and WLAN bandwidth were investigated. The experiments show the relationship between VM

processing speed and WLAN bandwidth. If the WLAN bandwidth is increased, WLAN delay improves, but no improvement in failed tasks exists from the initial experiment as the percentage of failed tasks remains unfavourable for all architectures, especially for high load conditions. However, if VM processing speed increases, significant improvement in failed tasks is realised, but no improvement in network delay. Hence, increasing both key parameters means that increased WLAN bandwidth provides a faster rate of task request arrival, while increased VM processing speed provides sufficient computational resources to handle all requests effectively. Combining significant parameters equates to further improved performance in terms of scalability.

3. **The trade-off between performance and computational resources** – The potential of edge computing fulfils different applications demands. For example, specific applications are latency-sensitive and delay-sensitive, whereas some real-time applications demand a near-zero percentage of failed tasks. This investigation proposes an edge configuration of 900 Mbps WLAN bandwidth and 4000 MIPS processing speed to achieve an exceptionally low average percentage of failed tasks and zero percentage of failed tasks due to VM capacity. However, the configuration of 700 Mbps WLAN bandwidth and 3000 MIPS can achieve a similar percentage of failed tasks with slightly higher service time and VM utilisation, which may be adequate to support certain applications. Thus, the consideration for using simulation for strategic deployment of edge resources depending on the application's requirements and cost/energy factors is recommended.

4. **Handling the mobility of IoT devices** – Arising mobility of users and devices in the wireless network requires low-latency processing of tasks. Therefore, task failure due to mobility is a vital evaluation metric. Depending on the type of IoT devices at the edge, different demands are required by the application to support mobility. By acknowledging IoT devices' behaviour and wait times at locations, the dynamic migration of resources across multiple edge nodes is recommended to provide sufficient resources to maintain performance. Furthermore, for IoT devices, such as mobile devices, it is suggested to provide sufficient computational resources to ensure tasks are fulfilled in time before devices move outside the dedicated WLAN area. But, for IoT devices with high mobility, such as driverless vehicles, efficient handoff of processing requests to the next edge server is advised to maintain performance.

Chapter 8
Project Evaluation

8.1 Evaluation

8.1.1 Aims and Objectives Evaluation

Aim. The project investigated several types of edge architectures, principally single-tier, two-tier, and two-tier with Edge Orchestrator (EO) deployed in a specific domain to determine the effect of computational and networking system parameters on performance results. A fundamentally sound methodology was followed to generate accurate and truthful results based on iterations of an initial experiment to fulfil the aim. In addition, a deeper understanding of the scenario and simulation representation was achieved after each iteration and modification of configuration values promoting further experiments to test each hypothesis thoroughly. Finally, the aim was accomplished as the project outcome successfully presented findings and conclusions to propose practical recommendations for future Edge architecture design. One criticism is that recommendations proposed in this study apply to the specific architecture and application investigated. Therefore, while findings are backed by literature, further study on different applications is required to validate generalised claims.

Objectives. 5 SMART objectives were defined initially. The justifications clarify the reasons why each objective has been met.

Table 5. *Justifications for project objectives*

Objective	Justification
Identify the critical issues about edge architecture simulation and edge computing performance.	By way of comprehensive background research and literature review of relevant resources, critical issues for investigation were proposed. By evaluating key parameters and the performance potential of specific architectures, the scope was defined, and hypotheses were set to provide valuable findings. Further, the suitability of EdgeCloudSim for scenario representation and the visualisation of data was considered.
Experiment design and implementation by configuring key parameters to investigate performance-based on single-tier,	An initial experiment was conducted through the agile and iterative experiment design process to derive key parameters for investigation. A specific application scenario was determined that was accurately represented by the EdgeCloudSim simulation tool.

two-tier, and two-tier with EO architectures.	A further set of experiments were conducted iteratively to produce results for interpretation and analysis.
Interpret experiment results by conducting an analysis of results/data from the produced graphs.	Experiment results were formatted graphically into readable graphs for ease of interpretation. Clear findings were determined by analysing the numerical data and trends of each graph.
Based on a thorough technical evaluation of results, present findings and justifications using existing literature. Verify hypothesis using strong evidence.	Justifications for results were provided, and findings were verified using existing literature or expert knowledge where appropriate. Additionally, hypothesis verification and evidence of meeting expected outcomes were proposed, including the reasoning for any shortcomings.
Provide recommendations based on the architecture of edge computing applications concerning scalability.	The simulation experiments and technical evaluation uncovered key parameters to reveal the effect on performance and scalability. Additional recommendations based on the single-tier, two-tier, and two-tier with Edge Orchestrator (EO) deployed in a specific domain in specific scenarios to potentially address the demands of Edge Computing. Further recommendations for edge orchestration, efficient resource management, and IoT composition were asserted.

8.1.2 Project Management

Aims and objectives were asserted early in the project, ensuring a clear vision of the project's deliverables, and expected outcome. Additionally, the research stage and literature review was highly organised as obtaining relevant resources enabled a comprehensive understanding of the study area. Wider reading into the potential of Edge Computing, Parallel and Distributed Simulation, Workload Orchestration, and Energy-Efficient Models helped form experiment ideas. A clear-cut agile methodology was followed by designing an initial experiment, collecting the data, and interpreting/analysing the results – the methodology allowed for iterations of the initial experiment to assist purposeful experiment design. Representing the application using the simulator required many iterations to ensure all configurations were accurate, so results were reliable. Repeating of experiments numerous times ensured anomalous results were rectified.

In terms of time management, the background research and literature review were completed within the allocated timeframe. However, experiment design and implementation required extended time to understand how to represent the application. Warm-up exercises were conducted to familiarise the simulation tool, but understanding modules and scripts required extensive learning. Anticipating the intense allocation of modules during the first

semester, conducting most of the experimentation in the second semester was the strategy. However, due to exceptional mitigating circumstances, strictly following the project timeline was not possible. A revised project timeline and setting of SMART tasks enabled work to continue for this project.

8.1.3 Limitations

1. The experiments examined increasing both WLAN bandwidth and VM processing time incrementally. With more time, investigating all possible intermediary sub experiments, including 300 Mbps – 2000 MIPS, 300 Mbps – 3000 MIPS, 300 Mbps – 4000 MIPS, to find the optimal parameter values for scalability could benefit edge architecture research.
2. NEXT_FIT orchestration policy was used during the design of the initial experiment. Yet, further exploration of orchestration policies could help produce a more realistic edge scenario.
3. An additional aspect of scalability is for the network to remain effective and somewhat resilient to failure. Liu et al. [60] deduce that most active connections occur on short routes under high load, leading to multiple link failures. Subsequently, link failures and edge device failures may occur when systems are deployed, affecting the availability of edge nodes and servers. These considerations were not simulated in this study.

8.2 Related Work

8.2.1 Previous Work

Previous work has explored edge computing architecture for different applications. But many performance parameters are yet to be investigated. This investigation extends edge architecture research by combining parameters to achieve better performance and scalability.

8.2.2 Future Work

Future work entails:

1. Conducting further experiments using a wider combination of scalability parameters.
2. Considering energy-aware edge server placement to improve scalability.
3. Performing a reliability analysis of edge scenarios to understand the exact behaviour and impact of high mobility of IoT devices on edge infrastructure.
4. A deeper analysis of simulation results for the edge servers and the cloud separately to realise instances where offloading to the cloud provides favourable performance.

For instance, finding whether local processing, edge server offloading, or remote cloud server offloading provides enhanced performance.

5. Further encompassing EdgeCloudSim to evaluate the computational cost of increasing scalability parameters.
6. Utilising different edge orchestration policies to investigate better scalability.
7. Evaluating the effect of different tasks sizes and task migration among edge nodes.
8. Real-world deployment to confirm the validity and reliability of simulation results.

8.3 Reflection and Self-appraisal

By completing this project, I achieved a profound understanding of the subject area and have gained admiration for the potential and capability of Edge Computing paradigms and their application in the world. In addition, my supervisor instilled great interest and passion into the subject area, which elevated my motivation. Furthermore, supervision meetings helped make critical project decisions and clarify understanding of concepts. Moreover, after every supervision meeting, I gained vast new knowledge, which required conducting further research throughout the project, which improved the study's quality. By reviewing meetings, I aligned my work to my supervisor's advice whilst retaining the freedom to progress toward my interests within the project's scope. Modules studied, such as Distributed Systems, Networks, and Artificial Intelligence, were beneficial.

Technical challenges of implementation and coding were overcome through EdgeCloudSim GitHub resources, videos, and forums. However, graph production was complex and required more time as further processing was needed for readability. Finally, by following a systematic literature review process, I learned to assess technical work critically.

Chapter 9
Conclusion

In conclusion, this study has investigated the effect of computational and network system parameters on the scalability of single-tier, two-tier, and two-tier with EO architecture for a face recognition / augmented reality application. The simulation results emphasised the importance of edge orchestration and the ability to consume any edge server. Furthermore, a combination of scalability parameters identified the importance of edge architecture design to support the application's requirements.

The findings are important as the future of IoT presents unique opportunities in which face recognition applications require real-time analytics. Some use cases include authentication systems for industrial IoT organisations, unlocking smartphones, validating identity at ATMs, accurate recognition of people wearing masks through mask detection, searching for missing people, and face recognition for smart supermarkets.

As such, this investigation proposes design recommendations for applications requiring a low percentage of failed tasks, low service time, minimal network delay, and enhanced VM utilisation. Furthermore, the simulation experiments represent the significantly increasing number of IoT devices as experiments with increased load conditions evaluated edge architecture capabilities.

Last, of all, it is hoped that this study inspires future research where additional scalability parameters and evaluation metrics are evaluated.

List of References

[1] Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K. and Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience*, 47(9), pp.1275–1296.

[2] What *are IoT Devices*. [online] Arm | The Architecture for the Digital World. Available at: https://www.arm.com/glossary/iot-devices.

[3] IBM. *The edge computing advantage*. [online] Available at: https://www.ibm.com/thought-leadership/institute-business-value/report/edge-computing-advantage.

[4] Hamdan, S., Ayyash, M. and Almajali, S. (2020). Edge-Computing Architectures for Internet of Things Applications: A Survey. *Sensors*, 20(22), p.6441.

[5] Sheltami, T.R., Shahra, E.Q. and Shakshuki, E.M. (2018). Fog Computing: Data Streaming Services for Mobile End-Users. *Procedia Computer Science*, 134, pp.289–296.

[6] Ray, P.P., Dash, D. and De, D. (2019). Edge computing for Internet of Things: A survey, e-healthcare case study and future direction. *Journal of Network and Computer Applications*, [online] 140, pp.1–22. Available at: https://www.sciencedirect.com/science/article/pii/S1084804519301651.

[7] Ai, Y., Peng, M., Zhang K. (2017). Edge cloud computing technologies for internet of things: A primer. Digital Communications and Networks. 4. 10.1016/j.dcan.2017.07.001.

[8] Ray, P.P. (2015). *Notice of Removal: Internet of Things for Sports (IoTSport): An architectural framework for sports and recreational activity*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/7 253963.

[9] Shi, W. and Dustdar, S. (2016). The Promise of Edge Computing. *Computer*, 49(5), pp.78–81.

[10] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H. and Zhao, W. (2017). A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. *IEEE Internet of Things Journal*, 4(5), pp.1125–1142.

[11] Sonmez, C., Ozgovde, A. and Ersoy, C. (2018). EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11), p.e3493.

[12] Mell, P.M. and Grance, T. (2011). The NIST definition of cloud computing. [online] Available at: https://csrc.nist.gov/publications/detail/sp/800-145/final.

[13] Aljulayfi, A.F. and Djemame, K. (2018). Simulation of an augmented reality application for driverless cars in an edge computing environment. *2018 Fifth International Symposium on Innovation in Information and Communication Technology (ISIICT)*. [online] Available at: https://eprints.whiterose.ac.uk/143898/7/52_Leeds_Uni.pdf.

[14] Akherfi, K., Gerndt, M. and Harroud, H. (2018). Mobile cloud computing for computation offloading: Issues and challenges. *Applied Computing and Informatics*, 14(1), pp.1–16.

[15] Liu, Y., Lee, M.J. and Zheng, Y. (2016). Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System. *IEEE Transactions on Mobile Computing*, 15(10), pp.2398–2410.

[16] Chiang, M. and Zhang, T. (2016). Fog and IoT: An Overview of Research Opportunities. *IEEE Internet of Things Journal*, 3(6), pp.854–864.

[17] Lyu, B., Yuan, H., Lu, L. and Zhang, Y. (2021). Resource-constrained Neural Architecture Search on Edge Devices. *IEEE Transactions on Network Science and Engineering*, pp.1–1.

[18] Ryu, D.-H. (2015). Development of IoT Gateway based on Open Source H/W. *The Journal of the Korea institute of electronic communication sciences*, 10(9), pp.1065–1070.

[19] Thangiah, L., Ramanathan, C. and Chodisetty, L.S. (2019). *Distribution Transformer Condition Monitoring based on Edge Intelligence for Industrial IoT*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/8767272.

[20] Goel, K., Bhaumick, A., Kaushal, D. and Bagchi, S. (2020). *Reliability Analysis of Edge Scenarios Using Pedestrian Mobility*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/9159132.

[21] Mahmoudi, C., Mourlin, F. and Battou, A. (n.d.). *Formal Definition of Edge Computing: An Emphasis on Mobile Cloud and IoT Composition*. [online] Available at: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=919490.

[22] Sonmez, C., Ozgovde, A. and Ersoy, C. (2019). Fuzzy Workload Orchestration for Edge Computing. *IEEE Transactions on Network and Service Management*, 16(2), pp.769–782.

[23] Mao, Y., Zhang, J. and Letaief, K.B. (2016). Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications*, [online] 34(12), pp.3590–3605. Available at: https://arxiv.org/pdf/1605.05488.pdf.

[24] You, C., Huang, K., Chae, H. and Kim, B.-H. (2017). Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading. *IEEE Transactions on Wireless Communications*, 16(3), pp.1397–1411.

[25] Zhou, Z., Liu, P., Chang, Z., Xu, C. and Zhang, Y. (2018). *Energy-efficient workload offloading and power control in vehicular edge computing*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/8368975.

[26] Zhang, K., Mao, Y., Leng, S., He, Y. and ZHANG, Y. (2017b). Mobile-Edge Computing for Vehicular Networks: A Promising Network Paradigm with Predictive Off-Loading. *IEEE Vehicular Technology Magazine*, 12(2), pp.36–44.

[27] Kumar, N., Zeadally, S. and Rodrigues, JJPC (2016). Vehicular delay-tolerant networks for smart grid data management using mobile edge computing. *IEEE Communications Magazine*, 54(10), pp.60–66.

[28] Huawei, Patel, M., Hu, Y., Hédé, P., Joubert, J., Thornton, C., Naughton, B. and Julian, I.: (2014). *Mobile-Edge Computing Introductory Technical White Paper*. [online] Available at: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf.

[29] Tianze, L., Muqing, W. and Min, Z. (2016). *Consumption considered optimal scheme for task offloading in mobile edge computing*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/7500484.

[30] Yu, W., Liang, F., He, X., Hatcher, W.G., Lu, C., Lin, J. and Yang, X. (2018). A Survey on the Edge Computing for the Internet of Things. *IEEE Access*, 6, pp.6900–6919.

[31] Sajjad, H.P., Danniswara, K., Al-Shishtawy, A. and Vlassov, V. (2016). *SpanEdge: Towards Unifying Stream Processing over Central and Near-the-Edge Data Centers*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/7774704.

[32] Rahimi, H., Picaud, Y., Singh, K.D., Madhusudan, G., Costanzo, S. and Boissier, O. (2021). Design and Simulation of a Hybrid Architecture for Edge Computing in 5G and Beyond. *IEEE Transactions on Computers*, 70(8), pp.1213–1224.

[33] Babar, M. and Sohail Khan, M. (2021). ScalEdge: A framework for scalable edge computing in Internet of things–based smart systems. *International Journal of Distributed Sensor Networks*, 17(7), p.155014772110353.

[34] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P. and Riviere, E. (2015). Edge-centric Computing. *ACM SIGCOMM Computer Communication Review*, 45(5), pp.37–42.

[35] Premsankar, G., Di Francesco, M. and Taleb, T. (2018). Edge Computing for the Internet of Things: A Case Study. *IEEE Internet of Things Journal*, 5(2), pp.1275–1284.

[36] Ouyang, T., Zhou, Z. and Chen, X. (2018). Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing. *IEEE Journal on Selected Areas in Communications*, 36(10), pp.2333–2345.

[37] Wang, S., Zhao, Y., Xu, J., Yuan, J. and Hsu, C.-H. (2019). Edge server placement in mobile edge computing. *Journal of Parallel and Distributed Computing*, [online] 127, pp.160–168. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0743731518304398?via%3Dihub.

