ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

An Adaptive Offloading Decision Scheme in Two-Class Mobile Edge Computing Systems

M.Sc. THESIS

Kahlan HASAN

Informatics Institute

Computer Science Program

JUNE 2018

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

An Adaptive Offloading Decision Scheme in Two-Class Mobile Edge Computing Systems

M.Sc. THESIS

Kahlan HASAN
(704161009)

Informatics Institute

Computer Science Program

Thesis Advisor: Asst. Prof. Dr. Mehmet Akif YAZICI

JUNE 2018

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

İki-Sınıflı Mobil Kenar Bilişim Sistemleri için Uyarlanır Bir Aktarma Karar Yöntemi

YÜKSEK LİSANS TEZİ

Kahlan HASAN
(704161009)

Bilişim Enistitüsü

Bilgisayar Bilimleri

Tez Danışmanı: Asst. Prof. Dr. Mehmet Akif YAZICI

HAZIRAN 2018

Kahlan HASAN, a M.Sc. student of ITU Informatics Institute Engineering and Technology 704161009 successfully defended the thesis entitled "An Adaptive Offloading Decision Scheme in Two-Class Mobile Edge Computing Systems", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Dr. Mehmet Akif YAZICI** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Lütfiye DURAK ATA** Istanbul Technical University	.....
	Asst. Prof. Dr. Kasım ÖZTOPRAK KTO Karatay University	.....
	Asst. Prof. Dr. Mehmet Akif YAZICI Istanbul Technical University	.....

Date of Submission : **04 May 2018**
Date of Defense : **06 June 2018**

To my family, whose love is the fuel of my life

FOREWORD

In the name of ALLAH the most merciful and compassionate
Praise be to ALLAH foremost and last Who gave me strength, determination, and patience to accomplish this work. To My Family I say thank you all for the endless support and love, Mother and Father you are the only stars I have in my life's sky (*I LOVE YOU ALL*). To all of my friends with gladly mentioning Şen FERHAT, Maryam MAJIDI (SHIMA), and Gregorio SPINELLI, I am Happy to have you in my life and thank you for your help and support throughout my study. Intentionally mentioning my eldest brother and advisor Asst. Prof. Dr. Mehmet Akif YAZICI lastly so as to dedicate all of my appreciation and sincere thankfulness for his great effort and support along with his priceless guidance and motivation. Indeed, my words are ashamed of being incapable of expressing how happy I am to be your student. From you I learned countless things, without your advising and recommendations I would never hope to achieve what I have achieved , THANK YOU.

JUNE 2018 Kahlan HASAN

TABLE OF CONTENTS

Page

ABBREVIATIONS

CC	**:** Cloud Computing
SC	**:** Small Cell
MCC	**:** Mobile Cloud Computing
ETSI	**:** European Telecommunications Standard Institute
MEC	**:** Mobile Edge Computing
BS	**:** Base Station
AP	**:** Access Point
HN	**:** Heterogeneous Network
MS	**:** Macro Station
MiS	**:** Micro Station
PiS	**:** Pico Station
FS	**:** Femto Station
SP	**:** Service Provider
NFV	**:** Network Function Virtualization
ICN	**:** Information Centric Network
SDN	**:** Software Defined Network
SP	**:** Signal Processing
ML	**:** Machine Learning
AR	**:** Augmented Reality
IoT	**:** Internet of Things
HODA	**:** Heuristic Offloading Decision Algorithm
QoS	**:** Quality of Service
EECO	**:** Energy Efficient Computation Offloading
WRR	**:** Weighted Round Robin
CPU	**:** Central Processing Unit
GPS	**:** Generalized Processor Sharing
MC	**:** Machine Cycle
IT	**:** Information Technology
FIFO	**:** First-In First-Out
FCFS	**:** First Come First Served
PS	**:** Processor Sharing
RR	**:**Round Robin
HU	**:** High Priority User
LU	**:** Low Priority User

SYMBOLS

λ_H	: Arrival rate of high priority task per user
λ_L	: Arrival rate of low priority task per user
L	: Task size
X	: Number of required MC per task
f_m	: Mobile's CPU frequency
f_s	: Server's CPU frequency
n_c	: Number of cores in server's CPU
s	: Speed-up factor
r	: Data rate
h	: Ratio of high service to low service
α	: Encoding rate
θ	: Damped averaging parameter
T_l	: Sojourn time of local execution
T_o	: Sojourn time of remote execution

LIST OF TABLES

Page

LIST OF FIGURES

Page

An Adaptive Offloading Decision Scheme in Two-Class Mobile Edge Computing Systems

SUMMARY

With the huge growth of data exchange and the increasing number of connected devices to the Internet, Mobile Cloud Computing (MCC) paradigm with its centralized approach will face tremendous loads. Therefore; the European Telecommunication Standard Institute (ETSI) came up with a new idea to overcome the problem of latency mainly and decrease the consumed energy of transmission. This new approach, Mobile Edge Computing (MEC), is proposed to use the capacity at the edge of the network such as Base Station (BS) or Access Point (AP). In this way, the amount of sent data to the cloud will be significantly reduced by having the edge server executing all the offloaded tasks on behalf of mobile users.

In mobile edge computing, the system can be modelled as a single-class MEC system which considers all the tasks as the same (no differentiation in terms of priorities), or a multi-class system. In a multi-class system, the tasks originating from different mobiles may have different priorities. In case of a two-class system, there will be high and low priorities.

In this study, we investigate the offloading problem in the presence of two user classes: one is high priority and the other is low priority. The tasks are generated by mobile users and are offloaded to the edge server to be executed. We assume that the server is in charge of making offloading decisions. A task is decided to be offloaded or not based on the comparison between sojourn time in case of local execution (at the mobile device) and sojourn time in case of remote execution (at the edge server). The task is offloaded if the sojourn time in case of remote execution is smaller.

We assume that the edge server employs weighted Round-Robin (WRR) processor scheduling, which can be modeled as a priority Proessor Sharing (PS) queue if the time slots are considered to be small. (WRR) uses the same principle of Round Robin, which is basically sharing the CPU service among all the packets for a specific number of time slots (Machine Cycles (MC) in case of CPU scheduling) without considering the priorities of arrived tasks. On the other hand, WRR takes into account the priorities of the tasks. In WRR, a high priority task is served more frequently than others. To put it another way, a high priority task receives more MCs than lower priority tasks. It is known that the expected sojourn time in a PS queue is linearly proportional to the task size. Therefore; we use a factor (queueing delay multiplier) that accurately captures the sojourn time of an offloaded task. The queueing delay multiplier both models and affects the queueing delay, and thus acts as an admission rule parameter. If the multiplier is too high, tasks are discouraged to offload, resulting in a lightly loaded MEC server which entails a low multiplier value. On the other hand, a too low value for the multiplier will result in more offloads, yielding high queueing delays. Therefore, there should be an optimal queueing delay multiplier value that balances the demand for the MEC server.

We propose an adaptive scheme that finds an optimum value for the queueing delay multiplier on the fly, using damped averaging. We show that high priority users experience much lower average sojourn times compared to the low priority users. We also illustrate the effect of our method on the energy consumption of the mobile. Using a stand-alone simulator, we demonstrate the performance of the proposed method in several different scenarios with numerical experimentation.

İki-Sınıflı Mobil Kenar Bilişim Sistemleri için
Uyarlanır Bir Aktarma Karar Yöntemi

ÖZET

İnternet'e bağlı cihaz sayısında ve toplam veri trafiğinde görülen önemli artışla birlikte bulut servislerinin yakın gelecekte ciddi yüklerle karşı karşıya kalması beklenmektedir. Bunun üstesinden gelinebilmesi amacıyla kenar bilişim sistemleri önerilmiştir. Kenar bilişim sistemlerinde, ağın kenarında yer alan erişim noktası veya baz istasyonu gibi cihazların mobil cihazlarda ortaya çıkan hesaplama işlerinin bir bölümünü yürüterek, sonucunu cihazda kullanılması gerektiği durumda cihaza, buluta gönderilmesi gerektiği durumda ise buluta yönlendirmesi öngörülmektedir. Hesaplama kaynaklarının merkezi buluttan ağ kenarına doğru getirildiği bu yöntemle birlikte hem buluttaki iş yükünün azaltılması, hem de gecikme sürelerinin azaltılması mümkün olabilecektir.

Aktarma problemi, bir mobil cihazda ortaya çıkan bir hesaplama işinin lokal olarak mobil cihazda mı yoksa kenar sunucuda mı yürütüleceği kararının verilmesi problemidir. Literatürde çeşitli parametreleri göz önüne alan aktarma algoritmaları önerilmiştir. Sık kullanılan parametreler arasında gecikme zamanı, enerji tüketimi, ve bunların hibrit olarak eniyilenmesi sayılabilir. Literatürde yer alan çalışmaları ayıran farklılıklar, eniyileme hedef fonksiyonunun yanı sıra, incelenen sistemde de yatmaktadır. Bir işlem, aktarım yapıldığında tamamen kenar sunucusuna devrediliyorsa, bu ikili aktarım olarak adlandırılır. Öte yandan, bazı işlemler, parçalara ayrılarak kısmen mobil cihazda, kısmen de kenar sunucusunda yürütülebilir. Bu yöntem ise kısmi aktarım olarak adlandırılır. Kısmi aktarımın verimli işlemesi için hesaplama işinin karakteristiğinin detaylı olarak bilinmesi gereklidir. Bu amaç için, hesaplama işinin alt parçalarının birbirlerine bağımlılıklarını gösteren çağrı çizgeleri kullanılır. Verili bir çağrı çizgesi için en iyi kısmi aktarım çözümü üretmeyi hedefleyen algoritmalar literatürde mevcuttur. Bunun yanısıra, birden fazla kenar sunucusu olduğunu varsayan, mobil kullanıcı davranışının statik veya rassal olduğunu varsayan, sunucunun ve/veya mobil kullanıcıların enerji tüketimini azaltmayı önceleyen, ve farklı enerji tüketim modelleri kullanan çeşitli çalışmalar da bulunmaktadır. Tüm bu çalışmalarda matematiksel araç olarak genellikle eniyileme kuramı veya oyun kuramından yararlanıldığı görülmektedir.

Bu çalışmada, tek kenar sunucusu olan bir sistemde, hesaplama işi üretme bakımından rassal davranan mobil cihazların, yüksek ve düşük öncelikli iki sınıfa ayrıldıkları bir senaryo incelenmiştir. İkili aktarım uygulandığı varsayılmıştır. Aktarım kriteri olarak hesaplama işinin üretildikteki andan itibaren, hesaplanıp kenar sunucusuna aktarıldığı ana kadar geçen sürenin enküçüklenmesi seçilmiştir. Bu senaryoda artırılmış gerçeklik, sanal gerçeklik, çevrimiçi oyun, veya video görüşme gibi bir çokluortam uygulamasının video kodlama işleri ürettiği varsayılmıştır. Kodlama sonucu ortaya çıkan veri, ham veriden küçüktür. Dolayısıyla, aktarım yapılmadığı durumda hesaplama işinin sistemde geçirdiği süre, videonun mobil cihazda kodlanması ve

kodlanmış verinin kenar sunucusuna gönderilmesi için geçen süredir. Buna karşılık, aktarım yapıldığında bu süre, ham videonun kenar sunucusuna gönderilmesi ve burada kodlanmasından oluşacaktır.

Kenar sunucusunda ağırlıklı round-robin çizelgeleme yapıldığı varsayılmıştır. Buna göre sunucuda servis alan aynı sınıfa ait hesaplama işleri, küçük periyotlarda eşit miktar servis alırlar. Buna karşılık yüksek öncelikli işler, her roundda düşük öncelikli olanların h katı kadar hizmet alır. Round robin çizelgeleme, zaman biriminin sıfıra gittiği limit durumunda işlemci-paylaşımı modeline dönüşür. Kuyruk kuramından bilindiği üzere, işlemci-paylaşımı sistemlerinde herbir işin sistemde kalma zamanı, işin uzunluğu ile doğru orantılıdır. Bu bakımdan, her bir iş için aktarım kararı verilirken hesaplanması gereken sunucuda geçen hesaplama zamanı, bir kuyruklama katsayısı ile modellenebilir. Bu kuyruklama katsayısı, düşük öncelikli kullanıcılar için yüksek öncelikli kullanıcılara göre h kat büyük alınmaktadır. Böylece bu katsayı ile hem bir kabul mekanizması, hem de servis ayrımı gerçekleştirilmiştir.

Bu sistem modeli için bir benzetim programı yazılmış ve çeşitli senaryolar için sayısal sonuçlar elde edilmiştir. Öncelikli olarak iki sınıf arasında hizmet kalitesi ayrımını sağlayabilecek, ancak yüksek öncelikli işlerin performansını düşürmeyecek en iyi kuyruklama katsayısının bulunması için, bu katsayını çeşitli değerleri için sistemde kalınan ortalama süre istatistiği elde edilmiştir. Buna göre örnek bir senaryoda bu katsayının alması gereken değer hakkında çıkarımlar yapılmıştır. Ayrıca aynı senaryo için mobil cihazın enerji tüketimi grafikleri de elde edilmiştir. Bu senaryoda h parametresinin başarıma etkisi de incelenmiş ve bir üst sınır bulunmuştur.

