UNIVERSITY OF CALIFORNIA, SAN DIEGO

Adaptive Rendering and Scheduling Techniques to Enable Cloud Mobile Gaming

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Computer Engineering)

by

Shaoxuan Wang

Committee in charge:

Professor Sujit Dey, Chair
Professor David Kriegman
Professor Bill Lin
Professor Ramesh Rao
Professor Geoffrey Voelker

2013

This Dissertation of Shaoxuan Wang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2013

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

There have been a large number of people whose support and guidance throughout years at UCSD made this dissertation possible. I take this opportunity to express my sincere gratitude for all of them. Though I cannot hope to enumerate, let alone repay, all of whom I am indebted to, I still insist on naming a few in print.

The first people I would like to thank is my advisor, Prof. Sujit Dey, who has been a constant source of encouragement and support over these years. I thank him for being extremely patient with me, for teaching me how to conduct research and present ideas, and for being enthusiastic in venturing into new topics of research. His involvement in every aspect of my work, from discussing algorithms to shaping my writing, was a defining factor in ensuring the quality of the final manuscript.

In addition, I would like to thank Yao Liu, my collaborator and friend, for his contributions, insights and feedback, especially for the work presented in Chapter 4. I also thank the members of my thesis committee, Professors Bill Lin, David Kriegman, Geoffrey Voelker, and Ramesh Rao for providing valuable suggestions and feedback. This work has also improved in quality thanks to detailed feedback provided by the anonymous reviewers of the related conference and journal papers.

I gratefully acknowledge the financial support provided by the Jacobs Engineering Fellowship that made my doctoral studies possible. Throughout my graduate studies, I have been fortunate to have the chance to work with a wonderful group of co-workers in the MESDAT lab at UCSD. I am very grateful to all the past and present members of our lab for providing an interesting and encouraging working experience. Our interactions and discussions made the day-to-day graduate student life enjoyable.

I owe a special debt of gratitude to all my friends whose encouragement and moral support kept me sane for all these years and helped me bear the various ups and downs of graduate life. I take this opportunity to thank all the new friends who made me feel at home in a foreign land, and old ones who, often from across continents, made me think that I had never left home. The list of all who deserve mention is far too long to include here, but special thanks go to Chong Zhao, Wenbo Zhao, Shoubhik Mukhopadhyay, and Naomi Ramos for being incredibly patient listeners and providing valuable feedback on everything from research, writing and life.

I dedicate this dissertation to my parents. They are the ones who taught me all the really important things in life, ensured that I got a good education against considerable odds, and were the first to believe in me. I would not have been able to dream of embarking upon this journey if it were not for their encouragement and support.

At last, I want to express my special thanks to my dear wife Hanlu Lu. I am so lucky to know my wife in the first year of my Ph.D study. Because of her, my last six years have been so great, especially that we had our lovely daughter Julia Wang born in 2010. Yes, I have ups and downs during my Ph.D study. And I have faced challenges that I never dreamed I'd face. But my wife has always been there to support me and to support my family. She have strengthened me when my faith was weak. Without her love, sacrifice, and unfathomable patience, I would never have been able to complete my Ph.D.

The text of the following chapters, in part or in full, is based on material that has been published in conference proceedings or journals, or is pending publication in journals, or is in review.

Chapter 2 is based on material that has been published in ACM Mobile Computing and Communications Review (MC2R) (S. Wang, S. Dey, “Cloud Mobile Gaming: Modeling and Measuring User Experience in Mobile Wireless Networks,” ACM SIGMOBILE MC2R, Jan. 2012) and material published in IEEE Global Communications Conference (Globecom) (S. Wang, S. Dey, “Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach,” IEEE Globecom, Honolulu, Dec. 2009).

Chapter 3 is based on material that has been published in IEEE Transactions on Multimedia (TMM) (S. Wang, S. Dey, “ Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications,” IEEE Transactions on Multimedia, Jun. 2013) and material published in IEEE Global Communi1cations Conference (Globecom) (S. Wang, S. Dey, "Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming,” IEEE Globecom, Miami, Dec. 2010).

Chapter 4 is based on material that has been published in IEEE International Conference on Communications (ICC) (S. Wang, Y. Liu, S. Dey, “Wireless Network Aware Cloud Scheduler for Scalable Cloud Mobile Gaming”, IEEE ICC, Jun. 2012) and material submitted to IEEE Transactions on Networking (S. Wang, Y. Liu, S. Dey, “Mobile Cloud Scheduling: Scheduling Heterogeneous Network and Cloud Resources to Enable Scalable Mobile Cloud Computing”).

I was the primary researcher and author of each of the above publications, and the co-authors listed in these publications collaborated on, or supervised the research which forms the basis for these chapters.

VITA

1999-2003	B.S., Electrical Engineering, Peking University, Beijing, China
2004-2007	M.S., Electrical Engineering (in Wireless Communications), Peking University, Beijing, China
2007-2010	Research Assistant, Dept. of Electrical and Computer Engineering, University of California, San Diego
2008	Summer Intern, Palm Corp., Sunnyvale, California
2009	Summer Intern, Broadcom Corp., San Diego, California
2010-2013	Senior Staff Engineer, Broadcom Corp., San Diego, California
2013	Ph.D, Electrical Engineering (Computer Engineering), University of California, San Diego

PUBLICATIONS

S. Wang, Y. Liu, S. Dey, "Mobile Cloud Scheduling: Scheduling Heterogeneous Network and Cloud Resources to Enable Scalable Mobile Cloud Computing," *IEEE Transactions on Networking*, 2013 (in review).

S. Dey, Y. Liu, S. Wang, Y. Lu, "Addressing Response Time of Cloud-based Mobile Applications," *ACM Mobile Cloud 2013*, (in review).

S. Wang, S. Dey, "Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications," *IEEE Transactions on Multimedia*, vol. 15, no. 4, Jun. 2013.

S. Wang, Y. Liu, S. Dey, "Wireless Network Aware Cloud Scheduler for Scalable Cloud Mobile Gaming", *in Proc. of IEEE ICC,* Ottawa, Jun. 2012.

S. Wang, S. Dey, “Cloud Mobile Gaming: Modeling and Measuring User Experience in Mobile Wireless Networks,” *ACM SIGMOBILE MC2R*, vol. 16, issue 1, Jan. 2012, pp. 10-21.

Y. Liu, S. Wang, S. Dey, “Modeling, characterizing, and enhancing user experience in Cloud Mobile Rendering”, *in Proc. of IEEE ICNC, Maui,* Jan. 2012.

S. Wang, S. Dey, "Addressing Response Time and Video Quality in Remote Server Based Internet Mobile Gaming," in *Proc. of IEEE WCNC*, Sydney, Mar. 2010.

S. Wang, S. Dey, "Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming,” *in Proc. of IEEE GLOBECOM,* Miami, Dec. 2010.

S. Wang, S. Dey, “Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach,” *in Proc. of IEEE GLOBECOM*, Honolulu, Dec. 2009.

ABSTRACT OF THE DISSERTATION

Adaptive Rendering and Scheduling Techniques to Enable Cloud Mobile Gaming

by

Shaoxuan Wang

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California, San Diego, 2013

Professor Sujit Dey, Chair

This dissertation studies a new cloud server based approach for mobile gaming, termed Cloud Mobile Gaming (CMG), that enables 3D, multiplayer, Internet video games on mobile phones — the kinds of games that today require a powerful PC tethered to the Internet. We flipped the traditional client-server architecture for PC-based Internet games on its head and looked to cloud computing, instead of putting most of the storage and computational burden of the games on the mobile device. Though promising, the new CMG approach will impose other challenges, namely to ensure high mobile gaming user experience (as two-way gaming data including video will need to be transmitted through wireless networks in real time), and ensure scalability in terms of cloud servers and network bandwidth, so as to ensure economic viability of the approach. In our work, we have developed key new technologies to address these challenges.

We first developed and validated a Mobile Gaming User Experience model, which can quantitatively measure user perceived mobile gaming experience. Next, we develop an adaptive rendering technique which can simultaneously vary the richness and complexity of graphic rendering to adapt the communication and computing needs of each CMG session in responding to the dynamic conditions of the wireless networks and cloud server. Finally, we present a mobile cloud scheduling approach which can allocate resources to meet user experience requirements while maximizing the number of users which can be scheduled concurrently and minimizing cloud cost.

The experimental results presented demonstrate that our proposed adaptive rendering technique and the mobile cloud scheduling approach can efficiently address the challenges imposed by the wireless network and cloud server, ensuring the user perceived gaming quality as well as the scalability for the CMG approach. We believe the techniques proposed in this work will not only enable rich Internet cloud gaming, but also other rich mobile multimedia applications using the cloud.

Chapter 1

Introduction

The emergence of new and more capable mobile devices, including smart phones, tablets, and netbooks, along with the steady deployment of broadband wireless networks is making mobile access to rich Internet sites a reality. This technological progress opens up a new possibility: the ability to play rich Internet games produced for PCs on wireline networks from mobile devices. Enabling mobile Internet gaming will significantly change the experience of mobile users from the thin, single player gaming possible today to a rich, multi-player Internet gaming experience of users' familiar games from any location with the proper access. It will also open up the possibility for mobile service providers and Internet game developers to translate the tremendous growth experienced in recent years in Internet PC games to the fast emerging mobile eco-system. However, due to the inherent hardware constraint of mobile devices such as memory and graphics processing, the goal might be difficult to achieve using the current client-server gaming architecture for PC-based Internet games, since most of the storage and computational burden of the game lies with the client device.