[38] Abbas, N., Zhang, Y., Taherkordi, A. and Skeie, T. (2018). Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal*, 5(1), pp.450–465.

[39] Salama, M., Elkhatib, Y. and Blair, G. (2019). IoTNetSim. *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*.

[40] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F. and Buyya, R. (2010). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1), pp.23–50.

[41] Bahwaireth, K., Tawalbeh, L., Benkhelifa, E., Jararweh, Y. and Tawalbeh, M.A. (2016). Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications. *EURASIP Journal on Information Security*, [online] 2016(1). Available at: https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-016-0039-y.

[42] Zeng, X., Garg, S.K., Strazdins, P., Jayaraman, P.P., Georgakopoulos, D. and Ranjan, R. (2017). IOTSim: A simulator for analysing IoT applications. *Journal of Systems Architecture*, 72, pp.93–107.

[43] Malik, A., Qayyum, T., Rahman, A., Khattak, M., Khalid, O. and Khan, S. (2020). xFogSim: A Distributed Fog Resource Management Framework for Sustainable IoT Services. *IEEE Transactions on Sustainable Computing*, pp.1–1.

[44] Svorobej, S., Takako Endo, P., Bendechache, M., Filelis-Papadopoulos, C., Giannoutakis, K., Gravvanis, G., Tzovaras, D., Byrne, J. and Lynn, T. (2019). Simulating Fog and Edge Computing Scenarios: An Overview and Research Challenges. *Future Internet*, 11(3), p.55.

[45] Qayyum, T., Malik, A.W., Khan Khattak, M.A., Khalid, O., and Khan, S.U. (2018). FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment. *IEEE Access*, 6, pp.63570–63583.

[46] Margariti, S.V., Dimakopoulos, V.V. and Tsoumanis, G. (2020). Modeling and Simulation Tools for Fog Computing—A Comprehensive Survey from a Cost Perspective. *Future Internet*, 12(5), p.89.

[47] A comparative analysis of simulators for the Cloud to Fog continuum. (2020). *Simulation Modelling Practice and Theory*, [online] 101, p.102029. Available at: https://www.sciencedirect.com/science/article/pii/S1569190X19301601.

[48] Sonmez, C., Ozgovde, A. and Ersoy, C. (2017). *Performance evaluation of single-tier and two-tier cloudlet assisted applications*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/abstract/document/7962674.

[49] Jasim, M., Siasi, N., Malapaka, S., Oliveira, D. and Ugweje, O. (2020). *A Single-Tier Fog Architecture for Delay-Sensitive and Computation-Intensive S.F.C. Requests*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/abstract/document/9284860.

[50] Suryavansh, S., Bothra, C., Chiang, M., Peng, C. and Bagchi, S. (2019). Tango of edge and cloud execution for reliability. *Proceedings of the 4th Workshop on Middleware for Edge Clouds & Cloudlets - MECC '19*.

[51] Jha, D.N., Alwasel, K., Alshoshan, A., Huang, X., Naha, R.K., Battula, S.K., Garg, S., Puthal, D., James, P., Zomaya, A., Dustdar, S. and Ranjan, R. (2020). IoTSim-Edge: A simulation framework for modeling the behavior of Internet of Things and edge computing environments. *Software: Practice and Experience*, 50(6), pp.844–867.

[52] Tan, J., Chang, T.-H., Guo, K. and Quek, TQS (2021). Robust Computation Offloading in Fog Radio Access Network with Fronthaul Compression. *IEEE Transactions on Wireless Communications*, pp.1–1.

[53] Zaitoun, T.A., Issa, M.B., Banat, S. and Mardini, W. (2018). *Evaluation and Enhancement of the EdgeCloudSim using Poisson Interarrival time and Load capacity*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/abstract/document/8486288?casa_token=e_ZSltbax_QAAAAA:qEUWgEdx7v9ueqh-X0D6h26MImHQFp-zj1JJ1a9DHhjpHpKZ8xAGO-FS32kLsT0W5BmNiOQ.

[54] Hossain, M.D., Sultana, T., Hossain, M.A., Hossain, M.I., Huynh, L.N.T., Park, J. and Huh, E.-N. (2021). Fuzzy Decision-Based Efficient Task Offloading Management Scheme in Multi-Tier MEC-Enabled Networks. *Sensors*, 21(4), p.1484.

[55] Svorobej, S., Bendechache, M., Griesinger, G. (2020). springerprofessional.de. *Orchestration from the Cloud to the Edge*. [online] Available at: https://www.springerprofessional.de/en/orchestration-from-the-cloud-to-the-edge/18155450.

[56] Xia, W., Zhao, P., Wen, Y. and Xie, H. (2017). A Survey on Data Center Networking (DCN): Infrastructure and Operations. *IEEE Communications Surveys & Tutorials*, 19(1), pp.640–656.

[57] Jalalpour, E. (2018). *Dynamic Security Orchestration System Leveraging Machine Learning*. [online] undefined. Available at: https://www.semanticscholar.org/paper/Dynamic-Security-Orchestration-System-Leveraging-Jalalpour/efb993c5f96ed116250766609c184567249f67c6.

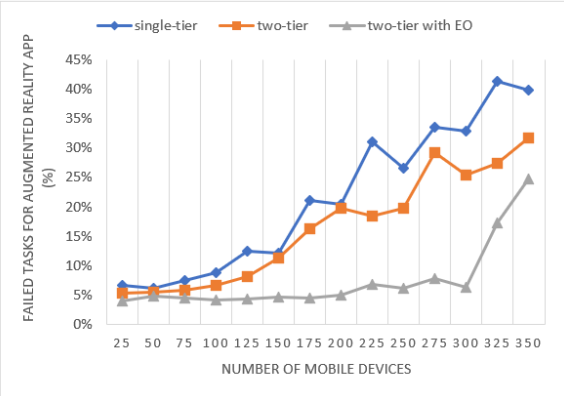
[58] Association for Computing Machinery (2018). *ACM Code of Ethics and Professional Conduct*. [online] Acm.org. Available at: https://www.acm.org/code-of-ethics.

[59] Farris, I., Taleb, T., Flinck, H. and Iera, A. (2017). Providing ultra-short latency to user-centric 5G applications at the mobile network edge. *Transactions on Emerging Telecommunications Technologies*, 29(4), p.e3169.

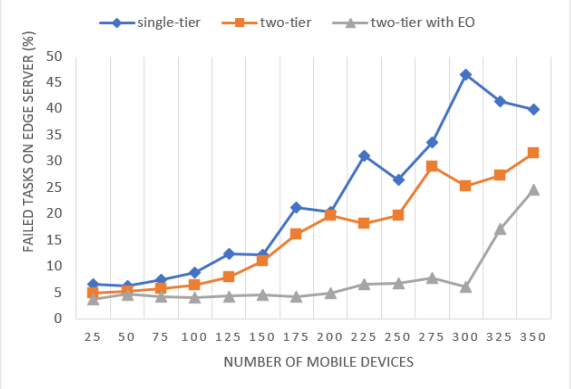
[60] Liu, G, and Ji, C., (2009). Scalability of Network-Failure Resilience: Analysis Using Multi-Layer Probabilistic Graphical Models. IEEE/ACM Transactions on Networking, 17(1), pp.319–331.

Appendix A
Final Simulation Results

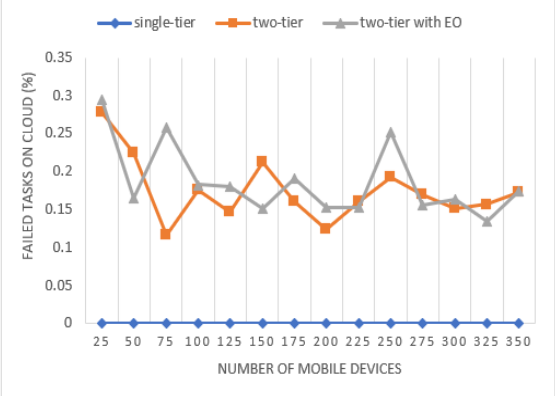
Initial experiment



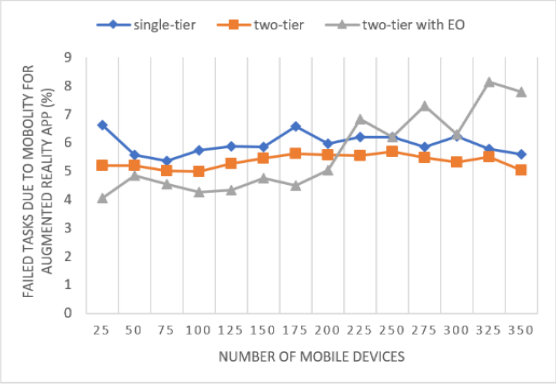
(A) Average failed tasks



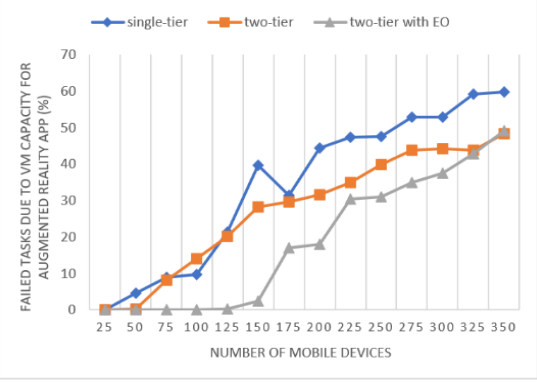
(B) Edge Server



(C) Cloud Server

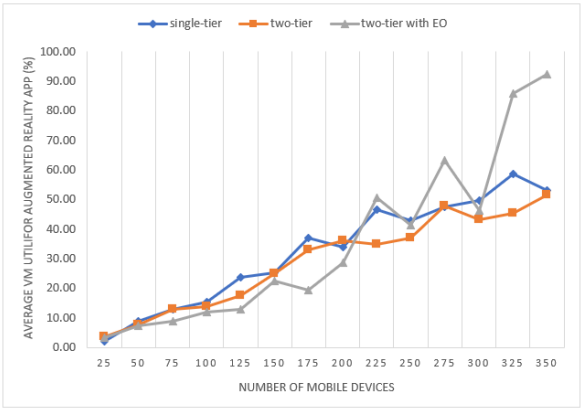


(D) Failed task due to mobility



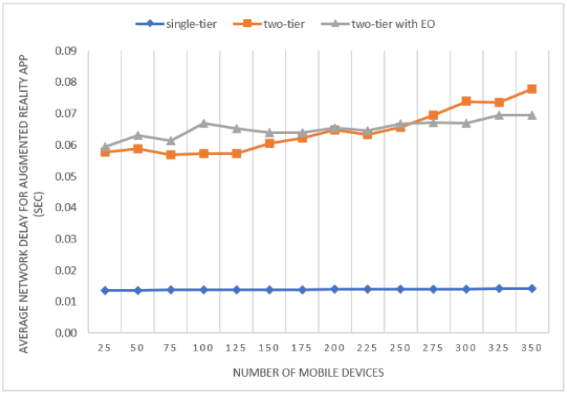
(E) Failed task due to VM capacity

Figure 1. Percentage of failed tasks with reason – Initial experiment

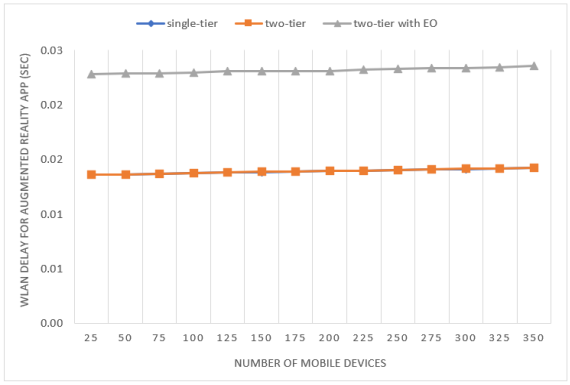


(A) VM Utilisation

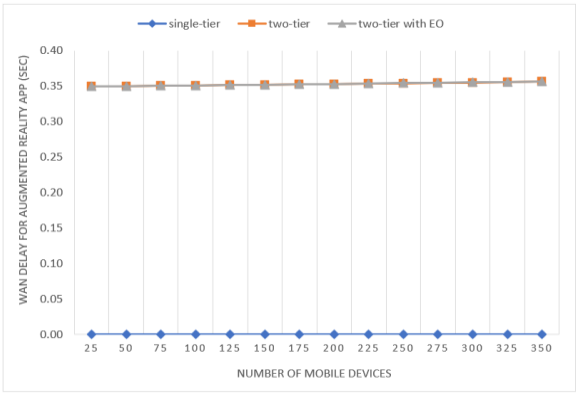
Figure 2. VM utilisation – Initial experiment