Her senaryo için bu katsayının en iyi değerinin değişeceği açıktır. Bu bakımdan bu en iyi değeri yakalayan uyarlanır bir algoritma önerilmiştir. Bu algoritmaya göre, bu katsayı ilk değeri 1 olmak üzere çalışmaya başlanır. Her bir işlem sistemi terk ettiğinde, sistemde kaldığı süre, işin uzunluğuna bölünerek bu işin deneyimlediği katsayı değeri elde edilir. Bu değer kullanılarak, katsayı değeri güncellenir. Bu yöntemin sistem dinamiklerini yakaladığı ve değişken senaryolarada uyum gösterdiği gösterilmiştir. Bu senaryoda, değişen iş yükleri, yüksek öncelikli mobil kullanıcı sayısı ve h parametresi değerleri için sonuçlar elde edilmiştir.

Daha gerçekçi iş üretim modellerinin yer aldığı modeller, ve enerji tüketiminin de aktarım kararına entegre edildiği algoritmalar, gelecek çalışmaların konusu olacaktır.

1. INTRODUCTION

The previous decade has witnessed the emergence of Cloud Computing (CC) as a new paradigm of computing. Storage and network management in the cloud were the main visions of this new paradigm. With the vast resources available in the cloud and the ability of leveraging these resources, elastic and powerful computing can be offered to the end users. The Internet has seen a rapid growth, cloud businesses have increased to become a profitable section. Such as Amazon and Dropbox [3]. The popularity of mobile devices has been increasing in recent years along with the exponential growth of mobile internet traffic resulting in a huge progress in wireless communication and networks. The users were promised gigabit access in the next wireless generation by the Small Cells (SCs) network and multi-antenna communication [4]. The high data-rate allows mobile users to run computing services remotely at the data centre of the cloud, resulting in a new area of research called Mobile Cloud Computing (MCC). However; the long propagation distance between the mobile device and the remote cloud data centre, lack of mobility support, and location-awareness are serious inherent problems of cloud computing. The number of connected devices to the Internet is expected to reach 50 billion by 2020, and the people, machines, and things are estimated to produce data around 500 zettabytes by 2019 [5]. With this huge growth of data exchange and the increasing number of connected devices, MCC paradigm with its centralized approach will face tremendous loads in the coming years. Hence; European Telecommunications Standard Institute (ETSI) came up with a new idea to overcome this problem in 2014 [6]. This approach, namely Mobile Edge Computing (MEC), is proposed to use the capacity at the edge of the network such as Base Stations (BSs) or Access Points (APs) to significantly reduce the amount of data sent to the cloud. In other words, the BS and /or AP executes the mobile users' computation tasks on behalf of them. Figure 1.1 illustrates the architecture of MEC.

The basic idea behind MEC is to bring down the resources from the cloud into the edge to have the resources in close proximity to the end user. In this manner, any edge

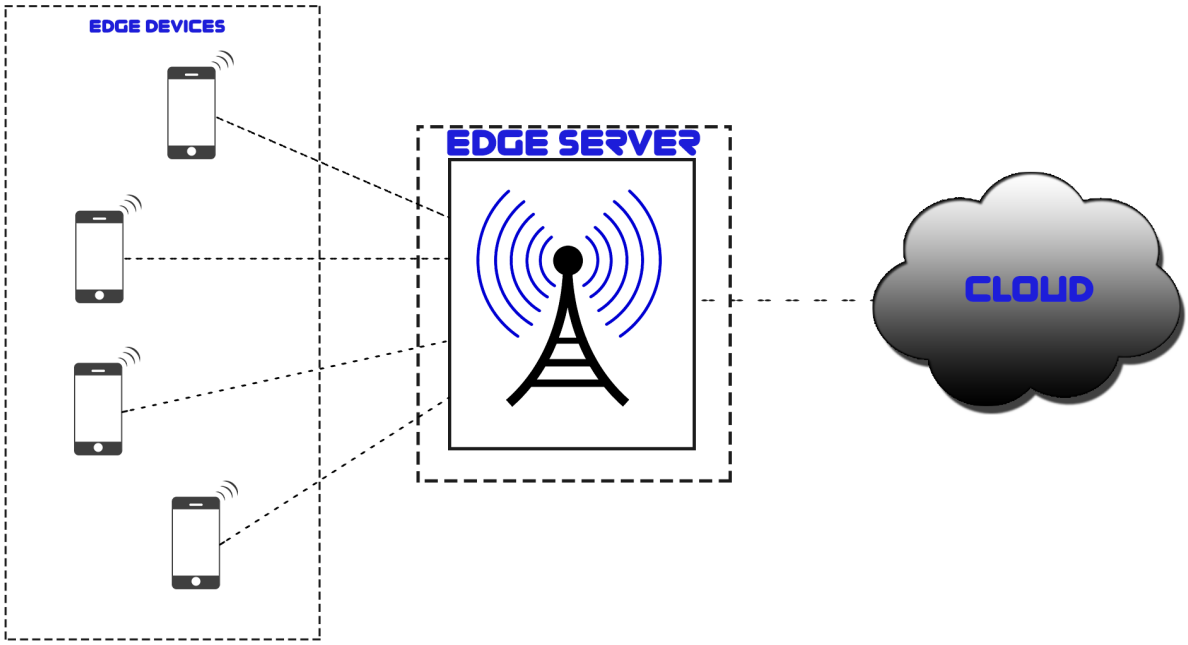

Figure 1.1 : The architecture of Mobile Edge Computing.

device will be able to use resources available in the edge servers, thus dramatically diminishing communication latency. On the other hand, a variety of challenges awaits MEC systems:

(1) *Resource management and allocation:* It is well known that with this new age of computing, resource management has become more challenging. It is expected that an individual MEC would have a limited storage and computing resources, it can serve a limited number of users, and there is a variety of mobile devices that require a variety of resources. Therefore; there has to be a method to know which resources are where and how they can be located. This could be a problem to be solved in the coming days.

(2) *Mobility:* As MEC will be deployed in Heterogeneous Networks (HetNet) which in turn has different BSs such as Macro Station (MS), Micro Station (MiS), Pico Station (PiS), and Femto Station (FS), and these BSs have different coverage areas and thus; mobile devices that experience frequent hand-offs due to roaming may see significantly reduced performance due to the need of conveying raw data and/or computation results to and from an edge server with which they no longer have direct contact.

(3) *Security:* The conventional mechanisms may not be sufficient to achieve the desired level of privacy [6]. Several security threats might arise up such as cyber-attacks due to the integration of the communication and the cloud system. Protecting the privacy of mobile users' data is a major concern [7].

(4) *Interoperability:* The variety of the Service Providers (SPs) who own the infrastructure of MEC makes it a necessity to have a collaboration between the SPs to have common specifications [8].

(5) *Robustness and Resilience:* It is important to guarantee the robustness of the MEC when it is incorporated into mobile base station. Moreover; it is essential that the incorporation is not affecting the availability of the mobile network.

(6) *Offloading:* Which refers to the usage of Edge server in MEC or (cloud in case of MCC) to execute the mobile users' tasks on behalf of the mobiles. Offloading is one of the most important challenges that need to be tackled. Decision making of whether to partition the task or not and what task to offload and where are concerns in MEC.

Recently, even a newer concept was proposed by Cisco called Fog Computing and networking that is considered as an extension of the cloud computing in the perspective of Cisco [9]. It is a generalized form of MEC that is a highly virtualized platform that provides storage, networking services, and computation between the mobile devices and the traditional cloud server. The two concepts of fog computing and mobile edge computing are somewhat overlapping and the terminologies are sometimes interchangeably used, and many technologies are applicable to both.

MEC is based on virtualized platforms that leverage new advances in networking such as *Network Functions Virtualization* (NFV), *Information Centric Network* (ICN), and *Software Defined Network* (SDN). Within NFV, a single Edge device can be enabled and by creating multiple *virtual machines* (VMs), providing computing services for multiple mobile devices so as to perform different tasks simultaneously is possible [10]. On the other hand, the ICN provides an alternative paradigm for an end-to-end service recognition for MEC. Last but not least, MEC network management is allowed by SDN to manage services via function abstraction, achieving scalable, and dynamic computing [11]. Developing these general network technologies is one of the main focuses of MEC with a view to implement them at the network edge.

The emerging mobile applications that will benefit from MEC by offloading the computation-intensive tasks to be remotely executed at the edge server are increasing. Two examples are provided in the following to illustrate the very basic principles of MEC. Figure 1.2 shows a face recognition application as the first example that has five essential computation components, starting with image acquisition and followed by face detection, pre-processing, feature extraction, and ending with classification [1].

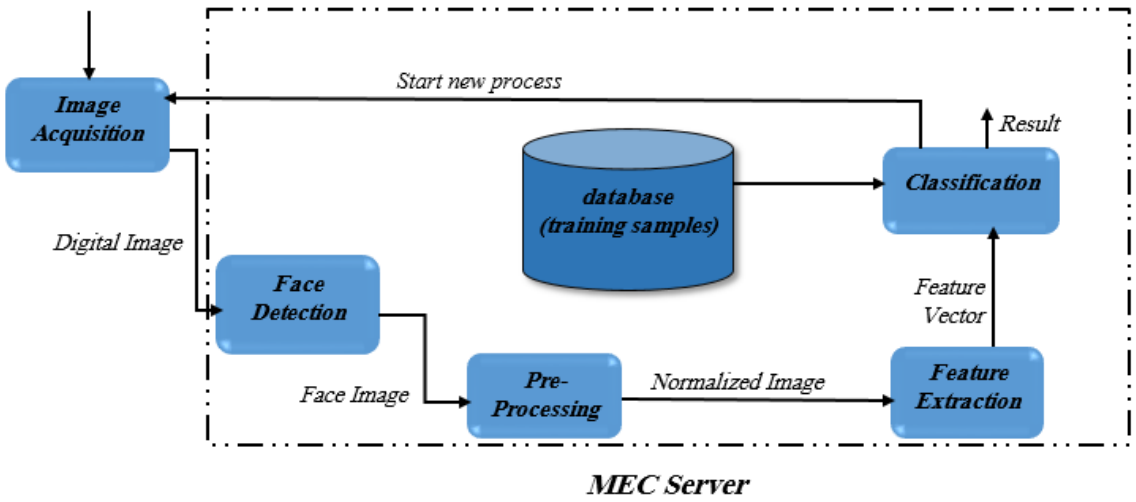

Figure 1.2 : The main Processes of Face Recognition Application [1].

Image acquisition should be executed at the mobile device so as to support the user interface and it is not a computational intensive process and thus there is no need for offloading. The rest of the components could all be offloaded to be executed remotely at the edge server because each one contains complex algorithms, Signal Processing (SP), or Machine Learning (ML), and is a very computationally intensive process that needs heavy resources. For instance face detection requires image processing to improve the input image whereas feature extraction has very complex algorithms to distinguish between the faces of the people [1].

Secondly, Augmented Reality (AR) applications which overlie an artificial image into the reality through the screen of the mobile are very popular applications that can leverage rich resources at the edge of the network. The procedure of AR applications is shown in Figure 1.3. The procedure of AR applications consists of five major components starting with video resource and ending with Renderer [2]. Video resource along with renderer processes need to be executed locally at the mobile device whereas the rest of the components that are computationally intensive such as Tracker, Mapper, and Object recognizer may be offloaded for remote execution at the edge of the network.

In this study we investigate the offloading problem in the presence of two user classes: one is high priority and the other is low priority. The rest of the thesis is organized as follows. In chapter 2, we present a literature review on mobile edge computing and offloading, as well as background information on weighted round-robin service scheduling. Chapter 3 contains the description of the proposed method, the system

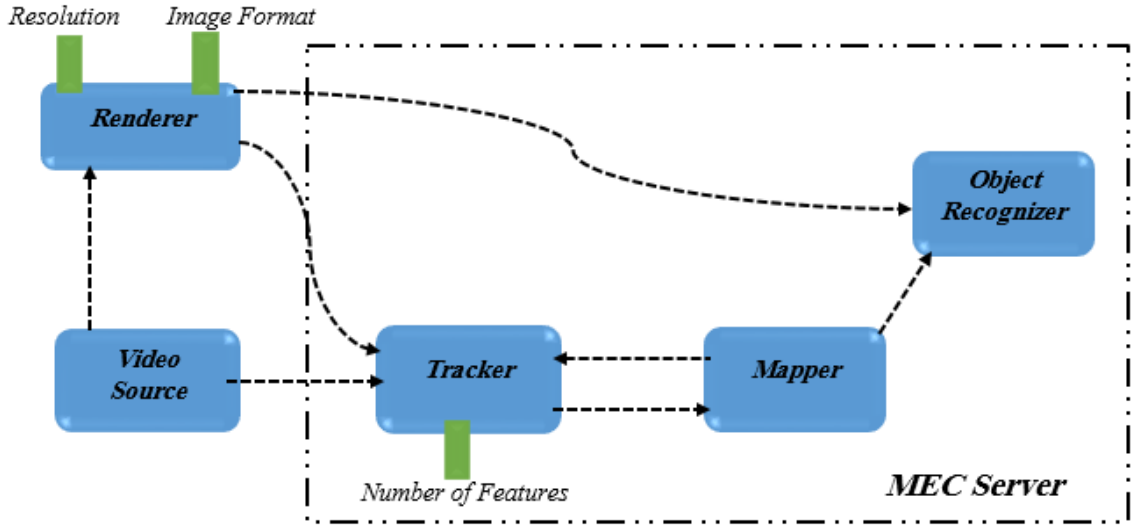

Figure 1.3 : The main processes of Augmented Reality Application [2].

model, and experimentation results. Finally, the conclusion and future works are presented in chapter 4.

