

School of Computing
FACULTY OF ENGINEERING



Edge Architectures Simulation

Chae Rim Kim

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

2019/2020

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) Chaeim

Summary

The rapid growth of the Internet of Things (IoT) has brought edge computing paradigm under the spotlight. Edge computing brings computing resources closer to end-devices, to meet the increasing needs of performance requirements. This report aims to investigate the effect of computational and networking system parameters on the performance of a face recognition application. The project will evaluate the performance of single-tier, two-tier, and two-tier with Edge Orchestrator (EO) architecture through an empirical investigation, using EdgeCloudSim. The results showed that the best performance could be achieved by utilising two-tier with EO architecture, with high VM processing speed and numerous edge servers. Based on the investigation of the parameters, recommendations for future application design and deployment are made, which will help to design a scalable and effective application.

Acknowledgements

I would like to express my gratitude to Professor Karim Djemame for the valuable support, guidance, and encouragement he has given me throughout the project. It has been a great help in the development and management of the project.

I would also like to thank my project assessor, Professor Jie Xu for his useful and constructive recommendations during the progress meeting.

List of Acronyms

Abbreviation	Term	Definition
AR	Augmented Reality	Computer-generated content that is overlaid on a real-world environment
AWS	Amazon Web Service	Amazon's cloud computing service
CSV	Comma Separated Value	A delimited plain text file that stores data in a tabular format
EO	Edge Orchestrator	A module that decides how and where to handle incoming client requests in edge computing
GB	Gigabytes	A unit of information used
GHz	Gigahertz	A unit to measure clock frequency of CPU
IoT	Internet of Things	Devices that are interconnected through the internet, collecting and sharing data
KB	Kilobyte	A unit of digital information
Mbps	Megabits per second	A measure of internet bandwidth
MIPS	Million Instructions Per Second	A measure of the execution speed of the computer
RAM	Random Access Memory	Computer's short-term memory that can be accessed randomly
VM	Virtual Machine	Emulation of a physical computer, running its own applications and an operating system
VR	Virtual Reality	A three-dimensional computer-generated environment that can be interacted by a person
WAN	Wide Area Network	A network that extends over a large area, interconnecting multiple local area networks
WLAN	Wireless Local Area Network	A network that allows devices to connect and communicate wirelessly via Wi-Fi

Table of Contents

Summary	iii
Acknowledgements	iv
List of Acronyms	v
Table of Contents	vi
1. Introduction	1
1.1 Project Background	1
1.2 Problem Statement	1
1.3 Possible Solution	2
1.4 How to demonstrate the quality of solution	2
1.5 Aim	2
1.6 Objectives	2
1.7 Deliverables	3
1.8 Project Management.....	3
1.8.1 Methodology	3
1.8.2 Tasks, Milestones and Timeline	4
1.8.3 Risk Assessment	5
2. Background Research	6
2.1 Cloud Computing	6
2.2 Internet of Things (IoT)	6
2.2.1 Internet of Things.....	6
2.2.2 IoT and Edge Computing.....	7
2.3 Edge Computing	7
2.3.1 Edge Computing Architectures.....	7
2.3.2 Advantages of Edge Computing.....	8
2.4 Edge Computing Simulator.....	9
3. Literature review	11
3.1 Single-tier	11
3.2 Two-tier	13
3.3 Two-tier with EO	15
4. Design	17
4.1 Hypotheses	17
4.2 Experimental Design.....	17

4.2.1 Experiment's Assumptions	17
4.2.2 Initial Experiment	18
4.2.3 Experiment Design	18
4.3 Performance Metrics.....	19
5. Implementation	20
5.1 Implementation Details	20
5.2 Evaluation Metrics	20
5.3 Result Visualisation	20
6. Technical Evaluation	21
6.1 Initial Experiment	21
6.2 Experiment Results.....	24
6.2.1 Effect of WLAN bandwidth.....	24
6.2.2 Effect of VM processing speed.....	25
6.2.3 Effect of edge servers.....	27
6.2.4 Effect of VM capacity	29
6.3 Findings.....	31
6.3.1 Findings Summary.....	31
6.3.2 Hypotheses Verification	32
6.3.3 Significant Parameters	33
6.4 Recommendations	33
7. Project evaluation	35
7.1 Evaluation	35
7.1.1 Aims and Objectives	35
7.1.2 Project Management.....	36
7.1.3 Limitations	36
7.2 Related work	37
7.2.1 Prior Work	37
7.2.2 Future work	37
7.3 Personal Reflection.....	38
7.4 Legal, Ethical, Social and Professional Issues	39

8. Conclusion.....	40
List of References.....	41
Appendix A. External Material.....	43
Appendix B. Literature Review – Summary.....	44
Appendix C. Initial Gantt Chart	45
Appendix D. Full Simulation Results.....	46

1. Introduction

1.1 Project Background

With the Internet of Things (IoT), an enormous amount of data gets generated from end-user devices including mobile phones, wearable devices, sensors and vehicles every second [1]. In order for these large volumes of data to provide meaningful insight, data processing is necessary, which may require larger computational resources than the device's capabilities. The emergence of cloud computing addressed the issue of limited computational and energy resources of devices with its virtualised resources and dynamically reconfigurable nature [2]. However, cloud computing comes with the cost of high latency and limited bandwidth issues, which could be a serious bottleneck for applications requiring real-time data processing and responses [3].

Edge computing allows for data processing to be done at the network edge, closer to the data source, resulting in shorter response time and more efficient processing [3]. It places computational resources at the logical extremes of a network, with the ability to store, cache, process and load balance the data to be sent to the cloud.

Edge architecture is being utilized in many different fields where the latency of data processing is a crucial factor of the system, such as in Autonomous Vehicles, Augmented reality (AR) / Virtual Reality (VR), and facial recognition for security, where the processing of large data is a prominent feature of the application [4]. Edge computing does not necessarily replace cloud computing but rather complements it by offloading tasks to servers in close proximity, rather than a distant cloud datacentre [4].

1.2 Problem Statement

Fundamentally, edge computing allows us to bring computing resources closer to end-devices, hence providing a better quality of service [2]. This is directly related to application performance; hence it is important to investigate the effect of different computational and networking system parameters on the performance results. An empirical investigation of the evaluation of different edge architectures will be carried out using the EdgeCloudSim simulator, which addresses the particular demands of edge computing [5].

1.3 Possible Solution

Different edge architectures deployed in different domains can be simulated by altering the parameters and metrics of the EdgeCloudSim simulator. The simulator provides a design space with configurable parameters to allow for testing various architectures of edge computing. This provides an insight regarding the connection between the computational parameters and the performance results [5].

Parameters will be chosen by first conducting an initial experiment to test and determine the key variables. The project will further be conducted by designing experiments, implementing and collecting data, followed by results interpretation [6]. The results will be used to further imply and recommend future directions of the performance of edge computing applications.

1.4 How to demonstrate the quality of solution

The quality of the solution could be demonstrated by incorporating a thorough and systematic methodology. By outlining how the research is undertaken, the project stays focused, helping to deliver a precise and accurate solution to the research question.

The validity of the simulation results can be confirmed with existing works of literature and experiments. One can compare and contrast the overall results with the literature's findings to see where the solution lies, to gain some extent of confidence in the solution. Furthermore, novel experiments could be designed by choosing parameters and metrics that have not been researched yet, thereby bringing the scope of the solution forward.

1.5 Aim

The aim of the project is to investigate the effect of computational and networking system parameters on the application performance results, through EdgeCloudSim simulator. It will evaluate the performance of different architectures, including single-tier, two-tier and two-tier with Edge Orchestrator (EO).

1.6 Objectives

The objectives of the project are as follows:

1. Identify the key issues in relation to edge computing performance
2. Design and implement different experiments through the configuration of key parameters
3. Analyse and interpret the performance results
4. Propose further recommendations on the architecture of edge computing applications

1.7 Deliverables

The project delivers a report including:

1. **Background research** – outlines relevant areas for the investigation and critically appraises literature that addresses similar problems.
2. **Simulation experiment design** – proposes the details of the simulation.
3. **Experiment implementation** – details the execution of the experiment design, outlining the tools and techniques used for the implementation. This will be in the form of software and scripts.
4. **Technical evaluation** – entails the analysis and interpretation of the simulation results, verifying the hypotheses of the experiment.
5. **Project evaluation** – presents a discussion regarding the success of the project, self-evaluation and future works of the research.

1.8 Project Management

1.8.1 Methodology

The project utilises simulation and is comprised of three sequential phases. Simulation offers the possibility to investigate novel and complex systems with limited resources in the laboratory [7]. It provides empirical results for specific scenarios [8] in a controlled and repeatable environment [9], which is suitable for subjects like edge computing with no standardized architecture [10]. The project adopts an agile approach, where several iterations of each step are performed. The iterative process allows for the elimination of uncertainties arising from the experiment, ensuring the accuracy and integrity of the results.

1. Experimental design

Following a thorough literature review, design a suite of systematic experiments in advance to clearly define the focus of the research; states the parameters and evaluation metrics to be explored.

2. Implementation and data collection

Execution of the experiment design using EdgeCloudSim to collect a set of data, which is to be translated into meaningful results by generating visual plots that illustrate the relationship between the variables chosen. It details the tools and methods used for data collection.

3. Analysis and interpretation

Examining and moulding the data collected, producing a result set describing the general trend of the experiments. It also identifies the significance and implications of the analysis and discovers linkages to the objectives of the investigation.

1.8.2 Tasks, Milestones and Timeline

A Gantt chart in Figure 1 outlines the tasks and milestones of the project. Christmas break and exam period have been omitted as it is less likely that a substantial amount of work could be done over the period.

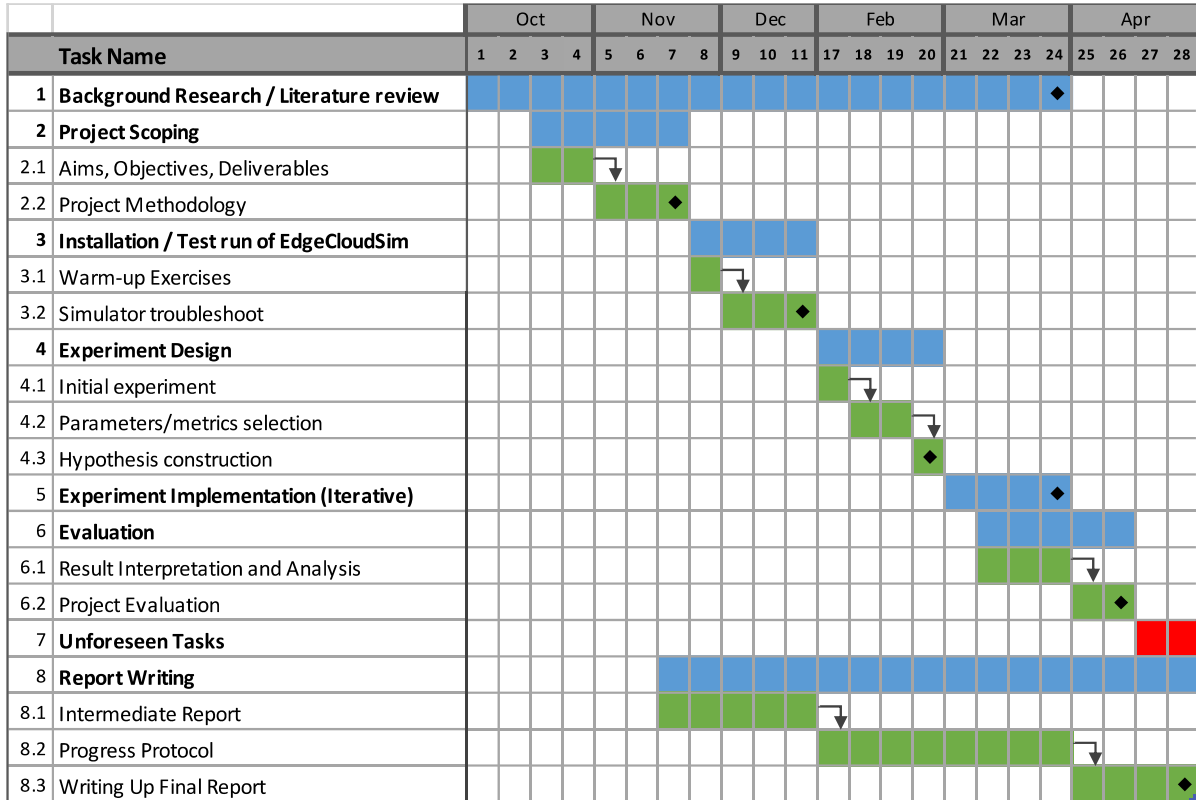


Figure 1. Project Gantt chart

1. Background Research / Literature review

To conduct research and develop a critical analysis of relevant literature essential for the understanding of the project scope, identifying new ways of interpreting the topic and to validate experimental results. A continuous task throughout the project.

2. Project scoping

Develop a scoping document outlining and planning the project; includes problem context, aims, objectives, deliverables, project plan, methodology and a preliminary literature review.

3. Installation / Test run of EdgeCloudSim

Install and run the default configuration of EdgeCloudSim for better understanding of the possible solution and experiment design.

4. Experiment Design

(Milestone 1) Conduct an initial experiment to identify key parameters.

(Milestone 2) Design the system architecture to be tested based on the initial experiment – constructing hypotheses, selecting computational parameters and metrics of the experiment.

5. Experiment Implementation

(Milestone 3) Implement and run experiments on EdgeCloudSim, then generate plots through MATLAB.

6. Evaluation

The evaluation is composed of two parts – technical and project evaluation.

(Milestone 4) Conduct a technical evaluation that interprets and analyses the experiment results for hypotheses verification.

Carry out a project evaluation that reflects on project management and objectives.

7. Unforeseen Tasks

Extra time used to handle unexpected tasks or experiments.

8. Report Writing

Writing up the report – intermediate and final report, including a complete literature review and detailed documentation of the experiment implementation and analysis.

1.8.3 Risk Assessment

Table 1. Risk assessment and mitigation

Possible Risk	Outcome	Likelihood (1-5)	Impact (1-5)	Risk (1-25)	Risk mitigation
Software unavailable from GitHub	Unable to run the simulation and generate output	1	5	5	Keep a personal copy of the original simulator
Technical issues with the simulator	Unable to generate simulation results	2	4	8	Seek support from simulator maintainer and supervisor
Loss of simulation results	Loss of empirical data to be studied	1	4	4	Backup the simulation results in a cloud storage
Limited literature on the topic	Less resource to compare the project findings to; limitation in confirming the result validity	1	3	3	Perform a more thorough analysis of literature and seek help from supervisor or experts in the field
Absence of the supervisor	Unable to receive feedback	1	2	2	A weekly update of progress and discussion via email
Illness and unforeseen personal issues	Delay in project delivery and progress	1	3	3	Apply for mitigating circumstances
Poor time management skills	Unable to meet the project deadline	1	3	3	Set short term goals and rearrange priorities if the deadline could not be met

1-6 - Low Risk

8-12 - Moderate Risk

15-25 - High Risk

2. Background Research

This chapter provides the context of the work involved in the project. It defines key concepts and technologies regarding the subject, creating a general foundation for the problem.