(A) Average network delay

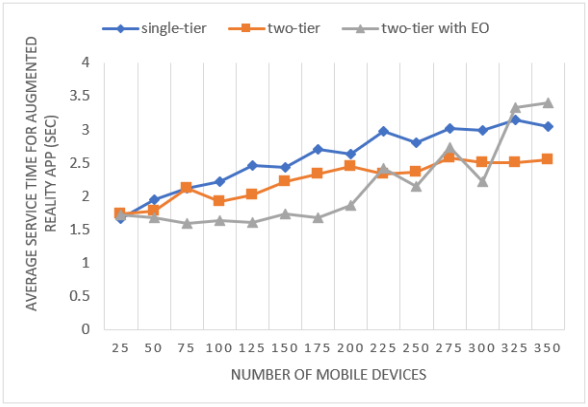


(B) WLAN delay

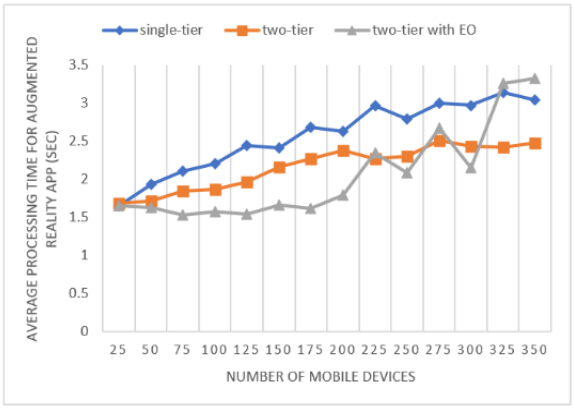


(C) WAN delay

Figure 3. Network delay – Initial experiment



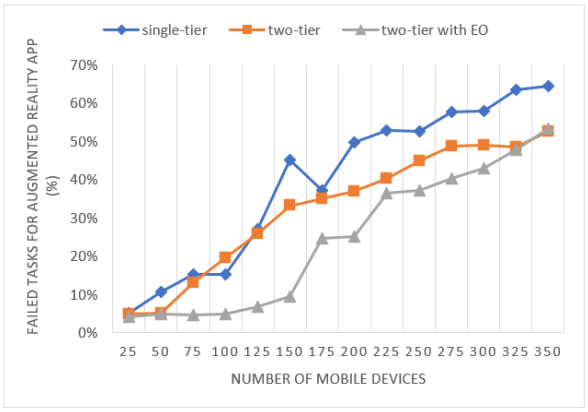
(A) Service time



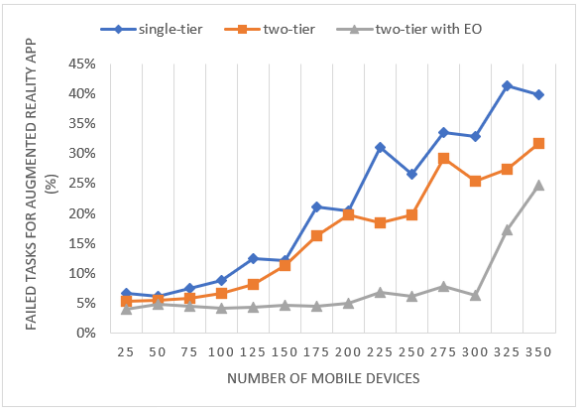
(B) Processing time

Figure 4. Processing time and service time – Initial experiment

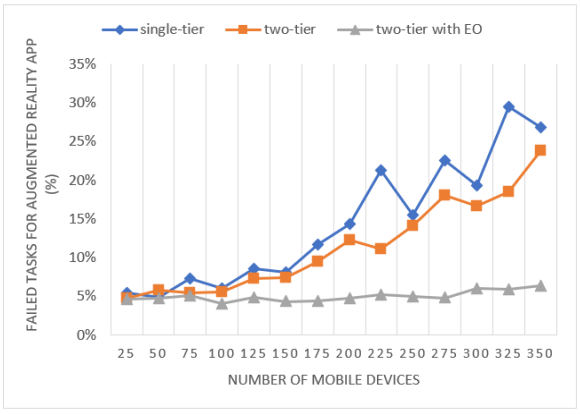
Effect of Edge Servers



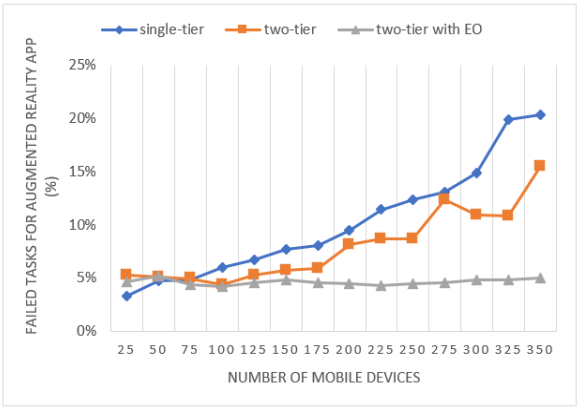
(A) 2 Servers



(B) 4 Servers

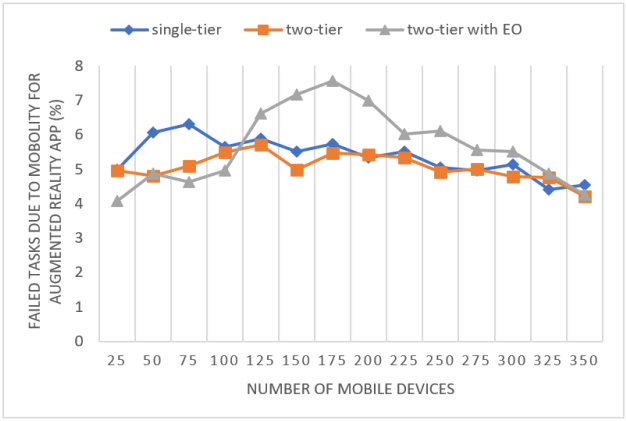


(C) 6 Servers

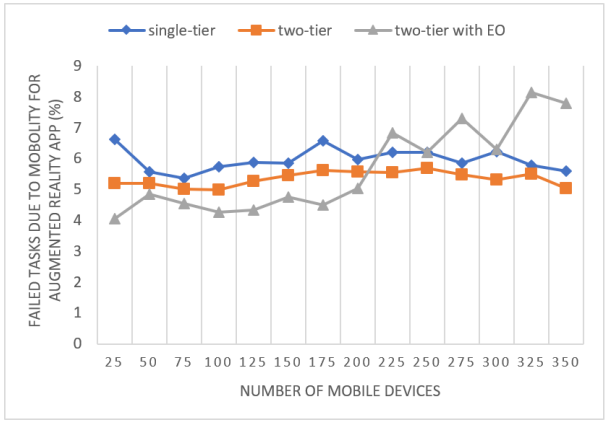


(D) 8 Servers

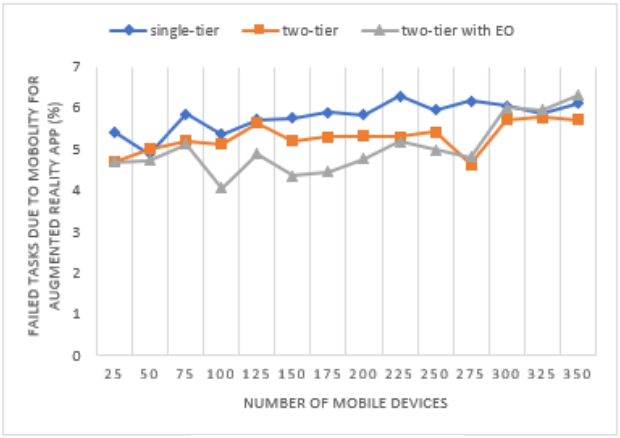
Figure 5. Average percentage of failed tasks – Effect of Edge Server



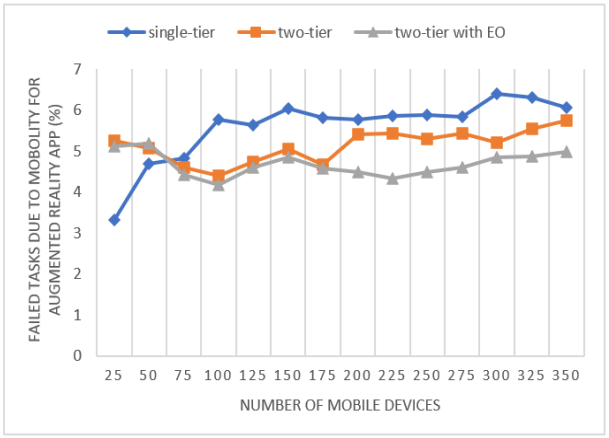
(A) 2 Servers



(B) 4 Servers

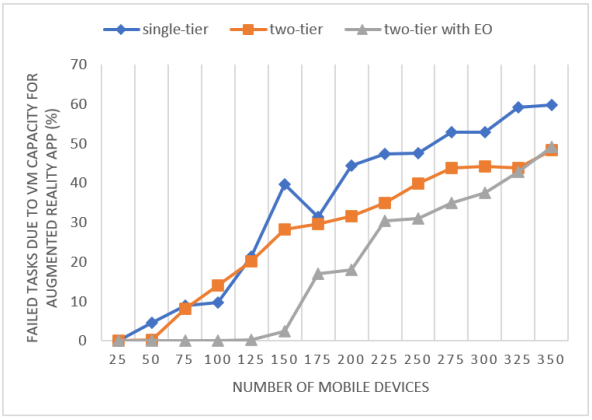


(C) 6 Servers

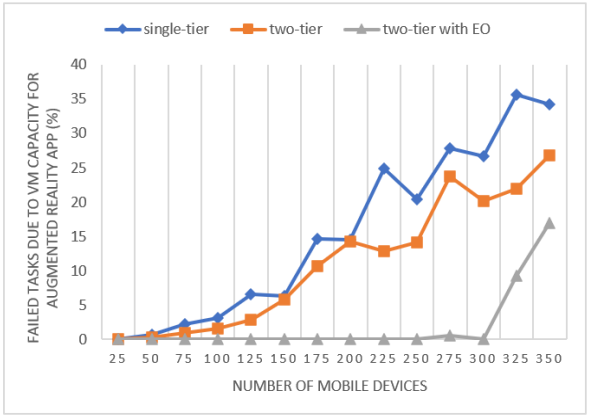


(D) 8 Servers

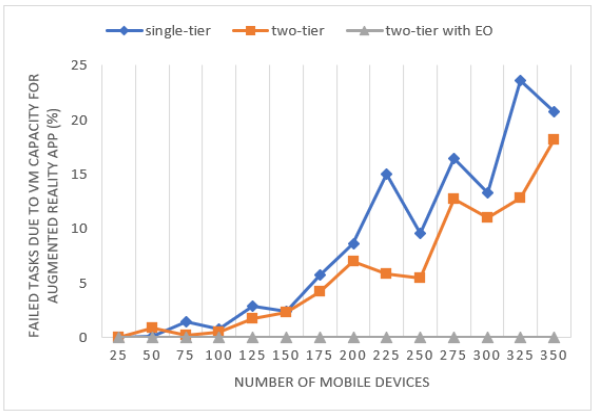
Figure 6. Percentage of failed tasks due to mobility – Effect of Edge Server



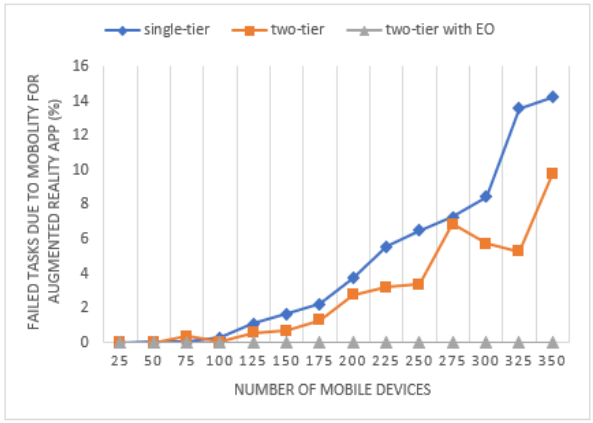
(A) 2 Servers



(B) 4 Servers

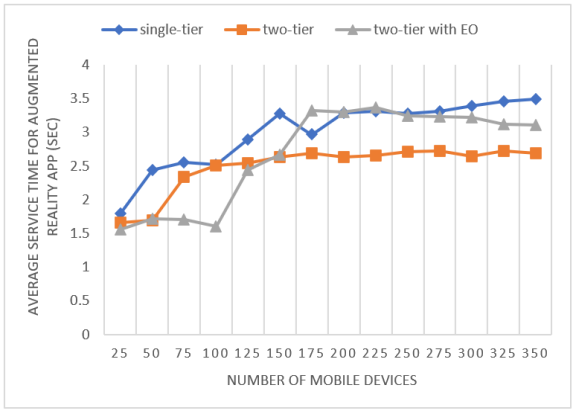


(C) 6 Servers

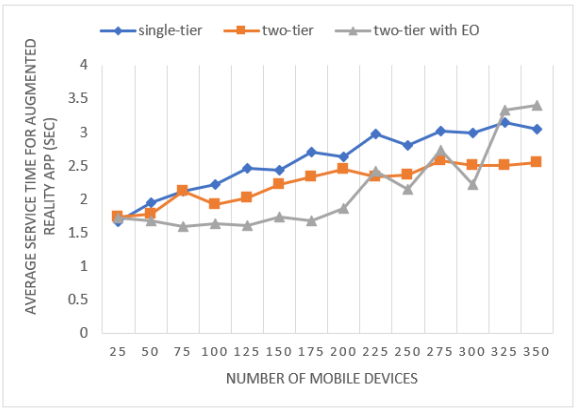


(D) 8 Servers

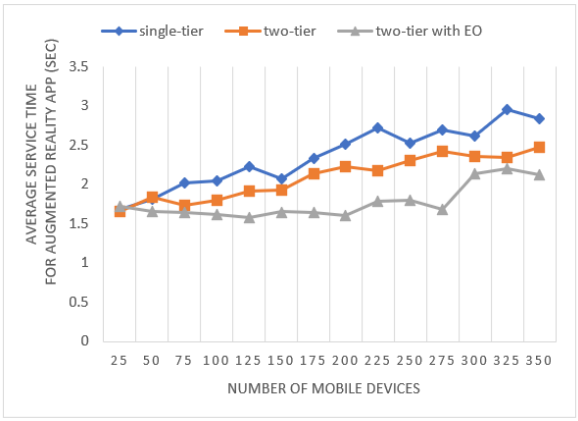
Figure 7. Percentage of failed tasks due to VM capacity – Effect of Edge Server



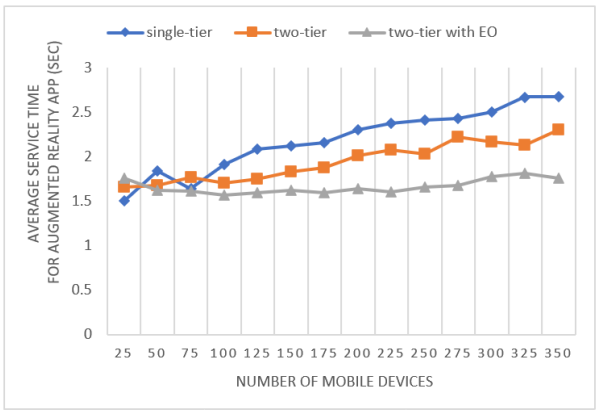
(A) 2 Servers



(B) 4 Servers

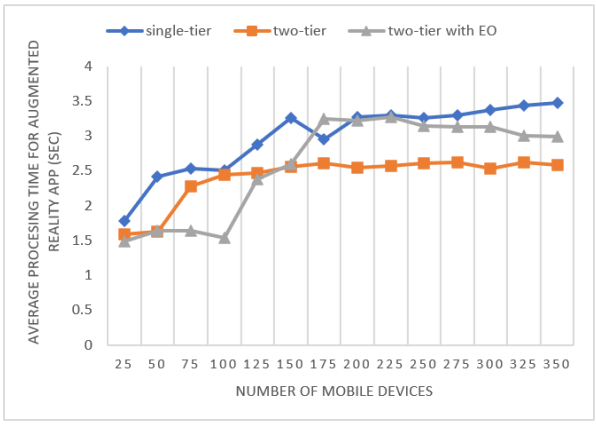


(C) 6 Servers

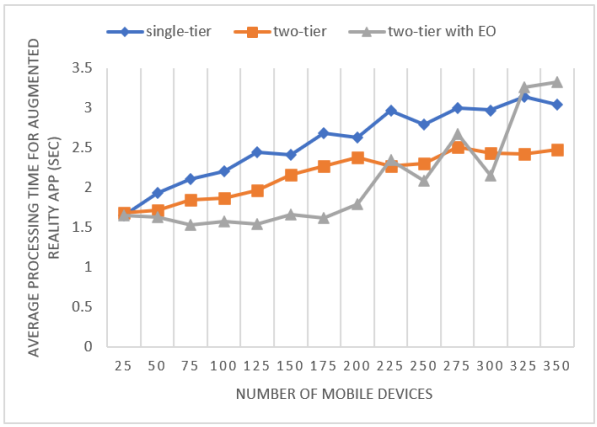


(D) 8 Servers

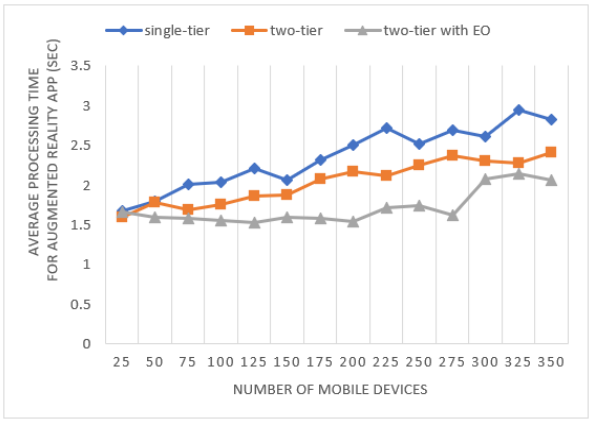
Figure 8. Average Service time – Effect of Edge Server



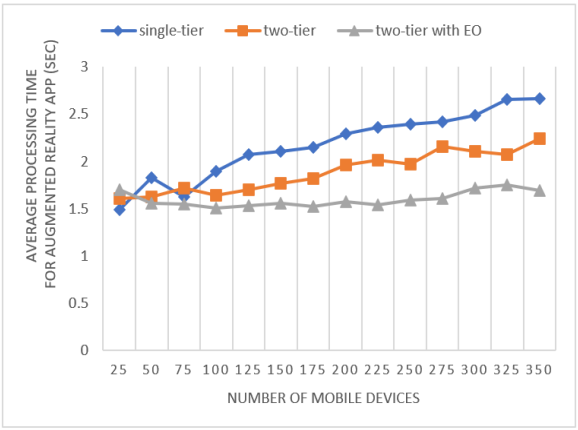
(A) 2 Servers



(B) 4 Servers

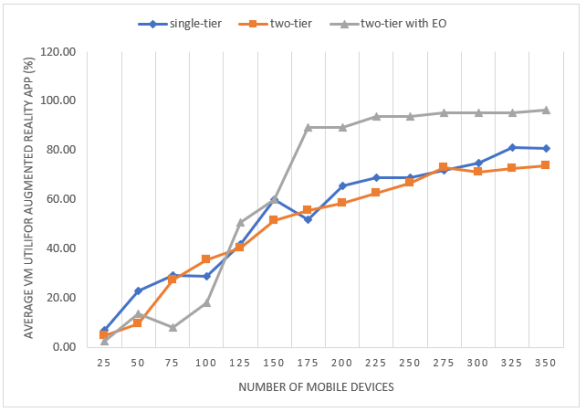


(C) 6 Servers

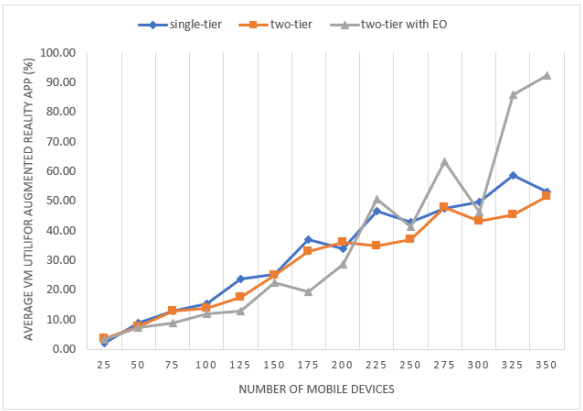


(D) 8 Servers

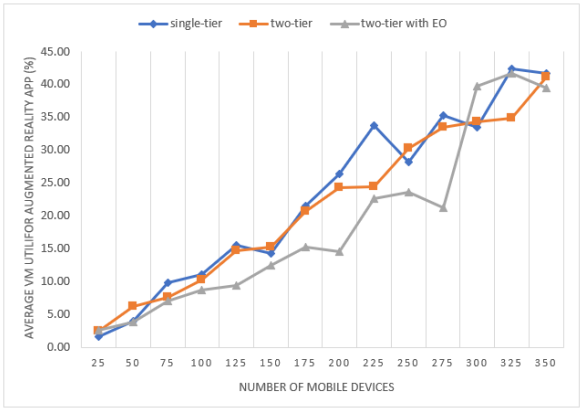
Figure 9. Average processing time – Effect of Edge Server