2. LITERATURE REVIEW

2.1 Offloading in Mobile Edge Computing

Many researchers have investigated a wide range of issues related to MEC in recent years from both academia and the industry. Standardization, implementation, multi-user resource allocation, and system and network modelling were all investigated in those studies. Thereafter, many surveys have been published aiming to provide overviews of different aspects of MEC including applications, enabling techniques, edge computation offloading, energy efficiency, single and multi-server systems, and connections with 5G and Internet of Things (IoT) [8], [12], [13], [14]. In [13], an overview of MEC was presented along with a discussion of some promising real time application scenarios of MEC services and a taxonomy that describes the key attributes of MEC. The survey in [15] provides an overview of MEC and focuses on the key enabling techniques in addition to the analysis of some deployment scenarios that offer multi-tenancy support for applications' developers. A comprehensive survey of MEC systems was presented in [16] including the concept, technical enablers, and architecture along with some categorized deployment scenarios and services models, whereas the survey in [17] provides technical analysis and limitations of MEC through identifying, discussing, and classifying a variety of applications. [12] gives an overview of MEC networks with the definition, architecture, and advantages. Moreover; it surveys the MEC issues of computing, communications technologies, and caching at the network edge beside key enablers like SDN/NFV. In addition, [18] describes the role of MEC in IoT through providing some MEC deployment examples with the essential benefits and challenges of MEC moving toward 5G.

Generally a task generated at a mobile device is either executed in the mobile device or offloaded to the cloud or to the edge server to be executed there and then, the outcomes is sent back to the mobile device that generated it. The operation of transmitting the application to be remotely executed inside the cloud or the edge is called offloading.

Tasks offloading can be divided into three different types according to the way the task is executed:

1−*Binary Offloading:* In this method the task must be executed either in the mobile device itself completely or transmitted completely to the MEC server to be executed there. In another words, it cannot be partitioned and processed [6]. Basically the simple tasks are not preferred to be offloaded.

2−*Partial Offloading:* In this method, the task can be partitioned into two or more parts. After the partitioning, some partitions will be executed in the mobile device, and the rest will be offloaded to be executed at the edge server [6].The procedure of offloading and computing should be performed carefully. Parts' offloading and execution order cannot be arbitrary, for several reasons for instance some parts are fed by the output of others, some parts cannot be offloaded and must be executed locally [6].

3−*Full Offloading:* In full offloading, the task is always executed at the edge server [31].

As mentioned in the challenges, offloading is one of the most critical issues that need to be tackled, hence; there are plenty of studies on the challenges related to offloading. These studies differ in their choice of offloading strategy, and objective functions when considering offloading optimization. In [19], the authors study the computational offloading problem of multi-user in MEC in wireless interference channels and attempt to achieve efficient computation offloading in a distributed manner through adopting a game theoretic approach. A strategy of offloading computation for MEC is proposed in [20] with aiming to derive an optimal partial offloading on a given call graph, choosing partitions to be executed locally or remotely. Investigation of three different computation models is carried out in [21], namely local compression, edge cloud compression, and partial compression offloading with aiming to develop an algorithm of optimal joint communication and computation resource allocation. In [22] an efficient computation offloading method for mobile cloud computing is proposed using a game theoretic approach. A scenario where a full application's tasks are offloaded to a computationally enhanced small cell base station is explained in [23], where the goal is to provide a strategy for uplink, downlink, and remote computational

resource allocation and to improve the quality of experience of users. In [24], the authors' contribution is the optimization of the offloading decision, communication, and computation resources via a Heuristic Offloading Decision Algorithm (HODA).

On the other hand, researchers tried to enhance the energy efficiency by achieving an optimal way of offloading. For instance in [25], an endeavour to minimize the overall cost of energy by jointly optimizing the offloading decision as well as the resource communication allocation is presented. The contribution in [26] is similar to [25]. However; [27] proposes to conserve energy and preserve the Quality of Service (QoS) of the users by using a game theoretical approach in which the mobile makes its decision of offloading distributively and communication allocation resources computation are decided by the computing access point.

Energy efficiency of both mobile devices and edge servers deployed at the edge of the network is an issue to be solved in MEC systems. Therefore; researchers in both academia and industry are working to find optimal solutions to reduce the consumed energy by the server and prolong the life time of the mobiles' battery. In the literature, many works on this challenge exist aiming to minimize the energy consumed by tasks offloading taking into account task computing and file transmission. The authors of [28] propose Energy Efficient Computation Offloading (EECO) mechanism for MEC in 5G heterogeneous networks. In [29], a novel approach is introduced which is the green MEC system with harvesting devices and developing an effective strategy of computational offloading.

To the best of our knowledge, no study considers multi-class users except [30], where an admission control policy and adaptive resource allocation are proposed to offload mobile applications' tasks into a cloudlet in the presence two classes namely "*members*" and "*non-members*" with high and low priorities consecutively. The high priority user is allowed to utilize more resources compared to the low priority.

2.2 Queueing Systems

Queueing theory is concerned with the analysis of traffic congestion and queueing and scheduling of services. Models of queueing systems are used in the field of Information Technology (IT) to analyze the performance of computing systems such

as CPU scheduling and router/switch buffering.

When a customer arrives and finds the server(s) busy, the customer joins the queue and then based on some rule or policy, the customer is selected for service in a queueing system. After the customer is selected, it will be served by any of the system's server(s). This event is referred to as *service discipline* or *scheduling discipline*. *holding time* is indicated as the service time of that customer, After the customer is completely served, the customer leaves the system [32].

An important rule is played by the way the scheduling discipline is modelled. First-in First-out (FIFO) or First Come First Served (FCFS) is the simplest choice of scheduling discipline, however; it might not be the most desirable method. For instance, in a multi-class system which contains dissimilar customers in terms of priorities or the required service time, choosing the best service discipline should take into account: (i) minimizing the mean latency, (ii) increasing the number of served customers per time unit (*increasing the throughput*), or (iii) fairness achievement among different customers from different classes [32, p.43].

In [33] the notion of Processor Sharing (PS) was proposed as a specific case of Round-Robin (RR) scheduling where the time portion is allowed to approach zero. Under such circumstances, when n customers exist in the system, each customer receives the service with rate K/n, where K indicates the overall capacity of system's server.

2.3 Weighted Round-Robin (WRR) Service Discipline

Weighted Round-Robin is an approximation of Generalized Processor Sharing (GPS) and a discipline in scheduling where all tasks are classified into a number of service classes with respectively high and low priorities. For holding the packets temporarily, a set of queues are established, while the tasks await service. WRR uses the very same principle of Round Robin which basically shares the CPU service among all the packets for a specific number of time slots (Machine Cycles (MC) in case of CPU scheduling) with out considering the priorities of arrived tasks. WRR uses the very same principle of Round Robin except considering the priorities of the tasks. In WRR, a high priority task is served more frequently than others. To put it another way, a high priority task receives more MCs than lower priority tasks.

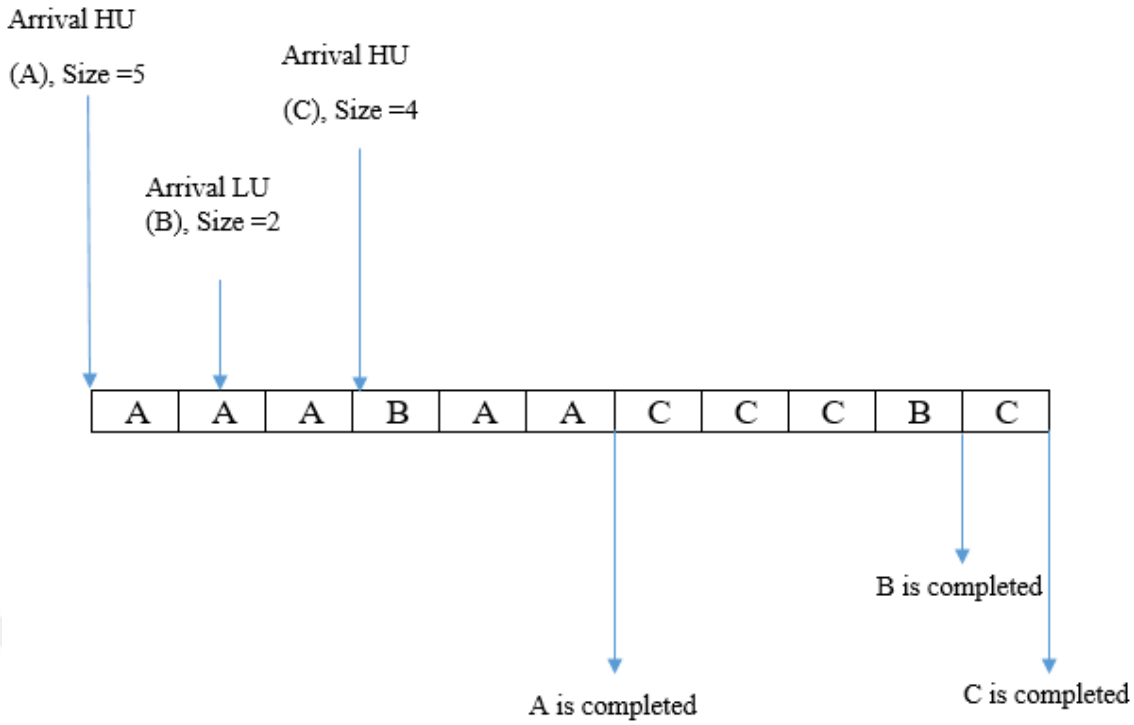

Figure 2.1 : A sample scenario with WRR

Figure 2.1 illustrates the WRR service discipline. In this scenario, three different tasks with different sizes (different MC requirements) arrive to the system in different times where h factor is 3 (high to low priority service ratio). Task A with high priority and size 5 arrives first and directly starts receiving the service as the system is empty. During the second MC, B arrives to the system with low priority and size 2. Just after the third MC, another high priority task, C, arrives with size 4. B starts receiving the service for one MC after A receives $h = 3$ MCs of service, because it arrived before C, while both A and C are waiting in the queue. After that, A takes the place to be completely executed as it needs only 2 MCs. Then, C, being the high priority task remaining, takes precedence in the turn to receive service for $h = 3$ MCs (high priority task) while B is waiting in the queue. After C is processed for $h = 3$ MCs, B comes again to receive the remaining required MC and then leaves the system. Finally, C is served for the rest of required MCs and leaves the system.

3. SYSTEM MODEL AND NUMERICAL RESULTS

3.1 System Model

We consider a MEC system with a single MEC server and N subscribers. There are two user classes: high priority users (HU) and low priority users (LU). Each mobile user generates tasks according to Poisson process with intensities λ_H and λ_L for HU and LU, respectively. The tasks' sizes in bits, denoted L_H and L_L, are assumed to have uniform distributions. The CPU of the mobiles are assumed to have a single core with an operating frequency of f_m whereas the server has a CPU with frequency f_s, n_c cores, and a speed-up multiplier s due to additional memory, better processor design and so forth as described in [34]. This yields an effective total speed-up of $s n_c f_s / f_m$ for a task that is run on the edge server as opposed to a task run locally on the mobile.

A task that is decided to be offloaded is entirely sent to the MEC server (Binary Offloading). For each task, the required number of Machine Cycles (MCs) are equal to $L_i X$; $i \in \{H, L\}$, where X is the MCs-per-bit and depends on the individual task [35]. To get an idea about commonplace values, we refer the reader to Table 3 in [36]. Some studies, such as [37], assume load balancing is done at the edge server, that is, "the CPU cycles are proportionally allocated for each user such that all users experience the same computing time". Instead, we assume weighted round-robin scheduling for tasks where HU tasks are executed h cycles for each cycle of LU tasks.

The energy consumption of the mobiles are modelled as having two major components: (i) *energy consumption due to transmission* [36], and (ii) *energy consumption due to computation* [37]. Although both components are affected by instantaneous conditions, we opted for a model with constant transmission energy per-bit and constant computation energy per-MC. These can be viewed as average values acting over long windows of operation.

The offloading decision mainly aims to minimize the sojourn time of each task. We denote with $\alpha \in (0,1)$ the encoding ratio, that is, the encoded data size is $L \times \alpha$ bits

Table 3.1 : Simulation Parameters. U[·,·] denotes uniform distribution.