2.1 Cloud Computing

Cloud computing, a virtualised datacentre, offers rich computation and storage capabilities with effective economies of scale [11]. Cloud computing processes a vast amount of data and heavy computation tasks through on-demand and configurable computing resources. Cloud computing allows for various tasks to be offloaded to a datacentre, which solves the problem of end-devices' lack of computational and energy resources limitations.

However, cloud computing comes with high latency and limited bandwidth, as datacentres are likely to be located distantly from end-user devices. In situations requiring instant data processing such as Augmented Reality with real-time constraints [12], this factor cannot be tolerated as the response could be tied to a vital decision of the application. In addition, the architecture concerns the loss of privacy as it releases personal and social data to centralised services. The emergence of edge computing supplements the limitations of the traditional cloud approach, by deploying resource-rich servers at the edge [2].

2.2 Internet of Things (IoT)

This section illustrates the basic concepts of the Internet of Things (IoT) and its conventional approach. In addition, the integration of IoT with edge computing will be introduced, to help mitigate the challenges of the existing architecture.

2.2.1 Internet of Things

Internet of Things (IoT) is a network of connected sensors and devices, which continuously produce and exchange data via complex networks to provide intelligent analytics [13]. Due to the resource-constrained nature of the devices, data collected is offloaded to the cloud datacentre for further processing [2].

This process introduces new challenges such as high communication latency, network bandwidth requirements, resource constraints and security challenges, which arises from large data transmission [14]. IoT applications that require real-time data analytics, such as safety and health applications, cannot tolerate the response time of the conventional cloud approach [15].

2.2.2 IoT and Edge Computing

To help address the aforementioned issues, the concept of edge computing has emerged. It places computing resources, referred to as micro datacentres or cloudlets, in proximity to end devices at the network edge [16]. Based on the request, one or more Virtual Machines (VMs) are launched to execute application tasks remotely on edge servers [12].


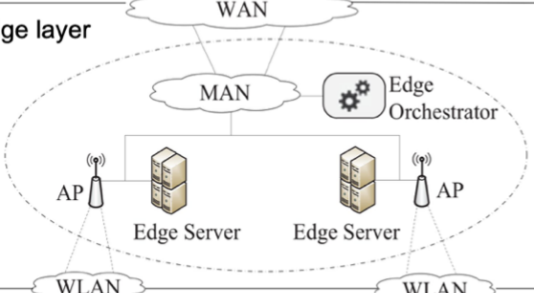

Edge computing supports the mobility and geographical dispersion of IoT applications, where devices can offload and process data in the closest computing resource. By taking advantages of these intermediate nodes and storage capabilities, reduced bandwidth demands on the network and faster response time can be achieved [1].

The computational capacity in proximity, enough storage space, and fast response time that edge computing offers, can help IoT solve its critical issues [15]. With the substantial growth of the number of IoT devices, the selection and management of edge nodes will be a crucial aspect for the successful deployment of the future IoT applications [2].

2.3 Edge Computing

Edge computing places computational resources at the network edge, allowing for proximate data processing [3]. By distributing the edge servers near the end devices, devices can offload their tasks, ensuring short response time and better quality of service [17].

2.3.1 Edge Computing Architectures

Layer	Parameters	
Cloud layer 	<ul style="list-style-type: none"> • WAN bandwidth • WAN propagation delay Datacentre <ul style="list-style-type: none"> • Number of VM on cloud host • Cloud Processing speed 	
Edge layer 	Edge server <ul style="list-style-type: none"> • Number of edge server per location • Number of cores • Edge processing speed • Edge server storage and RAM • Number of VMs per server 	Architecture <ul style="list-style-type: none"> • Single tier • Two tier • Two tier with EO Virtual Machine <ul style="list-style-type: none"> • VM processing speed • VM storage and RAM
IoT layer 	<ul style="list-style-type: none"> • WLAN bandwidth • WLAN propagation delay IoT device <ul style="list-style-type: none"> • Number of IoT devices • Task interarrival time • Active / Idle period • Dwell time in each location 	Application <ul style="list-style-type: none"> • Usage percentage • Cloud selection probability • Data upload and download size • Task length • VM utilisation

[5]

Figure 2. Edge architectures and parameters in each layer (inspired by [6])

Figure 2 illustrates the three-layer architecture of edge computing, including the cloud layer, edge layer, and IoT device layer.

1. Single-tier

Single tier architecture only utilizes the edge layer to perform its computations [6]. Devices communicate with edge servers over Wireless Local Area Network (WLAN) for data transmission and processing [18]. Edge servers allow mobile devices to perform complicated tasks which otherwise would be unable due to the resource constraints of the devices.

2. Two-tier

The two-tier architecture utilizes both edge and cloud layers to perform computations. If edge devices require more computational power to process the data [9], or based on a probability selection mechanism, the cloud datacentre is used [6]. Mobile devices offload their tasks to the global cloud through Wide Area Network (WAN).

3. Two-tier with Edge Orchestrator (EO)

If the edge servers belong to the same administrative organisation, the system can be expanded by sharing the federation of edge servers [18]. Tasks are first sent to the edge orchestrator (EO), where EO load balances and redirects tasks among multiple edge servers based on their status [19]. This requires data transfer from one node to another, which might result in a slight transmission delay [18]. Despite the concern, EO architecture provides better performance than single or two-tier architecture as the workload is distributed between computational resources, in which the VMs do not experience overload and congestion [6].

2.3.2 Advantages of Edge Computing

The proximity of computational resources helps relieve many issues of the existing architecture [3]. The advantages of integrating Internet of Things with edge computing are discussed next.

1. Reduced bandwidth costs

Placing edge servers close to end-user devices serves as a platform for filtering and analysing the data to be sent to the cloud [1]. Uploading large volumes of data produced by IoT devices to a cloud datacentre is a bandwidth-intensive computation, which leads to network congestion and transmission delay [3]. Edge helps to relieve the pressure on the network traffic, thereby reducing the bandwidth demand and increasing network performance.

2. Lower transmission latency

Edge computing enables data processing at the network edge, supporting real-time analytics for time-sensitive systems [14]. As clients do not encounter Wide Area Network (WAN) delay to access cloud services such as Amazon Web Services (AWS) or Google Cloud [15], edge computing provides a lower transmission delay in data communication [5]. Satyanarayanan et al. [16] propose that the use of edge servers for wearable cognitive-assistance systems improves response times by between 80 and 200 milliseconds (ms) [20].

3. Reduced energy consumption

IoT devices can mitigate their energy consumption by offloading the tasks to edge servers [15]. Research by Michalák and Watson [21] identified that for applications, communication consumes a lot more energy compared to processing [21]. Therefore, by performing in-network processing, the communication can be reduced hence the energy per unit data can be reduced [22]. Satyanarayanan et al. [16] propose that task offloading reduces 30-40% of energy consumption on mobile devices [3]. On the other hand, Noor et al claim that reduced battery consumption comes with higher bandwidth usage and power consumption, which is a trade-off to be investigated further [23].

4. Privacy policy enforcement

Private data collected from IoT devices are pre-processed in edge resources [2], enforcing the privacy policies of its owner before releasing to centralised cloud services [16]. The data is fragmented and distributed across edge nodes, allowing for enhanced security management and data protection [17].

5. Scalability and availability

Data being processed and filtered in dispersed edge nodes leads to distributed data processing architecture, thereby scalable [1]. Moreover, if cloud services become unavailable, the redundant edge servers can replace the service to mask the failure, making the service more available [16].

2.4 Edge Computing Simulator

To simulate the performance of different edge architectures, a simulator that fulfils the edge computing domain is required. With the development of edge computing paradigm, various simulation tools have been proposed to model the edge and IoT environment [9]. This section will discuss the capabilities and features of various simulators and propose a justification of the chosen simulator, EdgeCloudSim.

iFogSim [1], proposed by Gupta et al. models an edge and fog computing environment to evaluate the resource management techniques in latency, network congestion, energy

consumption and cost. However, it does not support failure modelling, which is essential as it serves as a guide to building a better performing edge architecture.

IoTsim [24] simulates IoT big data processing, supporting heterogeneous IoT and edge systems. However, it does not provide networking support and edge processing, limiting the investigation scope of edge computing.

FogNetSim++ [25] models a large fog network, incorporating various mobility models and scheduling algorithms. While it supports suitable metrics for performance evaluation, it does not enable interoperability between fog federations, which is needed to investigate the two-tier with EO architecture.

EdgeCloudSim [5], based on CloudSim [26], provides a design space for configuring computational and networking parameters for performance evaluation [5]. It provides mobility support with geographic awareness, which characterises the nature of end-user devices in IoT, such as moving cars and users. EdgeCloudSim has been selected due to its suitability of supported metrics and parameters in regard to the project aim stated in section 1.5. The simulator uses three configuration files to compose different edge architectures, which are as follows:

1. **default_configuration.properties** - contains key simulation parameters such as the simulation time, number of mobile devices, network configuration, cloud datacentre settings and simulation scenarios.
2. **edge_devices.xml** - consists of key-pair values that model the edge server topology. It defines edge server characteristics such as its access points and server hosts [10]. Specifications of the VMs are available for configuring, such as its storage, processing speed and the number of cores.
3. **applications.xml** - models the characteristics of IoT applications used. It defines application properties such as task length, data size, active/idle period, and the probability of cloud selection.

These configurations together provide full control of different edge computing scenarios. Following the configuration is the execution of the experiment, which is done by stating related files in a command line.

3. Literature review

This chapter aims to present a critical appraisal of relevant literature and to summarise existing solutions that address similar problems. The findings are classified according to the architecture considered in the research – single-tier, two-tier, and two-tier with EO.

3.1 Single-tier

Single tier architecture exclusively uses the edge layer to serve its requests, where the end-user device and the edge server is within the same WLAN [18].

Service time

Sonmez et al. [18] and Suryavansh et al. [19] designed a simulation-based experiment using EdgeCloudSim for performance evaluation of a face recognition application and an augmented reality application. Both papers proposed that for single-tier architecture, the service time increases as the number of mobile devices increases. Suryavansh et al. indicated that the service time increased from 2 to 4 seconds as the number of mobile devices grew from 100 to 600. Single tier architecture has a limited number of edge servers - therefore, no new tasks can be admitted if bottleneck occurs in popular locations due to exhaustion of resources, which increases the service time [12].

The results of the above investigation agree with another study performed by Jha et al. [9], where a self-driving application was simulated on an IoTsim-Edge simulator. The study identified that as the processing is done in a time-shared manner, it leads to increased service time.

Suryavansh et al. [19] studied the impact of the VM capacity on the service time of the application. The finding shows that the service time increases with the reduction of the VM capacity. With low VM capacity, edge servers can handle fewer tasks, hence it takes longer for tasks to get processed.

Failed task

Sonmez et al. [5] proposed that the percentage of failed tasks due to VM capacity increases as long as the WLAN bandwidth increases. As the bandwidth increases, tasks arrive at a faster rate which causes the shortage of network resources, incurring network congestions [12]. The edge servers face limitation in its VM resources, hence the percentage of failed tasks increases.

Additionally, Suryavansh et al. [19] stated that the decrease in the number of edge servers due to device failure results in a computational overload in the functional servers. This increases the percentage of failed tasks as it does not have sufficient computational capacity, leading to a degradation of the application performance.

Latency

Gupta et al. [1] investigated the network congestion for a latency-sensitive online game using iFogSim simulator. As the number of mobile devices increases, the load on the network increases, leading to network congestion. Sonmez et al. [18] identified that the communication latency for single-tier architecture is solely caused by WLAN delay, as it does not encounter Wide Area Network (WAN) communication. WAN congestion introduces significant communication latency, which makes real-time communication impossible for time-sensitive applications.

Premsankar et al. [2] designed a testbed for GamingAnywhere cloud gaming platform, which identified that the use of network edge servers gives significantly low latency of under 25ms. The study identified that the public Amazon Cloud in Germany and Ireland incurs a delay that is at least twice as much, compared to the edge servers. The use of closely located servers is beneficial as it does not encounter a significant WAN delay caused by the long-latency links.

Summary

It is concluded that offloading the workload to the edge nodes certainly improves the application performance due to the benefit provided by its proximity. The resource provided by the edge server is adequate, but not as sufficient as what is available from the infinitely scalable cloud datacentre [12]. Hence, further offloading to a more resourceful facility is desired.

3.2 Two-tier

Service time

Jha et al. [9] proposed an IoTsim-Edge simulator, where they simulated a self-driving application that utilises both the edge and the cloud server for processing. If a car moves and goes out of the range of the current edge server, the processed data is transmitted to an appropriate edge server. The communication time between the edge servers increases the service time of the application, although it is balanced by the faster processing speed of the two-tier architecture.

Sonmez et al. [18] examined the effect of task size on face recognition application's service time. The application requires high computational resources; hence the processing time dominates the service time, especially when the task size is large. When the task size is 4000 Million Instructions (MI), single and two-tier architecture encounters computational resource congestion, leading to longer processing time. For smaller task size of 250 MI, the WAN delay accounts for most of the service time, and almost no failure happens during processing as it has enough processing power.

Aljulayfi and Djemame [6] identified that increasing both the WLAN bandwidth and VM processing speed leads to the reduced processing time as there is no congestion on edge servers, hence no overload on the VM.

Failed task

A study by Aljulayfi and Djemame [6] showed that an increase in the number of edge servers leads to a decreased percentage of failed tasks. More requests can be accepted, which are then distributed across the edge nodes for efficient processing. Moreover, the percentage of failed tasks decreases in the cloud as well, as fewer tasks are offloaded to the cloud - the edge layer has sufficient computational resources, hence the need for cloud offloading decreases.

However, more tasks being accepted increases the load on the VMs, leading to a saturation of the VMs. There exists a trade-off between WLAN bandwidth and the task failure, as tasks arriving at a faster rate causes more failure due to VM capacity.

The study showed that the failure due to the VM capacity can be mitigated by increasing the VM processing speed. As long as the VM processing speed increases, the processing gets done faster, preventing the queuing of the tasks. Sonmez et al. [5] argue that if the VM utilisation is too high, the requested task cannot be accepted, leading to a high percentage of failed tasks. Aljulayfi and Djemame [6] stated that increasing both the bandwidth and the

VM processing speed improves the overall performance of the architecture, due to no congestion in the WLAN and no overload on the VM.

Sonmez et al. [5] identified that the task failure increases dramatically for two-tier when the WAN bandwidth is very low of 4 Mbps, due to WAN congestion. The communication for cloud data processing can lead to WAN congestion, increasing the task failure by 25% in the worst case.

Latency

According to Gupta et al. [1], the two-tier architecture has managed to keep its latency low despite the increase in the number of devices, as most of the data-intensive communication happens through low-latency links. In two-tier architecture, most tasks are sent to the edge devices through WLAN, which Sonmez et al. [18] identified it to be a trivial reason to task failure. On the other hand, the WAN delay dominates the network congestion, accounting for more than 30% of the overall latency.

Gupta et al. [1] also identified that as the size of the edge server topology increases, the average network delay reduces. More computational resources are made available for processing; hence no specific edge node gets overly saturated. Aljulayfi and Djemame's [6] investigation indicated that the WLAN delay improves with the addition of the edge servers, as there is no network congestion. In addition, the study stated that cloud datacentre also benefits from the increase in the number of edge servers, as a smaller number of tasks are sent to the cloud, reducing the WAN delay.