Instead, it may be promising to investigate a cloud server based mobile gaming approach termed *Cloud Mobile Gaming (CMG)*, where a gaming server is responsible for executing the appropriate gaming engine and streaming the resulting gaming video to the client device, while the mobile devices only communicate the user's gaming commands to the cloud server. In our work, we first investigate the potential major challenges for this new CMG approach focusing on response time, user experience, mobile network bandwidth, cloud computing cost, and scalability to large number of CMG users. Subsequently, we propose and develop several techniques to address the challenges encountered. We believe the techniques proposed in this work will help to successfully launch the CMG approach and lead to a quantum leap in the perception and use of mobile devices for Internet gaming.

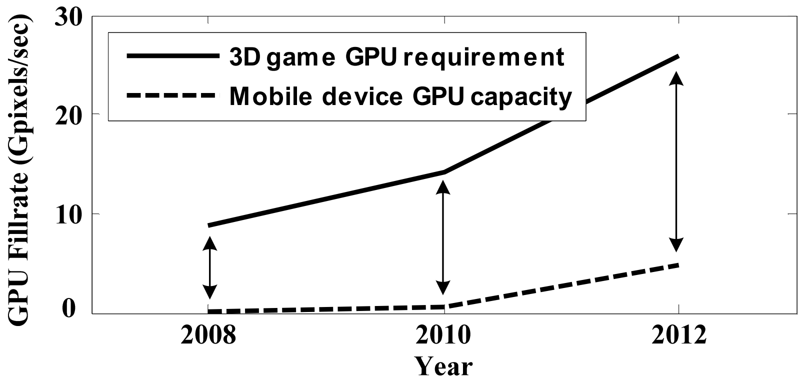
In this chapter, we first look at the early trends and advantages for this new CMG approach, and present an overview of the CMG approach. Next we analyze the challenges that need to be addressed to make the CMG approach viable. Finally, we outline the contributions made by this thesis, in applying the proposed techniques to the CMG approach. We conclude with an overview of the remaining chapters.

1.1 Cloud Mobile Gaming: Advantages and Overview

Over the last few years, there has been an increased number of applications which have "migrated to the cloud", as well as new cloud-based applications that have become recently popular. Most of the early adopters of the cloud have been enterprise applications and IT departments. According to the recent research report from Analysis Mason [AM13], revenue from mobile enterprise cloud-based applications and services is expected to rise from nearly $18.3 billion in 2012 to $31.9 billion in 2017. Similar motivations which have driven mobile enterprise cloud services are also driving adoption of mobile consumer cloud services: the ability to access media from anywhere- any device, platform, and network. According to the Cisco virtual network index mobile forecast [CIS13], the cloud video applications and services such as Netflix, YouTube, Pandora, and Spotify will account for 84 percent of total mobile data traffic in 2017, as compared to 74 percent at the end of 2012. In other words, mobile cloud video traffic will grow 14-fold from 2012 to 2017. And according to Juniper Research, revenues from consumer cloud mobility services, initially driven by cloud based music and video storage and download services like the ones recently launched by Amazon's Cloud Drive and Apple's iCloud, are expected to reach $6.5 billion per year by 2016 [JUN11].

Besides such storage and download services, a big boost to mobile consumer cloud services will come from a major shift in the mobile applications market, primarily from native applications to ones based on *Cloud Mobile Computing*: utilizing the computing and storage resources available in the cloud, thereby enabling the use of cutting edge multimedia technologies that are much more computing and storage intensive than what mobile devices can offer, and thus enabling a much richer media experiences than what current native applications can offer.

One promising cloud mobile computing application with the potential to significantly enhance the media experience of mobile users is *Cloud Mobile Gaming* (*CMG*). Despite the progress in the capabilities of mobile devices, there is a widening gap with the growing requirements of the latest 3D video games from what can be supported by today's and the near future's mobile devices, including tablets. Figure 1.1 shows this widening gap from 2008 to 2012, in terms of the recommended GPU requirements [3DG] of the most demanding games in those years such as Call of Duty 4 in 2008, Call of Duty 7 in 2010, and Battlefield 3 in 2012, and the GPU capabilities [ANA] of the popular smartphones in those years: iPhone 3G, iPhone 4, and iPhone 5 respectively. CMG can bridge this gap by allowing game rendering to be executed in the cloud instead of on the mobile device, thereby, potentially enabling mobile users to play the same rich Internet games available to high-end PC users.



	2008	2010	2012
3D game GPU requirements	Call of Duty 4	Call of Duty 7	Battlefield 3
	8.8 GPixels/sec	14.16 GPixels/sec	25.9 GPixels/sec
Smartphone GPU capability	Iphone 3G	Iphone 4	Iphone 5
	0.135 pixels/sec	0.5 Gpixels/sec	4.875 Gpixels/sec

Figure 1.1. Growing gap between (recommended) GPU requirement of rendering-based applications and GPU capability of mobile devices.

Besides eliminating the hardware constraint of mobile devices, the CMG approach also provides a new capability to solve the cross-platform issue for mobile gaming, a fundamental problem which constrains mobile gaming from being very successful. It allows mobile users to play an Internet game without the installation of a game engine on his/her platform. With one game software installed on the cloud server, heterogeneous mobile devices with different operating systems and hardware capacities can play the same game via the CMG approach. This will significantly relieve game developers from the expensive cycle of developing device and platform specific mobile versions for the same game.

Figure 1.2 shows the overall CMG approach. In the conventional Internet multiplayer game architecture, the game synchronization server is the key component that runs the fundamental game logic in a single process, and maintains the game database. Besides, it

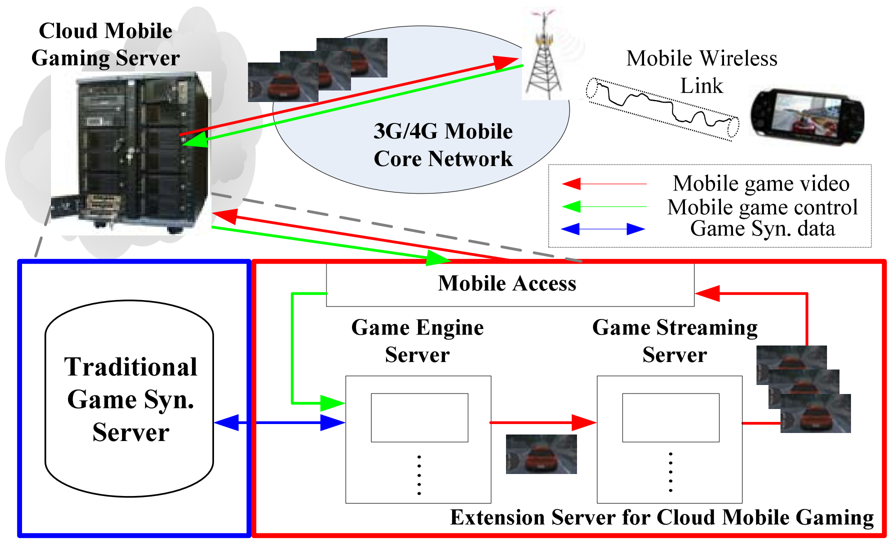


Figure 1.2. Overview of Cloud Mobile Gaming architecture and data/control flow.

maintains the connections for users, delivers the interactive messages as well as updates the active users' game data simultaneously. To support mobile gaming on thin clients, in the CMG system we extend the conventional game synchronization server with two key components: game engine server and game streaming server. When a player accesses the game server from a mobile device, the connection will first be confirmed by the game synchronization server. Subsequently, it initializes a game engine server and a game streaming server for this mobile device/user. The game engine server then loads the client's account information and game data from game synchronization server, and begins to process the game logic and user data to render the raw game video. The generated raw game video is encoded by the game streaming server, and finally sent to the mobile client via the wireless connection. On the other hand, the mobile user's inputs are delivered to the CMG server and accepted by the game synchronization server directly. Figure 1.2 shows the control flow (green) and data/video flow (red) for a CMG video game session.

1.2 Challenges of Cloud Mobile Gaming

Though the CMG approach is promising in terms of enabling mobile users to play rich Internet games on any mobile device without the need to download the game engines, several challenges exist and must be addressed in order to make the CMG approach feasible.

1.2.1 Mobile Gaming User Experience

The first challenge for the CMG approach is to ensure *Mobile Gaming User Experience (MGUE)*. Firstly, unlike traditional gaming, the game video in the CMG approach must first be compressed and transmitted to mobile user devices over bandwidth constrained and error-prone wireless networks which may cause network packet loss. The user perceived gaming video quality may suffer an unexpected quality loss besides the compression loss introduced when gaming video is compressed. Secondly, and more importantly, unlike other applications such as video streaming and video download, video games are highly interactive applications, demanding very fast response times. As opposed to conventional server-client gaming architecture, where the game is executed right on the client, the new CMG approach introduces the possibility of a significantly higher response time, from the time a gaming command is issued on a mobile device, to the time the video is streamed back to the device. This is evidenced by the data presented in Figure 1.3, which shows the measured uplink delay, downlink delay, and round-trip response time.

The experiments for Figure 1.3 are conducted in a commercial 3G network and under three different conditions: when the network was not loaded (data collected at midnight), when the network was loaded (data collected at 5 pm), and when the network was loaded and signal conditions were not strong (data collected at 6 pm, and inside a building). From Figure 1.3, we can observe that when the network is not loaded and signal strength is strong, the

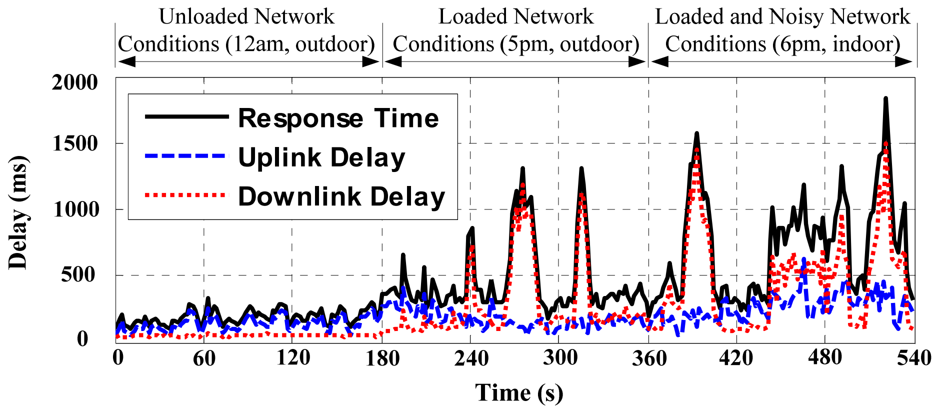


Figure 1.3. Delay and response time measured in a CMG session at different time during a day.