Parameter	Value
Number of Mobile users	100
Number of HL	30
Number of LU	70
$\lambda_H = \lambda_L$	1 task / 2 hours
L_H, L_L	U [20 MB,100 MB]
X	U[200 , 1000]
f_m	1 GHz
f_s	10 GHz
n_c	32
s	10
Computation energy per MC	20×10^{-11} j/MC
Transmission energy per bit	2.0833×10^{-9} j/bit
r	U[200kbps , 20Mbps]
h	10
α	$\in \{0.1, 0.01, 0.001\}$
Number of tasks simulated	5000

for raw data of L bits. The sojourn time in case of local execution can be written as:

$$T_l = \frac{LX}{f_m} + \frac{L\alpha}{r} \tag{3.1}$$

where r is the data rate of the associated mobile user. In (3.1), the first term corresponds to execution time, whereas the second term corresponds to the transmission time of the encoded data. The sojourn time in case of offloading is expressed as:

$$T_o = \frac{L}{r} + q_i \frac{LX}{s n_c f_s} \tag{3.2}$$

where $q_i > 1$; $i \in \{H, L\}$, is the factor of queueing delay. It is clear that the MEC server will be running multiple concurrent tasks. Since we employ weighted round-robin scheduling, the queueing discipline can be modelled as a priority processor-sharing (PS) discipline. It is well known that the expected sojourn time in a PS queue is linearly proportional to the task size [32, p.215]. Therefore using a factor such as q_i (with $q_L = h q_H$) accurately captures the sojourn time of an offloaded task. In local computation, the load on the mobile is expected to be much lower. Thus, we ignore queueing delay on the mobile device, although a second multiplier for local queueing delay can also be defined.

The offloading decision can be assumed to take place either at the mobile or at the MEC server. In either case, signalling between the mobile and the MEC server should

take place to exchange relevant information such as task size, or queue backlog. We will assume that the MEC server is in charge of the offloading decision and informs the mobile whether to offload or not based on the comparison of the T_l value to the T_o value. An important parameter in this setting is q_H, which not only models but also affects the queueing delay, and thus acts as an admission rule parameter. If q_H is too high, tasks are discouraged to offload, resulting in a lightly loaded MEC server which entails a low q_H value. On the other hand, a too low value for q_H will result in more offloads yielding high queueing delays. Therefore, there should be an optimal q_H value that balances the demand for the MEC server. This also acts as a service differentiator between HU and LU through the relation $q_L = h q_H$.

3.2 Numerical Experimentation

We wrote a stand-alone simulation program in Matlab that simulates the described system. Simulation parameters are summarized in Table 3.1. Average task sojourn times for different values of q_H are given in Figure 3.1 for $\alpha = 0.01$, and in Figure 3.3 for $\alpha = 0.001$. If the objective is to provide the best performance for HU, then the optimal q_H value can said to be around 1000 for these two cases. It is clear that LU starts getting discouraged from offloading when $q_H > 100$, and HU starts getting discouraged from offloading when $q_H > 1000$. To see the behavior in more detail, we plot the average task sojourn times and energy consumptions for $100 \leq q_H \leq 500$ in Figures 3.5 and 3.6, and for $1000 \leq q_H \leq 5000$ in Figures 3.7 and 3.8.

On the other hand, looking at the average energy consumptions that are given in Figure 3.2 for $\alpha = 0.01$, and in Figure 3.4 for $\alpha = 0.001$, it is evident that $q_H = 1000$ does not provide the best performance in terms of energy efficiency especially for LU, and thus the overall population of users. Taking this into account, a more modest value of $q_H = 100$ seems to provide a much better overall performance all around without sacrificing much from HU service quality.

The performance degradation in energy efficiency as q_H increases beyond a point is clearly due to the fact that less and less tasks are offloaded with increasing q_H, which is demonstrated in Figure 3.9.

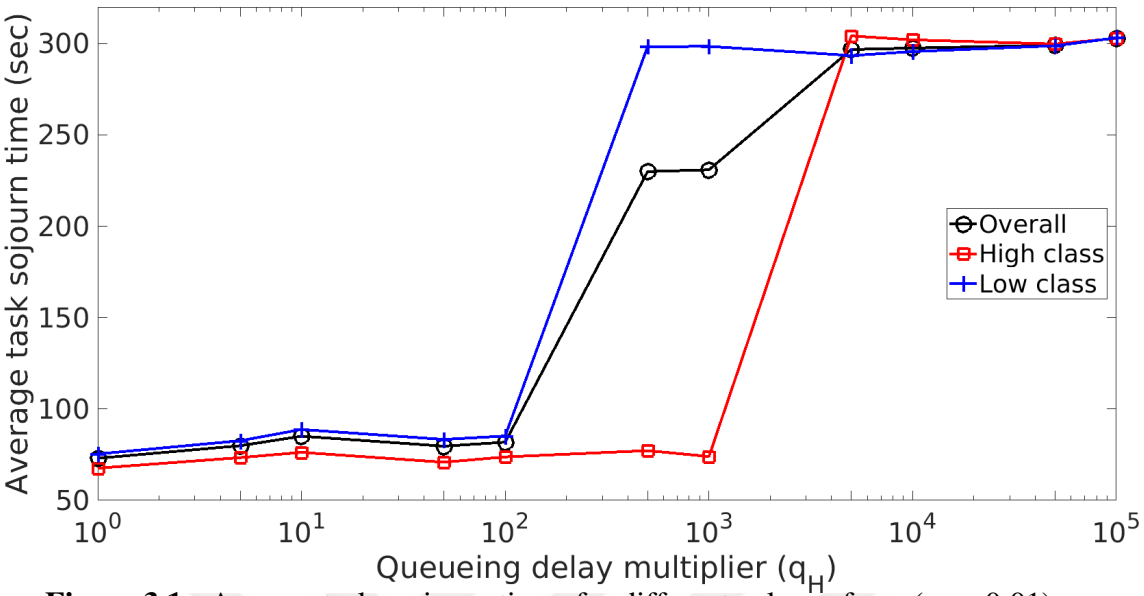

Figure 3.1 : Average tasks sojourn times for different values of q_H, ($\alpha = 0.01$).

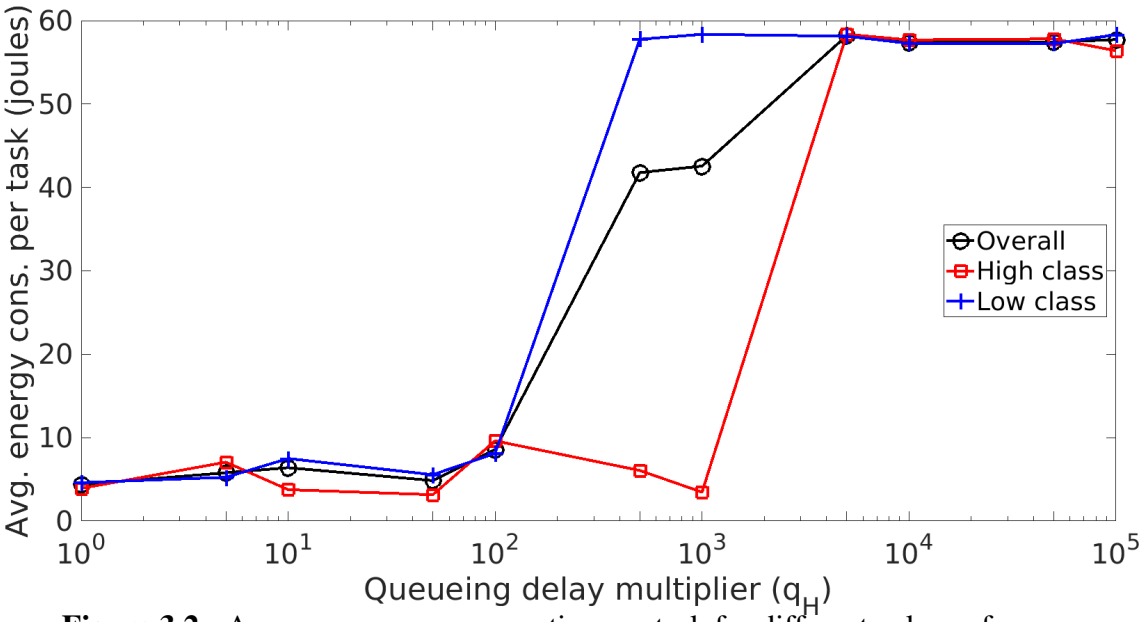

Figure 3.2 : Average energy consumption per task for different values of q_H, ($\alpha = 0.01$).

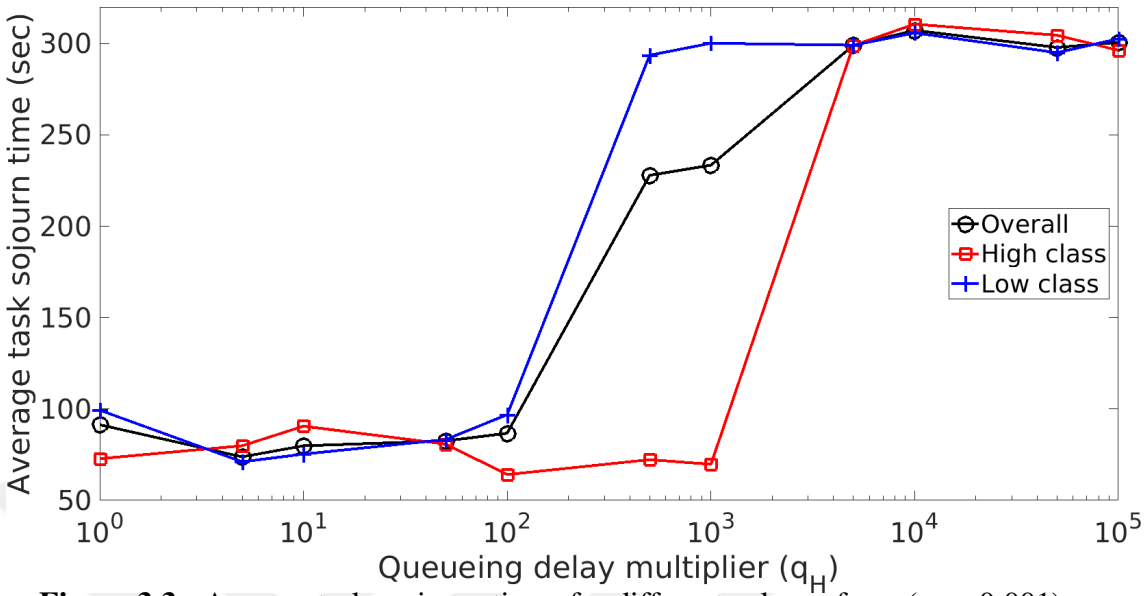

Figure 3.3 : Average tasks sojourn times for different values of q_H, ($\alpha = 0.001$).

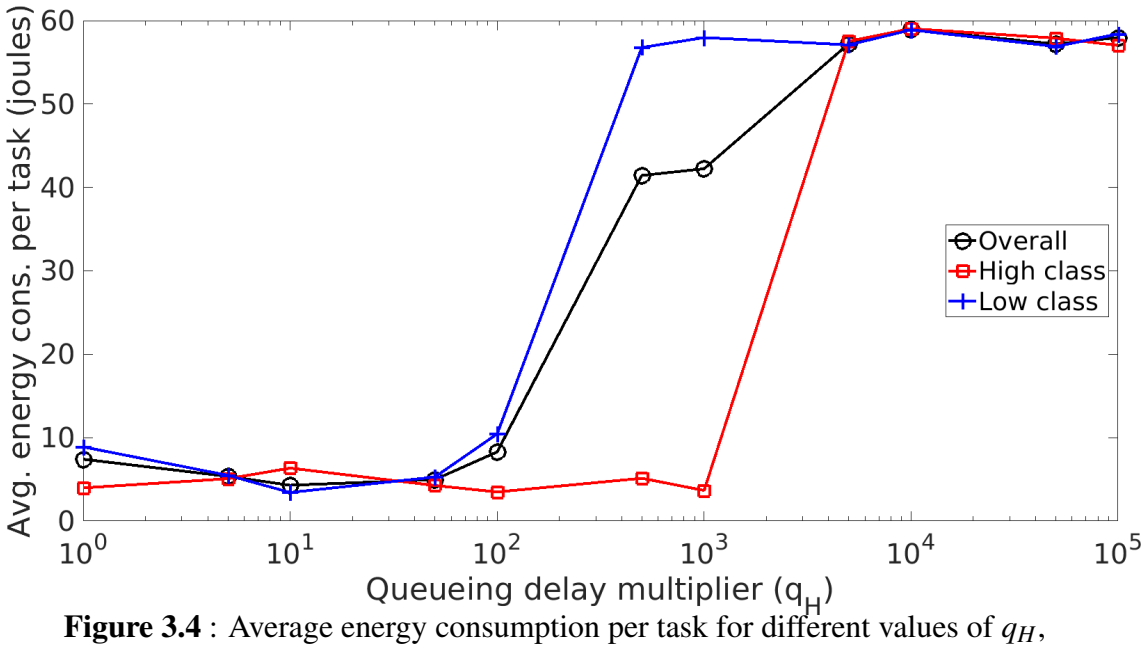

Figure 3.4 : Average energy consumption per task for different values of q_H, ($\alpha = 0.001$).