Summary

Overall, it can be concluded that the two-tier architecture outperforms the single-tier architecture as it offloads some of the tasks to the cloud datacentre, relieving some of the computational and networking resource limitations. However, a study by Liu et al. [12] revealed the insufficiency of the two-tier architecture for compute-intensive and time-sensitive mobile applications, proposing the need for a coordinated allocation of computing and network resources.

3.3 Two-tier with EO

To exploit the capabilities of computing resources, two-tier with EO architecture has been proposed - it utilises the full resources of both the edge servers and the cloud datacentre through effective coordination [12]. The orchestrator helps to provide different service to various users, according to the requirements and the availabilities of the resources [20].

Service time

A study by Suryavansh et al. [19] identified the two-tier with EO architecture utilises the edge servers as much as possible by load balancing, so that the data transmission to the cloud only happens when the edge capacity is exhausted. The minimization of the requests sent to the distant datacentre offsets the unsuitability of the cloud that arises due to the WAN delay, reducing the service time. On the other hand, if the probability of task offload to the cloud is high, the performance can be negatively impacted by the WAN delay that arises due to long-latency link communication.

Aljulayfi and Djemame [6] pointed out that when the number of mobile devices is high, the two-tier with EO architecture has significant improvement in the processing time compared to single and two-tier architecture, as it balances the load across the VMs. In addition, Sonmez et al. [5] examined that increased task size does not affect the service time as it has enough computational resources for processing.

Failed task

Sonmez et al. [5] proposed that in two-tier with EO architecture, average task failure due to mobility does not increase, as tasks are distributed and executed in a fast manner across the edge nodes.

Latency

Premsankar et al. [2] proposed that a traffic flow control for the edge servers reduces the service latency of the architecture while supporting for scalability. Firstly, the Edge Orchestrator can forward computationally intensive tasks to VMs with higher processing capabilities based on the need. Moreover, edge computing resources can be migrated to support the mobility of end-user devices. The orchestrator enables a scalable and desirable performance of the application by load balancing the tasks and resources based on the requirements.

Sonmez et al. [5] identified that as the orchestrator balances the load across computational resources, the utilisation of the VMs is consistent among the nodes without any overloading in particular VMs.

Summary

Overall, the studies highlighted the importance of two-tier with EO architecture consuming all available edge servers, outperforming the single and two-tier architectures as the VMs are not overloaded and there exists no resource congestion.

This project will contribute to the research by addressing further relationships between the parameters and the performance that have not been studied previously. Based on the understanding and skills gained from this chapter, simulation experiments will be designed in the next chapter.

4. Design

This chapter details the design of the experiments that were constructed to investigate the edge computing performance. Hypotheses are formulated, and experiment scenario with its parameters are presented.

4.1 Hypotheses

The research conducts a performance evaluation of edge computing in relation to scalability. A system is said to be scalable if it remains effective with no performance loss when additional resources are added [27]. In other words, as the load condition increases, the service should remain effective. In the scope of this investigation, the 'effectiveness' is measured by evaluating the system's service time and percentage of failed tasks.

Two hypotheses have been formulated to help demonstrate the aim of the project. The results of the experiments will validate the correctness of the hypotheses, identifying the factors that affect the performance of the application. Hypotheses are as follows:

Hypothesis 1 – Increase in WLAN bandwidth and VM processing speed reduces the service time.

Hypothesis 2 – Increase in the number of edge servers and VM capacity reduces the percentage of failed tasks.

The combination of reduced service time and failed tasks will prove for the scalability of the application, where the tasks are handled in a reasonable time and accuracy.

4.2 Experimental Design

4.2.1 Experiment's Assumptions

The study will focus on the scenario of the face recognition application, as the parameter values modelled in Table 2 is the most compatible with this scenario. Face recognition application is a compute-intensive task, which requires a high volume of data upload [28]. The edge server processes the data and returns a relatively small-sized response back to mobile devices [18]. It is assumed that the user sends a task and waits for the reply, instead of sending successive requests [18]. The environment will be simulated for 30 minutes, which is considered enough for the main events to have taken place. Moreover, each experiment will be repeated five times to minimize the variation in the dataset to gain statistical significance.

4.2.2 Initial Experiment

An initial experiment will be conducted in order to identify the key parameters for the specific scenario used. The application uploads 1500 Kilobytes (KB) of data and downloads 15 KB of data. The users are assumed to be active for 45 seconds and idle for 15 seconds. Additionally, the WAN and WLAN delay reflects the average latency values of real life, which are 100 milliseconds (ms) and 5ms, respectively.

In order to avoid discrepancies arising from EdgeCloudSim, the initial experiment will reproduce Sonmez et al.'s [18] experiments. Hence the configuration values have been inspired by the literature [18], allowing for the verification and comparison of the results. Table 3 illustrates the initial experiment specific parameters and their values.

Table 2. Simulation parameters

Parameter (Unit)	Value
Simulation time (min)	30
Number of repetitions	5
Min. number of mobile devices	50
Max. number of mobile devices	250
Cloud processing speed (MIPS)	20000
WAN propagation delay (ms)	100
WLAN internal delay (ms)	5
Probability of cloud selection	10%
Data upload size (KB)	1500
Data download size	15
Active period (sec)	45
Idle period (sec)	15
Dwell time in L1, L2, L3 (sec)	60,30,15

Table 3. Initial experiment parameters

Parameter (Unit)	Value
WAN/ WLAN bandwidth (Mbps)	20 / 300
Number of edge servers in L1, L2, L3	2,4,8
VM processor speed (MIPS)	1000
VM capacity (KB)	50000

4.2.3 Experiment Design

In edge computing, there exists a number of parameters that needs investigation in three different layers of the architecture, as illustrated in figure 2. The initial experiment identified the significance of the computational power at the network edge, highlighting the importance of VM capacity and the number of edge servers, see section 6.1. In addition, the WLAN bandwidth and the VM processing speed has shown its influence on the network delay and processing time, which will be examined further.

Four main parameters identified from the initial experiment will be used to test against the hypotheses proposed in section 4.1. Table 4 illustrates the details of the configuration values.

Table 4. Description of parameters for hypotheses testing

Hypothesis	Experiment	Parameter	Values		
-	1	Initial experiment	-		
Hypothesis 1	2	Bandwidth (Mbps)	100	300	500
	3	VM processing speed (MIPS)	1000	2000	3000
Hypothesis 2	4	Number of edge servers per location	2	4	6
	5	VM capacity (KB)	50000	75000	100000

1. An initial experiment is designed to identify the key parameters for the investigation and to understand the behaviour of different architectures in edge computing.
2. Effect of **WLAN bandwidth**: is designed to investigate the effect of increasing the bandwidth on network delay and the service time in a bigger scope.
3. Effect of **VM processing speed**: is designed to investigate the effect of increasing the VM processing speed on the processing time.
4. Effect of **edge servers**: is designed to investigate the effect of increasing the number of edge servers per location on the percentage of task failure due to mobility. It examines the impact of increased server availability.
5. Effect of **VM edge capacity**: is designed to investigate the effect of increasing the VM capacity on the number of failures, which will affect the server's handling of incoming tasks.

The combination of experiment 2 and 3 will test Hypothesis 1, and experiment 4 and 5 Hypothesis 2. By doing so, the study aims to evaluate the scalability, where it provides better performance through reduced service time and percentage of failed tasks.

4.3 Performance Metrics

Considering various performance metrics is important as it helps us understand the underlying behaviour and relationships in the data. In time-constrained Augmented Reality application, the time taken for the provision of service plays a crucial role. Hence, one of the performance metrics that will be investigated is the **service time**, which refers to the processing time and the network delay. Furthermore, the **percentage of failed tasks** is selected as another metric as the successfulness of the task processing is another important factor in application performance. Moreover, the **VM utilisation** is considered to monitor the workload on the VMs and how this impacts the performance.

5. Implementation

This chapter illustrates how the experiments detailed in the previous section will be implemented. It outlines the implementation environment, simulator details, and the program used to analyse the results collected from the simulation.

5.1 Implementation Details

The experiment will run on a MacBook Pro with 2.5GHz Intel Core i5 and 8GB memory. The IntelliJ platform will be used to configure, compile and run the experiment.

EdgeCloudSim version 2.0 will be used for the implementation, where different edge computing scenarios can be modelled and tuned through the configuration files. A runner script will be used to run the simulations in parallel, specifying the number of available cores and iterations. The simulation results are saved as Comma Separated Values (CSV) per iteration, as it eases the exporting and processing of the data [5].

5.2 Evaluation Metrics

Evaluation metrics for the experiment have been identified in section 4.3. They are used to measure how well the scenario performed under different criteria. The primary reason for using EdgeCloudSim [5] is its support for the various performance metrics, meaning that the values of the chosen metric can be obtained directly in the form of numerical data.

In EdgeCloudSim, further decomposition of the main evaluation metric, service time, is available as a combination of 'network delay' and 'processing time' metric. The network delay is further divided into WAN and WLAN delay, which identifies the effect of different networking parameters more evidently. Another important metric is the task failure, which is supported via the percentage of failed tasks along with the failure reason. The VM utilisation is illustrated as a percentage of how loaded the VMs are.

5.3 Result Visualisation

To visualise the results produced by the EdgeCloudSim simulator, MATLAB is used. Running the simulation produces numerous log files, where individual files are specific to the architecture, the number of mobile devices, and an application used. MATLAB plotter files included in the simulator take the log files and generate graphs based on a chosen performance metric. Various style, formats and error plotting schemes are deployed in MATLAB, generating graphs that are appropriate for the data.

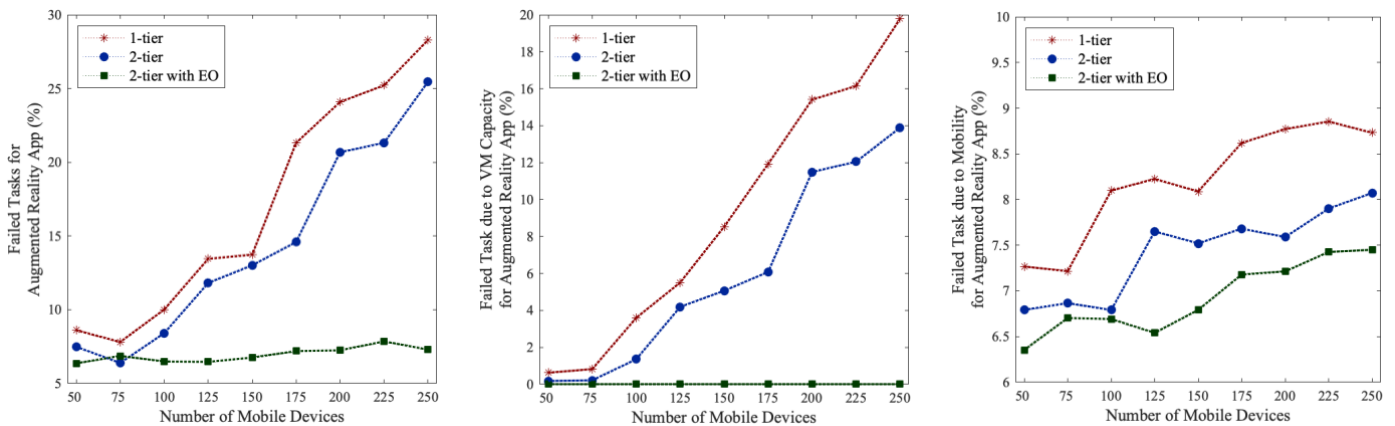
6. Technical Evaluation

This chapter provides a technical evaluation of the simulation results of a face recognition application. Graphical analysis of plots followed by justification and explanation will be carried out. It tests the hypotheses formulated in the design chapter by contrasting it to the results obtained. The y-axis of the graphs represents the evaluation metric and the x-axis represents the load condition. In the remainder of the report, all graphs will display the average of 5 iterations.

6.1 Initial Experiment

The graph displays the average of 5 iterations. From the statistical point of view, the calculated standard deviation was 0.1472, which indicates that the data is centred around the mean and hence is less variant. The standard deviation was used to show the indication of errors, as error bars were too small to be seen in the graphs.

Percentage of failed tasks



(A) Percentage of failed tasks

(B) Failure due to VM capacity

(C) Failure due to mobility

Figure 3.1. Percentage of failed tasks and failure reason - Initial experiment

The results of the initial experiment show that the percentage of failed tasks for single and two-tier increases rapidly as the load condition increases. In figure 3.1A, a steep increase in failure can be observed for 150 and 175 devices for single and two-tier architecture, respectively. The two-tier with EO architecture's failed tasks remained constant around 7%, providing a stable service.

The failure reason can be seen in figure 3.1B and 3.1C. Through the initial experiment, it was identified that the network delay had no significant impact on the task failure for the

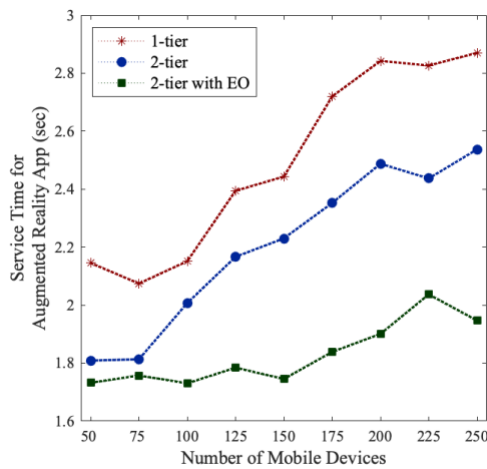
particular configuration used. However, the percentage of failed tasks due to VM capacity increased with the growth of the load condition. For single-tier architecture, 20% of tasks failed due to VM capacity when the load condition was the highest. Two-tier architecture faced a similar trend, although the failure percentage was slightly lower than the single-tier architecture due to its additional computing resource, the cloud datacentre.

Two-tier with EO architecture was not affected by the VM capacity as tasks were distributed to different processing facilities based on the application's requirements and the system's capabilities.

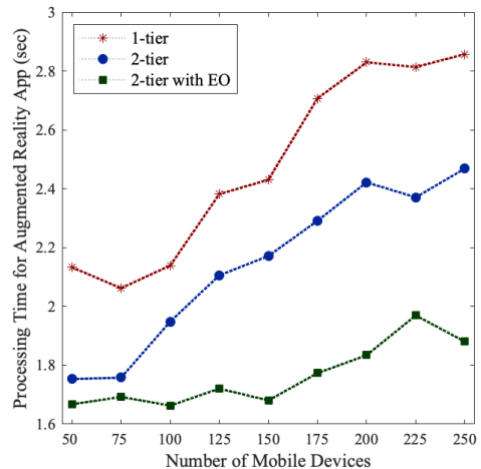
Failure due to mobility happens when the user requiring face recognition leaves the WLAN coverage area before receiving the response [5]. There is an increasing trend for all architectures as the load increases, with single-tier having the highest failure percentage.

The VM capacity and the number of edge servers have been identified as important, as it could reduce the percentage of failed tasks. Therefore, experiments will be designed to investigate the effect of increasing the number of edge servers and VM capacity.