CMG application can achieve a low response time. However, when the 3G network is loaded, or when the user is in a noisy network condition with poor signal strength, there are significant increases in uplink, downlink, and round-trip response time, which will lead to a significantly adverse impact on the quality of gaming experience.

Besides the video quality and network response time, the CMG server over-utilization encountered and characteristics of the mobile device may add to the entire gaming response time, and hence affect user experience. To ensure proper understanding and consideration of the effects of different networks, video compression parameters, as well as CMG server and mobile device factors on CMG application performance, we need to develop techniques to model, and quantitatively measure and monitor the mobile gaming user experience in real time.

1.2.2 Fluctuating and Constrained Mobile Network Bandwidth.

It has been well established that wireless networks are characterized by rapid fluctuations of the network bandwidth experienced by users. For example, in 3G and 4G cellular networks, while various techniques significantly reduce channel errors experienced by

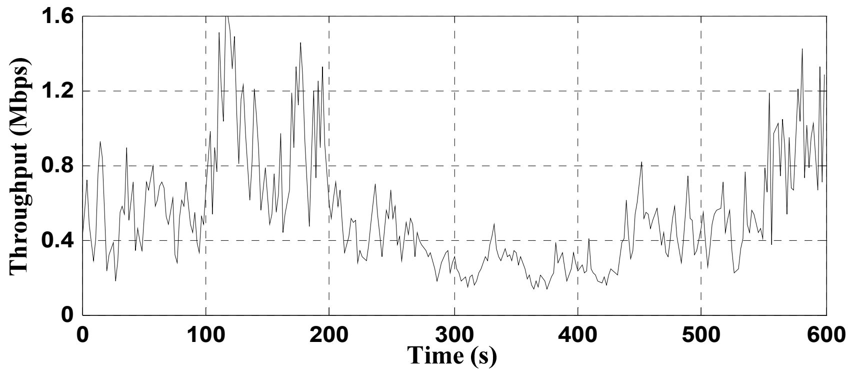


Figure 1.4. Maximum network downloading throughput measured in the test environment.

applications, they produce rapidly changing channel rates, leading to significant spatial and temporal variations of the mobile network bandwidth [BLB06] [LSM08] [TLL07]. To understand and characterize the available network bandwidth for a mobile user, we have conducted experiments using a commercial 3G mobile network. During the experiment, we measured the maximum network downloading throughput on a mobile client while we roamed with this device at different locations inside a building. These locations were selected as they displayed different network conditions such as signal-to-noise ratio received by the mobile device. Figure 1.4 shows a representative sample of data. From Figure 1.4, we have mainly two observations: a) the maximum network downloading throughput is changing rapidly. It can drop more than 60% in a few seconds; b) the available downloading bandwidth for a mobile user may become very constrained to as low as 0.2Mbps as shown in Figure 1.4.

This inherent characteristic (fluctuating and constrained network bandwidth) of the mobile network imposes a challenge for the CMG approach, as the gaming videos streamed from CMG servers to mobile devices can be subject to high and unpredictable congestion delay and packet loss, leading to an undesirable increase in response time, besides the adverse impact on the quality of the video streamed.

Table 1.1. Cloud pricing structures and cloud cost for a WoW CMG session.

Cloud Instances Type	Computing Price	Storage Price	Network Price	$\text{Cost}_{\text{CLOUD}}$
On-demand cluster compute instances	$0.048 /EC2 CU per hour	$1.39e-4 /GB per hour	$0.054 /Mb	$0.1285 per hour
Heavy utilization reserved cluster compute instances	$0.00887 /EC2 CU per hour	$1.39e-4 /GB per hour	$0.054 /Mb	$0.0503 per hour

1.2.3 Cloud Service Cost and Scalability

One of the primary advantages of using cloud services is to eliminate capital expenses, increase dependability on the elasticity of cloud computing, and determine the cloud utility or pricing model to scale to varying capacity needs. However, there will be challenges faced by computing intensive CMG application in terms of prohibitively high operating expenses when using current cloud pricing models.

Table 1.1 shows the cloud pricing structures of the Amazon Elastic Compute Cloud (EC2) [EC2] including computing price, storage price, and network price. It also shows the cost per hour of a VGA resolution cloud mobile gaming session of the popular Multiplayer Online Role-Playing Game (MMORPG), World of Warcraft (WoW), assuming each session needs 2 Compute Units (CU), 1GB storage space, and 600kbps of network bandwidth. AWS mainly provides two types of cloud instances: on-demand instances and reserved instances. On-demand instances allow customers to pay for compute capacity by the hour with no long-term commitments, while reserved instances give customers the option to make a low, one-time payment for each instance customers want to reserve and in turn receive a significant discount on the hourly charge for each instance. From Table 1.1, we can observe the

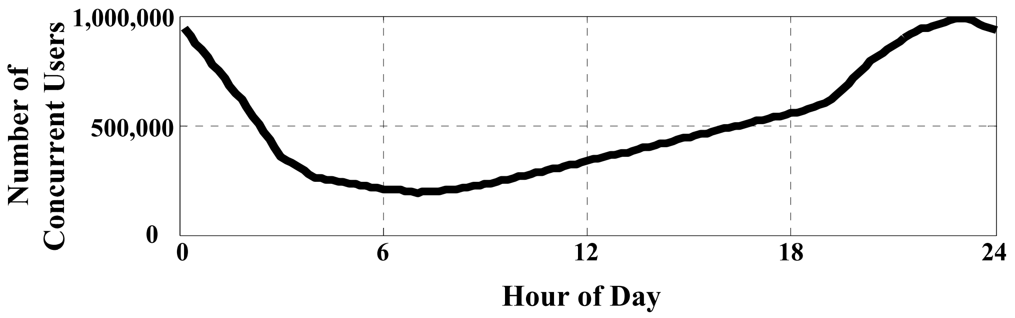


Figure 1.5. Daily concurrent user pattern for game WoW.

$\text{Cost}_{\text{CLOUD}}$ of using on-demand cloud instances is much higher (more than 2x) than the $\text{Cost}_{\text{CLOUD}}$ of using reserved cloud instances, due to the higher computing price of on-demand cloud instances. Cloud providers which consume a large amount of cloud resource will definitely want to purchase the reserved instances to get a better price discount. This infers that the capacity of reserved cloud resource by the CMG provider is limited.

Assuming an average playing time of 23 hours/week [WBD], from Table 1.1 the monthly operating expense for a cloud mobile gaming provider using public cloud platforms would be about \$12/month per WoW player if using on-demand cloud instance as compared to \$5/month if using reserved cloud instance. Considering the typical subscription prices (for example, current price of WoW prepaid card is \$15/month), this level of operating expense would be too high, even to support VGA resolution.

Moreover, considering the CMG provider must serve millions of users in a given day, the total cloud service cost could become extremely high. Figure 1.5 shows the estimate of concurrent WoW online gamers according to hours of day in China. Our estimation is based on a study showing daily usage patterns for WoW gamers [LCC11], and extrapolating with the number of WoW peak concurrent users in China, which has maintained steady at 1 million for several years [AT10]. The daily operating cost for the concurrent user profile shown in Figure

1.5 can be as high as \$1.53M even to support VGA resolution WoW gaming, which questions the scalability of cloud mobile gaming, as the level of concurrency needed support may be much higher to support all other popular games. Clearly, techniques must be developed to address the cloud cost and scalability challenges faced by CMG services in using public clouds.

1.3 Contributions and Overview

The main contributions made by this thesis are threefold. First, to ensure proper understanding of the effects of different impact factors on user perceived gaming experience, we develop and validate a *Mobile Gaming User Experience (MGUE)* model through controlled subjective testing. Several impact factors have been taken into account including the game video settings, the server and client conditions, the wireless network conditions, and the game genres. We also develop a software prototype which can in-service measure impact factors and simultaneously report MGUE. To the best of our knowledge, this is the first work to characterize and model mobile gaming user experience for a CMG approach. The MGUE model developed in this chapter is helpful for researchers or mobile gaming service providers to assess the performance of this new CMG technique. In addition, there is also a trend to deploy many interactive multimedia applications into the cloud server, such as virtual reality and augmented reality. We believe our approach to develop a MGUE model outlined in this chapter can be potentially used to develop Quality of Experience (QoE) models for these new cloud server based applications.

Second, we wish to ensure the scalability and address the challenges imposed by fluctuating and constrained wireless network bandwidth rates and expensive cloud service cost, thus we have developed several application adaptation techniques, including a dynamic game rendering adaptation and a joint rendering and encoding adaptation technique. There are two aspects to the novelty of this work. The proposed dynamic game rendering adaptation can vary the graphic rendering settings to adapt both communication and computation needs of a CMG session, thereby addressing both wireless network bandwidth constraint and cloud service cost challenges. To the best of our knowledge, the ability to adapt content source, here graphic rendering, is something that has not been attempted. The joint rendering and encoding adaptation can leverage our proposed rendering adaptation technique and any encoding adaptation in a proper manner as to optimally address the network and cloud cost challenges discussed in section 1.2, which again have not been attempted before for a video based interactive application like CMG.