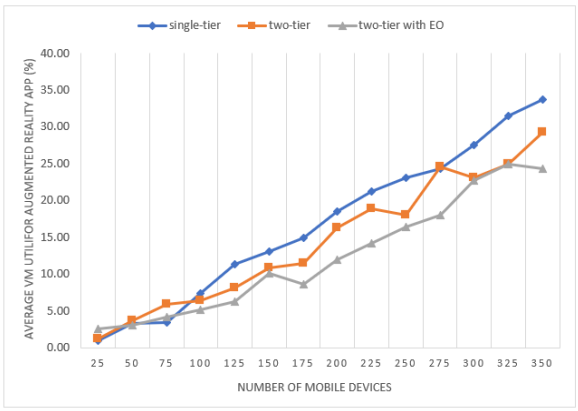
(A) 2 Servers



(B) 4 Servers

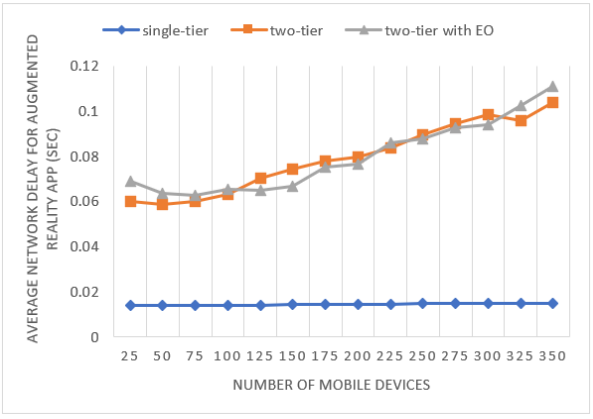


(C) 6 Servers

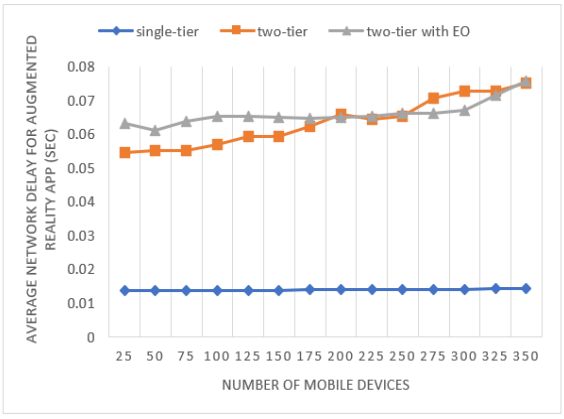


(D) 8 Servers

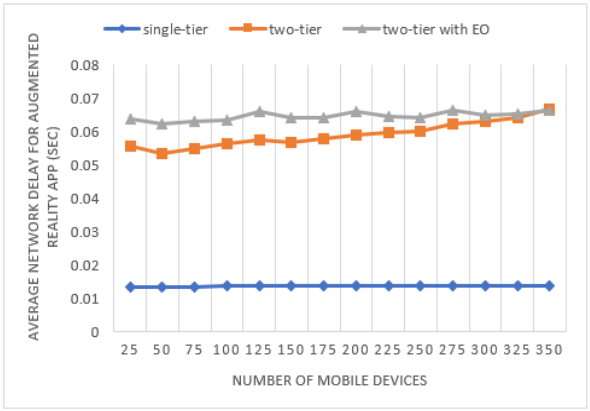
Figure 10. VM utilisation – Effect of Edge Server



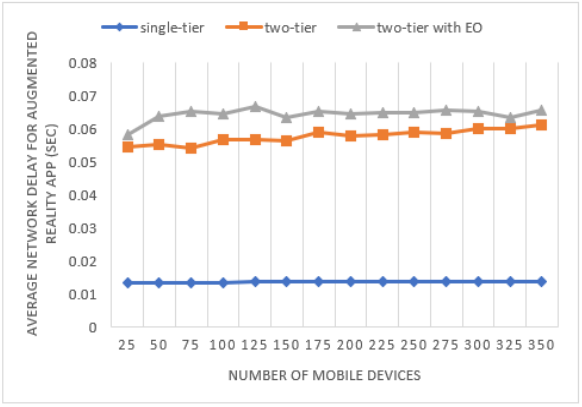
(B) 2 Servers



(A) 4 Servers



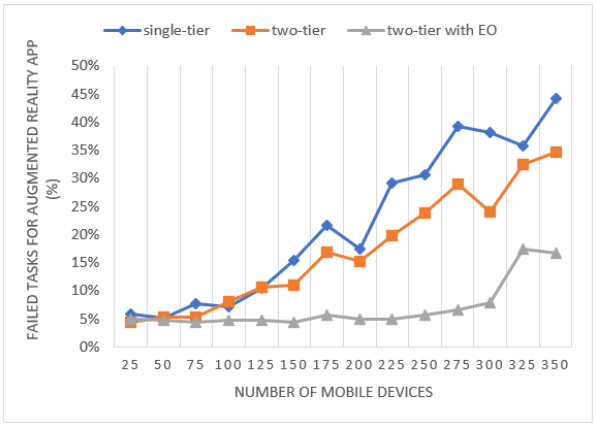
(C) 6 Servers



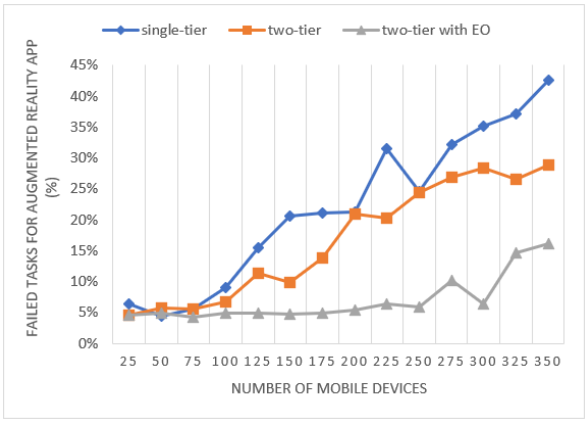
(D) 8 Servers

Figure 11. Network Delay – Effect of Edge Server

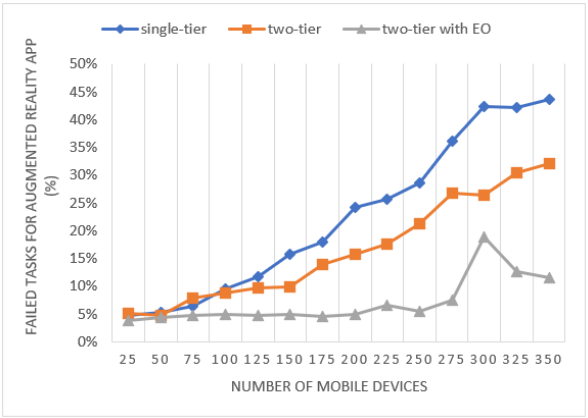
Effect of WLAN bandwidth



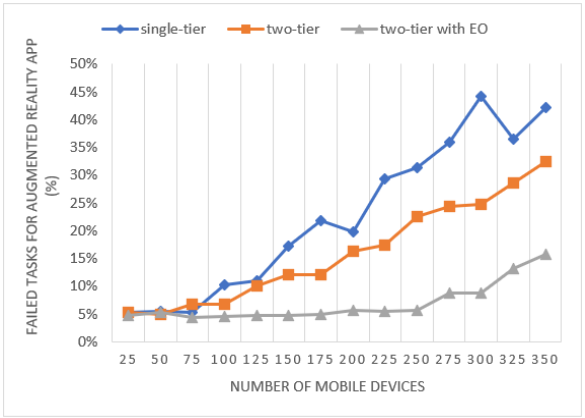
(A) 300 Mbps



(B) 500 Mbps

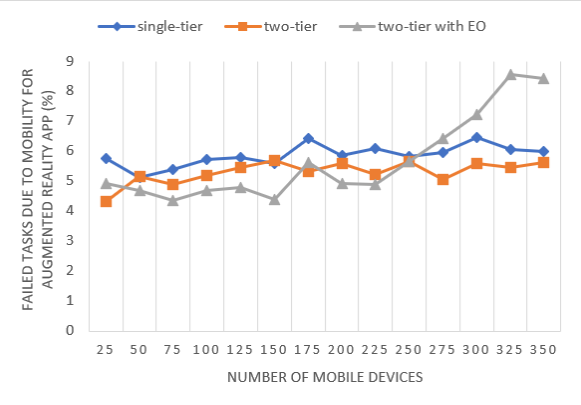


(C) 700 Mbps

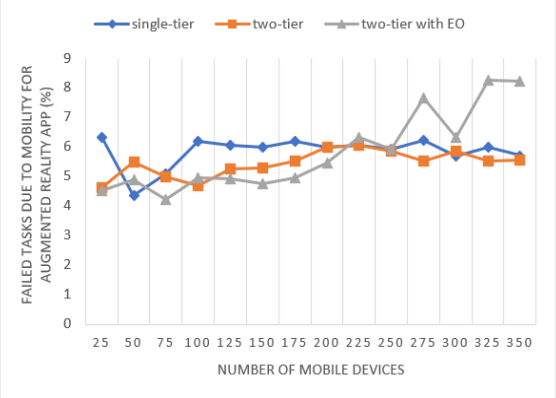


(D) 900 Mbps

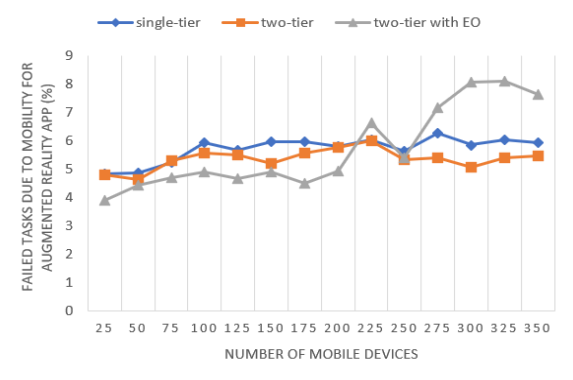
Figure 12. Average percentage of failed tasks – Effect of WLAN bandwidth



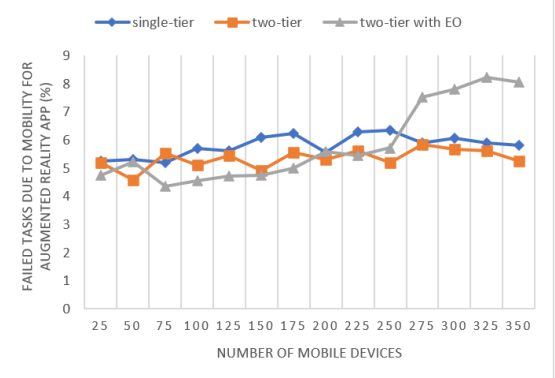
(A) 300 Mbps



(B) 500 Mbps

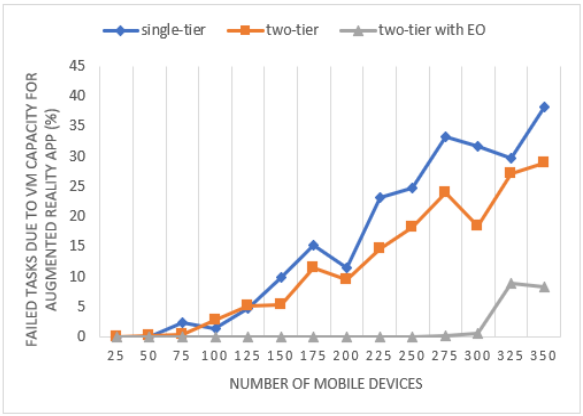


(C) 700 Mbps

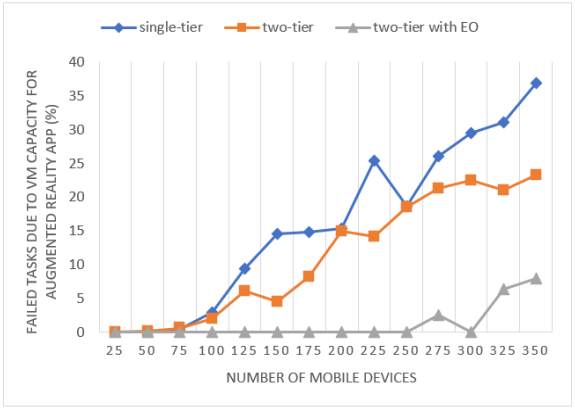


(D) 900 Mbps

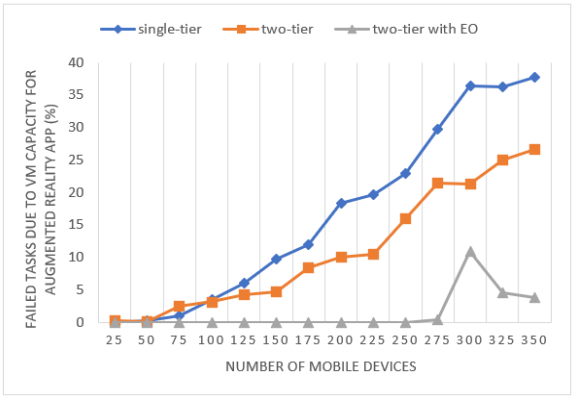
Figure 13. Percentage of failed tasks due to mobility – Effect of WLAN bandwidth



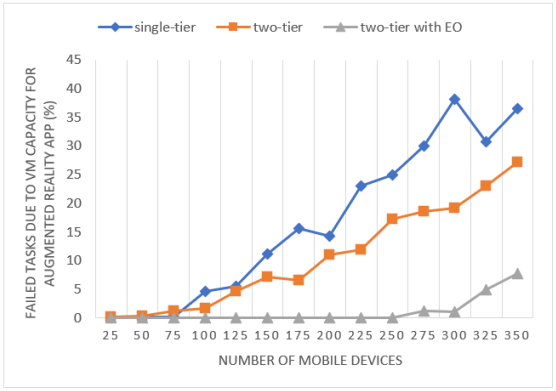
(A) 300 Mbps



(B) 500 Mbps

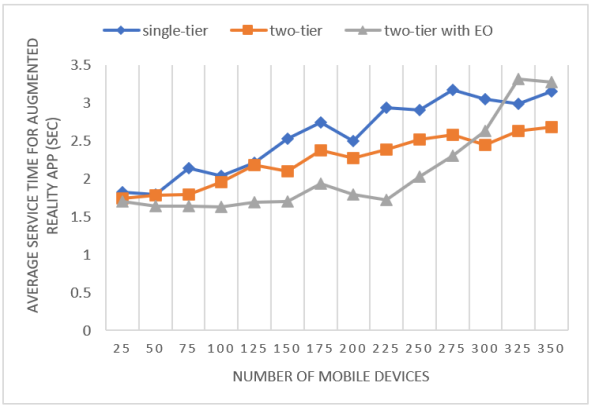


(C) 700 Mbps

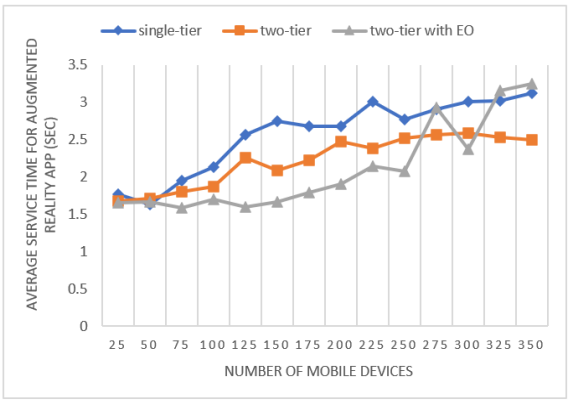


(D) 900 Mbps

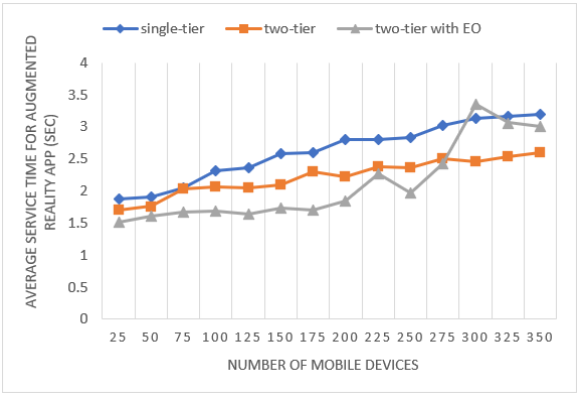
Figure 14. Percentage of failed tasks due to VM capacity – Effect of WLAN bandwidth