Service time



(A) Service time



(B) Processing time

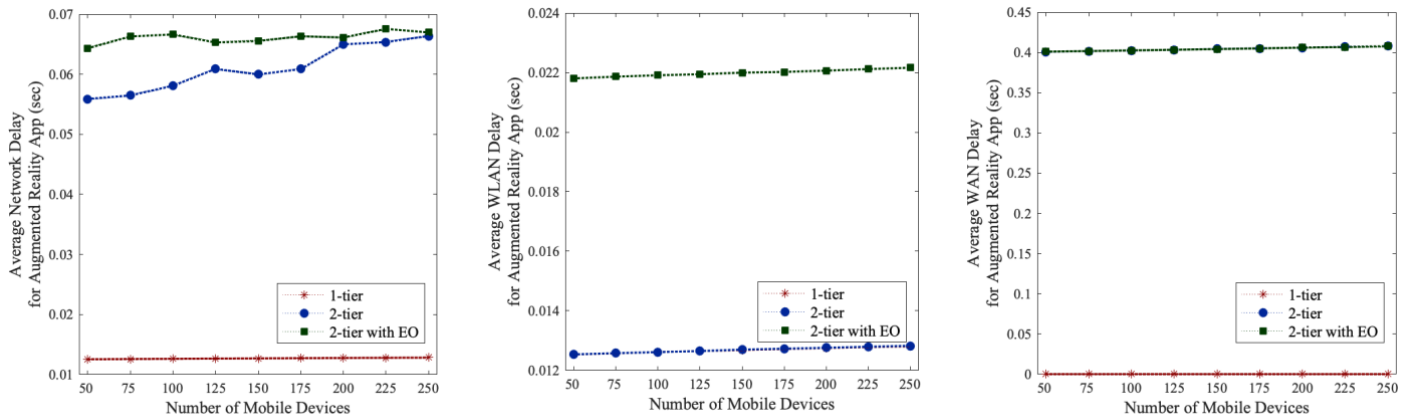
Figure 3.2. Average service time and processing time – Initial experiment

As figure 3.2A illustrates, the service time increases as long as the load condition increases. The single-tier architecture has the longest service time, followed by the two-tier architecture. Two-tier with EO architecture has a significantly lower service time, with a stable trend as the load increases. To investigate the individual components of the service time, two sub experiments exploring the processing time and the network delay have been designed.

Processing time

The processing time follows the same trend as the service time, showing an increasing trend for all architectures – see figure 3.2B. The EO architecture provides the fastest response with relatively low processing time compared to the other two architectures. It faces a slight increase in processing time when the load condition is 200, which after 225 devices, decreases again.

Network delay



(A) Average network delay

(B) WLAN delay

(C) WAN delay

Figure 3.3. Average network delay - Initial experiment

In figure 3.3, the network delay, which is composed of average WLAN and WAN delay, is prominent in two-tier and two-tier with EO architectures. The EO architecture has the highest WLAN delay of 0.022 seconds, whereas the single and two-tier architecture has a delay of 0.013 seconds, as seen in figure 3.3B. The single-tier architecture does not encounter WAN delay, hence is 0, whereas the two-tier architectures face a high delay of 0.4 seconds as it communicates with the cloud. Two-tier with EO architecture with faces both WLAN and WAN at a high rate, hence it has the highest network latency.

From this experiment, it can be derived that the main factor of service time lies in the processing time, although the network delay contributes to an increased latency to some extent. Even though the results indicate that the main cause of the communication latency is due to WAN delay, investigating WAN would be meaningless for the single-tier architecture. Therefore, the WLAN delay is selected as a key parameter for further investigation.

In the next section, the WLAN bandwidth and the VM processing speed will be further investigated to examine the impact on the network delay and processing time, respectively.

6.2 Experiment Results

6.2.1 Effect of WLAN bandwidth

The experiment was designed to investigate the effect of changing the WLAN bandwidth on the performance results. The simulation identified that the **network delay decreased slightly as the WLAN bandwidth increased but had a minor impact on other evaluation metrics.**

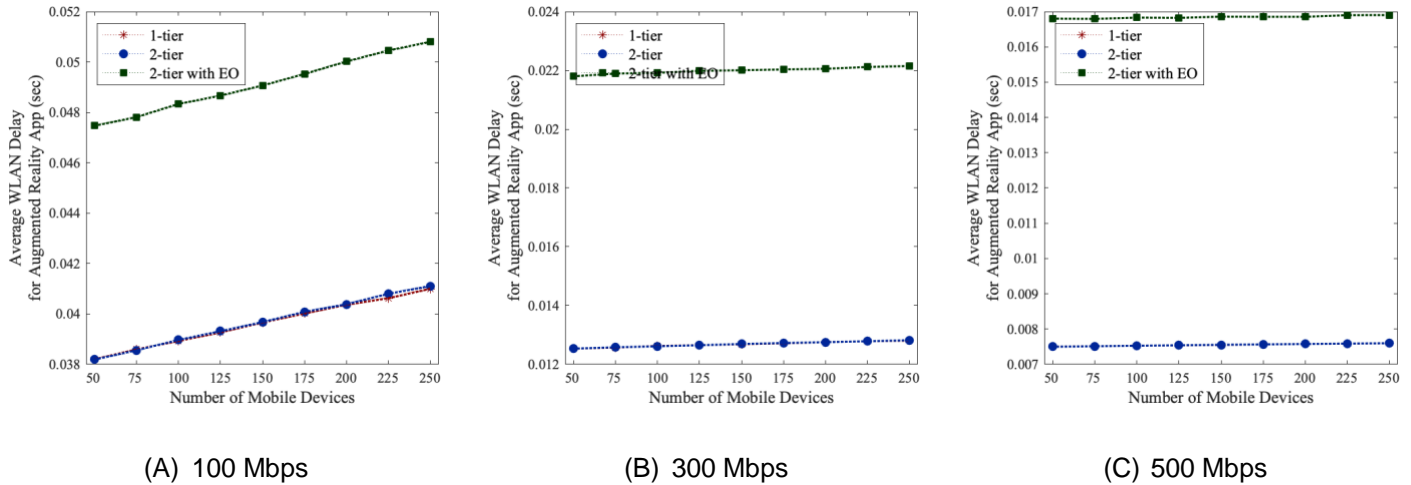


Figure 4.1. Average WLAN delay – effect of WLAN bandwidth

Network delay specifies how long the data took to travel from one point to another. In edge computing, the network delay is composed of two parts – WLAN used for edge node communication and WAN for cloud communication. Figure 4.1 shows the impact of WLAN delay as the load condition increases. For 100 Megabits per second (Mbps), all architectures show an increasing trend as more tasks are being sent to the edge servers and the cloud datacentre. Two-tier with EO architecture experiences 0.01 second higher delay than the single and two-tier architectures, due to the extra communication needed for load balancing.

As the bandwidth increases, the WLAN delay decreases, and remains stable even with an increasing load condition – see figure 4.1B. With increased WLAN bandwidth, the network was able to handle the task transmission stably with its available networking power. **The communication time can be decreased as long as the network bandwidth increases, leading to shorter service time.**

Nevertheless, **the percentage of failed tasks did not improve as the WLAN bandwidth increased.** A bottleneck has never occurred with the configuration used, concluding that the **WLAN bandwidth is not the main reason to failure.** Furthermore, decreased communication time did not lead to a significant reduction of service time as the WLAN delay is a matter of just 0.01 seconds – it does not account for a big part of the service time.

Summary of the effect of WLAN bandwidth

The effect of increasing the WLAN bandwidth was rather insignificant as the bandwidth values used were sufficient for the application's task load. Instead, the experiment suggests that WLAN bandwidth is combined with another parameter that will complementarily improve the performance. Alternatively, instead of reducing the communication delay, which is trivial, decreasing the processing time would provide better service time. Thus, the next experiment will be to increase the VM processing speed, aiming for lower processing time.

6.2.2 Effect of VM processing speed

The experiment studies the effect of increasing the VM processing speed. The processing speed has increased from 1000 to 3000, with 1000 increments each time.

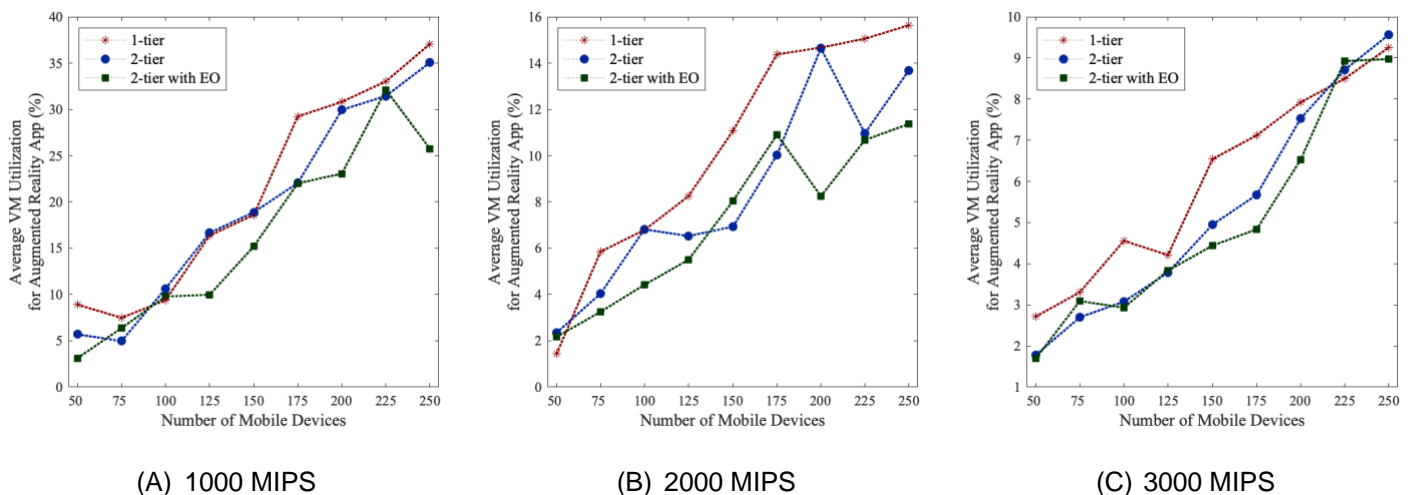


Figure 5.1. Average VM utilisation – effect of VM processing speed

It was observed that **the VM utilisation decreases as the VM processing speed increases**. When the processing speed was 1000 MIPS, the utilisation was high meaning that there were a lot of tasks being executed in the VMs, as seen in figure 5.1A. However, as the processing speed increases to 2000 MIPS, the highest VM utilisation for single-tier architecture is 16%, which compared to the 40% of 1000 MIPS, is significantly low. The tasks are executed in a fast manner which leads to low VM utilisation, hence there are no bottleneck or overload in the VMs, providing better performance.

Due to the better utilisation of VMs, the service time decreases for all tiers, as seen in figure 5.2. Two-tier with EO architecture provides a stable service time despite the increase in load condition, whereas the other architectures show an increasing trend as the number of mobile devices increases. When 250 load condition for single-tier is chosen, increasing the VM processing speed by 1000 MIPS led to reduced service time of 2.8 seconds to 1.22 seconds to 0.73 seconds.

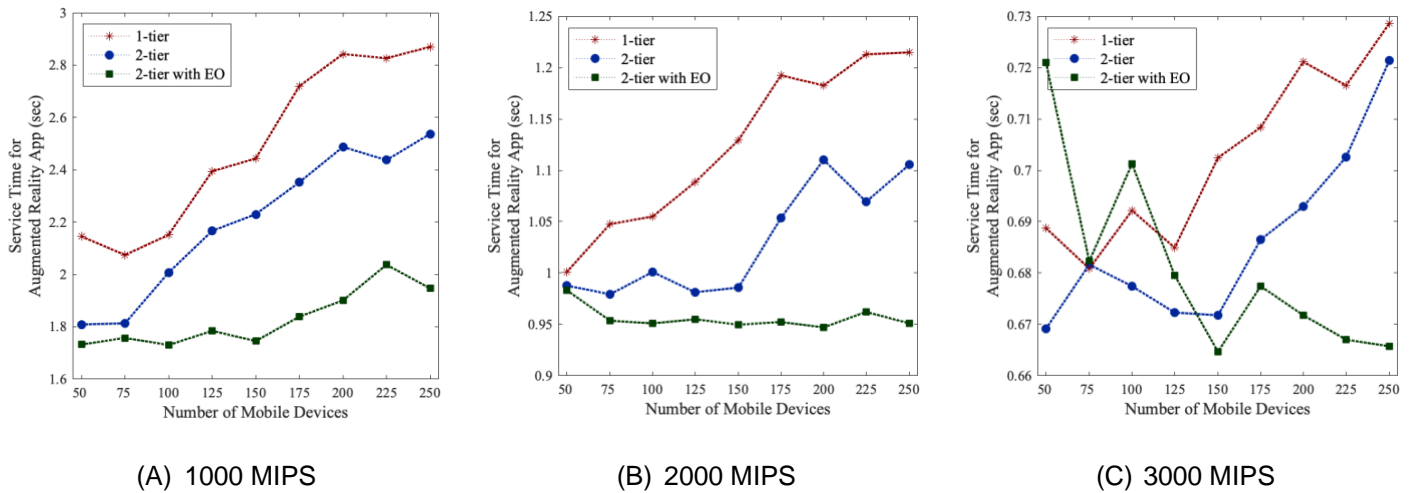


Figure 5.2. Average service time – effect of VM processing speed

However, in figure 5.2C, an unexpected trend is observed for two-tier with EO architecture for 3000 MIPS. The service time is the highest when the load condition is the smallest, and it gradually decreases. It has been examined that the network delay remains unchanged, hence the problem lies in the processing time. A bug in the edge orchestrator might have caused this odd behaviour when distributing and processing the request. Unfortunately, the exact reason as to why the problem arose could not be formulated as there is limited information about the load balancing scheme used in the simulator. However, a paper by Tham and Chattopadhyay [22] proposes a scheme that may control the workload more efficiently through a min-max optimisation problem.

In addition, **a decrease in the percentage of failed tasks can be observed as the processing speed increases.** The initial experiment identified the VM capacity to be a big part of the total failure. **As the VM processing speed increases, the task failure due to VM capacity became extremely low, under 1% for all architectures.** This shows that the increase in the VM processing speed results in a better successfulness of tasks, as the requests are processed with no contention in the edge servers.

Summary of the effect of VM processing speed

This section has proven that further improvements in performance can be made by increasing the edge server capability – by increasing the VM processing time. The next experiment will examine the effect of increasing the edge node availability by varying the number of edge servers.

6.2.3 Effect of edge servers

The experiment was designed to identify how the number of edge servers per location will impact the performance. There are three edge server locations for this experiment, and 2, 4 and 6 edge servers will be used for each sub-experiment. This results in a total of 6, 12, and 18 edge servers, which were varied from the initial experiment value to test for the architecture scalability.