Third, we have proposed a mobile cloud scheduling technique which takes into account the available capacity and usage probability of each heterogeneous access network and cloud server choice for each mobile user and the corresponding network round-trip delays, to minimize the average cloud service cost and maximize the number of concurrent mobile cloud users which can be served, while satisfying their user experience requirements, including response times. We furthermore, propose a joint adaptation and scheduling, where mobile cloud scheduling algorithm can leverage the application adaptation techniques, such that the capacity of the CMG system is dramatically increased. To the best of our knowledge, this is the first research work to solve the scheduling problem by jointly considering the heterogeneous access network and cloud server. We believe the scheduling technique developed in this thesis is potentially very useful for other new cloud server based mobile applications.

In the remainder of this dissertation, we take a detailed look at the novel techniques previously introduced. Note that we will compare our techniques with the previously published research work in each Chapter. In Chapter 2, we present the methods of how we develop the Mobile Gaming User Experience model. With this new proposed MGUE model, we characterize user achievable gaming experience in a commercial mobile cellular network. In Chapter 3, we elaborate the application adaptation techniques. We will also present experiment results to demonstrate the effectiveness of our proposed adaptation techniques to make the CMG approach feasible: ensuring protection against wireless network conditions, thereby, ensuring an acceptable mobile gaming user experience as well as ensuring scalability in regards to the mobile network and CMG cloud server availability. Chapter 4 will introduce our mobile cloud scheduling approach. We have also performed a set of simulation experiments to compare and characterize the performance of our proposed algorithms and the performance of the original CMG approaches where our scheduling techniques are not applied. The simulation results show that a) our proposed mobile cloud scheduling algorithm can help the CMG approach to maximize the number of blocked users and minimize cloud service cost while ensuring the user perceived gaming experience when comparing with the original CMG approach under the same conditions; b) the proposed joint adaptation and scheduling algorithm can dramatically increase the system capacity while ensuring acceptable user experience for each assigned user.

Chapter 2

Modeling and Characterizing User Experience in Cloud Mobile Gaming Approach

In the previous chapter, we presented an overview of the Cloud Mobile Gaming (CMG) approach and analyzed several important challenges we have to cope with for the success of the CMG approach. In this chapter, we develop a Mobile Gaming User Experience (MGUE) model to quantitatively measure the user experience of mobile gaming using the CMG approach. We validate the model using controlled subjective testing, and then use this model to characterize MGUE in various settings and commercially deployed wireless networks. Based on the characterization results, we furthermore suggested the possible solutions that may help improve the user experience achieved by the CMG approach, thereby leading to the feasibility of rich, multi-player, Internet games on mobile devices.

2.1 Mobile Gaming User Experience (MGUE) Model

The ability to model and evaluate the QoE of a network service is important to network operators and service providers, so that they can provision for the appropriate QoE levels, monitor the Quality of Experience (QoE) achieved, and take steps to improve the service as needed. QoE standardization has been actively pursued by International Telecommunications Union (ITU). Its existing standards cover the video quality metrics and tools (ITU-T J.144) [J144], and quality assessment approaches for multimedia services like IPTV (ITU-T SG12) [COM12], VoIP (ITU-T G.107) [G107], and Videophone (ITU-T G.1070) [G1070]. Unfortunately, there is no ITU standard quality measurement tool available for gaming. Moreover, many other video quality metrics and measurement techniques [Han04][TYK04][TW07][YH06][RNR08] cannot be directly applied to measure the user experience of (mobile) gaming, which is a highly interactive application, and where the round-trip response time has a significant impact on user experience, as opposed to the one-way nature of video. Though ITU standards of VoIP and Videophone have considered the impacts of network delay, they do not take into account round-trip delay, but only one-way delay. There has also been some work to analyze factors affecting user experience in gaming [BF10][CC06][DWW05][FCF$^+$05][FRS05][YII05]. However, they focus on conventional PC games and do not apply to the Cloud server based Mobile Gaming (CMG) approach, which is the subject of this dissertation.

Among existing QoE measurement methodologies, parametric model is the most commonly used way of measuring network multimedia QoE. It generally has three key components: model inputs, model assumptions, and quality estimation function. The inputs of parametric model are a group of impairment factors which affect user experience, while the model assumptions restrict the model working conditions. The quality estimation function is the essential part of the parametric model presenting the relationship between the impairment factors and model output of the predicted user experience. To develop such a parametric model that can quantitatively measure MGUE, we start this section by analyzing the various factors affecting MGUE. Through giving some model assumptions and discussions, we then identify several important impairment factors as model inputs, which are sub-sequentially used to formulate quality estimation function of MGUE model. The quality estimation function of MGUE model will be finally derived through subjective experiments introduced in the next section.

2.1.1 Impairment Factors Affecting MGUE

User perceived MGUE would mainly depend on two subjective factors: gaming visual experience and gaming response experience. The gaming visual experience depends on the resolution, smoothness, and image quality of gaming video received by mobile client, while the gaming response experience refers to the total delay from the user control command occurring to the corresponding video frame displaying on the mobile device. User perceived video quality needs to be measured from the off-line comparison of the video displayed in the client and the reference video in the server. The exact response time can be measured from the video obtained from recording the entire visual progress when the user played the game. Both measurements of these two subjective factors are time consuming and costly. To ensure feasibility, we decided to formulate MGUE model based on objective factors that can be in-service measured.

As shown in Figure 2.1, MGUE is affected by a number of objective factors, which can be categorized into four groups: source video factors, cloud server factors, wireless network factors, and client factors. Each of these objective factors affects the gaming visual

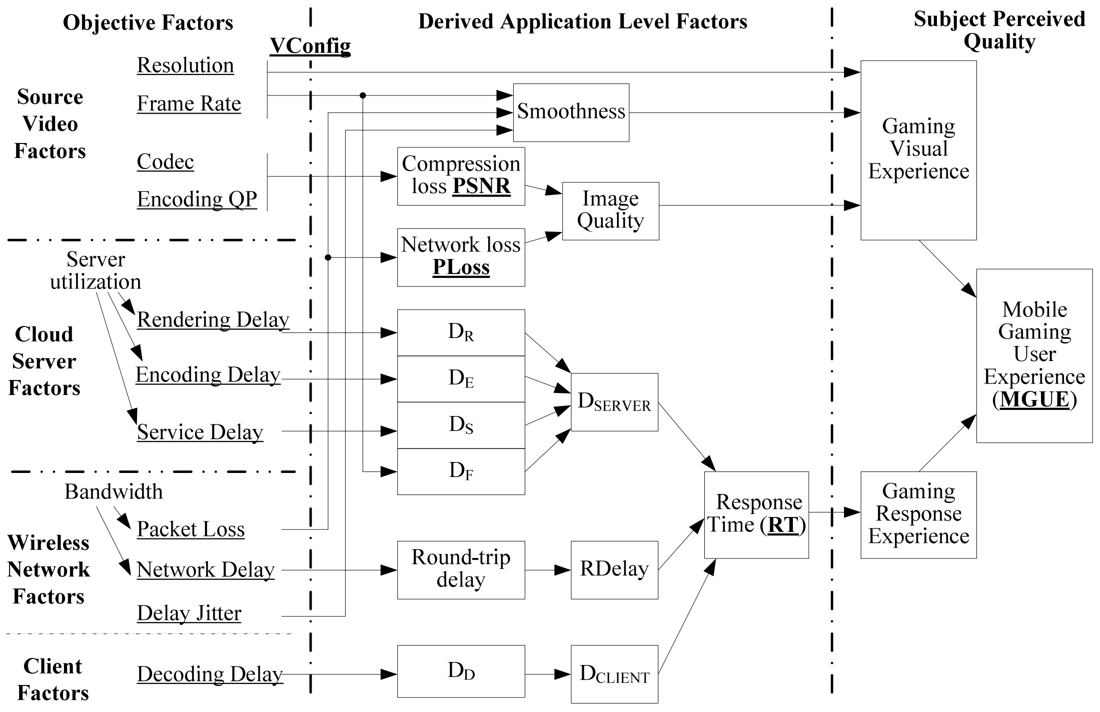


Figure 2.1. Impairment factors affecting mobile gaming user experience.

experience and gaming response experience in a complex manner. For example, quality degradation of user received gaming video may occur during the compression process from the source, depending on compression codec and compression Quantization Parameter (QP). Besides compression loss, network packet loss due to network congestion or wireless RF conditions could also reduce the video quality. The smoothness of the video is decided by the video frame rate, network packet loss, and network delay jitter. On the other hand, the gaming response experience will be affected by various delays created in cloud server, wireless network, and mobile client.

Among all the objective factors, server utilization and network bandwidth will only affect the MGUE when they are over-utilized. Moreover, the negative effects of these two factors could be represented by other factors. For example, server over-utilization will cause unexpected increase on rendering delay, encoding delay and service delay, while wireless channel over-utilization will cause network packet loss and unexpected delay. Therefore, we do not need to consider server utilization and network bandwidth into the MGUE model. Besides the factors shown in Figure 2.1, game genre is also an important factor that determines the contributions of all the other factors in determining the MGUE. For instance, in some games (e.g. racing games) fast response time is more crucial in determining the MGUE, while in some other games (e.g. Massively Multiplayer Online Role-Playing Game (MMORPG)) being able to clearly see the objects, and hence sufficient video quality, is more crucial in determining the MGUE. Therefore, parametric MGUE model can be formulated as:

$$\begin{aligned} MGUE = F(&Game, Resolution, FrameRate, Codec, EncodingQP, \\ &RenderingDelay, EncodingDelay, ServiceDelay, PacketLoss, \\ &NetworkDelay, DelayJitter, DecodingDelay) \end{aligned} \qquad (2.1).$$

However it is hard to integrate all the objective factors in Equation 2.1 as the model inputs to determine the MGUE. Thus we present the following analysis and model assumptions which allow us to reduce the complexity of the model by reducing the number of factors.