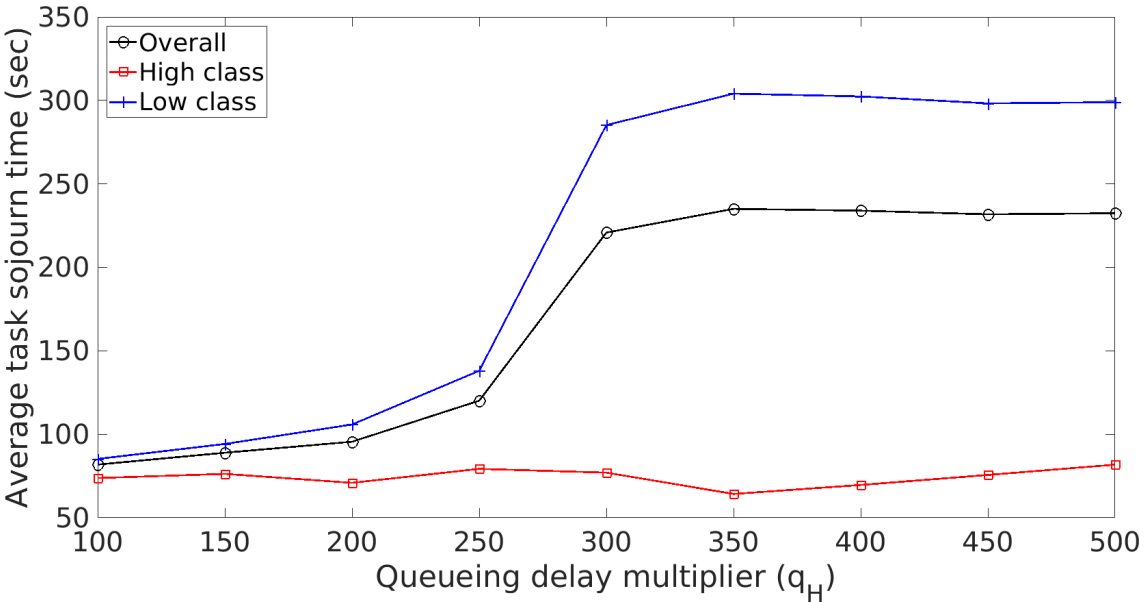

Figure 3.5 : Average tasks sojourn times for $100 \leq q_H \leq 500$, ($\alpha = 0.01$).

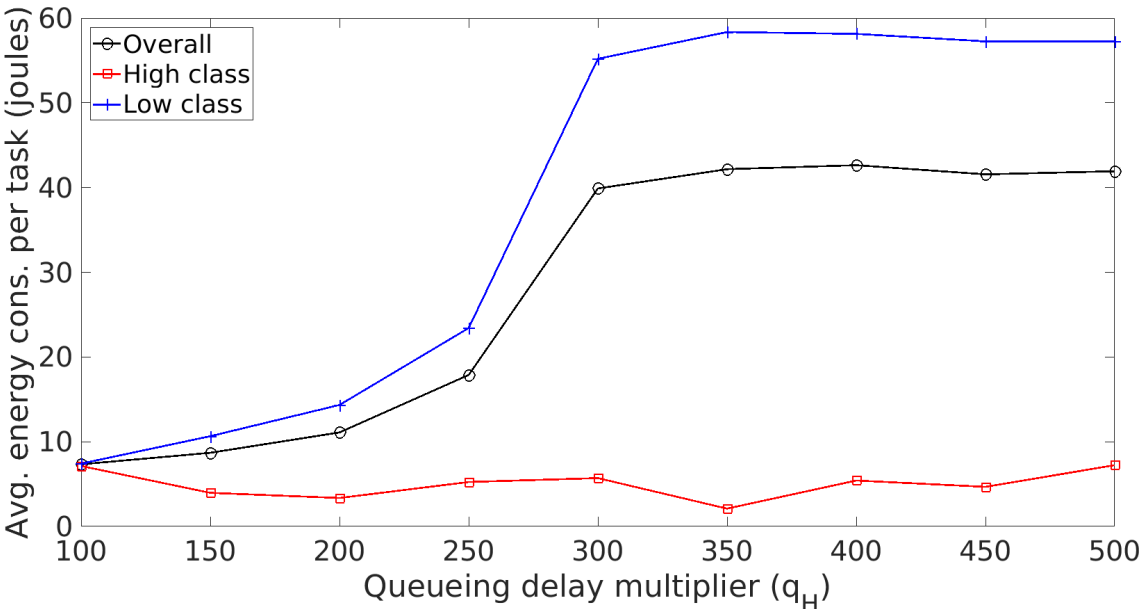

Figure 3.6 : Average energy consumption per task for $100 \leq q_H \leq 500$, ($\alpha = 0.01$).

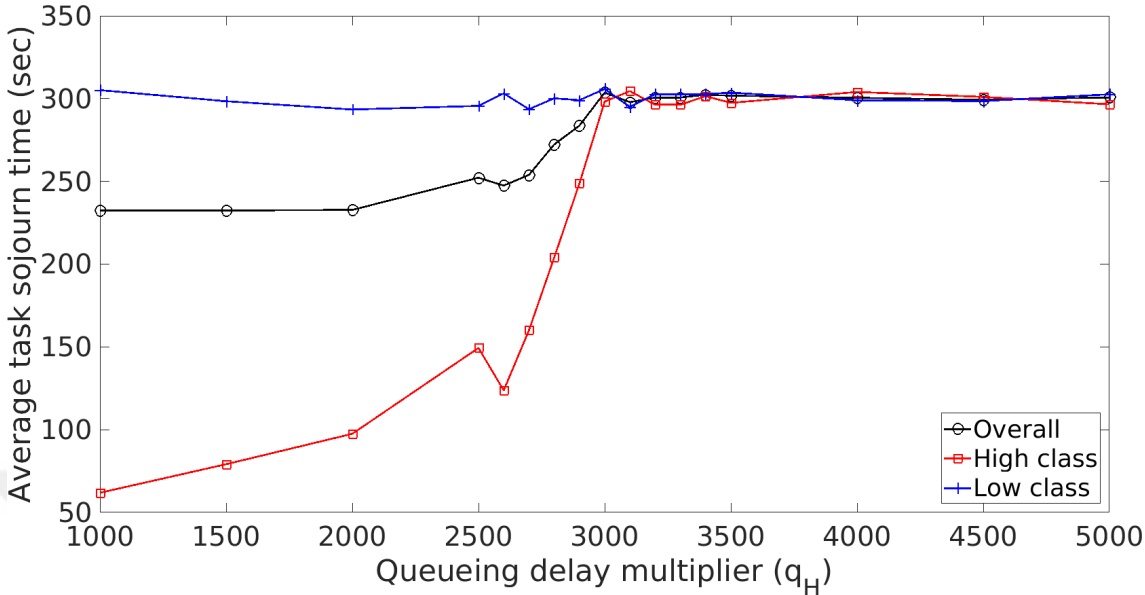

Figure 3.7 : Average tasks sojourn times for $1000 \leq q_H \leq 5000$, ($\alpha = 0.01$).

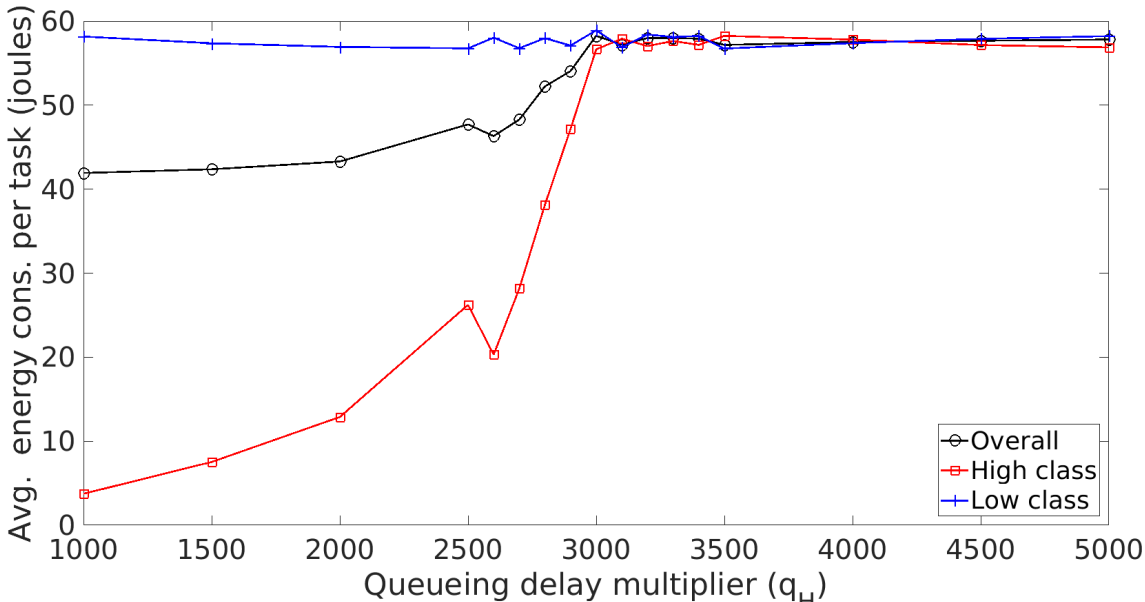

Figure 3.8 : Average energy consumption per task for $1000 \leq q_H \leq 5000$, ($\alpha = 0.01$).

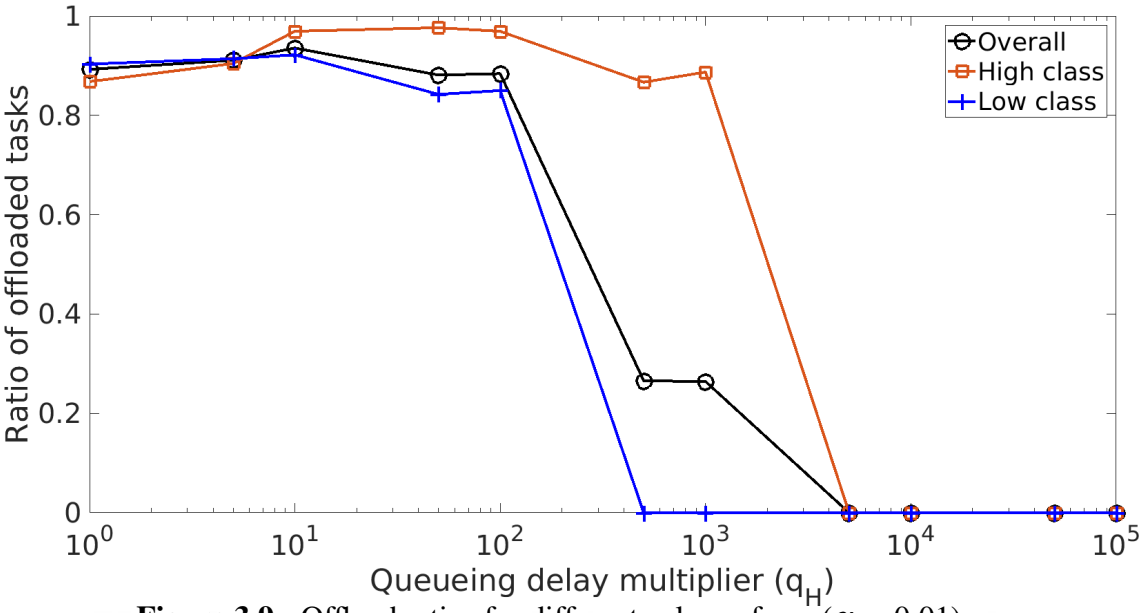

Figure 3.9 : Offload ratios for different values of q_H, ($\alpha = 0.01$).

Keeping the queueing delay multiplier $q_H = 100$, and the encoding ratio $\alpha = 0.01$, we plot the average task sojourn time and average energy consumption per task in Figure 3.10 and Figure 3.11, respectively, for varying values of the PS priority factor, h. Clearly, increasing h increases the service discrimination between HU and LU. The effect of increasing h on the service HU gets does not seem to be significant, whereas the impact it has on LU is evidently profound. This is obviously due to the fact that the sojourn time in case of offloading for LU is directly proportional to h, and increasing h increasingly forbids LU tasks from being offloaded. Thus, we conclude that since the effect on the HU service is slight, the service provider should choose to keep h modest (below 26 for the specific scenario investigated here) to be inclusive so as to improve overall system performance.