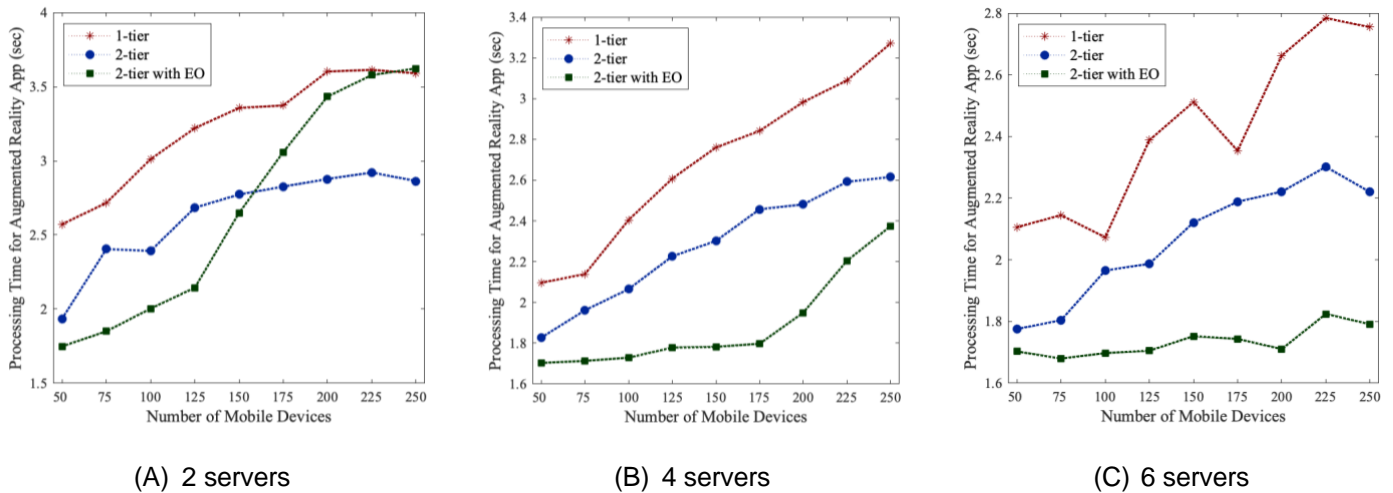


Figure 6.1. Average processing time – effect of edge servers

It has been identified that **as long as the number of edge server increases, the processing time decreases**. Two-tier architectures perform better than the single-tier, as it has an additional cloud resource to offload to. Hence the VM overload is not as problematic as the single-tier architecture. Moreover, the two-tier with EO architecture provides a significantly better processing time for increased load condition.

The two-tier with EO architecture faces a bottleneck after 125 devices in figure 6.1A. The processing time increased substantially, even when the orchestrator is supposedly balancing the load efficiently. Although the network delay of 0.065 seconds accounts for the processing time to some extent, the delay is trivial hence is negligible. The problem might have occurred when the orchestrator was moving the data from one node to another, causing communication overhead. A defect in the load-balancing scheme might have caused an overhead, leading to a spike in the processing time.

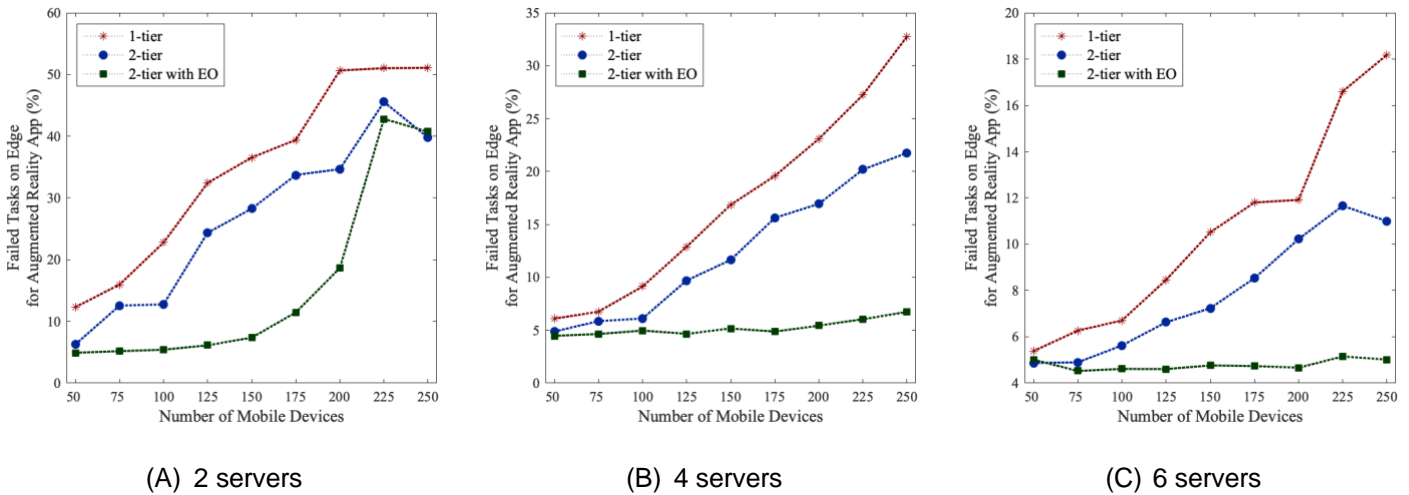


Figure 6.2. Percentage of failed tasks on edge – effect of edge servers

The results showed that although there are more edge servers available, **VM congestion is inevitable as the servers have limited storage capacity and processing power**. This is evident primarily when there is a small number of available edge servers, 6.2A, which leads to increased failed task on edge.

However, as the number of edge servers increased, the percentage of failed tasks on the edge was significantly reduced for all architectures. This is because the offloading points increases as the number of edge server increases, which means that there are additional computational resources for task offloading. Two-tier with EO architecture provides very low failed tasks, as it dynamically launches suitable VMs with sufficient power while considering the geographic positions of the user and the edge server [29].

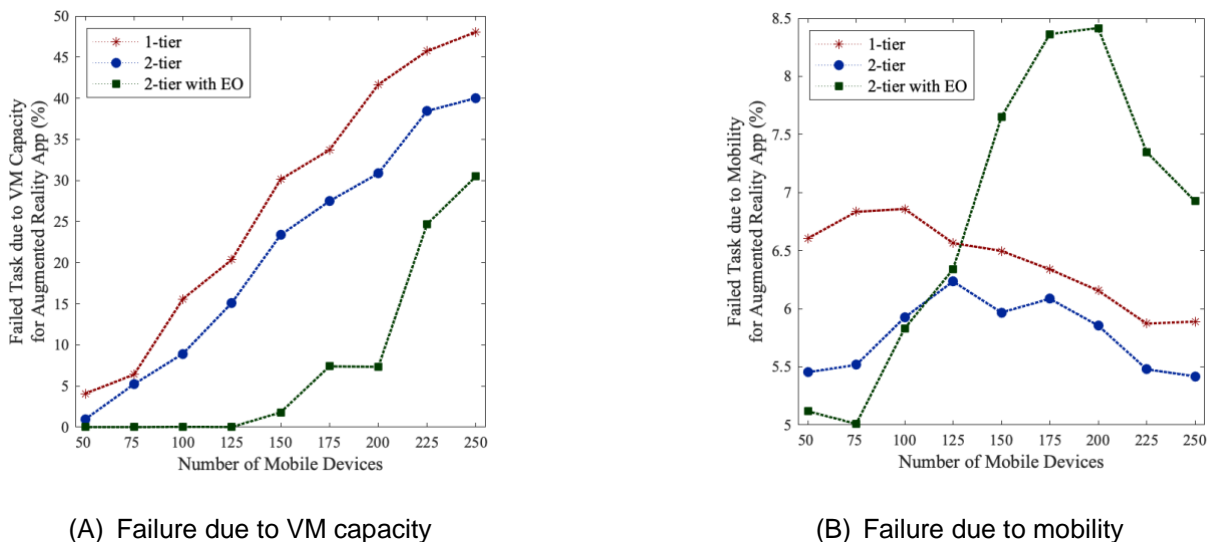


Figure 6.3. Failure reason for 2 edge servers per location

A bottleneck observed in 200 load condition for two-tier with EO architecture in figure 6.3A shows that even with a balanced and orchestrated workload, failure due to resource limitations could happen.

Figure 6.3 shows the decomposition of the failure reason. The EO architecture experienced a high rate of failure mainly due to the VM capacity, especially after 200 devices. A **computational burden was experienced on the VMs after it reached its capacity, leading to a task failure**. In addition, the failure due to mobility increased drastically after 125 mobile devices for the EO architecture. VM congestion led to longer processing time, causing the mobile devices to leave the area without receiving the response. **In other words, long processing time due to a lack of resources led to an increased probability of failure due to mobility.**

As for the VM utilisation, the single-tier architecture utilised 65% of its VM when the load condition was high. Usage of exclusive edge servers leads to the saturation of resources faster than the architectures that utilise the cloud, leading to overloading in VMs and inability to process incoming tasks. **Increasing the number of edge servers allowed for the balanced use of the VMs, solving the problem of the high failed task for two-tier with EO architecture.** The workload is balanced between the edge server federations, exploiting the full available resources.

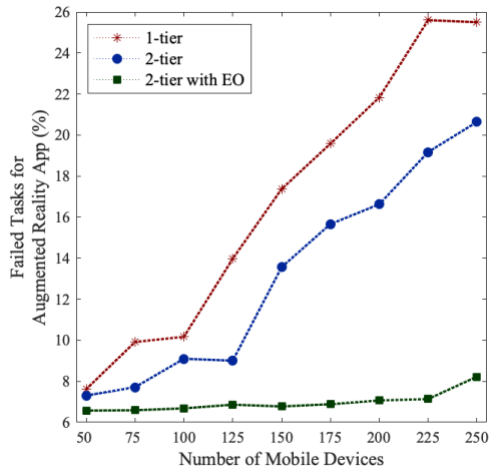
Summary of the effect of the number of edge servers

The results identified that significant performance improvement can be achieved by increasing the availability of the edge servers. However, the percentage of failed tasks for single and two-tier architecture still increased as the load condition grew, due to the limited capacity of the VMs. Therefore, further improvement in the edge server capacity is desired, which the next section will investigate.

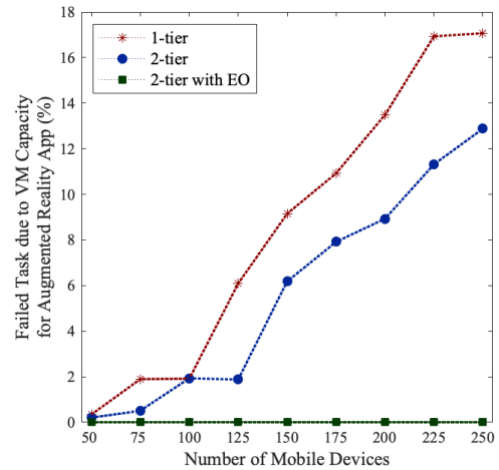
6.2.4 Effect of VM capacity

The VM capacity was adjusted to show the impact of improving the edge server capabilities on the performance results. The results indicated that **increasing the VM capacity by itself does not improve the application performance.**

The VM capacity used for the experiment were 50,000, 75,000, and 100,000 KB. As no clear improvement was seen, the experiment was rerun with a lower VM capacity of 25,000 KB to observe any change in performance.



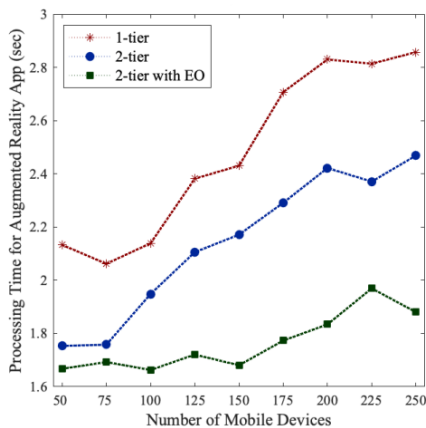
(A) Percentage of failed tasks



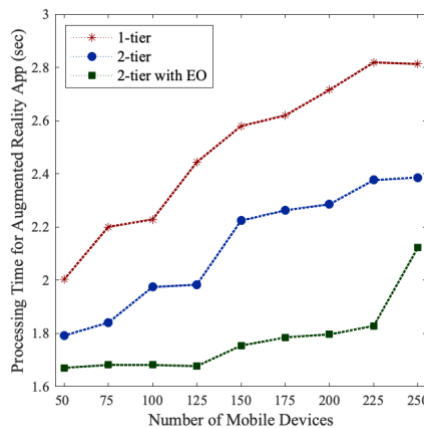
(B) Failure due to VM capacity

Figure 7.1. Percentage of failed tasks for 75000 KB – effect of VM capacity

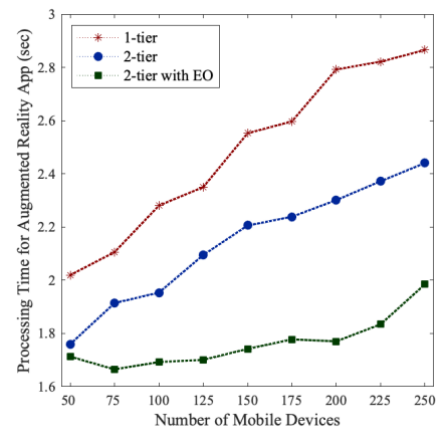
The change in VM capacity did not affect the percentage of failed tasks. All four configurations provided the same trend illustrated in figure 7.1 – in terms of both the percentage of failed tasks and failure due to VM capacity. The task failure for single-tier architecture increased up to 26% as the load increased, followed by the two-tier architecture with the same trend but with 20%. Two-tier with EO architecture has significantly low failure percentage despite the growth in load condition, which none of it was caused by the VM capacity as seen in figure 7.1B.



(A) 50000 KB



(B) 75000 KB



(C) 100000 KB

Figure 7.2. Average processing time – effect of VM capacity

The processing time increased as the load increased for all architectures, as seen in figure 7.2. **However, varying the VM capacity did not reduce the processing time** – three results are nearly identical to each other. Moreover, the VMs were utilised at a constant trend for all sub-experiments – thus, **increasing the VM capacity did not lead to better utilisation of VMs**. The utilisation increased with increased load for all architectures, meaning that more tasks are being processed inside the VMs.

Summary of the effect of VM capacity

The set of experiments identified that for the configuration used, the initial value of 50,000 KB, or the lowered version of 25,000 KB was sufficient to handle the incoming tasks and increasing it further did not affect the performance. EO architecture performed the best in terms of its failed tasks and processing time, as it has sufficient capacity and the ability to load balance the tasks, encountering less computational problems compared to the single and two-tier architecture.

In order to exploit the full advantage of increased VM capacity, increasing the edge server topology or increasing the bandwidth to allow for more incoming tasks in the edge is desired. This synergistic scheme will help to utilise the benefits that increased VM capacity has to offer, leading to a better performing application.

6.3 Findings

6.3.1 Findings Summary

Table 5 summarises the experimental results obtained from the investigation. It states the relationship between the parameter and the performance metric and compares the findings with the literature reviewed in chapter 3.

Table 5. Findings from the experiment and comparison with the literature review

Section	Parameter	Findings
6.2.1	WLAN bandwidth	<ul style="list-style-type: none"> - Increased bandwidth leads to decreased communication time, hence reduced service time – this agrees with the investigation performed by [6]. - The bandwidth values used were sufficient for the scenario hence no bottleneck occurred – confirmed by [5] and [9]. As the effect is minor, complementing it with another parameter will provide better performance - [6] showed an improved performance of the architecture by combining the WLAN bandwidth with increased VM processing speed. - EO architecture suffered from the highest network delay, but the overall service time was complemented by reduced processing time due to distributed processing power.
6.2.2	VM processing speed	<ul style="list-style-type: none"> - Increased VM processing speed leads to better utilisation of VMs, hence reduced service time – this agrees with [5], [12], [19]. - Increased VM processing speed also leads to less failure due to VM capacity. Therefore it achieves a reduced percentage of failed tasks – this is agreed by [6]. - EO architecture's quality of service remained stable with the increasing load condition.