First, video resolution and frame rate are video configurations, which are given when the video streaming starts. We group these two factors together termed as VConfig. Giving the VConfig, the encoding QP determines the video quality, but is conditional on the codec and video content. Therefore, it is hard to study and determine its impact in determining the MGUE. Instead, we use Peak Signal to Noise Ratio (PSNR), the most commonly used distortion metric, to measure the quality of compressed video at the gaming video source.

Second, as discussed earlier, user perceived gaming response experience is hard to measure. Therefore, we use a derived objective factor gaming Response Time (RT) to indicate the gaming response experience. To study and understand RT, we consider the round trip data

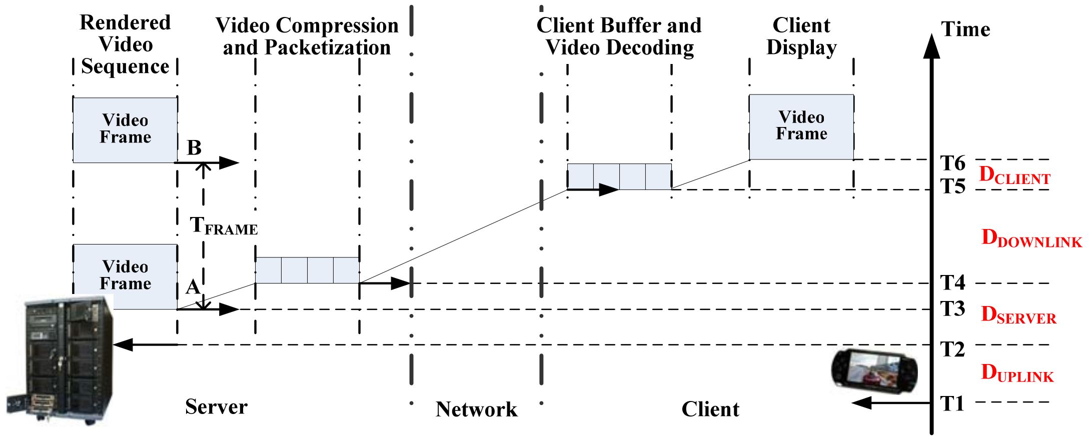


Figure 2.2. Round-trip flow of gaming response time.

flow in a CMG session shown in Figure 2.2. When a user command occurs on the mobile device, it will be sent to the gaming server with a network uplink delay D_{UPLINK} (T1-T2). This command may be held in the processing queue if there are plenty of commands from other clients waiting to be processed by the cloud server. Only the first command in the queue will be processed by the server, while the other commands have to wait for a period of time called service delay D_S. Then the game engine will process this command and then generate raw game video. This will take a period of time called rendering delay D_R. However, the raw game video is not always generated just after the command is processed. There is an interval T_{FRAME} between consequential video frames, which is equal to the reciprocal of Frame Rate (FR). Due to this T_{FRAME}, there might be a delay D_F with a range from 0 to T_{FRAME} depending on interval between the time when the server processes command and the time when the next raw game video is generated. Since command is processed randomly, D_F is uniformly distributed between 0 to T_{FRAME}. Therefore the average value of D_F is $1/(2 \times \text{FR})$. Once raw game video is generated, it will be encoded and packetized in a period of time called encoding delay D_E (T3-T4). The video packets will be received after a network downlink delay $D_{DOWLINK}$ (T4-T5), and displayed onto the mobile device after a client buffering and decoding delay D_C (T5-T6). Let RDelay denotes the round-trip network delay including D_{UPLINK} and $D_{DOWNLINK}$, RT can be formulated to:

$$\begin{aligned} &RT = RDelay + D_S + D_R + D_F + D_E + D_C \\ &\text{where } D_F = 1/(2 \times FR) \end{aligned} \qquad (2.2).$$

Some of delays in Equation 2.2 can be directly measured by CMG server and mobile client, like D_R, D_F, D_E, and D_C, while for some other delays like RDelay and D_S, we will measure them by a network probing mechanism (introduced in section 2.3.1).

Next, we analyze another factor, delay jitter, and explain why we will not consider it in modeling MGUE. Delay jitter denotes the network delay variation. Given an average network delay, there are two kinds of delay jitter, negative jitter and positive jitter. Negative jitter is caused by late arriving packets, and early arriving packets lead to positive jitter. The client buffer can eliminate positive jitter by caching the early arriving packets, while the effect of negative jitter can be represented by the larger network delay, captured in RDelay. Hence, we do not consider explicitly the effects of delay jitter in modeling MGUE.

Based on the above analysis and assumptions, we could reduce all the objective factors to five impairment factors: Game genre played, VConfig used, source video PSNR, network packet loss (PLoss), and gaming response time (RT). Thus MGUE can be formulated as:

$$MGUE = F(Game, VConfig, RT, PSNR, PLoss) \qquad (2.3).$$

2.1.2 Quantitative Measurement of MGUE

Having decided model inputs (impairment factors in Equation 2.3), next we need define a quantitative measurement metric for MGUE. In audio and video services, the most widely used subjective quality assessment methodology is opinion rating, which is defined in ITU-T Recommendation P.800 [P800]. In the subjective assessment tests, subjects are instructed to rate their perceived quality of the services according to the following opinion scales: 5(excellent), 4(good), 3(fair), 2(poor), and 1(bad). Subsequently, the arithmetic Mean of all the collected Opinion Scores, MOS [P800], is used as the measure of QoE. Similarly, we introduce a Game Mean Opinion Score (GMOS) as the measurement metric for MGUE, and later in this chapter, go on to develop a parametric MGUE model to quantitatively measure GMOS:

$$GMOS = F(Game, VConfig, RT, PSNR, PLoss) \qquad (2.4).$$

Since GMOS in Equation 2.4 is determined by 5 factors and its formulation can be a complex function, we attempt to derive simple individual functions of each factor, similar to the framework of ITU-T E-model [G107] for transmission planning. Although this E-model was originally proposed for the audio transmission planning, the framework of transmission rating factor R is helpful for any transmission planning because it makes the quality judgments for good or better and poor or worse in a good statistical mapping, hence can be applied in our study. The function of MOS formulated by R can be found in [G107]. We duplicate that function for our GMOS formulation:

$$GMOS = 1 + 0.035R + 7 \times 10^{-6} R(R-60)(100-R) \qquad (2.5).$$

The R-factor ranges from 0 to 100 and is related to GMOS through a non-linear mapping. We first derive MGUE model with only considering the individual effect of each impairment factor, while the cross-effects of different impairment factors will be considered and added into MGUE model later in section 2.3.3. The R-factor only considering the individual effect of each impairment factor can be formulated as:

$$R = \begin{cases} 100 - \sum_i I_i \quad (\text{if } \sum_i I_i < 100) \\ 0 \quad (\text{if } \sum_i I_i > 100) \end{cases} \tag{2.6}$$

I_i is the impairment function for each impairment factor, which indicates the individual impairment on MGUE of each impairment factor. As discussed earlier, Game genre will determine the contributions of all the other factors in determining the MGUE. Therefore, Game genre will be a parameter in every impairment function for each of the other four impairment factors (VConfig, RT, PSNR, and PLoss). Based on above discussions, we formulate the R-factor with individual impairment functions:

$$\begin{aligned} R = 100 - I_C(Game, VConfig) - I_R(Game, RT) \\ - I_P(Game, PSNR) - I_L(Game, PLoss) \end{aligned} \tag{2.7}$$

I_C includes the effect of the initial streaming video VConfig (resolution, frame rate); I_R indicates the impairment caused by Response Time; I_P represents the impairment caused by source streaming video quality PSNR; I_L covers the impairment caused by Packet Loss. The quality estimation Equations 2.5 and 2.7 indicate that GMOS can be evaluated by the Game played, initial VConfig and measurable factors: RT, PSNR, and PLoss. In this section, we have decided the MGUE model inputs based on some model assumptions, and introduced a metric GMOS as a measure for MGUE. Then we have derived the quality estimation equations of MGUE model and introduced its impairment functions. In the next section, we will derive the impairment functions in quality estimation Equation 2.7 to complete MGUE model.

2.2 Deriving Impairment Functions

In this section, we describe the approach we use to derive the impairment functions, I_C, I_R, I_P, and I_L. As a first step, we set up a controlled test environment, where each of the factors, VConfig, RT, PSNR, and PLoss, can be varied independently without affecting the settings of the other factors. Next, we conduct a series of MGUE subjective tests using a study group, where each test constitutes playing a game under a particular setting of one of the factors. Each study group participant provides an assessment of his/her gaming experience for each subjective test using a GMOS score. Base on our experiment results and regression analysis, we can derive the impairment function for each impairment factor. We start by describing the subjective testing process. Next, we describe how we derive the impairment functions to complete the MGUE model.