3.3 The Adaptive Scheme

The optimum value of q_H depends on the load on the server buffer, which is a function of traffic intensity of each class, task size distribution, MC required per bit, and the CPU speed. Hence, to use a fixed q_H, one needs to have all these information and pick q_H accordingly. Furthermore, this method will lead to sub-optimal performance when any of these parameters are time dependent and changes over the course of operation. Therefore, we propose to use an adaptive scheme for the value of q_H. We start the

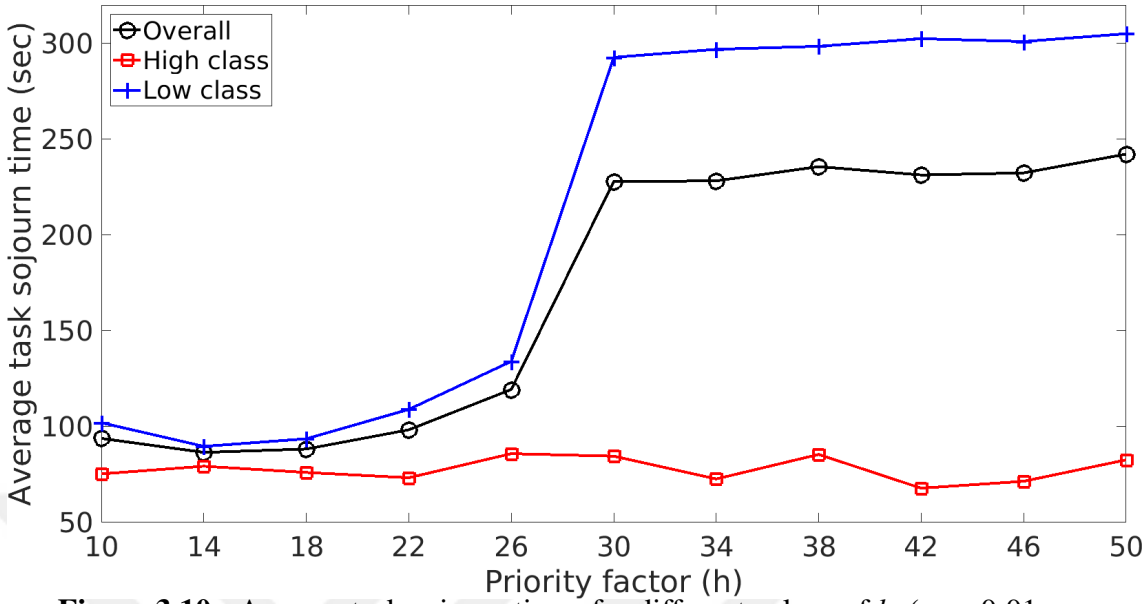

Figure 3.10 : Average task sojourn times for different values of h, ($\alpha = 0.01$, $q_H = 100$).

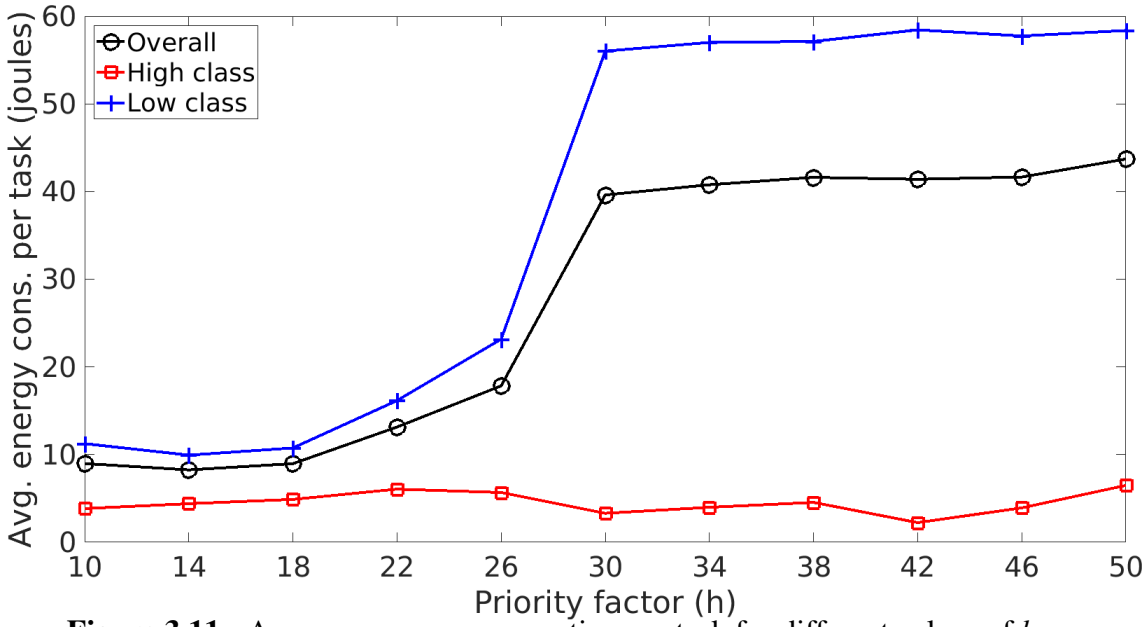

Figure 3.11 : Average energy consumption per task for different values of h, ($\alpha = 0.01$, $q_H = 100$).

Table 3.2 : Simulation Parameters. U[·, ·] denotes uniform distribution.

Parameter	Value
Number of Mobile users	250
Number of HL	50
Number of LU	200
$\lambda_H = \lambda_L$	1 task / 10 seconds
L	U [1 MB,10 MB]
X	U[10^3 , 10^4]
f_m	1 GHz
f_s	4 GHz
n_c	32
s	5
Computation energy per MC	20×10^{-11} j/MC
Transmission energy per bit	2.0833×10^{-9} j/bit
r	U[200kbps, 20Mbps]
h	10
α	0.2
θ	0.02
Number of tasks simulated	10^4

decision algorithm on the MEC server with an initial q_H value, and we update it after the completion of each task using damped averaging as follows:

$$q'_H = \begin{cases} (1-\theta)q_H + \theta\, T_s \dfrac{s\, n_c\, f_s}{LX}, & \text{for HU,} \\ (1-\theta)q_H + \theta\, T_s \max\left\{\dfrac{s\, n_c\, f_s}{LX\, h}, 1\right\}, & \text{for LU.} \end{cases} \tag{3.3}$$

Here, T_s denotes the time that the task spends on the MEC server, and $\theta \in \{0,1\}$ is the damping coefficient. Note that under a sufficiently loaded regime, LU tasks will see a runtime that is h-fold on the average compared to HU tasks. However, when the load is sufficiently low, an LU task might experience little or no queueing delay, resulting in a $(snc\,fs)/(LX)$ value that is close to 1. In such cases, we impose a $\max\{\cdot, 1\}$ function so that q_H is not pulled down more than necessary (to below 1). In this way, q_H value can adapt to changing conditions. Initially, the MEC server starts with an empty queue, which means that an arriving task will not experience queueing delay. Hence, it is reasonable to start with $qH = 1$.

3.4 Numerical Experimentation for the Adaptive Scheme

In this setting, we consider a video conference session in which a mobile user uses his/her device's camera to send video to other users. The task is the encoding of high-definition video captured by the camera of the mobile into a lower resolution video for transmission to the other end. The simulation parameters for this scenario (unless specified otherwise for some figures) are summarized in Table 3.2. To determine realistic job sizes, we consulted YouTube's "Live encoder settings, bit rates, and resolutions" documentation [38]. For HD videos such as 720p and 1080p, video bit rates range from 1500 Kbps to 9000 Kbps. Hence; we used the range from 1 Mb to 10 Mb for the job size, which is assumed to have a uniform distribution. For ease of modelling, we assumed Poisson arrivals for tasks at each mobile, resulting in a Poisson overall arrival stream. In this scenario, we may assume that the video becomes 240p or 360p, whose bit rates range from 300 Kbps to 1000 Kbps, after encoding. Thus, we opted for an encoding rate of $\alpha = 0.2$.

The evolutions of q_H as well as the number of active tasks on the server are plotted in Figure 3.12 for three different values of θ. It can be observed that starting from the initial value of 1, qH adapts to the incoming traffic as reaches steady-state behaviour after some initial transient behaviour. Smaller θ values yield smoother evolution, whereas larger θ values lead to faster response. In the remainder of this chapter, we use $\theta = 0.02$, as this value shows sufficient response speed with less fluctuations.

The high correlation between the number of tasks in the server and the q_H value is also evident from the plots. As jobs leave the system, the remaining ones see less queueing delay, and thus pull q_H down. This results in more offloading decisions, resulting in an increase in queueing delay, thus qH. This alternating behaviour is best seen in the first plot in Figure 3.12 with $\theta = 0.005$.

Next, we investigate the effect of the load on the server. The overall load can be expressed as $(N_H \lambda_H + N_L \lambda_L)\mathbf{E}[LX/f_s]$, where N_H and N_L denote the number of HU and LU, and λ_H and λ_L denote arrival rates for HU and LU, respectively. Although the system in consideration is one with rejections (non-offloading decisions) and thus stability is not a big concern, performance of the system is affected by the overall load. In Figure 3.13, the ratio of the tasks that are offloaded, mean task sojourn time, and mean energy consumption per task are plotted against varying mean interarrival times (1λ). As the mean inter-arrival time is increased, the load on the system is

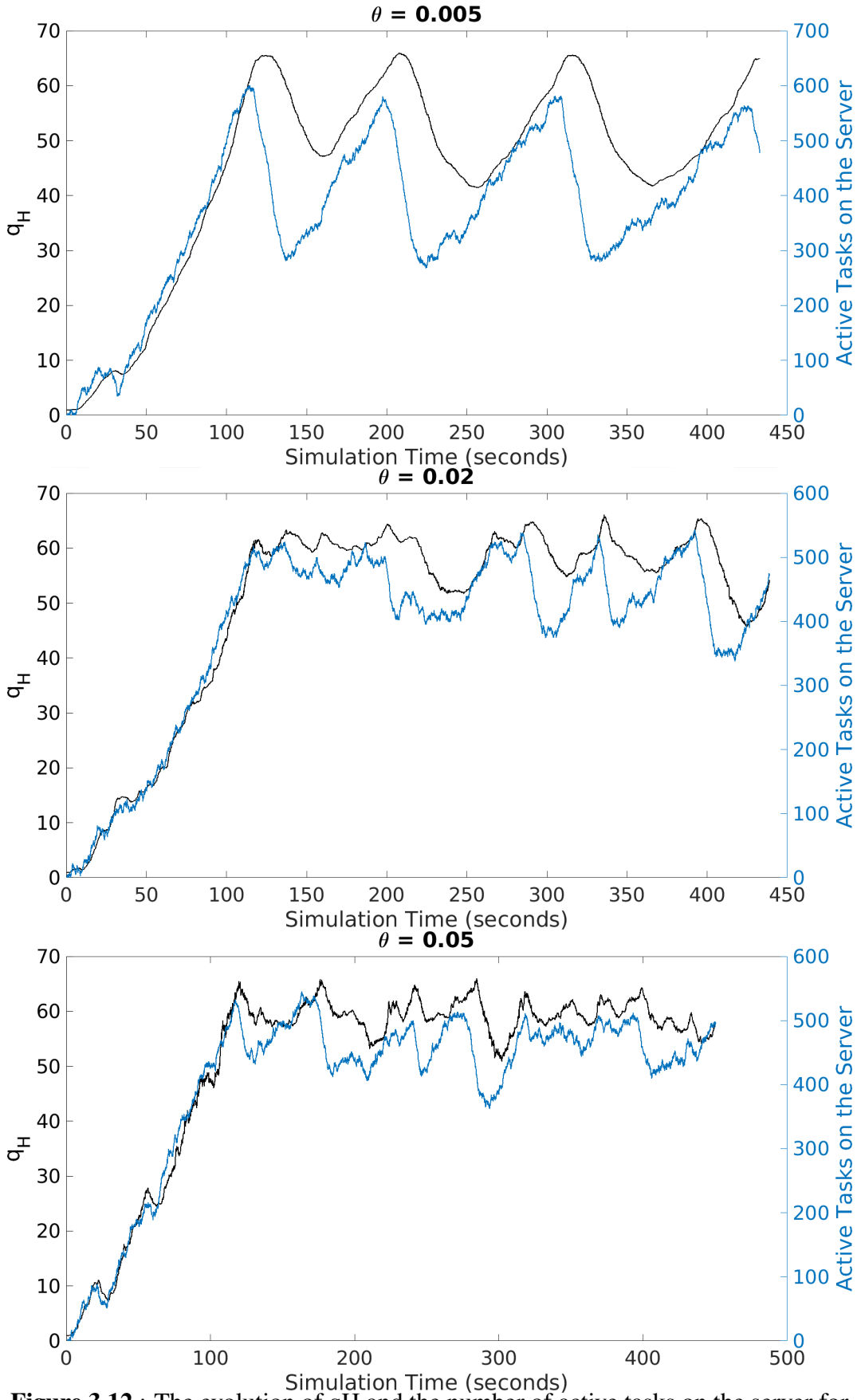

Figure 3.12 : The evolution of qH and the number of active tasks on the server for three different values of θ

decreased, and thus more and more LU tasks are offloaded. On the other hand, HU tasks are almost entirely offloaded. As the load is decreased, queueing delays are also decreased, leading to reduced mean task sojourn times. When it comes to energy consumption, we consider the consumption of the mobile user. Therefore, energy consumption consists of task execution energy and transmission energy for encoded data in case of local execution, whereas only transmission energy for raw data (which is larger than encoded data) is taken into account in case of offloading. Figure 3.13 shows that the performance of the system in terms of energy consumption is not trivial, and may also be incorporated into the offloading decision.