6.2.3	Number of edge servers	<ul style="list-style-type: none"> - Increased number of edge servers provides additional computing resources. There exists no more congestion leading to a reduced percentage of failed tasks and processing time – is studied in [6], [10], [19], which showed a performance improvement as the number increased. - EO architecture provided the best performance. However, when the available edge servers were low, it faced a computational burden, leading to performance degradation.
6.2.4	VM capacity	<ul style="list-style-type: none"> - Increased VM capacity had little effect on the performance due to the initial value's sufficiency. Could be combined with different parameters to maximise the advantage of large VM capacity. - EO architecture performed especially well in terms of a failed task. The processing time remained reasonably low, although it showed an increasing trend as the load increases.

6.3.2 Hypotheses Verification

The hypothesis stated in section 4.1 will be verified based on the findings of the experiment.

Table 6. Hypotheses verification

Number	Hypothesis	Verification
Hypothesis 1	Increase in WLAN bandwidth and VM processing speed reduces the service time.	<ul style="list-style-type: none"> - Increasing the bandwidth lead to the improvement of service time, although the reduction was not significant. - Increasing the VM processing speed contributed to reducing the service time to a great extent. - Both parameters have proven to be effective in reducing service time. Therefore, the hypothesis holds true for this experiment.
Hypothesis 2	Increase in the number of edge servers and VM capacity reduces the percentage of failed tasks.	<ul style="list-style-type: none"> - Increasing the number of edge servers significantly reduced the percentage of failed tasks, validating hypothesis 2. - VM capacity did not affect the percentage of failed tasks, disproving hypothesis 2. - The results partially confirm hypothesis 2.

6.3.3 Significant Parameters

To summarise, EdgeCloudSim simulator was used to investigate the effect of computational and networking system parameters on the scalability of application performance. Four different parameters have been investigated in this research – **1) WLAN bandwidth, 2) VM processing speed, 3) number of edge servers** and **4) VM capacity**. Three main performance metrics selected in section 4.3 were the percentage of failed tasks and service time.

The most apparent performance improvement in terms of both percentage of failed tasks and service time was achieved by increasing the **VM processing speed**, followed by the **number of edge servers**. In other words, increasing the computational power at the network edge through faster processing and low utilisation allowed for smoother and accurate processing of tasks.

On the other hand, the WLAN bandwidth and VM capacity had a minor impact on application performance. Although the WLAN bandwidth improved the network delay to some extent, as the percentage that network delay accounts for the total service time is small, it had little effect on the overall results. Furthermore, increasing the VM capacity by itself did not improve the failure rate nor the service time.

From a further investigation conducted to identify the parameter to be combined, it was determined that experiment 6.2.1, which varied the WLAN bandwidth, had the highest rate of failure due to VM capacity. This suggests that combining the **VM capacity and WLAN bandwidth** could fulfil the shortages of each other, improving the performance synergistically.

Increased WLAN bandwidth would lead to more tasks arriving at the edge server at a faster rate, allowing for the full exploitation of extra capacity the VM has. The edge servers will be able to handle a greater volume of tasks without the congestion in the node, providing a better quality of service in terms of both the service time and the percentage of failure.

6.4 Recommendations

1. Importance of EO

In general, using the two-tier with EO architecture always provides better performance, as it has the ability to balance the increasing workload. Hence, it is recommended that this architecture is utilised for the application to scale. However, it is recommended that two-tier architecture without the EO is utilised in the following scenarios:

- If the available WLAN bandwidth is small, the network delay that incurs when orchestrating the load might affect the performance. Hence, it is recommended not to

utilise the load-balancing scheme. The simulation experiment identified it to be 100 Mbps.

- If the number of edge servers is exceptionally low, to minimise communication overhead on already congested edge nodes. The simulation experiment identified it to be 2 servers.

2. Devising the Edge Architecture

When implementing the edge architecture, it is important to consider the characteristics of the application to configure the resources accordingly. Less latency and failed tasks are essentially achieved by having more resources. In other words, there exists a trade-off between available resources and the performance of the application. Therefore, depending on the application requirements and characteristics such as task size, the edge deployment and task offloading scheme should be considered carefully.

For applications with short dwell time per location, such as driverless cars and aircraft control, the service time is extremely important. Therefore, the focus should be on VM processing speed, which identified to have a significant impact on achieving reduced service time. Moreover, if the task load is heavy but there is not enough resource at the edge, it might be more valuable to handle the analysis centrally in the cloud [30] – which would give much more accuracy but comes with a slight latency.

3. Management of Edge Nodes

Because the project environment was simulated, it was assumed that the edge servers were not prone to failures. However, if the edge servers are up and running, various factors can influence the edge server availability, which is crucial for application performance. Due to the heterogeneity and distributed nature of the edge servers, they are not as well maintained as the centralised cloud datacentre [19]. The lack of maintenance could mean that the edge is prone to failures, which severely impacts the availability of edge servers. Hence, the management of the nodes through software updates rather than hardware is recommended.

7. Project evaluation

This chapter conducts an evaluation of the project as a whole, reflecting on the project plan that was devised in the introduction chapter. It includes related works and future recommendations as well.

7.1 Evaluation

7.1.1 Aims and Objectives

1. Identify the key issues in relation to edge computing performance.

Based on a detailed review of edge computing literature, important parameters for each of single-tier, two-tier and two-tier with EO architectures have been identified. The relationship between the parameters and the performance was proposed, which was further strengthened by an initial experiment that revealed 4 key parameters specific to the scenario used.

2. Design and implement different experiments through the configuration of key parameters.

A suite of experiment was designed based on the key parameters identified from the previous objective and was implemented iteratively through EdgeCloudSim.

3. Analyse and interpret the performance results.

Based on a graphical analysis of the simulation results, observed trends were explained and justified. Conclusions were drawn, and hypotheses were verified, fulfilling the objective.

4. Propose further recommendations on the architecture of edge computing applications.

The project identified significant parameters in relation to scalability and proposed a combination of parameters for better performance. However, it lacked to provide configuration values that are feasible and applicable in real-life, as it greatly depends on the application used and resources available. It further proposed a desirable architecture for different scenarios and provided further insight into the edge server management, widening the context of the research.

The investigation met the vast majority of the project objectives through an empirical investigation. By achieving these objectives, the aim of the experiment which was to investigate the effect of computational and networking system parameters on the application performance results, have been fulfilled.

7.1.2 Project Management

In addition to evaluating the success of the project aims and objectives, the project can also be evaluated in terms of the methodology used and project management.

An iterative methodology of repeating the experimental design, implementation and data collection, and analysis and interpretation was adopted for the study. Numerous revisions have been made in the design of the experiments to find a set of parameters that provide distinguishable performance results, thanks to the repeatable nature of the simulation. Based on the analysis of the results, extended experiments with different configurations were implemented to seek further inference of the phenomenon. For example, an experiment that showed little or no effect had been rerun with an extreme configuration value to observe for any changes or justification.

In terms of the technology used, EdgeCloudSim was utilised to conduct various simulations. It has addressed the needs of parameters and evaluation metrics used for the experiment and allowed for ease of producing graphical results.

Furthermore, the timeline shown in the Gantt chart (figure 1) was not strictly adhered to. The experiment design took longer than expected as it was constantly revisited to make improvements based on the experiment results. The design, implementation and analysis took place simultaneously and iteratively, in which the project benefited from the nature of an agile methodology allowing for an adaptive timeline.

7.1.3 Limitations

There exists a range of limitations related to the research problem, that has been identified, which are as follows:

1. The number of mobile devices used in the analysis was limited. Although it provided with some significant relationships from the data, it would have been more favourable if a larger number of mobile devices were used, to ensure the validity of generalisation when proving for scalability. Researching and configuring the values based on the estimated number of devices used for real IoT application could have contributed practical importance of the research.
2. More specific hypotheses could have been formulated in order to test and verify the findings more accurately. The current hypotheses combine two parameters each, leading to a formulation of less focused statements to investigate for.
3. The number of parameters and the values investigated was limited due to the time constraint of the research. Additional performance evaluation of the combination of parameters could have given more confidence to the claim made in 6.3.1, which is now formulated as a suggestion.

4. Some anomalies and unexpected behaviour concerning the two-tier with EO architecture could not be justified, due to the lack of understanding in the load balancing scheme. It could have been an important indication that is significant for the performance.

7.2 Related work

This section places the presented work in line with related work to identify the importance of the study within the field of investigation. The detailed positioning of the work and comparison with prior research is presented in appendix C.

7.2.1 Prior Work

The evaluation of the edge computing architectures has been conducted, as the subject is still on its developing stage. Numerous parameters need further investigation, hence extending the research and validating related works had shed some light on the better understanding of the performance of different edge architectures.

Although extensive work has been done for the generic idea of edge computing paradigm and scenarios of the application deployment, not a lot of simulation-based experiments have considered all three architectures of edge only, edge and cloud, and edge and cloud with an orchestrator. The study evaluates the varying performance of the application based on the parameter configuration, with consideration of the architecture and its scalability.

The performance evaluation in [6], [19] utilises the closest methodology, as it investigates the impact of increasing one parameter value at a time. This allows for the clear visualisation of the trend, as it isolates the parameter being tested from other factors that might affect the performance.

Essentially, the project contributes to the research in the field by exploring the impact of more parameters with various configuration values. In some cases, the parameters are combined to maximise the performance, suggesting possible considerations when designing such systems.

7.2.2 Future work

Future works that could further enhance the study of edge computing include:

1. **Further investigation into the load balancing scheme** - EO architecture's load balancing scheme could be investigated further by studying the effect of task migration between the edge servers, focusing on the network delay incurring from the orchestration.

2. **Extended experiments** – More experiments could be conducted using parameters that have not been explored in this report. Combination of parameters could be used to further identify the relationship and trade-offs between the parameters.
3. **Real-life implementation of the simulation** – The simulation could be deployed on a real experimental lab with appropriate resources, to test the application on physical datacentres and to compare the simulation results to real-life values.
4. **Consideration of various applications** – The performance evaluation carried out is highly related to specific application scenario used. Modelling of different applications is desired as they have different requirements, such as task size, dwell time, and the number of devices. Therefore, this will help to compare the performance of various application scenarios, enriching the understanding of edge computing and the deployment of IoT services.

7.3 Personal Reflection

The project started off as somewhat broad and vague, as I could not land myself a clear goal and an application scenario for the simulation. As I conducted more literature review and experiments, the concept consolidated, and I deeply delved into the project. The project's aim and goals slowly came together, which provided me with strong guidance and confidence in proceeding the research. Generally, the time was managed well, and milestones were achieved by setting up personal deadlines, helping me to stay on track and focused.

There were some unexpected technical challenges of using the simulation software. Due to the lack of documentation of EdgeCloudSim, to understand some features and construction, various attempts of source code inspection and modification were made. Moreover, MATLAB errors and failure of graph production was especially frustrating, as there were no troubleshooting guide or pre-existing solutions. By trial and error, proper error handling code had been added, which produced successful plots to be used for the project. The technical difficulties incurred in a slight delay of the experiment implementation, which was not desirable, but was handleable.

Upon the completion of the project, I have learned how to critically approach a research problem, in terms of both literature review and technical implementation. It has given me the confidence and skills for future research and insight into career plans. The opportunity to fully engage in a research project of my own interest was a grateful and rewarding experience, which will be valuable for my future endeavours.

7.4 Legal, Ethical, Social and Professional Issues

This section covers the legal, ethical, social and professional issues that might arise throughout the span of the project.

Legal Issues

The project involves EdgeCloudSim simulator, which is licensed under the GNU General Public License v3.0 [31]. The license states the freedom to share and modify the work, and that it remains free software to all users meaning that the same freedom is given to the recipients of the software. Copyright and license notices have been preserved in the repository, meeting the legal issues.

Ethical Issues

The project implementation does not involve any external data, as it is delivered through a simulator which eliminates the factor of human interaction. Sensitive data such as images of people, medical images or religious texts were not used, and no external data was analysed. However, when deployed in real-life, the storage and processing of the sensor data must be addressed ethically and responsibly.

Social Issues

The project was developed for academic and experimental purposes. However, the implications of edge computing, such as the reduction of energy consumption could have a great impact on building a more sustainable society.

Professional Issues

The project strived to achieve high quality in both the process and results of the work, following an adequate ethical practice. It took precautions to make reliable judgement and analysis throughout the process, which complies with the ACM code of ethics and professional conduct [32].

8. Conclusion

The project has thoroughly examined the performance of various edge computing architectures based on the computational and networking system parameters, in relation to the scalability of a face recognition application. Series of experiments were designed and implemented to test the effect of chosen parameters against the service time and percentage of failed tasks, demonstrating edge computing's capabilities.

The experiment results demonstrated that the best performance could be achieved by utilising the two-tier with EO architecture with high VM processing speed and many edge servers. The performance evaluation in relation to scalability serves for the rapidly growing nature of the data generated by the IoT devices. Hence, the interpreted simulation results will play an important role in the deployment of future IoT application design, allowing for a more effective and efficient service provision.

To conclude, the project has contributed to the extended research of the performance of edge computing domain by simulating different models, thereby identifying the ideal architecture.