2.2.1 Subjective Quality Assessment Experiments

To study the effect of each of the factors on MGUE, we conduct a group of subjective quality assessment experiments. Figure 2.3 shows our experiment test bed. We connect the mobile device, which will be used by the study group participants, to a CMG server, directly via a network emulator, which we can use to control the network RDelay and PLoss. The user perceived RT is measured by a network probing mechanism introduced later in section 2.3.1, and we vary the RT by changing the RDelay via network emulator. The VConfig is varied by changing the resolution and frame rate settings in the video encoder. Similarly, the PSNR of source video is varied by appropriately changing the compression QP of the video encoder

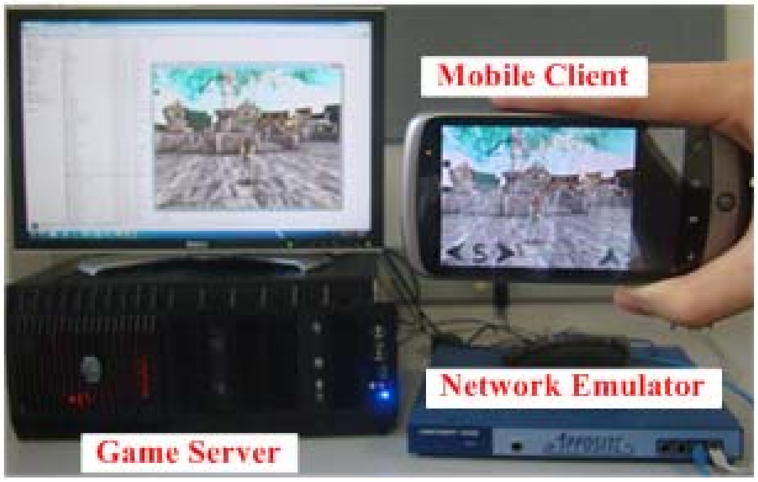


Figure 2.3. Test bed of subjective quality assessment experiments.

used by the CMG server. Table 2.1 shows the parameters of different factors that have been used in the subjective quality assessment experiments.

The study group was comprised of 25 students and staff at UCSD, who have prior experience of playing the selected games. Each study group participant played each of the three games under a certain test condition using the experimental parameters in Table 2.1, and provided assessment of their MGUE using a GMOS score rating system shown in Table 2.2. Finally, the results of the study group were tabulated for further analysis and derivation of the impairment functions.

2.2.2 Deriving Impairment Function I_C

To determine I_C, we consider the results of the subjective tests where only the Game and VConfig are changed, keeping all the other three factors at their best values, such that there is no impairment caused by them. For a given Game type, and a VConfig, we get the average GMOS score of all the participants, and use it to get the value of R from (5), and subsequently the value of I_C using (7), where the other impairment functions I_R, I_P and I_L are

Table 2.1. Experimental parameters of subjective quality assessment experiments.

Game (Type)		WoW(MMORPG), NFS(Racing), PES(Sports)
VConfig	Resolution	VGA, QVGA
	Frame Rate	[25:2:17], [15:1:5]
RDelay (ms)		[0:40:800]
PSNR (dB)		[26: 0.5: 38]
PLoss (%)		0, 0.5, 1, 2, 3, 4, 6, 8

Table 2.2. GMOS ratings and "R" values.

GMOS	**R**	**Description**
4.5—5.0	100	Excellent game, no impairment at all
4.0—4.5	80-100	Minor impairment, will not quit game
3.0—4.0	60-80	Impairment noticeable, might quit the game
2.0—3.0	40-60	Clearly impairment, usually quit the game
1.0—2.0	0-40	Annoying environment, definitely quit.

Table 2.3. Value of I_C in VGA resolution.

Game		25-16	15-13	12-11	10-9	8-7	6-5
WoW	VGA	3	3	8	14	25	41
	QVGA	10	10	16	23	35	54
NFS	VGA	0	0	7	15	26	51
	QVGA	3	3	10	18	28	53
PES	VGA	0	3	10	28	46	72
	QVGA	5	7	17	32	51	70

all 0 (as they do not cause any impairment). Table 2.3 shows the values of I_C for each of the VConfig used for each Game type. When adjacent two or more frame rates have close subjective GMOS score, we group them together (as their impacts on user experience are very close). For instance, for all three games, game users can hardly feel differences if we vary the frame rate from 25 to 16. Therefore there is only one average value of I_C of frame rate 25 to 16 for each game in different resolutions. It should also be noted that we did not study the MGUE where frame rate is below 5. One reason is that if frame rate is 5, user experience GMOS will drop below 3.0, where users cannot accept the gaming quality. Therefore, it is less necessary to study the user experience of frame rate below 5. Another important reason of not studying frame rate below 5 is related to the delay D_F in Response Time (RT). As discussed earlier, frame rate also affects the factor RT by D_F. When the frame rate is high, D_F is low, and thus cannot be felt by user. However when frame rate is below 5, the average D_F will be over 100ms. The experiment results of RT in next sub-section will show that such kind of delay will affect user perceived gaming experience, which we do not want happen in the derivation of I_C.

The impairment function I_C in Table 2.3 indicates how much the impairment of VConfig affects the user perceived quality. From Table 2.3, we find that all three games are not sensitive to changes in frame rate from 25 to 15. When frame rate is below 15, game PES is more sensitive to the frame rate than the other two games. For example, the impairment of VConfig (I_C) in game PES jumps over 40 at the frame rate of 8, while it is still around 30 in game WoW and NFS at the same frame rate. Regarding to the resolution changes, game WoW demands high resolution as the value of I_C increases dramatically (and hence MGUE suffers) while the resolution is reduced from VGA to QVGA. However, it seems that quality of the game NFS and PES are not affected significantly by the video resolution reduced from VGA to QVGA.

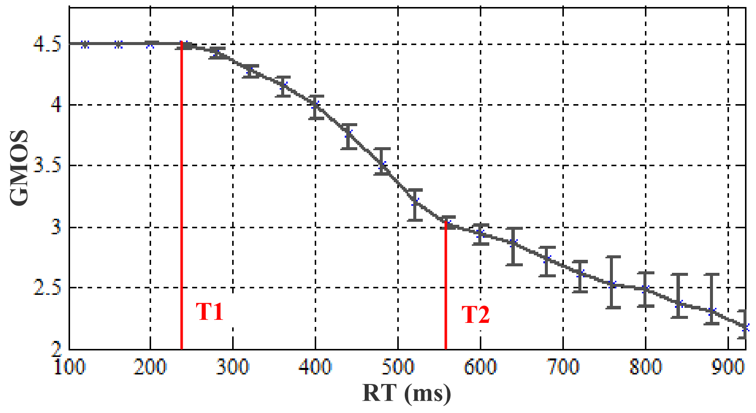


Figure 2.4. Subjective test results of RT versus GMOS for WoW.

2.2.3 Deriving Impairment Function I_R, I_P, I_L

To determine I_R, we use the results from the subjective tests, where only RT is varied and all the other factors are kept at their best values. As expected, the GMOS score goes down when the RT increases in all three games. As an example, Figure 2.4 shows the GMOS scoring by the study group for the game WoW. For each game genre, we define two delay points, T_1 and T_2. T_1 denotes the RT when the GMOS score starts to decrease below 4.5 (R=100), and T_2 denotes the RT where GMOS hits 3.1(R=60). Then T_1 and T_2 divide RT into three segments. In the first segment (T_1> RT), GMOS keeps at a constant value of 4.5, which implies the user experience remains unimpaired. Therefore, for this segment, the value of impairment function I_R should be 0. In the second segment (T_2> RT >T_1), GMOS decreases from highest 4.5 to 3.1 (minimum acceptable GMOS), while R-factor decreases from 100 to 60, which implies the value of I_R increases from 0 to 40. After T_2, the value of I_R keeps increasing from 40 with a slower slope, denoted by . We have tried several function models for regression analysis to derive the function of I_R. Based on the experiment results of regression analysis, the linear function as shown in Equation 2.8 is a simple and accurate model.

$$I_R = \begin{cases} 0 & (T_1 > RT > 0) \\ 40 \times [(RT - T_1)/(T_2 - T_1)] & (T_2 > RT > T_1) \\ 40 + \alpha \times (RT - T_2) & (RT > T_2) \end{cases} \quad (2.8).$$

Similarly, to derive the impairment function I_P, we analyze the GMOS scores of the corresponding subjective tests (where only PSNR is varied, keeping other factors at their best values). We notice similar trends like displayed by the Delay factor, except that the GMOS score increases while increasing PSNR. We derive I_P as:

$$I_P = \begin{cases} 40 + \beta \times (P_1 - PSNR) & (P_1 > PSNR > 0) \\ 40 \times [(P_2 - PSNR)/(P_2 - P_1)] & (P_2 > PSNR > P_1) \\ 0 & (PSNR > P_2) \end{cases} \quad (2.9).$$

From the subjective tests corresponding to PLoss, we see a different trend on how it affects user experience. We notice that even low packet loss rate tends to affect the quality and smoothness of the video received by the end device and perceived by the subject. And increasing PLoss will lead to a continuous drop in GMOS. Similar to the derivation of I_R and I_P, we have tried several function models for regression analysis to derive the function of I_L. Based on the experiment results of regression analysis, the linear Equation 2.10 is an accurate model to estimate the effect caused by packet Loss:

$$I_L = \gamma \times PLoss \quad (2.10).$$

The values of T_1, T_2, P_1, P_2, and coefficients α, β and γ, shown in Table 2.4, are determined by applying (8) (9) (10) to the subjective test results. This completes the derivation of the impairment functions in Equation 2.7.

Table 2.4. The value of variables in MGUE model for three different games.

Game	α	β	γ	T_2	T_1	P_2	P_1
WoW	0.05	5	8	560	240	34	30
NFS	0.08	6	13.5	440	200	33	29
PES	0.12	9	20	360	200	36	33

Equations 2.5-2.10, together with the Tables 2.3 and 2.4 provides the complete MGUE model, which can be used to quantitatively measure the quality of gaming experience over mobile wireless networks using the CMG approach. Note that the values in Tables 2.3 and 2.4 apply to the three game genres considered in this study. However, they can be easily extended to other game genres by repeating the approach (subjective study, regression analysis) outlined in this section.

2.3 MGUE Prototype, and Model Validation and Enhancement

In this section, we first introduce a software MGUE prototype for measuring the factors and calculating the corresponding GMOS score during live CMG sessions. Next, we validate the accuracy of the MGUE model by conducting another set of subjective experiments. Then we go on enhancing the accuracy of MGUE model with considering cross-effects of impairment factors.