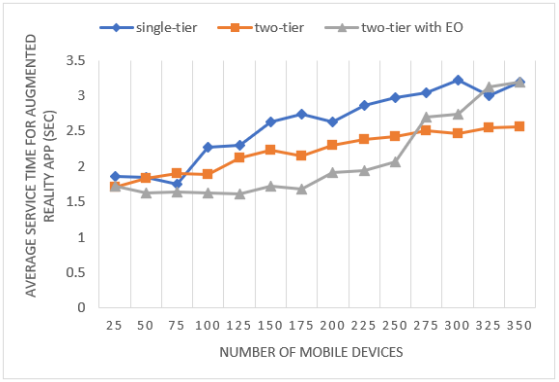
(A) 300 Mbps



(B) 500 Mbps

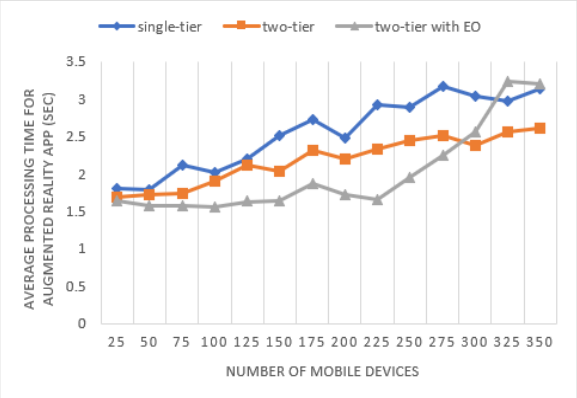


(C) 700 Mbps

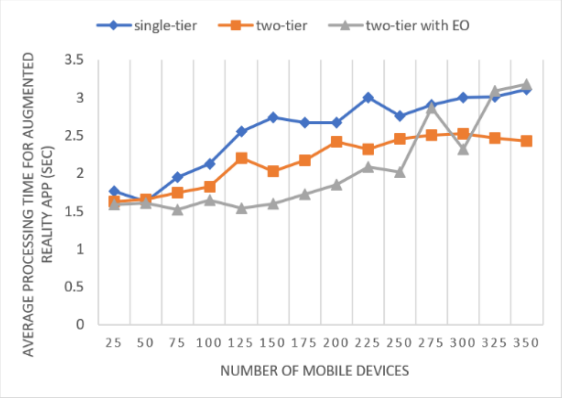


(D) 900 Mbps

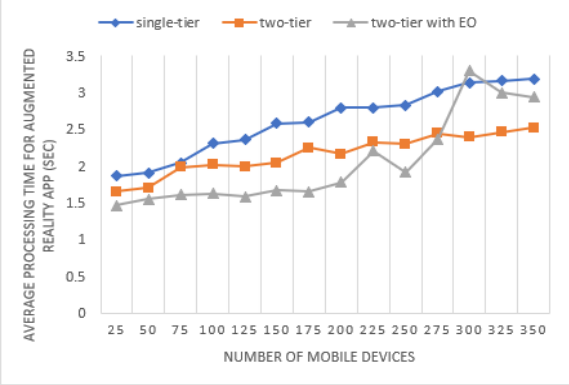
Figure 15. Average service time – Effect of WLAN bandwidth



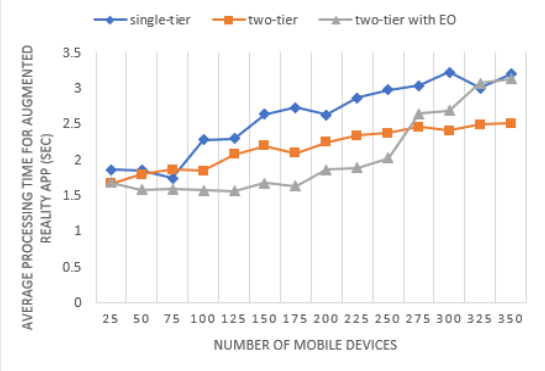
(A) 300 Mbps



(B) 500 Mbps

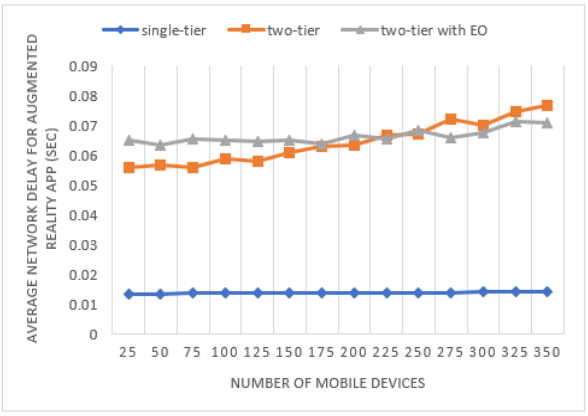


(C) 700 Mbps

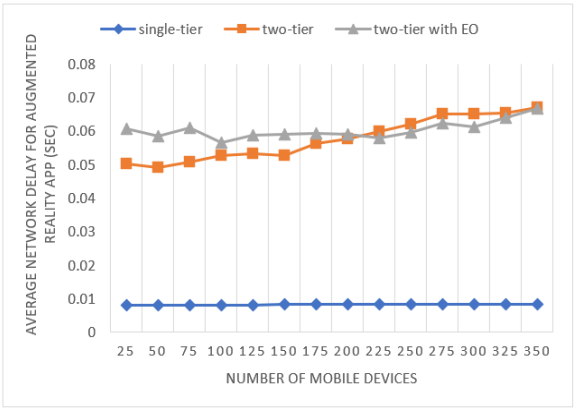


(D) 900 Mbps

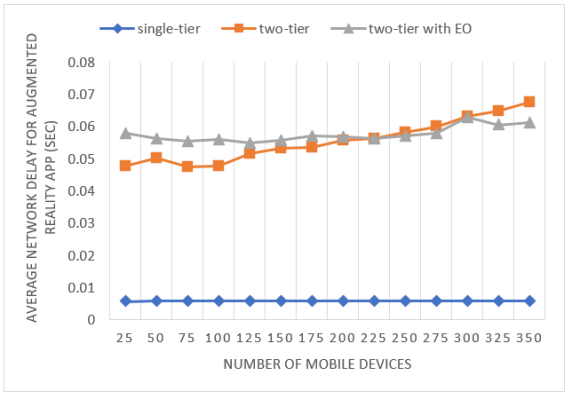
Figure 16. Average processing time – Effect of WLAN bandwidth



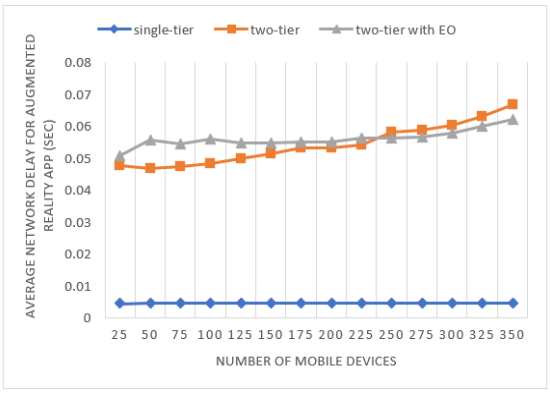
(A) 300 Mbps



(B) 500 Mbps

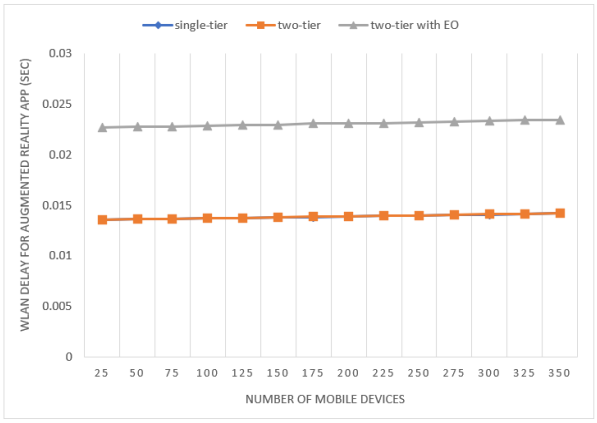


(C) 700 Mbps

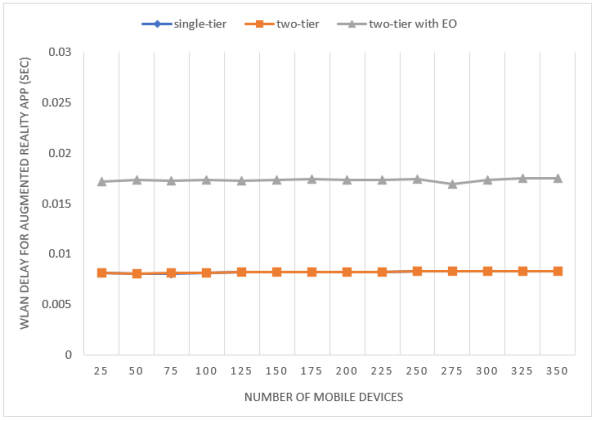


(D) 900 Mbps

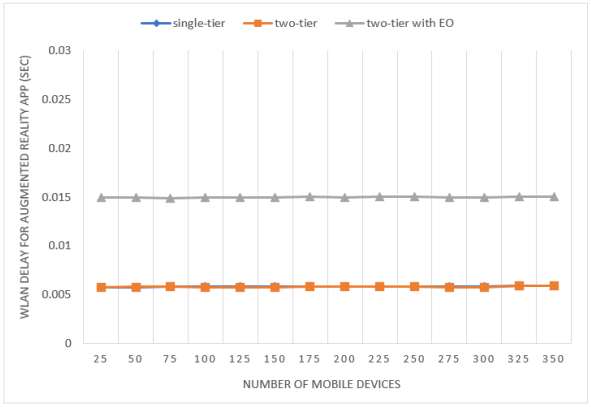
Figure 17. Network delay – Effect of WLAN bandwidth



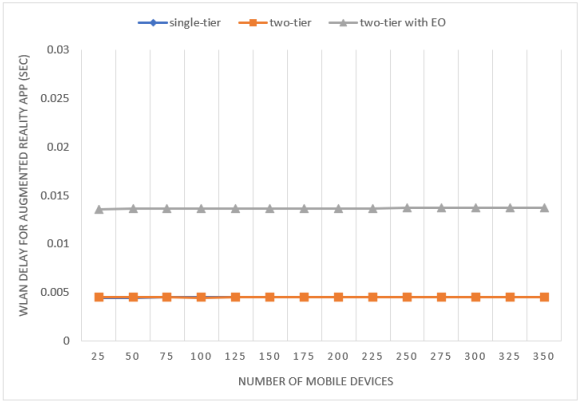
(B) 300 Mbps



(A) 500 Mbps



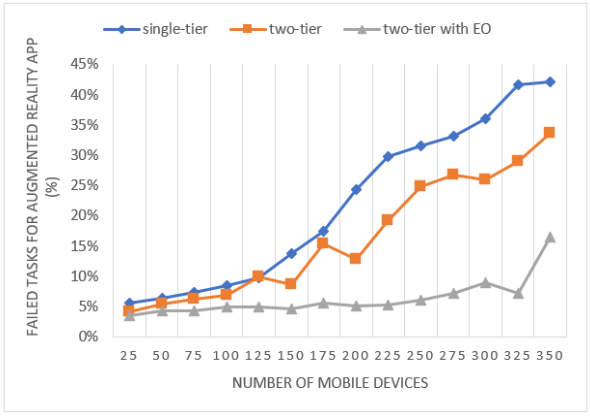
(D) 700 Mbps



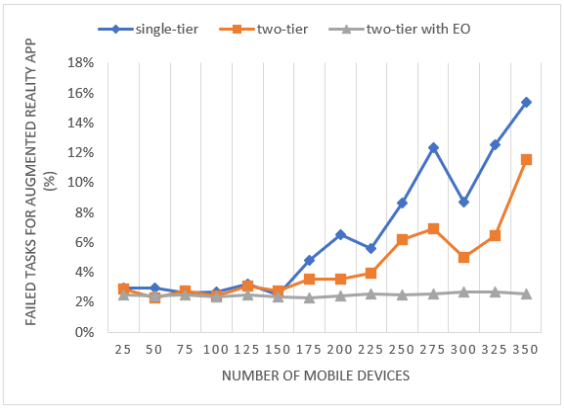
(C) 900 Mbps

Figure 18. WLAN delay – Effect of WLAN bandwidth

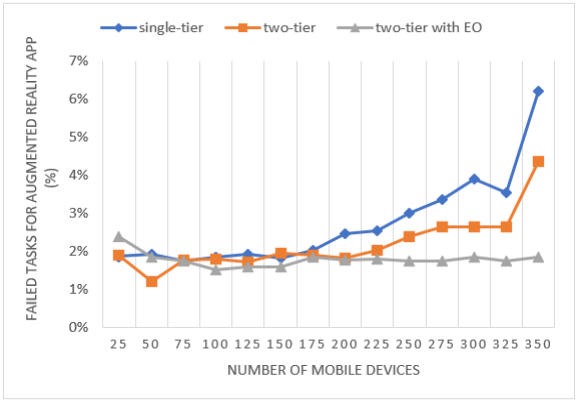
Effect of VM processing speed



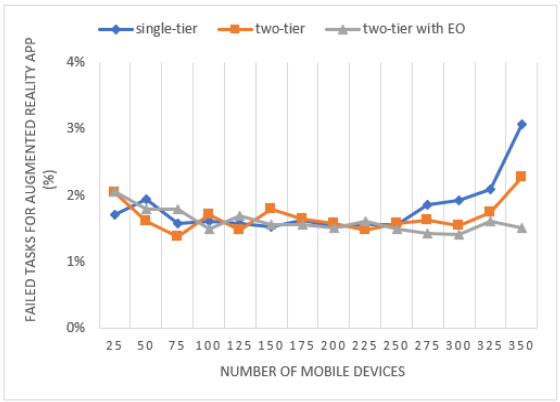
(A) 1000 MIPS



(B) 2000 MIPS

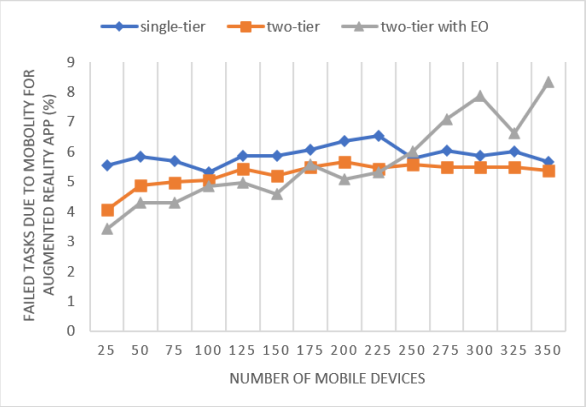


(C) 3000 MIPS

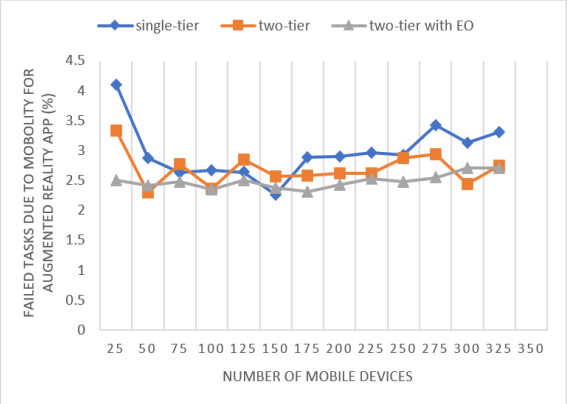


(D) 4000 MIPS

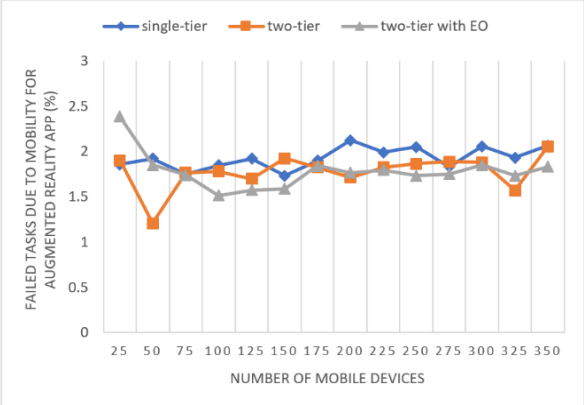
Figure 19. Average percentage of failed tasks– Effect of VM processing speed