List of References

- [1] H. Gupta, A. Vahid dastjerdi, S. K. Ghosh, and R. Buyya, '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*, vol. 47, no. 9, pp. 1275–1296, 2017, doi: 10.1002/spe.2509.
- [2] G. Premsankar, M. Di Francesco, and T. Taleb, 'Edge Computing for the Internet of Things: A Case Study', *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018, doi: 10.1109/JIOT.2018.2805263.
- [3] W. Shi and S. Dustdar, 'The Promise of Edge Computing', *Computer*, vol. 49, no. 5, pp. 78–81, May 2016, doi: 10.1109/MC.2016.145.
- [4] C. Sharma, 'Edge Computing Framework', p. 28, 2019.
- [5] C. Sonmez, A. Ozgovde, and C. Ersoy, 'EdgeCloudSim: An environment for performance evaluation of Edge Computing systems', in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 39–44, doi: 10.1109/FMEC.2017.7946405.
- [6] A. F. Aljulayfi and K. (orcid:- Djemame, 'Simulation of an augmented reality application for driverless cars in an edge computing environment', 2019.
- [7] G. Dodig-Crnkovic, 'Scientific Methods in Computer Science', p. 16.
- [8] T. P. Morris, I. R. White, and M. J. Crowther, 'Using simulation studies to evaluate statistical methods', *Statistics in Medicine*, vol. 38, no. 11, pp. 2074–2102, 2019, doi: 10.1002/sim.8086.
- [9] D. N. Jha *et al.*, 'IoT-Sim-Edge: A simulation framework for modeling the behavior of Internet of Things and edge computing environments', *Software: Practice and Experience*, vol. n/a, no. n/a, doi: 10.1002/spe.2787.
- [10] 'EdgeCloudSim: An environment for performance evaluation of edge computing systems - Sonmez - 2018 - Transactions on Emerging Telecommunications Technologies - Wiley Online Library'. <https://onlinelibrary.wiley.com/doi/10.1002/ett.3493> (accessed Feb. 21, 2020).
- [11] P. Mell and T. Grance, 'The NIST Definition of Cloud Computing', National Institute of Standards and Technology, NIST Special Publication (SP) 800-145, Sep. 2011. doi: <https://doi.org/10.6028/NIST.SP.800-145>.
- [12] Y. Liu, M. J. Lee, and Y. Zheng, 'Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System', *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2398–2410, Oct. 2016, doi: 10.1109/TMC.2015.2504091.
- [13] M. Salama, Y. Elkhatib, and G. S. Blair, 'IoTNetSim: A Modelling and Simulation Platform for End-to-End IoT Services and Networking', *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, pp. 251–261, Dec. 2019, doi: 10.1145/3344341.3368820.
- [14] M. Chiang and T. Zhang, 'Fog and IoT: An Overview of Research Opportunities', *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec. 2016, doi: 10.1109/JIOT.2016.2584538.
- [15] W. Yu *et al.*, 'A Survey on the Edge Computing for the Internet of Things', *IEEE Access*, vol. 6, pp. 6900–6919, 2018, doi: 10.1109/ACCESS.2017.2778504.
- [16] M. Satyanarayanan, 'The Emergence of Edge Computing', *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017, doi: 10.1109/MC.2017.9.
- [17] P. Garcia Lopez *et al.*, 'Edge-centric Computing: Vision and Challenges', *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, Sep. 2015, doi: 10.1145/2831347.2831354.
- [18] C. Sonmez, A. Ozgovde, and C. Ersoy, 'Performance evaluation of single-tier and two-tier cloudlet assisted applications', in *2017 IEEE International Conference on*

- Communications Workshops (ICC Workshops)*, May 2017, pp. 302–307, doi: 10.1109/ICCW.2017.7962674.
- [19] S. Suryavansh, C. Bothra, M. Chiang, C. Peng, and S. Bagchi, 'Tango of edge and cloud execution for reliability', in *MECC '19*, 2019, doi: 10.1145/3366614.3368103.
- [20] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, 'The Case for VM-Based Cloudlets in Mobile Computing', *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009, doi: 10.1109/MPRV.2009.82.
- [21] P. Michalák and P. Watson, 'PATH2iot: A Holistic, Distributed Stream Processing System', in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec. 2017, pp. 25–32, doi: 10.1109/CloudCom.2017.35.
- [22] C.-K. Tham and R. Chattopadhyay, 'A load balancing scheme for sensing and analytics on a mobile edge computing network', in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Jun. 2017, pp. 1–9, doi: 10.1109/WoWMoM.2017.7974307.
- [23] T. H. Noor, S. Zeadally, A. Alfazi, and Q. Z. Sheng, 'Mobile cloud computing: Challenges and future research directions', *Journal of Network and Computer Applications*, vol. 115, pp. 70–85, Aug. 2018, doi: 10.1016/j.jnca.2018.04.018.
- [24] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan, 'IOTSim: A simulator for analysing IoT applications', *Journal of Systems Architecture*, vol. 72, pp. 93–107, Jan. 2017, doi: 10.1016/j.sysarc.2016.06.008.
- [25] T. Qayyum, A. W. Malik, M. A. Khan Khattak, O. Khalid, and S. U. Khan, 'FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment', *IEEE Access*, vol. 6, pp. 63570–63583, 2018, doi: 10.1109/ACCESS.2018.2877696.
- [26] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, 'CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms', *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011, doi: 10.1002/spe.995.
- [27] M. van Steen and A. S. Tanenbaum, *Distributed systems*, Third edition (Version 3.01 (2017)). London: Pearson Education, 2017.
- [28] N. Muslim and S. Islam, 'Face recognition in the Edge Cloud', in *ICISPC 2017*, 2017, doi: 10.1145/3132300.3132310.
- [29] M. Waqas, Y. Niu, M. Ahmed, Y. Li, D. Jin, and Z. Han, 'Mobility-Aware Fog Computing in Dynamic Environments: Understandings and Implementation', *IEEE Access*, vol. 7, pp. 38867–38879, 2019, doi: 10.1109/ACCESS.2018.2883662.
- [30] N. Heath, 'What you need to know before implementing edge computing', *ZDNet*. <https://www.zdnet.com/article/what-you-need-to-know-before-implementing-edge-computing/> (accessed May 02, 2020).
- [31] C. Sonmez, *CagataySonmez/EdgeCloudSim*. 2020.
- [32] D. Gotterbarn, 'The Code affirms an obligation of computing professionals to use their skills for the benefit of society.', *ACM*. <https://www.acm.org/code-of-ethics> (accessed May 03, 2020).

Appendix A. External Material

EdgeCloudSim simulator developed by Sonmez et al. was used for the project, and can be accessed via:

<https://github.com/chaerim-kim/EdgeCloudSim>

To compile the application:

```
./compile.sh
```

1. To run the default configuration singly:

```
./runner.sh out_folder default_config edge_devices.xml applications.xml  $\alpha$ 
```

- ./runner.sh to run the shell script
- out_folder to define a folder for simulation result to be outputted to
- edge_devices.xml to define edge devices file to be used
- applications.xml to define application file to be used
- α to set the iteration number

2. Or to run the simulation in parallel:

```
./run_scenarios.sh  $\alpha$   $\beta$ 
```

- ./run_scenarios.sh takes the runner.sh and runs several iterations in parallel
- α defines the number of processors
- β defines the number of iterations to be performed

The simulator outputs the results of 5 different iterations as can be seen in Figure 1, where 'ite.log' files are provided as a human-readable log of the simulation results, and files in folder 'ite'n' to be fed to MATLAB for plot generation, as seen in Figure 2.

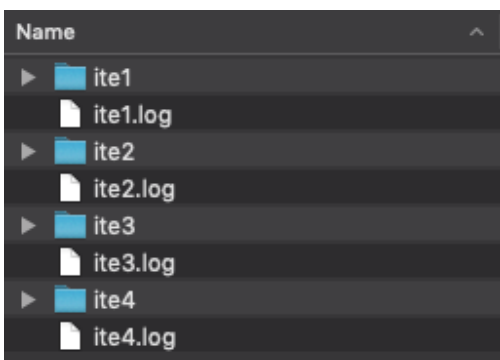


Figure 1. Simulation output

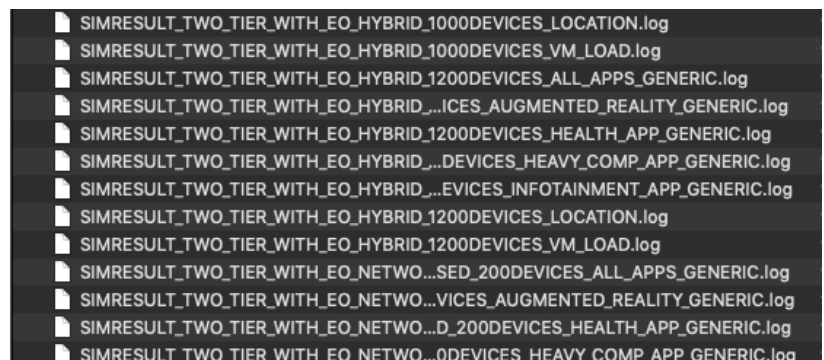


Figure 2. Simulation results to be fed into MATLAB

Appendix B. Literature Review – Summary

- 1 – single-tier architecture
- 2.1 – two-tier architecture
- 2.2 – two-tier with EO architecture

Table 1. The positioning of the project

Author	Simulator	Application	Metrics	Architecture		
				1	2.1	2.2
This report	EdgeCloudSim	- Augmented reality - face recognition	- Average service time - Average processing time - Average network delay - Percentage of failed tasks - Average VM utilisation	X	X	X
Sonmez et al. [18]	EdgeCloudSim	- Augmented reality - face recognition	- Percentage of failed tasks - Average service time	X	X	X
Aljulafi and Djemame [6]	EdgeCloudSim	- Augmented reality – driverless cars	- Percentage of failed tasks - Average network delay - Average processing time		X	X
Suryavansh et al. [19]	EdgeCloudSim	- Augmented reality application	- Percentage of failed tasks - Average service time	X	X	X
Gupta et al. [1]	iFogSim	- Latency sensitive online game - Intelligent surveillance through distributed camera networks	- Average latency - Network use - Energy consumption	X	X	
Jha et al. [9]	IoTsim-Edge	- Healthcare system - Smart buildings - Self-driving cars	- Average latency - Energy consumption - Average service time	X	X	X
PremSankar et al. [2]	GamingAnywhere cloud gaming platform	- Virtual and augmented reality – Neverball, a 3D arcade game	- Response delay (Processing delay + playout delay + network delay)	X		

Appendix C. Initial Gantt Chart

The original Gantt chart designed in October is illustrated below.

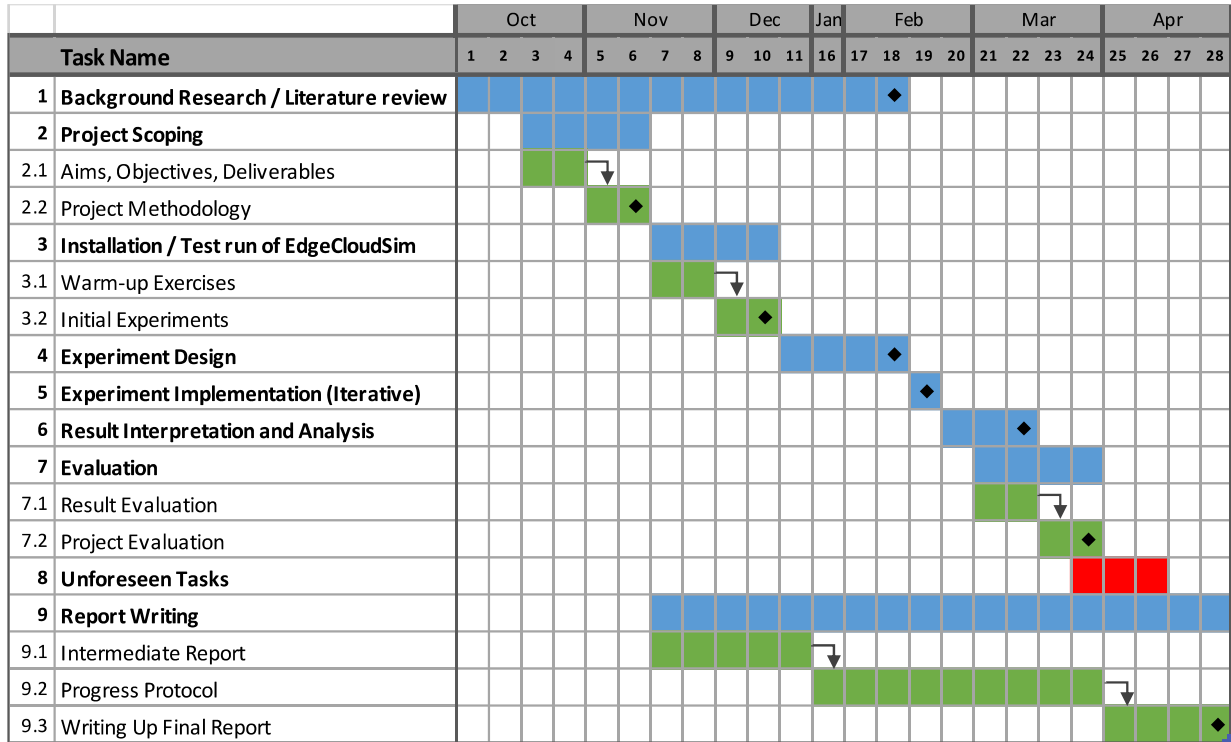
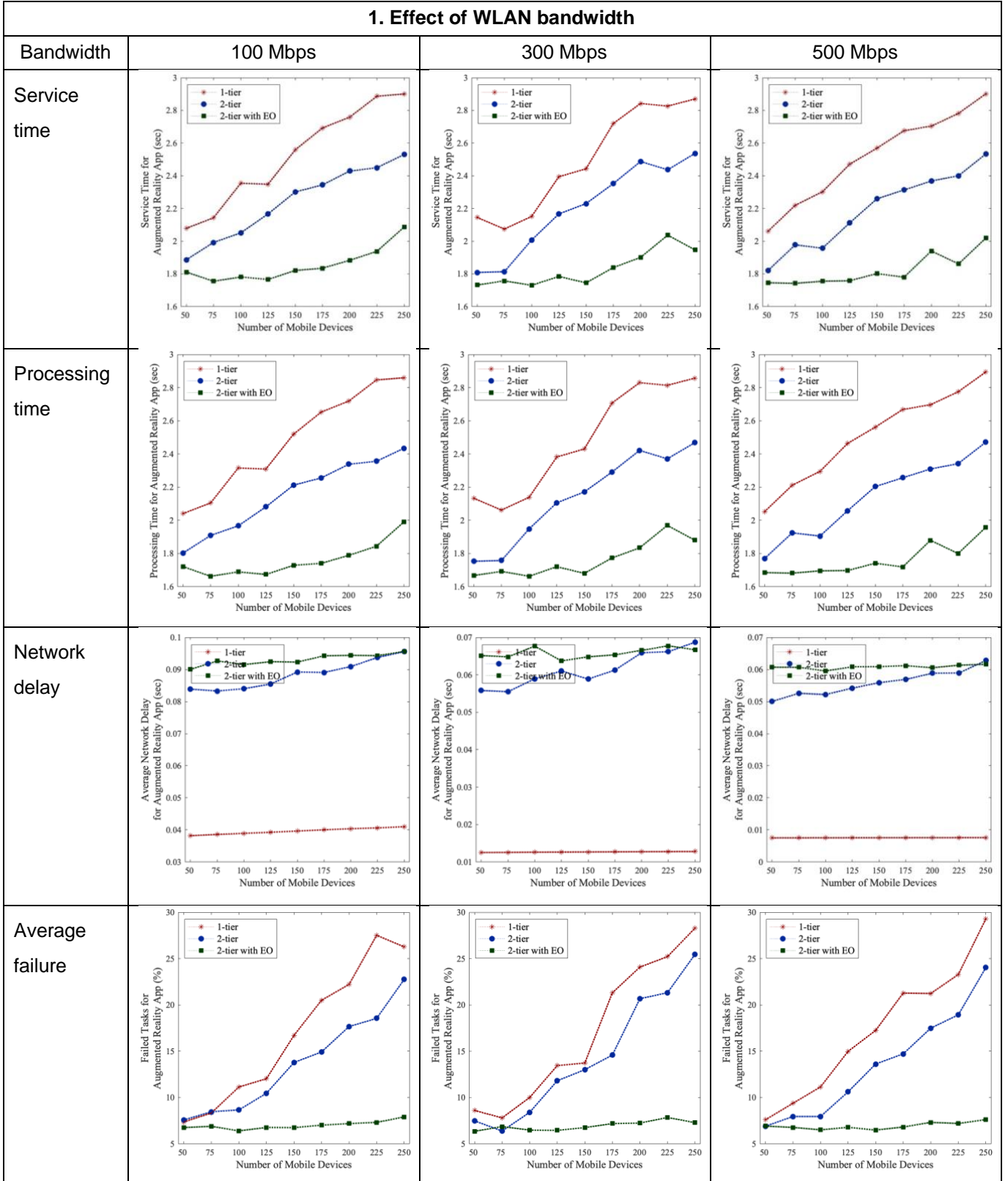
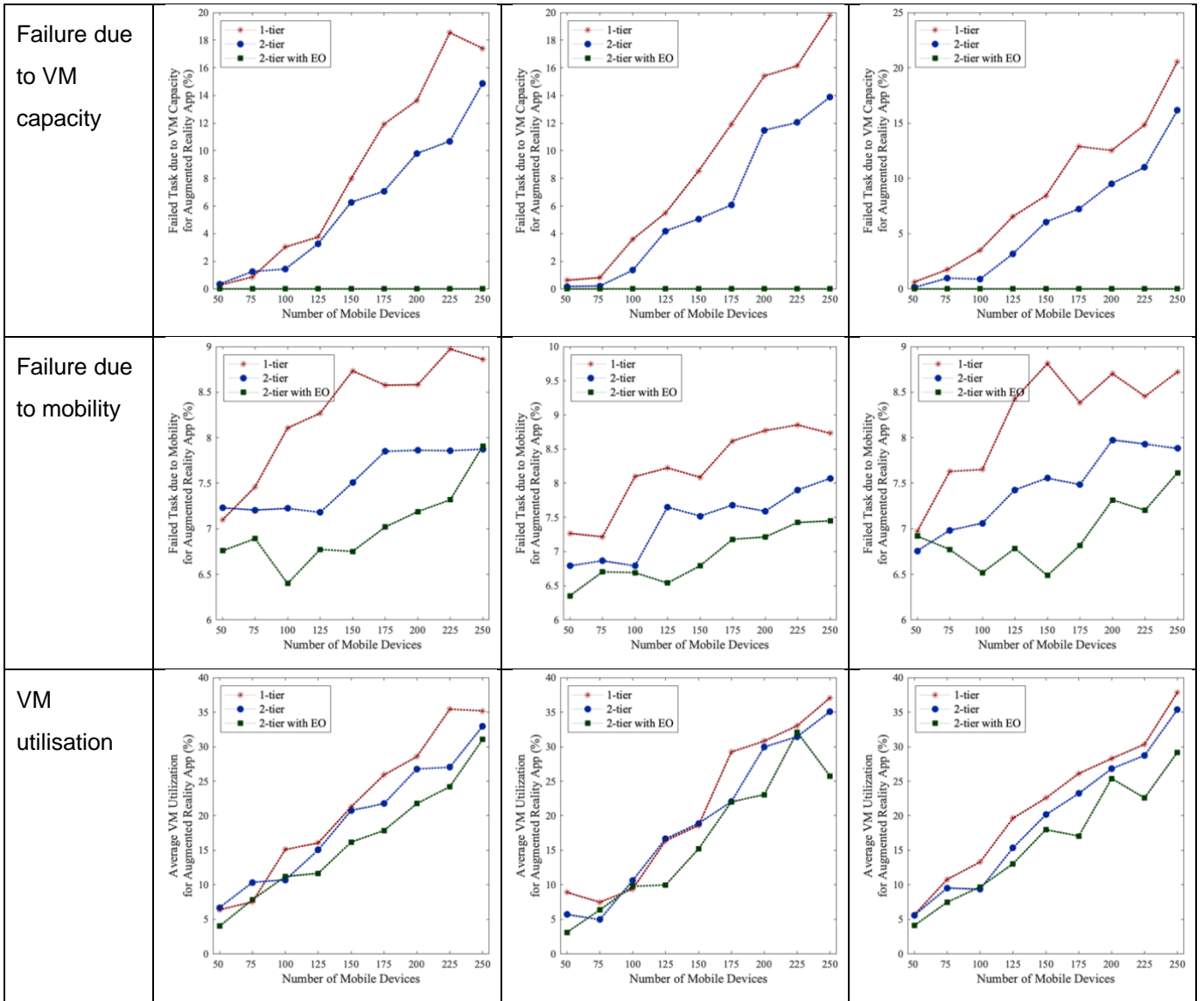