2.3.1 MGUE Prototype: Measuring Impairment Factors and Calculating GMOS

We have developed a client-server software MGUE prototype to automatically measure the objective factors (RT, PSNR, and PLoss) during a mobile gaming session, and calculate the corresponding GMOS score, using the MGUE model.

We first design a network probing mechanism that can help CMG server to obtain the client delay D_C and measure server queuing delay D_S, network RDelay, and network PLoss. The CMG server periodically sends a UDP probe to the mobile client (5 probes a second), which includes the probe send out time and probe sequence number. Once mobile client receives a probe, it puts the information of its buffering and decoding delay D_C, into the received probe, and sends it back to the server through the TCP connection. The difference of probe send out time and receive time can indicate the current network round-trip delay RDelay and server queuing delay D_S. And the packet loss rate PLoss can be calculated by checking the received probe sequence number.

Simultaneously, we let CMG server measure the source video PSNR, rendering delay D_R, encoding delay D_E, and frame interval D_F. With the RDelay, D_S, D_C obtained from the network probing mechanism, and D_R, D_F, D_E measured by CMG server, we can calculate the response time RT at the server side (as we have all the parameters in Equation 2.2).

The above design allows CMG server to real-time simultaneously obtain RT, PSNR, and PLoss. And with the additional information of VConfig, CMG server is able to calculate GMOS score in real time during a CMG session, by using (5) - (10), and the values of I_C (Table 2.3) and T_1, T_2, P_1, P_2, α, β, γ (Table 2.4) as appropriate for the type of Game being played. In the MGUE prototype, the client need to measure its decoding delay which can be obtained directly from the video player installed on the mobile client. Besides, the client will

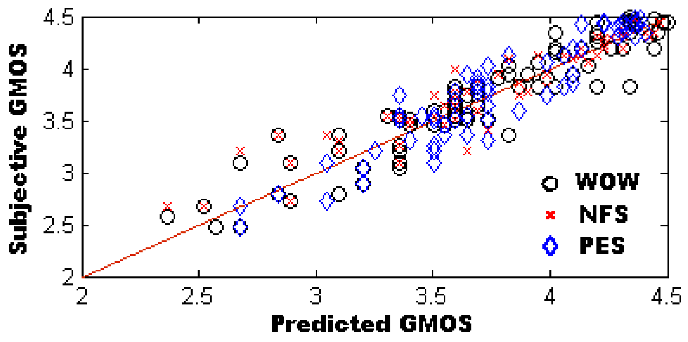


Figure 2.5. Relationship between predicted and subjective GMOS.

just forward back the probe packets sent from server. Therefore, the computing load added by MGUE prototype is extremely small (less than 0.01 percent per our test), such that it can be neglected.

2.3.2 MGUE Model Validation

Whereas the impairment functions of MGUE model is estimated using test results where only one factor is varying, the accuracy of the MGUE model needs to be validated by conducting another set of controlled experiments considering the effect of simultaneously varying all the factors. We use the same experimental framework as deriving the impairment functions, but a different study group consisting of 15 participants. As opposed to testing the effect of individual factors, we conduct 267 subjective tests where all the factors are varying randomly. It should be noted that during some of the validation tests, we also vary server and client utilization to introduce server and client delay. These tests will demonstrate that the way we estimate the impairment function of RT and the RT relation Equation 2.2 are correct. Figure 2.5 shows the relationship between the MGUE scores predicted by the MGUE model (x-axis) and the subjective (average) GMOS score by the participants (y-axis). The correlation between predicted and subjective user experience is 0.917. This result demonstrates the

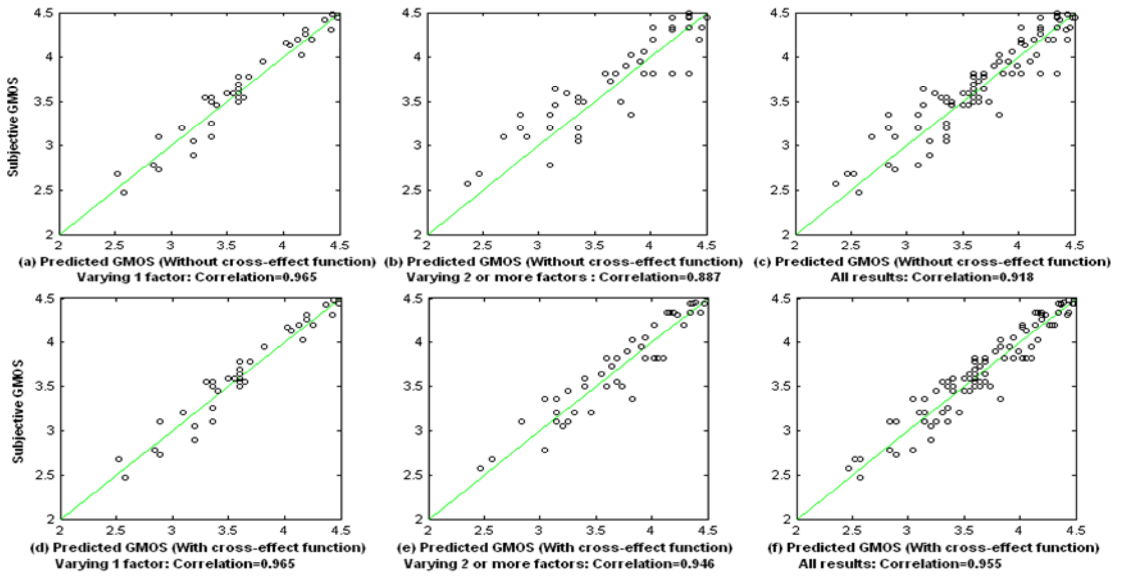


Figure 2.6. Correlation of predicted and subjective GMOS: (a) (b) (c) without cross-effect functions; (d) (e) (f) with cross-effect functions.

accuracy of our MGUE model in quantitatively measuring the Mobile Gaming User Experience of a user, given the Game, video configure VConfig used, video PSNR, and RT and Packet Loss experienced during the gaming session.

2.3.3 Enhancing MGUE Model by Cross-effect Functions

The above validation results demonstrate that MGUE model derived in section 2.2 has a good accuracy in predicating user perceived gaming quality. However, our further analysis on validation results discovers that the MGUE model developed in section 2.2 could perfectly predict GMOS when we only vary one factor but it is not very accurate in predicting GMOS if two or more factors are varied at the same time. Figure 2.6(a)(b) present the differences of validation results of game WoW between varying one factor and varying several factors. The correlation of predicted and subjective GMOS is 0.965 in Figure 2.6(a), while it is only 0.887

in Figure 2.1(b). Though the overall correlation is good (0.918) as shown in Figure 2.6(c), we believe the accuracy of MGUE can be potentially enhanced if we could address the problem in predicting GMOS when several factors vary simultaneously.

The main reason for the above problem is that we are not taking into account the cross-effects of impairment factors when deriving Equation 2.6. Obviously, the overall impairment of several factors is not exactly the same as the total of the individual impairments of all the factors, though it is related to them. For example, the impairment of 300ms RT and 2% PLoss is different (worse or better) than the total of individual impairment of 300ms RT and individual impairment of 2% PLoss. This difference is distinct especially when individual impairments of two or more factor are significant. To improve our MGUE model, we need to develop cross-effect functions and add them to Equation 2.6. Equation 2.11 below is the enhanced equation for R-factor considering cross-effects of impairment factors:

$$R = 100 - \sum_i I_i - \sum_{i \neq j} f_{ij}(I_i, I_j) - \sum_{i \neq j \neq k} g_{ijk}(I_i, I_j, I_k) \; - \ldots \; (100 > R > 0) \qquad (2.11),$$

where $f_{ij}(I_i, I_j)$ denotes the cross-effects of two impairment factors, while $g_{ijk}(I_i, I_j, I_k)$ denotes cross-effects of three impairment factors. More the impairment factors used, more cross-effect functions Equation 2.11 will have. However, the MGUE model would be very complex if we consider all kinds of cross-effect functions. Considering the feasibility, we decide to use only pair wise cross-effect functions $f_{ij}(I_i, I_j)$. Thus the enhanced equation for R-factor is:

$$R = 100 - \sum_i I_i - \sum_{i \neq j} f_{ij}(I_i, I_j) \; (100 > R > 0) \qquad (2.12).$$

We have trained the model for R (Equation 2.12) with many different types of functions for $f_{ij}(I_i, I_j)$, including $(I_i + a \times I_j)^n$, $I_i^n \times I_j^m$, and $e^{(I_i + a \times I_j)^n}$. We finally selected the function below (Equation 2.13), because the predicted GMOS scores using this function have the highest correlation with subjective scores obtained from our validation experiments:

$$f_{ij}(I_i, I_j) = k_{ij}\sqrt{I_i \times I_j} \quad (2.13).$$

The coefficients k_{ij} can be determined by applying Equation 2.13 to the validation test results. As a result, Equations 2.14-2.16 are the enhanced equations of R-factor for game WoW, NFS, and PES respectively:

$$\begin{aligned} R = 100 - \sum_{i=1}^{n} I_i + 0.05\sqrt{I_C \times I_R} - 0.42\sqrt{I_C \times I_P} + 0.10\sqrt{I_C \times I_L} \\ + 0.08\sqrt{I_R \times I_P} + 0.15\sqrt{I_R \times I_L} + 0.67\sqrt{I_P \times I_L} \end{aligned} \quad (2.14).$$

$$\begin{aligned} R = 100 - \sum_{i=1}^{n} I_i + 0.05\sqrt{I_C \times I_R} - 0.30\sqrt{I_C \times I_P} + 0.36\sqrt{I_C \times I_L} \\ + 0.06\sqrt{I_R \times I_P} + 0.13\sqrt{I_R \times I_L} + 0.76\sqrt{I_P \times I_L} \end{aligned} \quad (2.15).$$

$$\begin{aligned} R = 100 - \sum_{i=1}^{n} I_i + 0.02\sqrt{I_C \times I_R} - 0.24\sqrt{I_C \times I_P} + 0.12\sqrt{I_C \times I_L} \\ - 0.11\sqrt{I_R \times I_P} + 0.34\sqrt{I_R \times I_L} + 0.62\sqrt{I_P \times I_L} \end{aligned} \quad (2.16).$$

Figure 2.6(d)(e)(f) present the prediction results of using enhanced Equation 2.14 for game WoW. The correlation of predicted and subjective GMOS has been greatly improved, from 0.887 (Figure 2.6(b)) to 0.946 (Figure 2.6(e)), for the tests where two or more factors are varied. This leads to an improved correlation of total validation results which reaches as high as 0.955 (Figure 2.6(f)), as opposed to only 0.918 (Figure 2.6(c)) without using cross-effect functions.