Next, we look at h, the number of MCs each HU task is executed per one MC of an LU task. We plot the system performance under varying values of h in Figure 3.14. From these plots, we can conclude that the value of h does not drastically effect system performance, although the mean task sojourn time for HU tasks seems to be cut in half when $h = 15$ compared to $h = 5$.

Then, we examine the effect of the ratio of HU in the total population of the users. In Figure 3.15, we plot the average task sojourn time and energy consumption versus varying HU ratios. From this figure, we can conclude that increasing the number of HU has a big impact on the performance and the performance degrades very rapidly before settling down due to reduced number of offloads. This shows the significance of the total load on the system.

Finally, in Figure 3.16, we plot the evolution of q_H as well as the number of active tasks on the server when the system starts with 10 seconds of inter-arrival time, but it becomes 2.5 seconds later on. The figure clearly shows q_H adapting to changing incoming traffic (around 280 seconds). Due to the increase in arrival rate, the number of tasks at the server increases, but then is steadied by the increasing q_H.

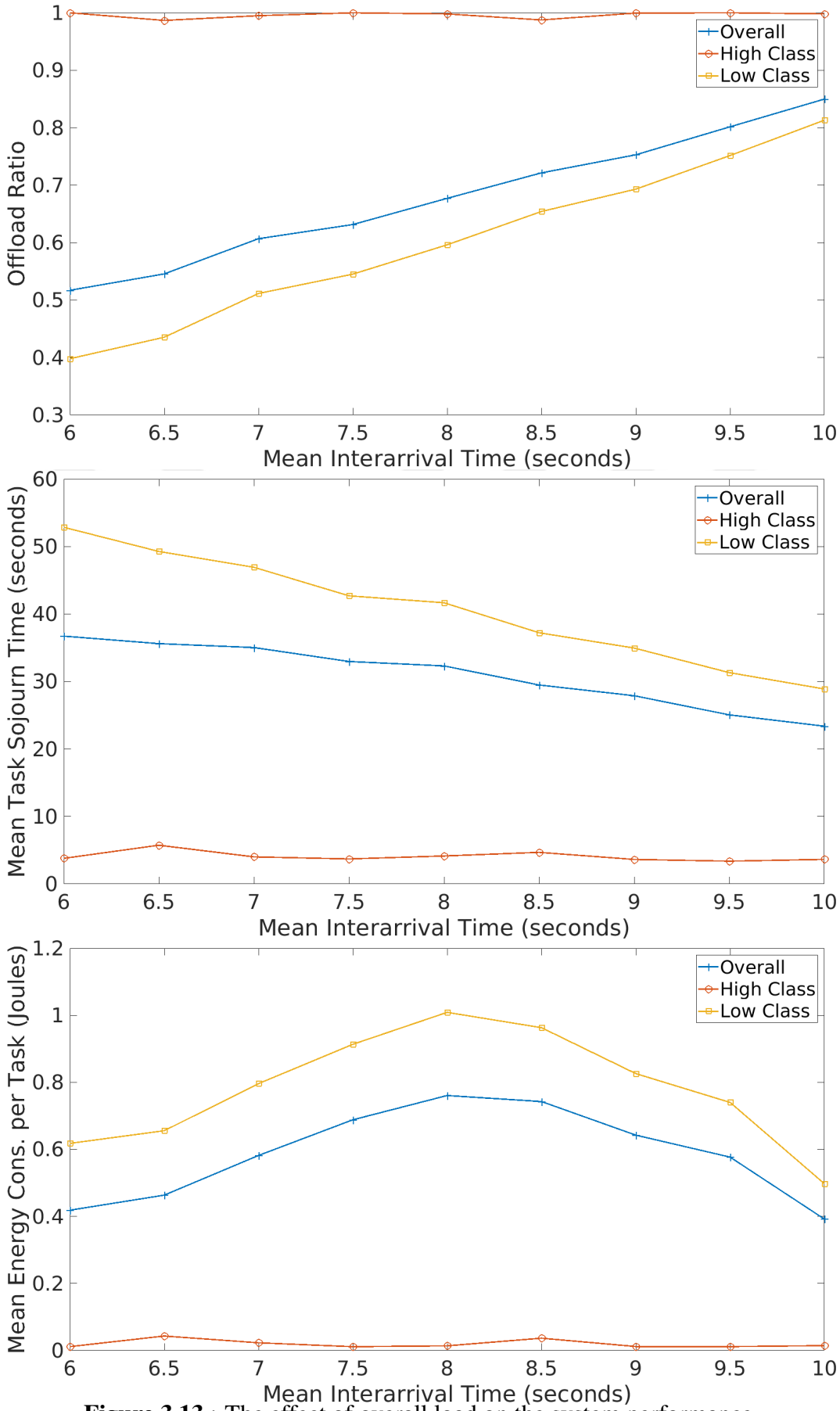

Figure 3.13 : The effect of overall load on the system performance.

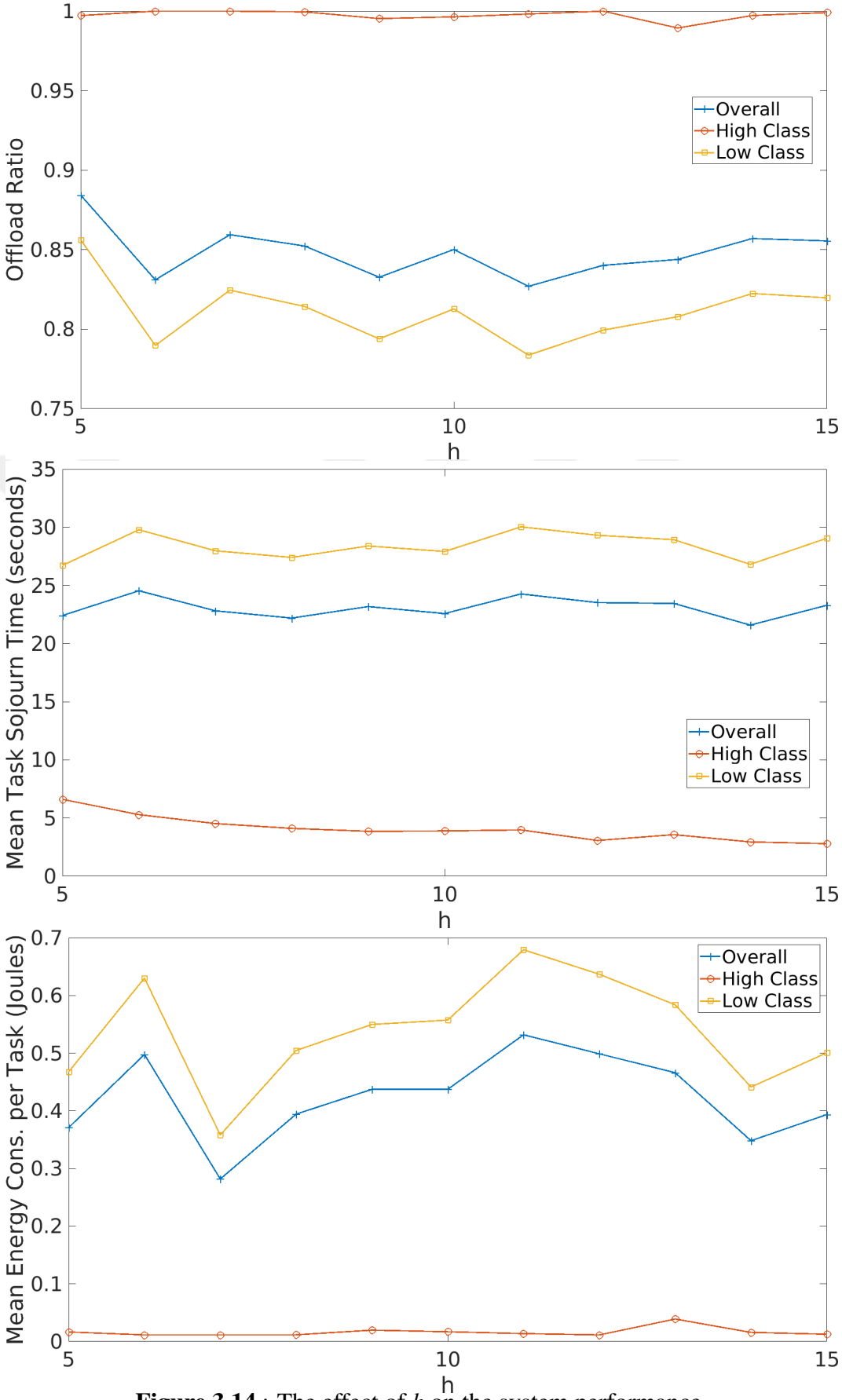

Figure 3.14 : The effect of h on the system performance.

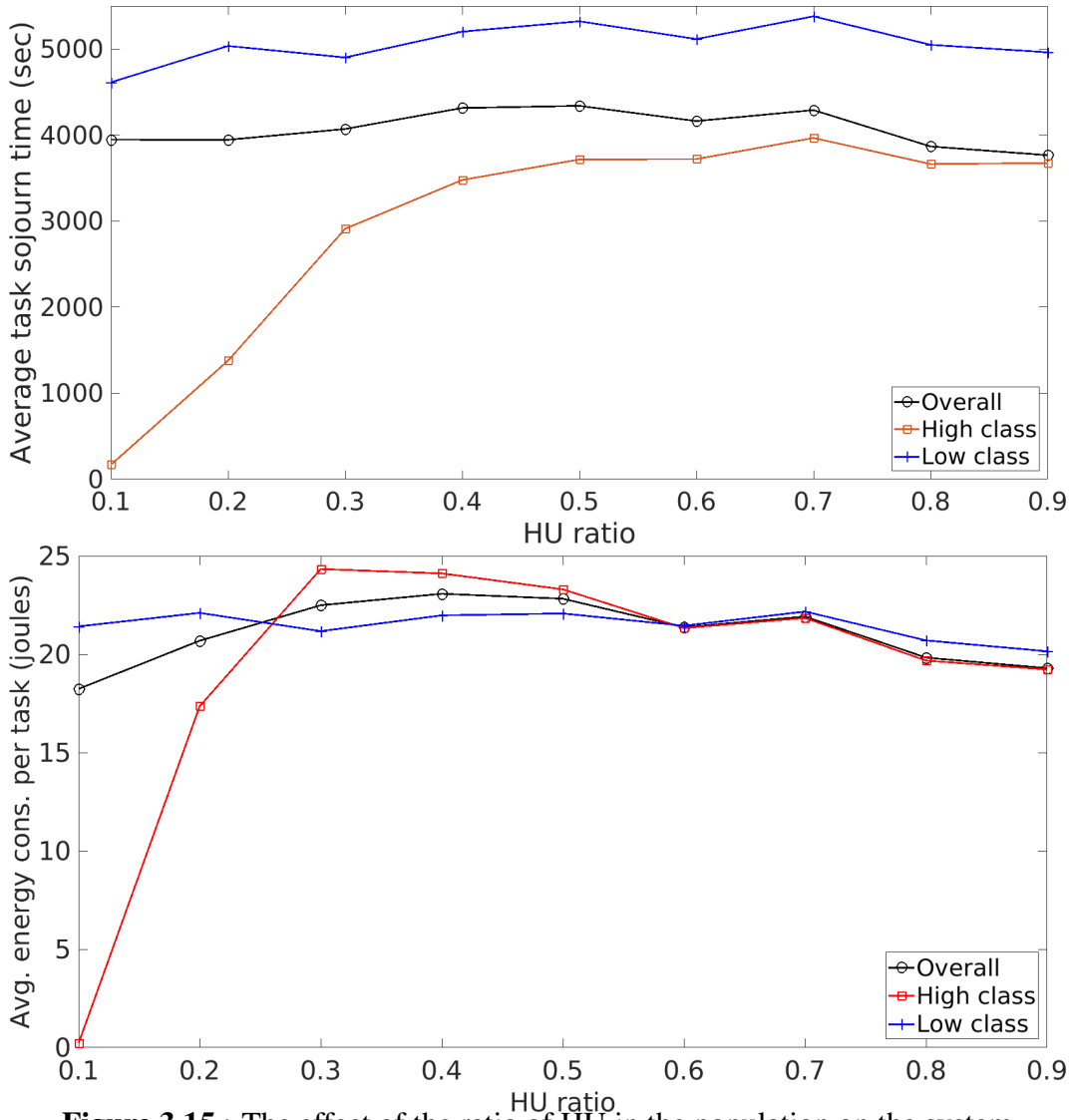

Figure 3.15 : The effect of the ratio of HU in the population on the system performance.