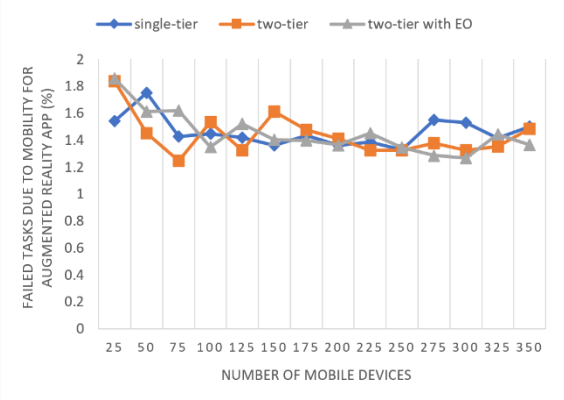
(A) 1000 MIPS



(B) 2000 MIPS

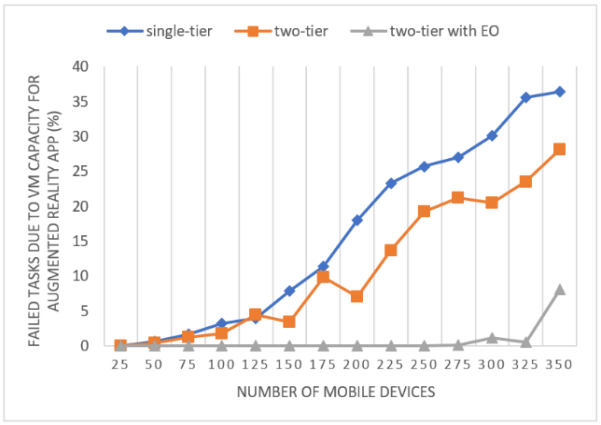


(C) 3000 MIPS

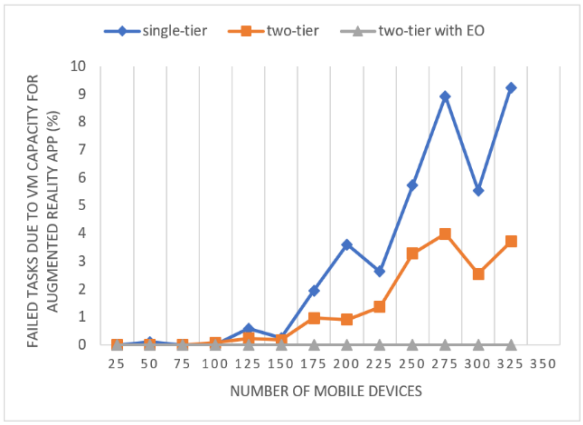


(D) 4000 MIPS

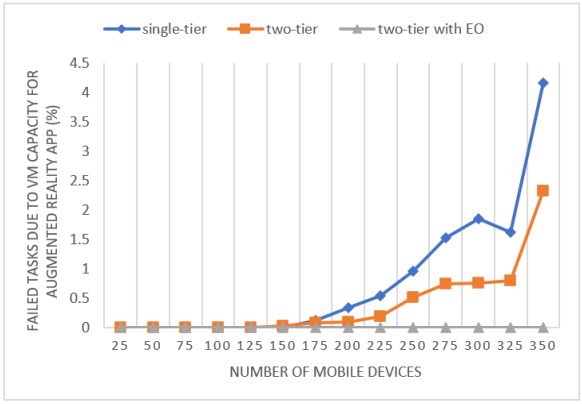
Figure 20. Percentage of failed tasks due to mobility– Effect of VM processing speed



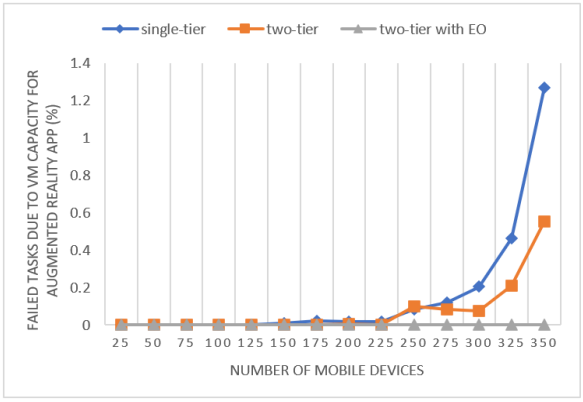
(A) 1000 MIPS



(B) 2000 MIPS

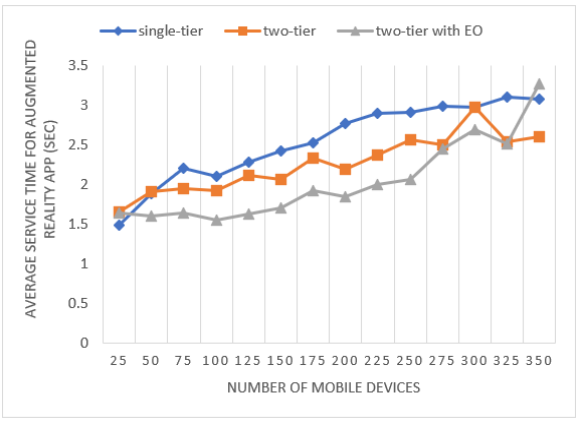


(C) 3000 MIPS

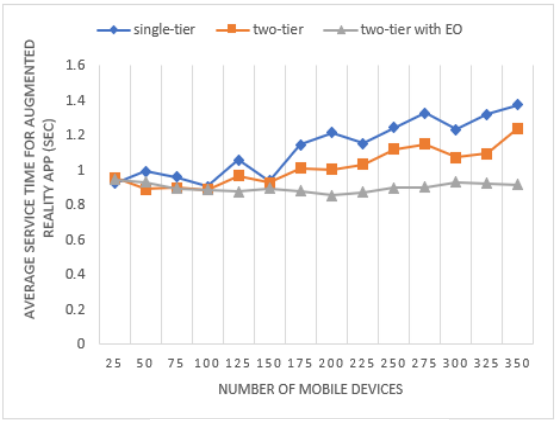


(C) 4000 MIPS

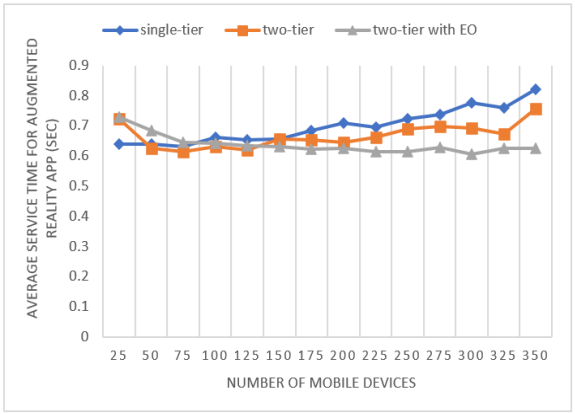
Figure 21. Percentage of failed tasks due to VM capacity– Effect of VM processing speed



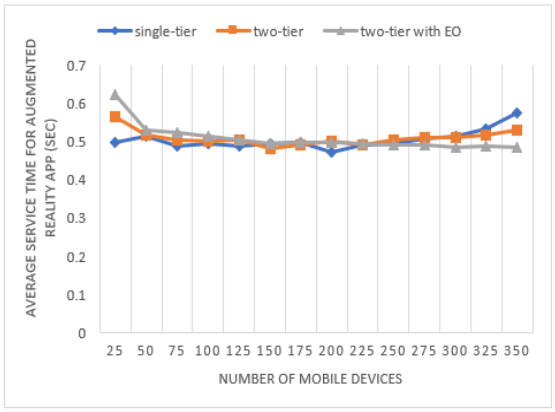
(A) 1000 MIPS



(B) 2000 MIPS

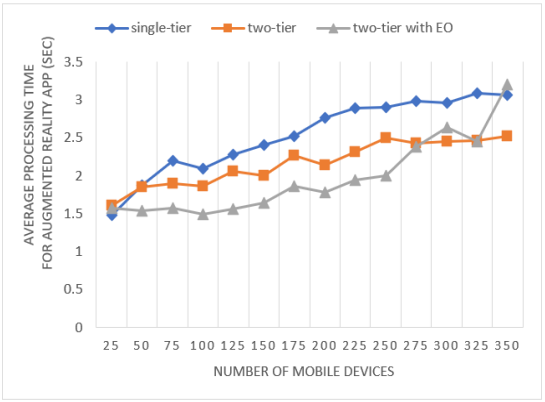


(C) 3000 MIPS

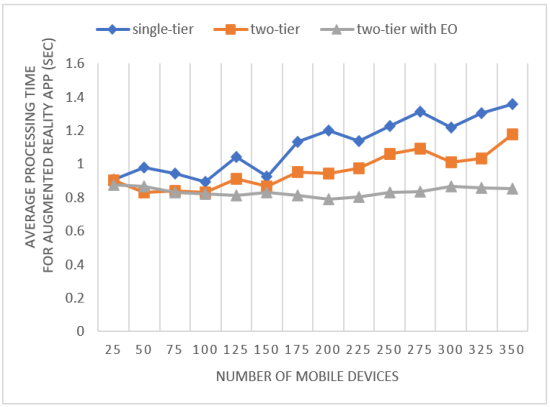


(D) 4000 MIPS

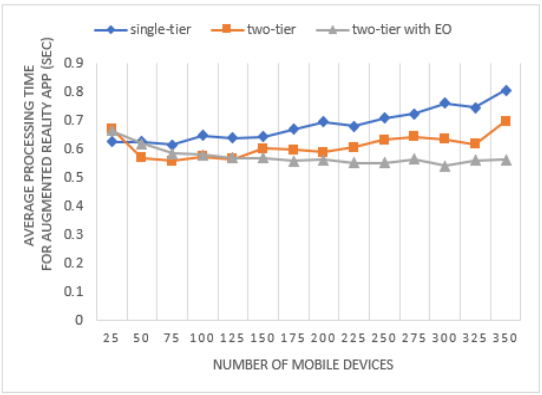
Figure 22. Average service time– Effect of VM processing speed



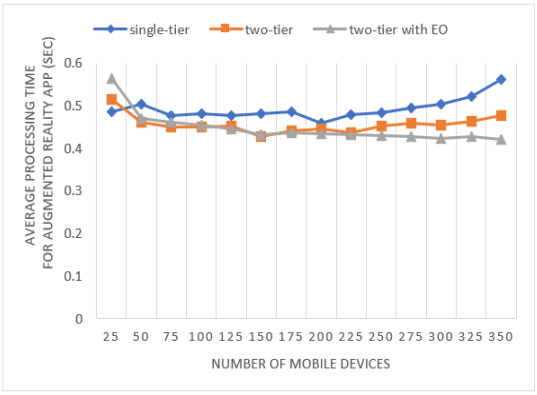
(A) 1000 MIPS



(B) 2000 MIPS

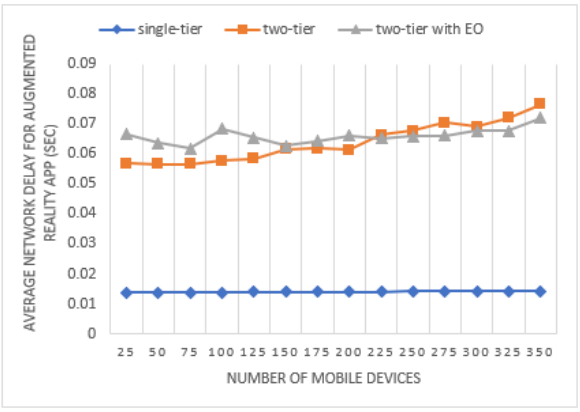


(C) 3000 MIPS

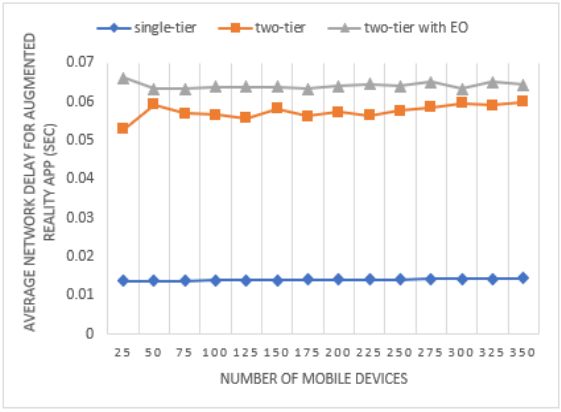


(D) 4000 MIPS

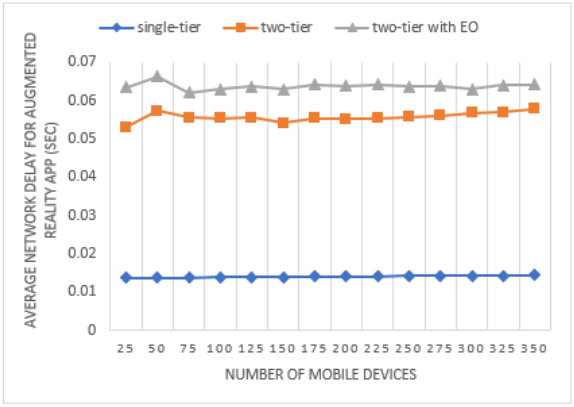
Figure 23. Average processing time– Effect of VM processing speed



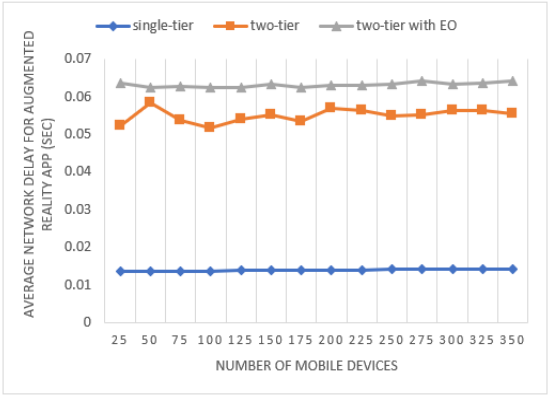
(A)1000 MIPS



(B) 2000 MIPS

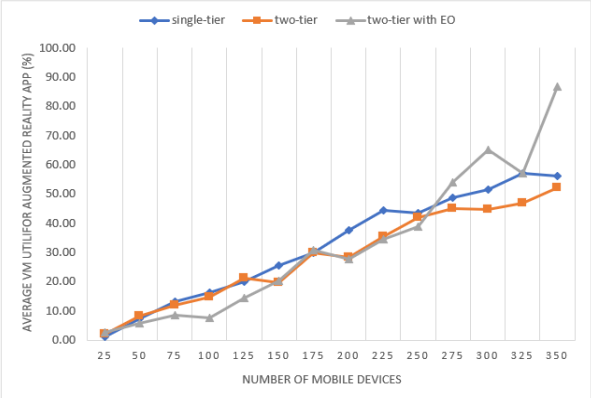


(C) 3000 MIPS

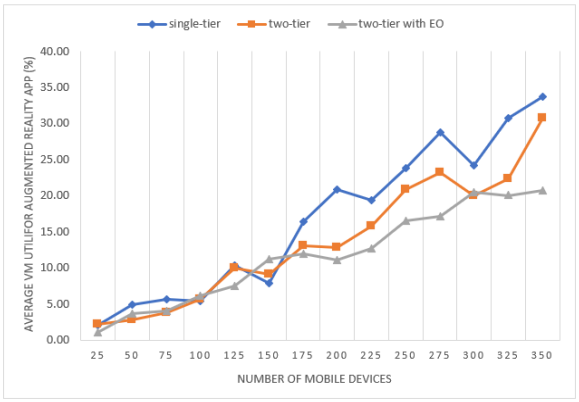


(D) 4000 MIPS

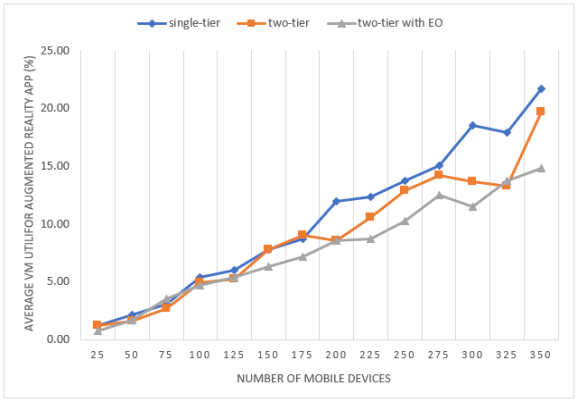
Figure 24. Network delay– Effect of VM processing speed