Figure 3. Original Gantt Chart

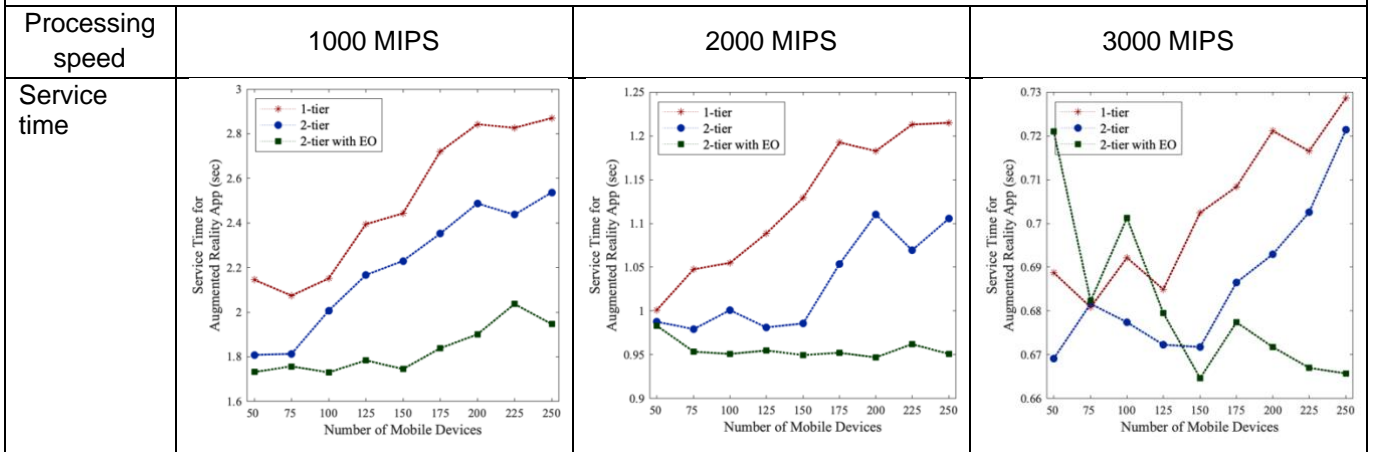
Appendix D. Full Simulation Results

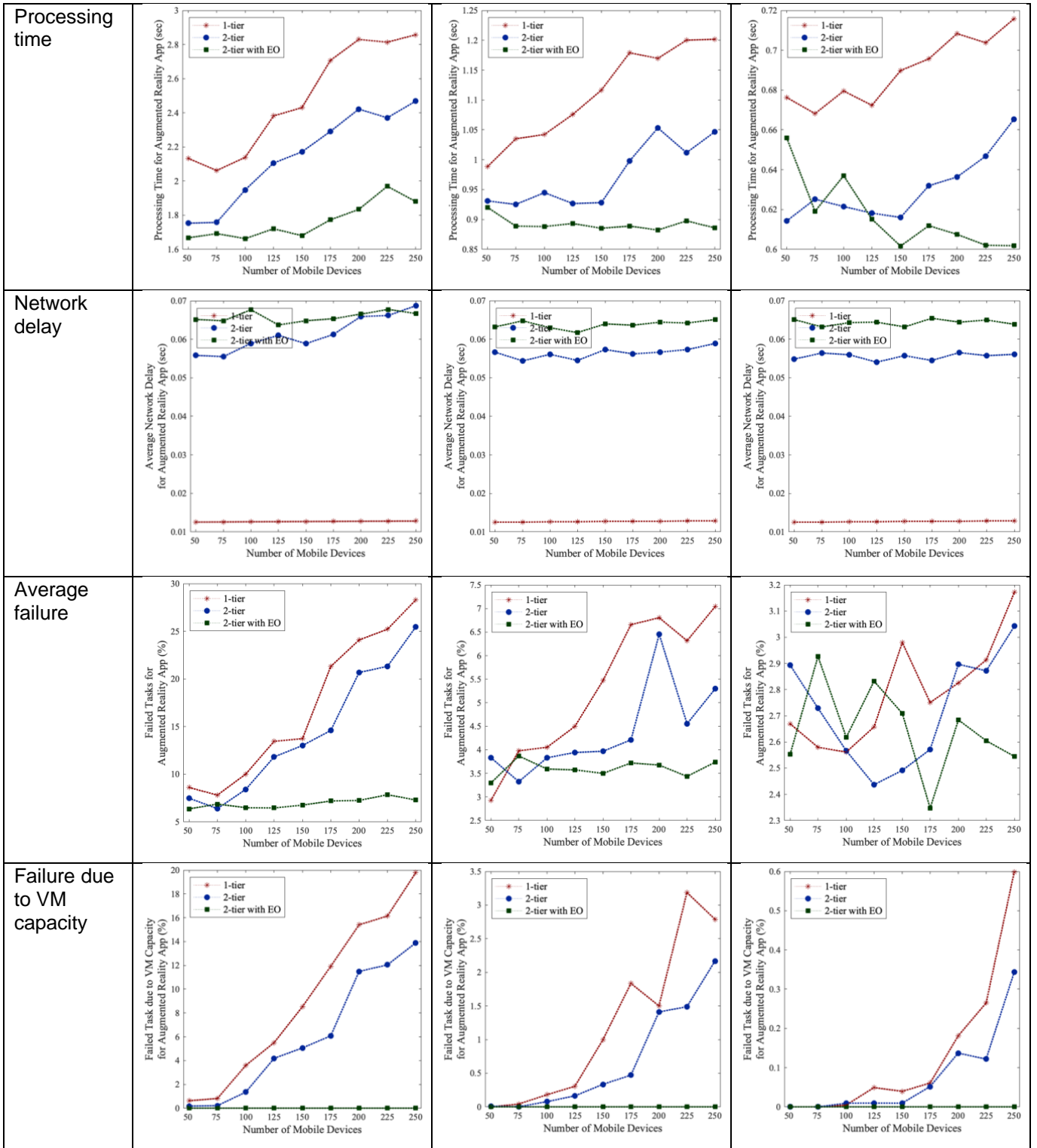
1. Effect of WLAN bandwidth

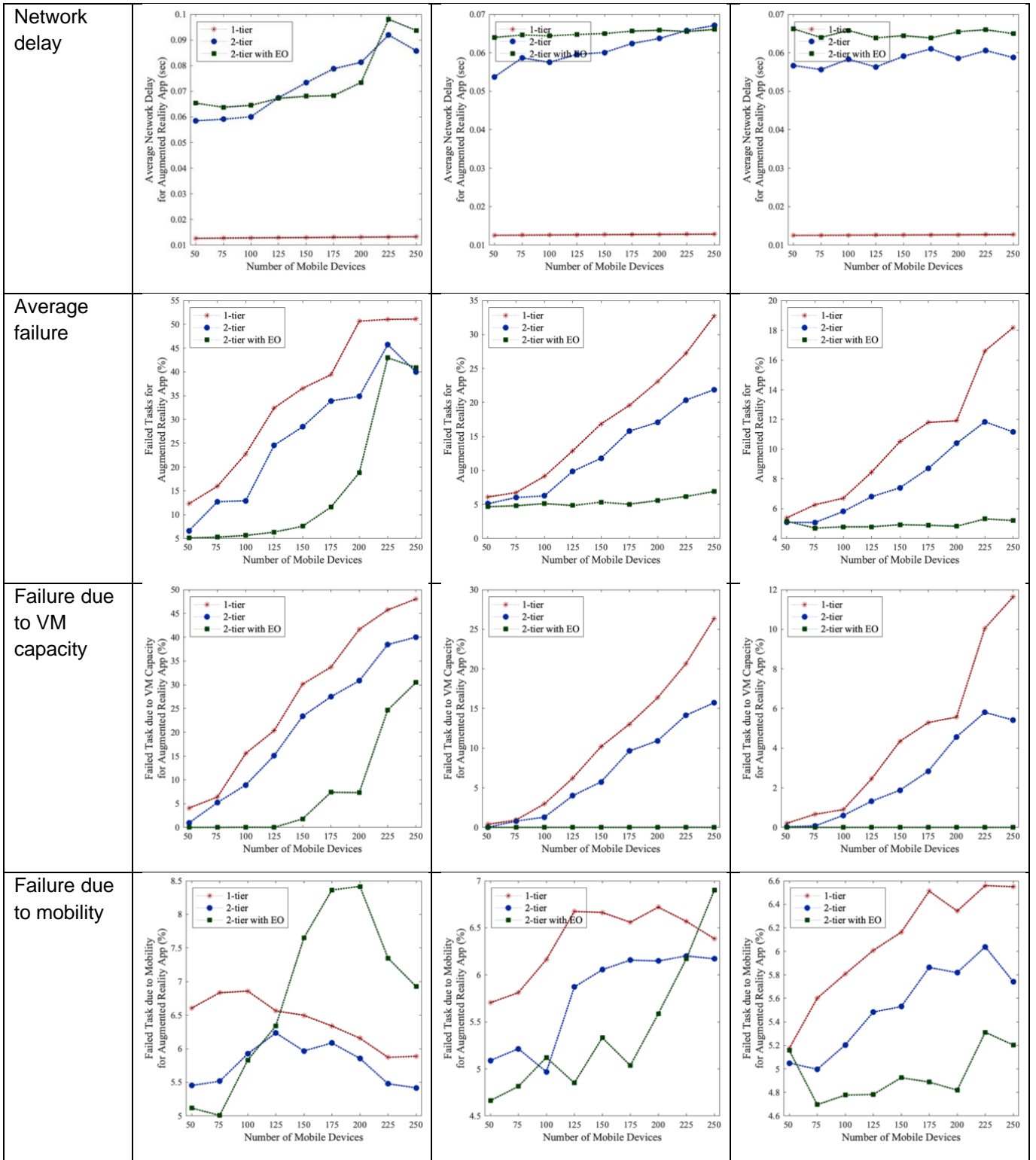


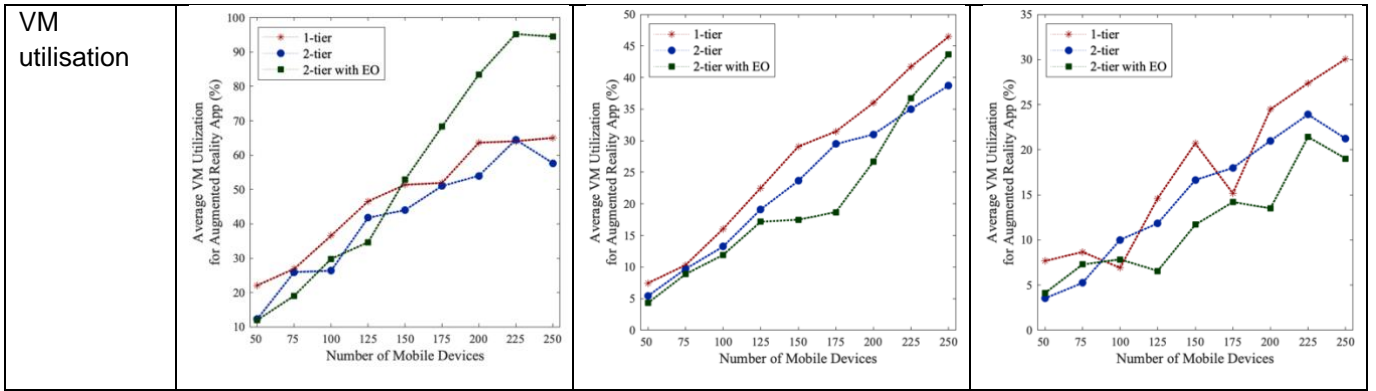


2. Effect of VM processing speed









4. Effect of VM capacity