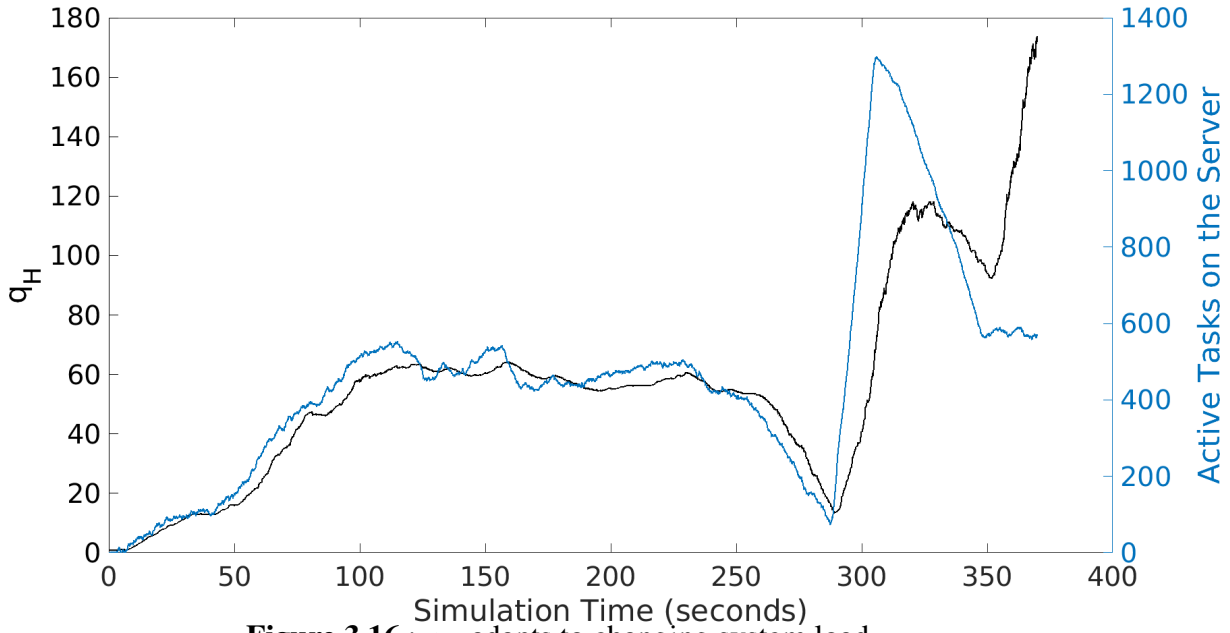

Figure 3.16 : q_H adapts to changing system load

4. CONCLUSION AND FUTURE WORK

In this study, we investigate a two-class MEC system where high-priority users (HU) and low-priority users (LU) have access to a single MEC server. The server executes the offloaded tasks in a weighted round-robin fashion so that the queueing performance can be modelled as a priority processor sharing (PS) system. A key property of the PS system is that the sojourn time of a task is proportional to its size, enabling us to formulate the sojourn time estimate of a job via a constant multiplier. This multiplier both models the queueing delay, and acts as an admission control parameter. Moreover, we propose an adaptive offloading decision algorithm so as to minimize the task sojourn time in the same system. The algorithm is based on the priority-processor sharing queueing discipline. Through numerical results presented, we conclude that there is an optimal value for this multiplier that also affects the energy efficiency of the entire system. Furthermore, we show that the prioritization factor that differentiates the service HU and LU obtain from the MEC server should be kept below a certain point to be inclusive to LU without harming HU performance. Conversely, one could argue that keeping HU and LU performance further apart gives incentive to clients to subscribe to the MEC service and thus, the service provider should increase the prioritization factor to obtain revenue. On the other hand, this can be refuted by claiming that such a policy may alienate customers and decrease revenue. Such economical and marketing related issues are beyond the scope, although we believe that service providers can devise pricing and operational policies based on the framework provided in this thesis.

Future work will be on incorporating energy consumption for both the mobiles and the server into the offloading decision, and investigation of the effect of parameters such as h and θ more thoroughly. Furthermore, a more realistic model would be on-off sources model, where a mobile generates tasks every second it is in the on state, and the

transitions between on-off states are modelled as a discrete or continuous-time Markov chain. This is left as future work.

REFERENCES

[1] **Jaber, Z.Q. and Younis, M.I.** (2014). Design and implementation of real time face recognition system (RTFRS), *International Journal of Computer Applications*, *94*(12).

[2] **Al-Shuwaili, A. and Simeone, O.** (2017). Energy-efficient resource allocation for mobile edge computing-based augmented reality applications, *IEEE Wireless Communications Letters*, *6*(3), 398–401.

[3] N. Wingfield, "Amazon's profits grow more than 800 percent, lifted by cloud services," The New York Times, Jul. 2016. [Online]. Available: http://www.nytimes.com/2016/07/29/technology/amazon-earnings-profit.html?r=0.

[4] **Andrews, J.G.**, **Buzzi, S.**, **Choi, W.**, **Hanly, S.V.**, **Lozano, A.**, **Soong, A.C. and Zhang, J.C.** (2014). What will 5G be?, *IEEE Journal on selected areas in communications*, *32*(6), 1065–1082.

[5] **Shi, W.**, **Cao, J.**, **Zhang, Q.**, **Li, Y. and Xu, L.** (2016). Edge computing: Vision and challenges, *IEEE Internet of Things Journal*, *3*(5), 637–646.

[6] **Mao, Y.**, **You, C.**, **Zhang, J.**, **Huang, K. and Letaief, K.B.** (2017). A survey on mobile edge computing: The communication perspective, *IEEE Communications Surveys & Tutorials*, *19*(4), 2322–2358.

[7] **Li, H.**, **Shou, G.**, **Hu, Y. and Guo, Z.** (2016). Mobile edge computing: progress and challenges, *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2016 4th IEEE International Conference on*, IEEE, pp.83–84.

[8] **Tran, T.X.**, **Hajisami, A.**, **Pandey, P. and Pompili, D.** (2017). Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges, *IEEE Communications Magazine*, *55*(4), 54–61.

[9] **Yi, S.**, **Li, C. and Li, Q.** (2015). A survey of fog computing: concepts, applications and issues, *Proceedings of the 2015 Workshop on Mobile Big Data*, ACM, pp.37–42.

[10] **Hu, Y.C.**, **Patel, M.**, **Sabella, D.**, **Sprecher, N. and Young, V.** (2015). Mobile edge computing—A key technology towards 5G, *ETSI white paper*, *11*(11), 1–16.

[11] **Chang, C.Y.**, **Alexandris, K.**, **Nikaein, N.**, **Katsalis, K. and Spyropoulos, T.** (2016). MEC architectural implications for LTE/LTE-A networks, *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*, ACM, pp.13–18.

[12] **Wang, S.**, **Zhang, X.**, **Zhang, Y.**, **Wang, L.**, **Yang, J. and Wang, W.** (2017). A survey on mobile edge networks: Convergence of computing, caching and communications, *IEEE Access*, *5*, 6757–6779.

[13] **Ahmed, A. and Ahmed, E.** (2016). A survey on mobile edge computing, *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pp.1–8.

[14] **Ahmed, E. and Rehmani, M.H.** (2017). Mobile Edge Computing: Opportunities, solutions, and challenges, *Future Generation Computer Systems*, *70*, 59 – 63.

[15] **Taleb, T.**, **Samdanis, K.**, **Mada, B.**, **Flinck, H.**, **Dutta, S. and Sabella, D.** (2017). On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration, *IEEE Communications Surveys & Tutorials*, *19*(3), 1657–1681.

[16] **Liu, H.**, **Eldarrat, F.**, **Alqahtani, H.**, **Reznik, A.**, **de Foy, X. and Zhang, Y.** (2017). Mobile edge cloud system: Architectures, challenges, and approaches, *IEEE Systems Journal*.

[17] **Beck, M.T.**, **Werner, M.**, **Feld, S. and Schimper, S.** (2014). Mobile edge computing: A taxonomy, *Proc. of the Sixth International Conference on Advances in Future Internet*, Citeseer, pp.48–55.

[18] **Sabella, D.**, **Vaillant, A.**, **Kuure, P.**, **Rauschenbach, U. and Giust, F.** (2016). Mobile-edge computing architecture: The role of MEC in the Internet of Things, *IEEE Consumer Electronics Magazine*, *5*(4), 84–91.

[19] **Chen, X.**, **Jiao, L.**, **Li, W. and Fu, X.** (2016). Efficient multi-user computation offloading for mobile-edge cloud computing, *IEEE/ACM Transactions on Networking*, *24*(5), 2795–2808.

[20] **Di Lorenzo, P.**, **Barbarossa, S. and Sardellitti, S.** (2013). Joint optimization of radio resources and code partitioning in mobile edge computing, *arXiv preprint arXiv:1307.3835*.

[21] **Ren, J.**, **Yu, G.**, **Cai, Y. and He, Y.** (2017). Latency optimization for resource allocation in mobile-edge computation offloading, *arXiv preprint arXiv:1704.00163*.

[22] **Chen, X.** (2015). Decentralized computation offloading game for mobile cloud computing, *IEEE Transactions on Parallel and Distributed Systems*, *26*(4), 974–983.

[23] **Molina, M.**, **Muñoz, O.**, **Pascual-Iserte, A. and Vidal, J.** (2014). Joint scheduling of communication and computation resources in multiuser wireless application offloading, *Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014 IEEE 25th Annual International Symposium on*, IEEE, pp.1093–1098.

[24] **Lyu, X.**, **Tian, H.**, **Sengul, C. and Zhang, P.** (2017). Multiuser joint task offloading and resource optimization in proximate clouds, *IEEE Transactions on Vehicular Technology*, *66*(4), 3435–3447.

[25] **Chen, M.H.**, **Liang, B. and Dong, M.** (2016). Joint offloading decision and resource allocation for multi-user multi-task mobile cloud, *Communications (ICC), 2016 IEEE International Conference on*, IEEE, pp.1–6.

[26] **Chen, M.H.**, **Liang, B. and Dong, M.** (2017). Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point, *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, IEEE, pp.1–9.

[27] **Chen, M.H.**, **Dong, M. and Liang, B.** (2016). Multi-user mobile cloud offloading game with computing access point, *Cloud Networking (Cloudnet), 2016 5th IEEE International Conference on*, IEEE, pp.64–69.

[28] **Zhang, K.**, **Mao, Y.**, **Leng, S.**, **Zhao, Q.**, **Li, L.**, **Peng, X.**, **Pan, L.**, **Maharjan, S. and Zhang, Y.** (2016). Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks, *IEEE Access*, *4*, 5896–5907.

[29] **Mao, Y.**, **Zhang, J. and Letaief, K.B.** (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices, *IEEE Journal on Selected Areas in Communications*, *34*(12), 3590–3605.

[30] **Hoang, D.T.**, **Niyato, D. and Wang, P.** (2012). Optimal admission control policy for mobile cloud computing hotspot with cloudlet, *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, IEEE, pp.3145–3149.

[31] https://www.igi-global.com/dictionary/offloading/54962.

[32] **Kobayashi, H. and Mark, B.L.** (2009). *System modeling and analysis: Foundations of system performance evaluation*, Pearson Education India.

[33] **Kleinrock, L.** (1967). Time-shared systems: A theoretical treatment, *Journal of the ACM (JACM)*, *14*(2), 242–261.

[34] **Kumar, K. and Lu, Y.H.** (2010). Cloud computing for mobile users: Can offloading computation save energy?, *Computer*, *43*(4), 51–56.

[35] **Melendez, S. and McGarry, M.P.** (2017). Computation offloading decisions for reducing completion time, *Consumer Communications & Networking Conference (CCNC), 2017 14th IEEE Annual*, IEEE, pp.160–164.

[36] **Miettinen, A.P. and Nurminen, J.K.** (2010). Energy Efficiency of Mobile Clients in Cloud Computing., *HotCloud*, *10*, 4–4.

[37] **You, C.**, **Huang, K.**, **Chae, H. and Kim, B.H.** (2017). Energy-efficient resource allocation for mobile-edge computation offloading, *IEEE Transactions on Wireless Communications*, *16*(3), 1397–1411.

[38] YouTube, "Live encoder settings, bitrates, and resolutions," Accessed: 17-March-2018. [Online]. Available: https://support.google.com/youtube/answer/2853702?hl=en.

APPENDICES

CURRICULUM VITAE

Name Surname: Kahlan Faaq Hasan

Place and Date of Birth: Iraq / April-1-1991

E-Mail: kahlan87@yahoo.com

EDUCATION:

- **B.Sc.:** 2013, University of Tikrit, College of Computer Science and Mathematics , Computer Science

PROFESSIONAL EXPERIENCE AND REWARDS:

- Certificate of participation for *participating in the Database programming with software Engineering Topics Class* presented by BALL STATE UNIVERSITY, April 2013

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Accepted paper entitled "Effect of Queueing Delay and Service Discrimination on Offloading Performance in Two-Class Mobile Edge Computing Systems", SIU conference, Izmir, TURKEY, May 2-5 2018.
- Accepted paper entitled "An Adaptive Offloading Decision Scheme in Two-Class Mobile Edge Computing Systems", TSP conference, Athens, GREECE, July 4-6 2018