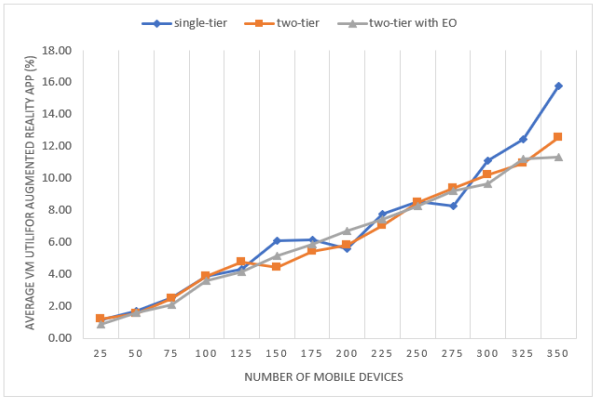
(A) 1000 MIPS



(B) 2000 MIPS



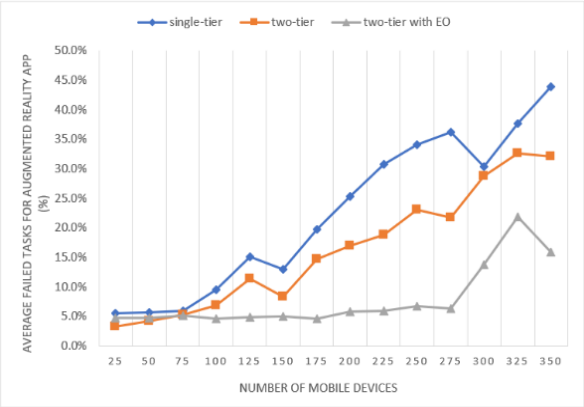
(C) 3000 MIPS



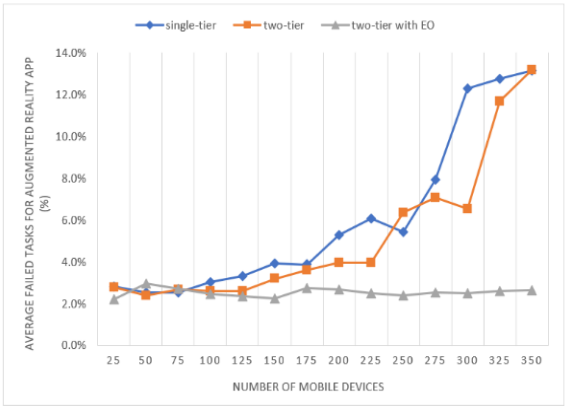
(D) 4000 MIPS

Figure 25. VM utilisation– Effect of VM processing speed

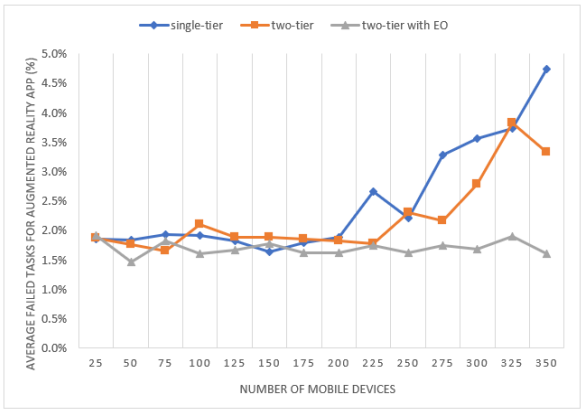
Effect of VM processing speed and WLAN bandwidth



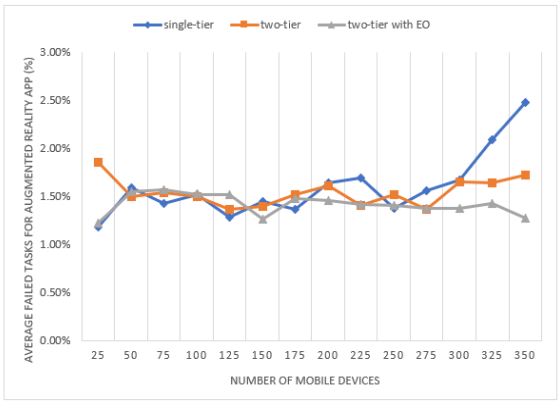
(A) 300 Mbps – 1000 MIPS



(B) 500 Mbps – 2000 MIPS

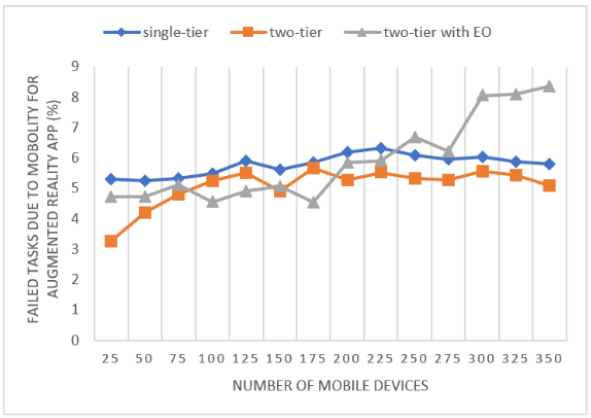


(C) 700 Mbps – 3000 MIPS

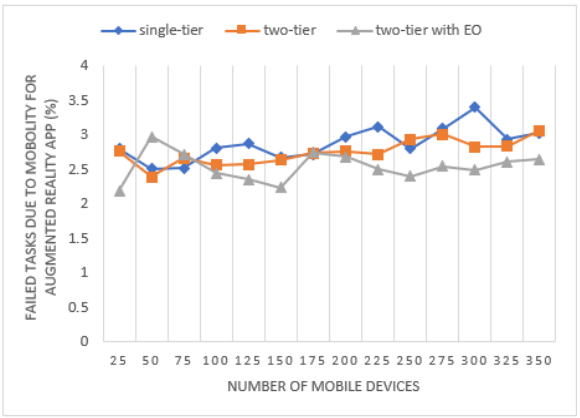


(D) 900 Mbps – 4000 MIPS

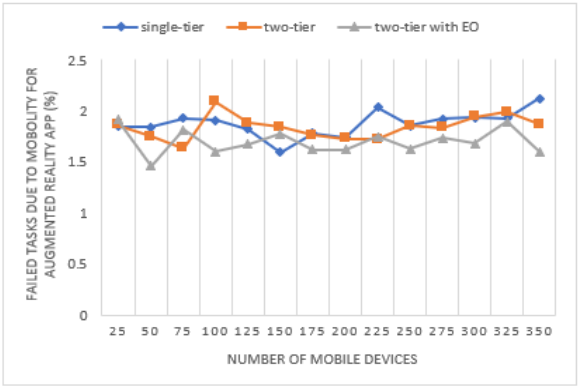
Figure 26. Average percentage of failed tasks – Effect of both VM processing speed and WLAN bandwidth



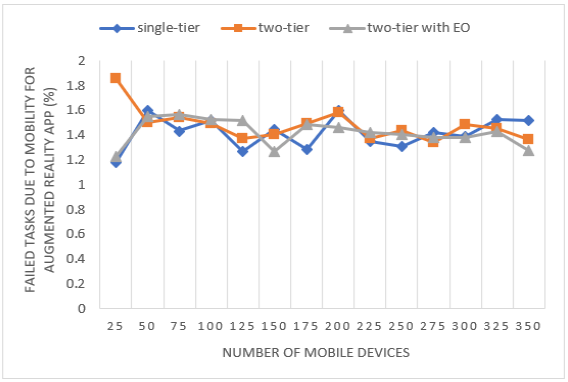
(A) 300 Mbps – 1000 MIPS



(B) 500 Mbps – 2000 MIPS

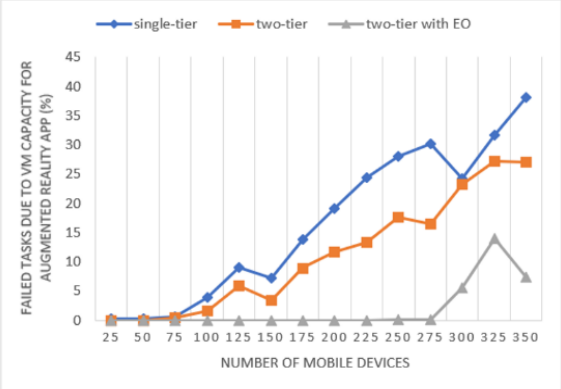


(C) 700 Mbps – 3000 MIPS

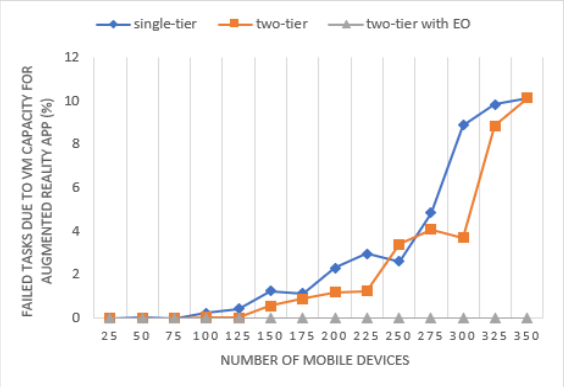


(D) 900 Mbps – 4000 MIPS

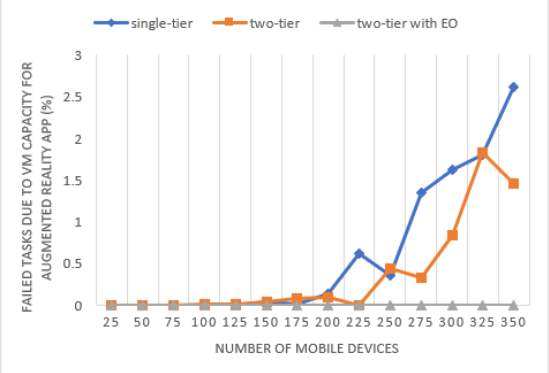
Figure 27. percentage of failed tasks due to mobility– Effect of both VM processing speed and WLAN bandwidth



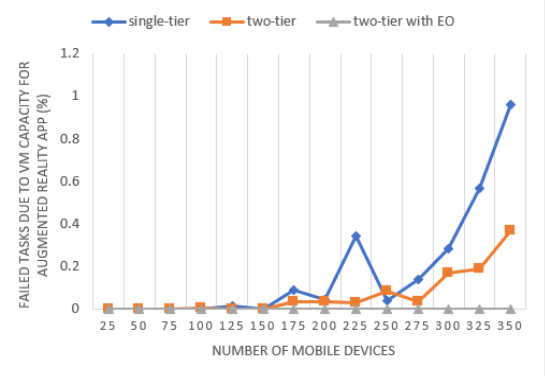
(A) 300 Mbps – 1000 MIPS



(B) 500 Mbps – 2000 MIPS

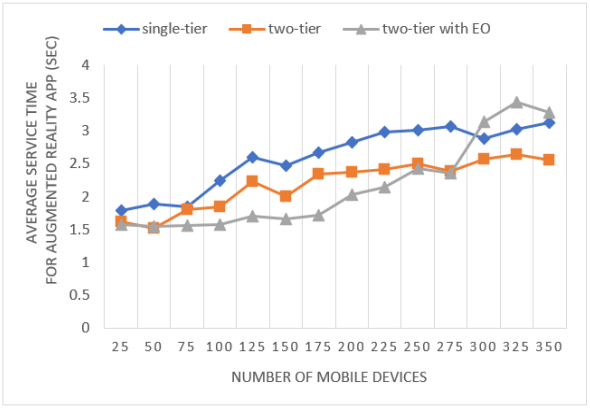


(C) 700 Mbps – 3000 MIPS

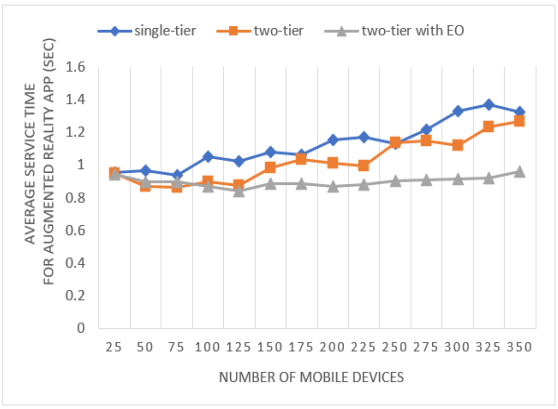


(D) 900 Mbps – 4000 MIPS

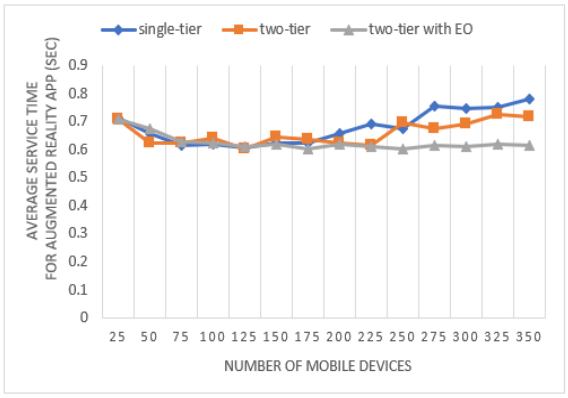
Figure 28. percentage of failed tasks due to VM capacity– Effect of both VM processing speed and WLAN bandwidth



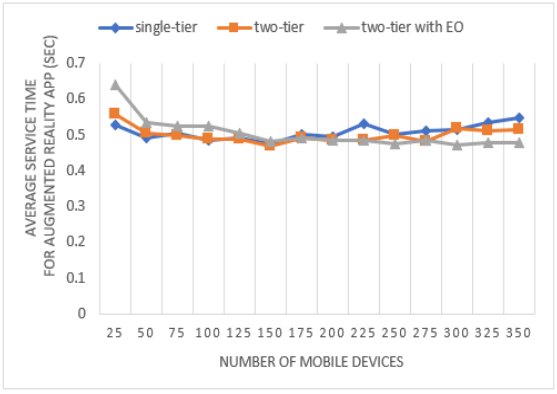
(A) 300 Mbps – 1000 MIPS



(B) 500 Mbps – 2000 MIPS

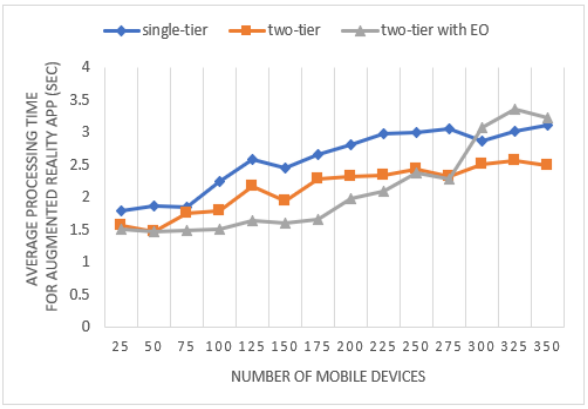


(C) 700 Mbps – 3000 MIPS

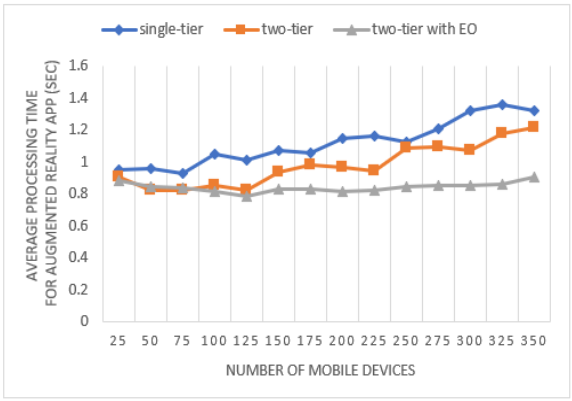


(D) 900 Mbps – 4000 MIPS

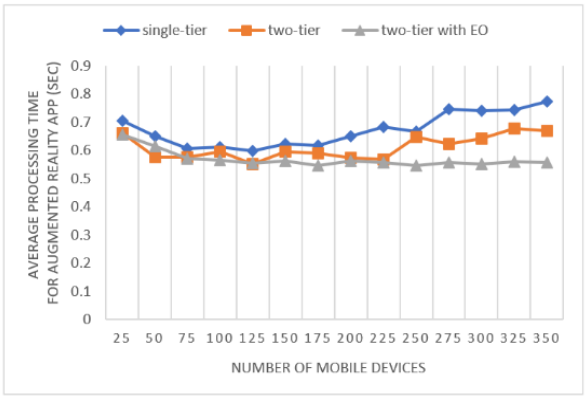
Figure 29. Average service time– Effect of both VM processing speed and WLAN bandwidth



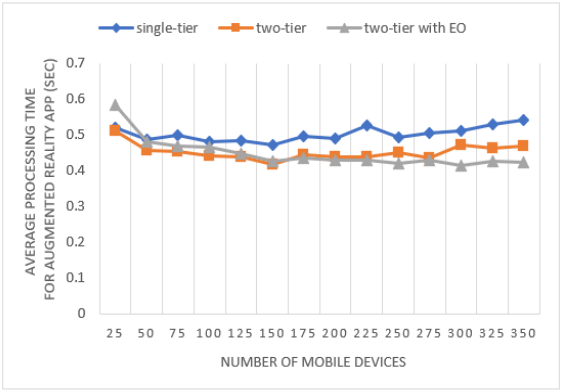
(A) 300 Mbps – 1000 MIPS



(B) 500 Mbps – 2000 MIPS

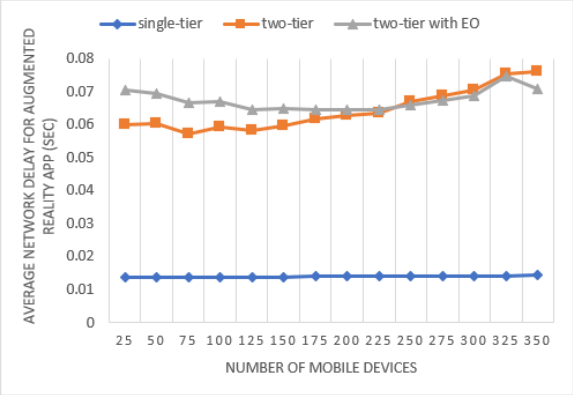


(C) 700 Mbps – 3000 MIPS

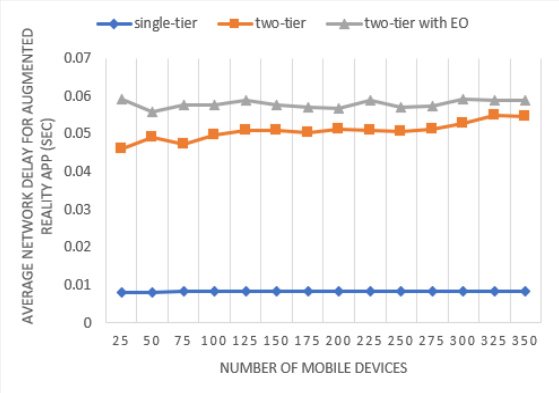


(D) 900 Mbps – 4000 MIPS

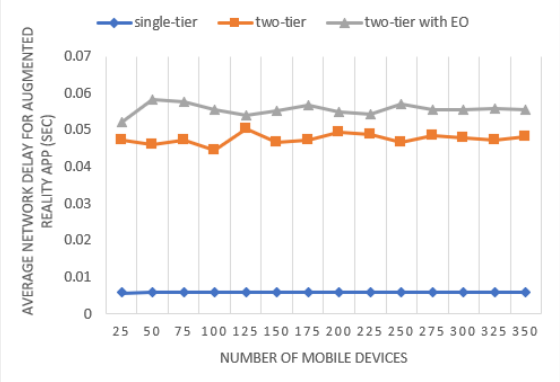
Figure 30. Average processing time– Effect of both VM processing speed and WLAN bandwidth



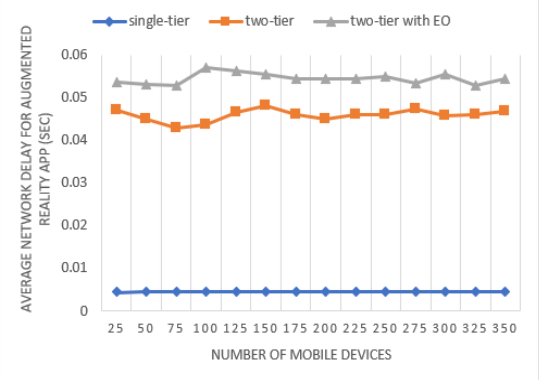
(A) 300 Mbps – 1000 MIPS



(B) 500 Mbps – 2000 MIPS



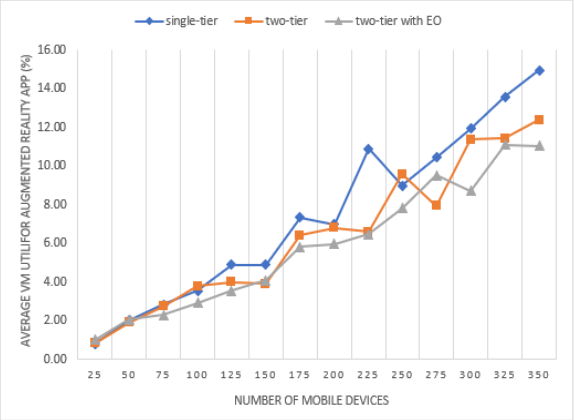
(C) 700 Mbps – 3000 MIPS



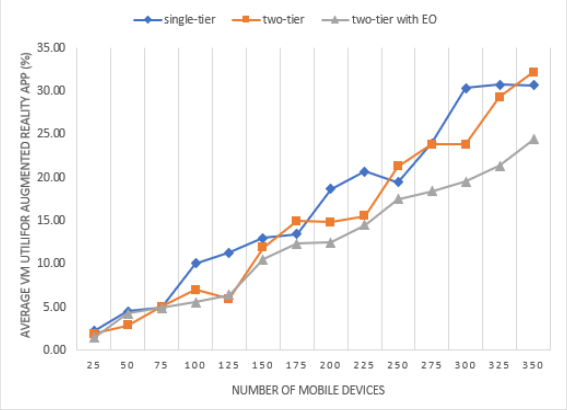
(D) 900 Mbps – 4000 MIPS

Figure 31. Network delay– Effect of both VM processing speed and WLAN bandwidth

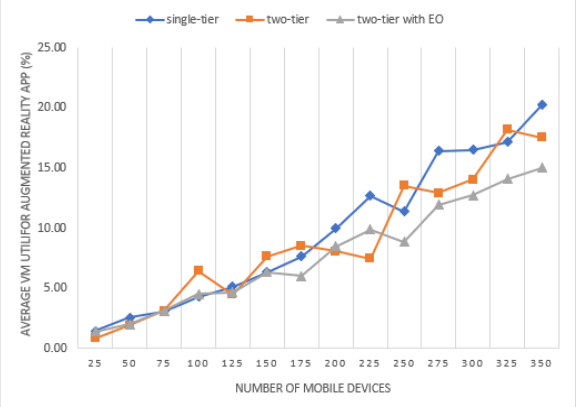
(A) 300 Mbps – 1000 MIPS



(B) 500 Mbps – 2000 MIPS



(C) 700 Mbps – 3000 MIPS



(D) 900 Mbps – 4000 MIPS

Figure 32. VM utilisation– Effect of both VM processing speed and WLAN bandwidth

Appendix B
Project Milestones

Milestone 1: Initial Research and Literature Review - To read and understand relevant literature provided by the supervisor and source more extensive resources to gain in-depth knowledge on edge computing and edge architecture.

Milestone 2: Aims, Objectives, and Methodology - Outline specific, measurable, achievable, relevant, and time-bound (SMART) objectives for the project. By discussing possible solutions, implement a transparent methodology to support achieving the goals of the project.

Milestone 3: Install EdgeCloudSim and Run Sample Experiments - Install and understand the simulation tool and set up the integrated development environment, conduct research on how the simulator operates and the purpose of each file and script.

Milestone 4: Design of Initial Experiment and Implementation - Design and document an initial experiment with simulation parameters representing the Augmented Reality application in the specific scenario. Then, implement the initial experiment for each architecture assessed in this study.

Milestone 5: Analysis and Interpretation of Initial Experiment Results - Analyse and interpret the initial experiment results to determine key parameters for the investigation and clear justification for evaluating selected parameters in relation to scalability and performance.

Milestone 6: Full Experiment Design and Implementation - Conduct experiment design of the system architecture and scenario representing the specific Augmented Reality application. Then, implement and execute simulation experiments using EdgeCloudSim and generate accurate and valid graphical representations of output data using MATLAB graph plotting tools.

Milestone 7: Technical Evaluation - Produce an evaluation and interpretation of complete simulation experiments. Using the graphical representation of results, deeply analyse results to determine explanation of findings. Interpret results within the context of the specific application to test and verify hypotheses.

Milestone 8: Project Evaluation - Evaluate the project's success and ascertain how well project aims and objectives are met. Evaluate the methodology and project management process and justify limitations of the project whilst recommending improvements and proposals for future investigation.

Milestone 9: Write up Final Report - Write up the final report and complete documentation of deliverables ready for submission.

Appendix C

Project Risk Assessment

Table 6. *Risk assessment strategy with risk ratings*

Risk Number	Risk Description	Effect of Risk	Severity (1-5)	Probability (1-5)	Risk score (1-25)	Risk Mitigation
1	Requirement for additional resources to gain the required depth of understanding on the subject of edge computing	Unable to find relevant resources and expert studies may effect the quality of the final solution	4	1	4	Gain relevant literature for review from my supervisor and present additional resources that may be relevant during meetings for wider understanding
2	Mitigating circumstances and unforseen health issues	Set backs in meeting milestones and delivery dates	4	2	8	Notify school and supervisor of any change of circumstances with evidence in order to revise project plan and milestone dates
3	Unable to gain a clear understanding of key concepts or technical terms	Incorrect use of technical terminology or misinterpretation of advanced concepts	3	1	3	Obtain expert insight from supervisor during regular progress meetings. Make notes or record voice during meetings, if permission given, to review discussions when needed
4	Difficulties understanding how the simulator represents the specific application	Unable to produce accurate and valid results	5	1	5	Ask my supervisor for help with scripts and understanding what each parameter represents regarding the specific application
5	Technical difficulties running the simulation experiment and recieving bugs and errors	Unable to produce accurate and valid results	5	2	10	Search Gitlab repository of EdgeCloudSim for previously reported bugs, access EdgeCloudSim developer forums and videos for help, ask supervisor for support
6	Corruption of scripts or simulation data including results graphs	Loss of important data for evaluation may reduce quality of the outcome	5	1	5	Create a backup using an external hardrive, cloud storage, and create a GitLab repository to ensure version control where code can be easily reverted back incase of unexpected errors
7	Management of time and accurate estimation of time required for each task	Delay in submitting project deliverables	5	3	15	Estimate time required using Gantt Chart, especially for implementation of experiments as this is the most complex milestone. Ask for guidance when troubleshooting code in order to print valid graphs

Key: Low Risk | Moderate risk | High Risk

Appendix D
Addressing Legal, Ethical, Social and Professional Concerns

Developing an understanding of ethical principles and abiding by the responsibilities of a computer scientist ensures a framework of principles to discern what is right and wrong.

Legal Issues. Ethics involves abiding by the law and ensuring the source code of licenced works are used fairly and legally. EdgeCloudSim is licensed under the GNU General Public License v3.0, which guarantees that it is legal to copy, change and share all versions of the software source code for free. Furthermore, the licence states that no warranty or liability for developers exists; therefore, the modified version is marked as edited as required legally. Finally, the license acknowledges fair use rights as provided by copyright law, so all repositories used for this investigation have a copy of this exact licence saved. Therefore, future developers have the same legal freedoms to run, edit, and test simulation scrips

Ethical Issues. Simulation experiments for this project do not utilise external data, and all data is generated through simulation, meaning consent is not required. Privacy and anonymity are unrelated concerns as no data used or generated can identify a person or constitute direct or indirect personal data. All ethical protocol was supported as no images of people, medical images, religious text were used. Furthermore, user testing was not required for meeting objectives as all data used for this project entails simulation data. Data used to back up findings have come from technical studies where the authors have been given full credit and reference.

Social Issues. This project purpose was to investigate edge architecture where exploration through simulation was practised. Simulations were run efficiently and often in parallel, ensuring less resource use. This project considered performance and scalability, but Performance and energy efficiency trade-offs must be considered when deploying edge architectures worldwide. In addition, considering the environmental cost and ecological impact on future generations is essential.

Professional Issues. Throughout the project, the obligation of computer scientists to act responsibly and consider broader impacts of work was endorsed. In addition, transparency and full disclosure of simulation capabilities, limitations and potential problems was provided. The project emphasises producing quality work whilst accepting professional reviews when

suitable. All computing resources were accessed with authorisation and with the belief of no harm as required by the ACM Code of Ethics and Professional Conduct [58].

Appendix E

External Materials

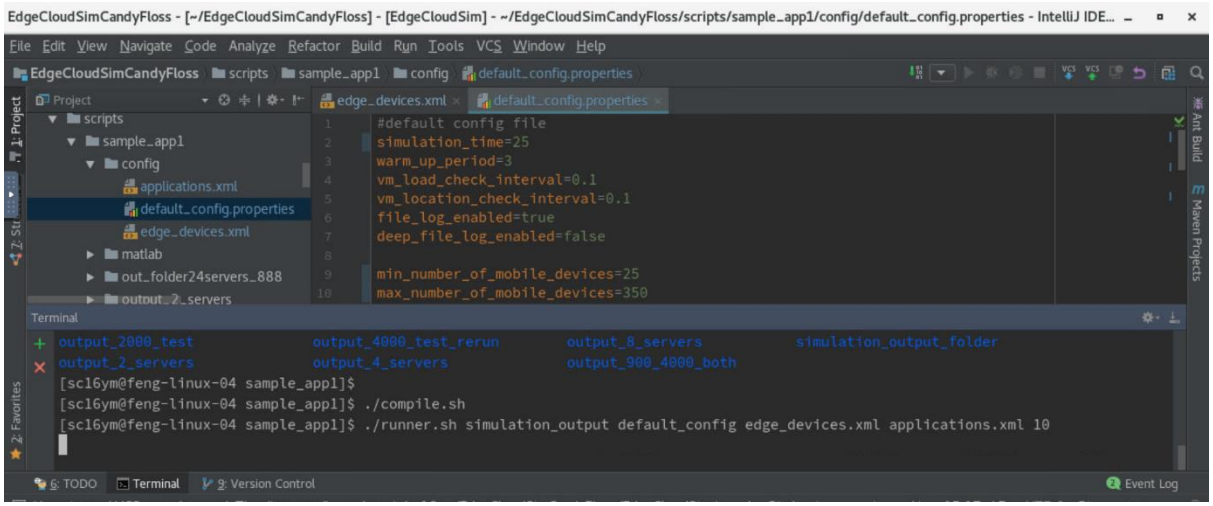


Figure 33. Terminal displaying commands to compile the simulation environment and run a simulation based on the default configuration file, edge devices file, application configuration, and the number of iterations.

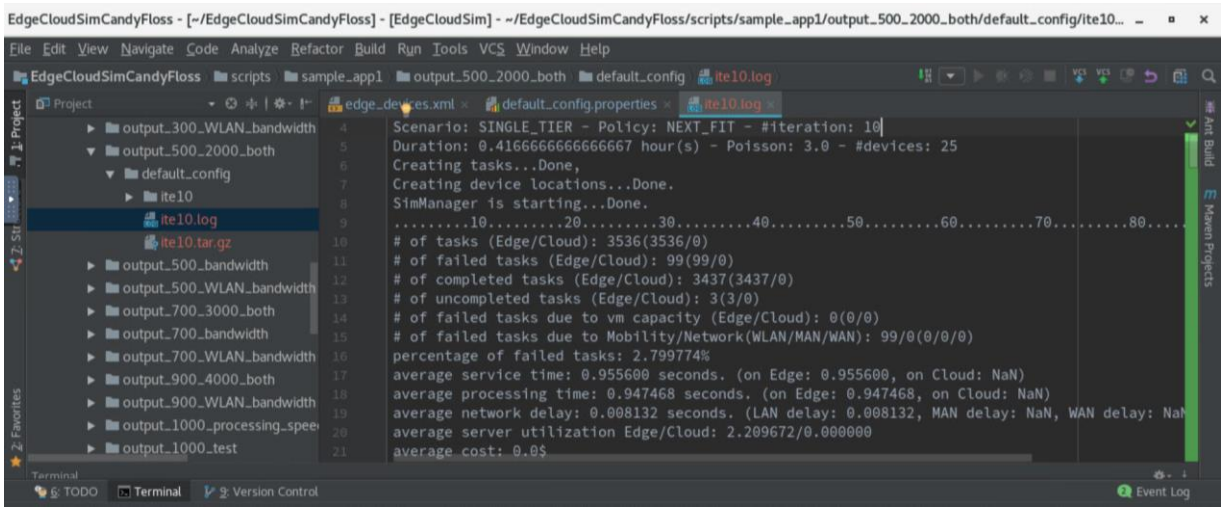


Figure 34. Example of the output of simulation results in the form of the ite10.log file.