2.4 Measuring MGUE in a Mobile Cellular Network

We have also applied enhanced MGUE model (section 2.3.3) and MGUE prototype (section 2.3.1) to measure MGUE in real wireless mobile networks. This will also help assess the feasibility and challenges of employing cloud mobile gaming to deliver desired gaming experience in today's mobile wireless networks. The experiments are conducted with a commercially available mobile cellular network. The game server is located in the UCSD campus, while the mobile game is played on a mobile device in four different scenarios: outdoor locations, indoor locations with poor coverage (low Carrier Interference Noise Ratio (CINR)), mobility conditions, and the conditions when cloud server is over-utilized. All three games (genres) and various video settings are used during the testing. Figure 2.7 shows a representative sample of the data (RT, PLoss, PSNR, and GMOS score) collected from numerous gaming sessions in the mobile cellular network in each of the four scenarios. We make the following observations from our experiments:

1) In outdoor locations, we stream the gaming video at 600kbps data rate, as this is sufficient to ensure very good source video quality (PSNR). The experiments are conducted in both midnight and noon during a day. As presented in Figure 2.7(a), CMG approach can provide the minimum acceptable MGUE (GMOS>3.0) but not very good MGUE (GMOS>4.0) at most times. This is mainly due to the high response time caused by round-trip network delay. As the gaming server is not located in mobile network system, gaming video has to be delivered via multiple hops and routes, including core network back-haul, carrier Ethernet, and wireless access link. Moreover, the gaming commands need to be sent from the mobile device to the server. Therefore, it is challenging to satisfy low response time, and thereby achieve high QoE, using current CMG approach and network architecture. It might be worthwhile to investigate the approaches to reduce this round-trip delay by deploying the gaming servers in

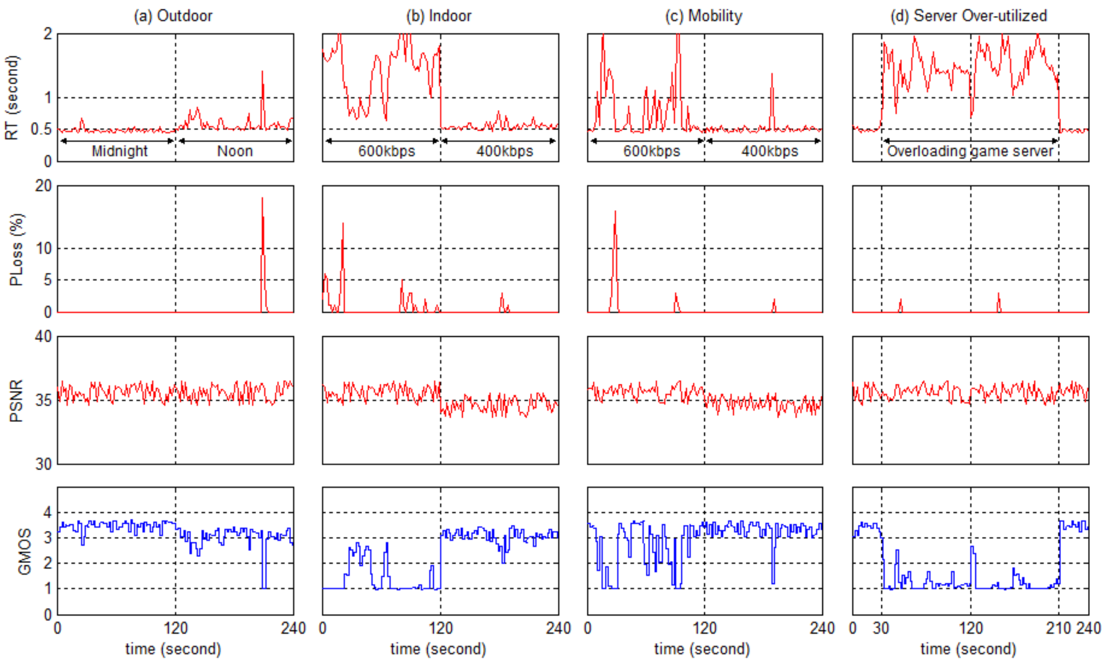


Figure 2.7. Test results for game WoW with video settings [VGA and 15 frames per second] in a mobile cellular network.

the wireless carrier network close to the base stations, as opposed to in the Internet cloud. From the results presented in Figure 2.7(a), we also notice that the user perceived GMOS varies during a day. For example, the average response time during noon is higher than during less busy midnight, leading to a relatively low GMOS score. In addition, the GMOS score during noon has occasional drops to less than 2.0 at times due to the severe packet losses experienced occasionally.

2) In indoor and in mobility conditions, we test with the gaming video streaming at two data rates: 600kbps with high source video quality as the outdoor test, and 400kbps with reduced source video quality. Figures 2.7(b) and (c) present sample results tested in both data rates. When we stream the gaming video at high data rate (600kbps), CMG approach cannot provide stable MGUE in both conditions, though GMOS score can reach above 3.0 during

brief periods in the mobility conditions. This unacceptable user experience is mainly caused by limited network bandwidth in the indoor and mobility conditions, where the wireless RF link quality can be poor or unstable at times. Unexpected network delay and packet loss will happen, as the wireless channel cannot provide adequate bandwidth for high data rate video streaming application. In contrast, when we stream the gaming video at 400kbps, though video quality is degraded a little at this data rate, the MGUE is more stable and overall better than the MGUE in high data rate (600kbps). The above results show the ability of our MGUE model to capture the networking impairments during cloud mobile gaming sessions on a live mobile wireless network, and provide feedback on the overall quality of user experience, considering the tradeoffs between source video quality impairments, and network impairments. The ability to quantitatively measure the overall user experience under different conditions may help in developing techniques to effect the right tradeoffs between different objective factors and improve user experience during cloud mobile gaming sessions.

3) We also conducted experiments to understand the effect of server utilization/overloading on the user experience during cloud mobile gaming sessions using our MGUE model. As we mentioned before, when cloud gaming server is over-utilized, the rendering delay D_R and encoding delay D_E will increase. In the meanwhile, the generated frame rate will dramatically decrease, which will affect I_C (Table 2.3) and D_F, thus leading to a deteriorating MGUE. As shown in Figure 2.7(d), the CMG approach cannot provide acceptable MGUE when the server computation resource is over-utilized, mainly due to the high response time and low frame rate. Like in the case of networking artifacts, the above results show the ability of our MGUE model to capture server related impairments in cloud mobile gaming user experience, which can be potentially used in the future to develop

techniques to appropriately schedule gaming sessions to cloud servers to minimize the server delays and reductions in rendering frame rates.

2.5 Conclusions

In this chapter, we develop and validate a Mobile Gaming User Experience (MGUE) model, to quantitatively measure user perceived gaming experience in the CMG approach. We also develop a MGUE prototype measurement tool to enable in-network monitoring of MGUE. Consequently, we measure and analyze the performance of the CMG approach in a mobile cellular network. Our analysis shows that while it is possible to achieve good quality gaming experience over mobile wireless networks, there are several network conditions in which MGUE may be incompatible/unacceptable. We suggest the investigation of possible solutions that may lead to significant improvement in user experience achieved by the CMG approach. The MGUE model discussed in this chapter can be potentially used by researchers to assess the performance of new CMG techniques, and by future mobile gaming service providers, including network operators to better plan and optimize their CMG services as well as monitor in-network the user experience of their mobile gaming subscribers. We also believe that the approach used to develop the MGUE model outlined in this chapter can be useful to develop QoE models for other new mobile cloud computing based interactive multimedia applications, such as virtual and augmented reality.

The text of this chapter, in part or in full, is based on material that has been published in ACM Mobile Computing and Communications Review (MC2R) (S. Wang, S. Dey, “Cloud Mobile Gaming: Modeling and Measuring User Experience in Mobile Wireless Networks,” ACM SIGMOBILE MC2R*,* Jan. 2012) and material published in IEEE Global

Communications Conference (Globecom) (S. Wang, S. Dey, “Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach,” IEEE Globecom, Honolulu, Dec. 2009). The dissertation author was the primary researcher and author in the publications, and the coauthors listed supervised the research that forms the basis of this chapter.

Chapter 3

Application Adaptation techniques to Address Wireless Communication and Cloud Computation Constraints in Cloud Mobile Gaming

To address the communication constraint imposed by the fluctuating and limited bandwidth of the mobile network and the computation constraint imposed by cost and availability of cloud servers, in this chapter we develop several application-adaptation techniques, including a dynamic game rendering adaptation and a joint rendering and encoding adaptation technique. With the help of these adaptation techniques, we can adapt the CMG application communication and computing needs to protect quality of service against dynamic variations of mobile network bandwidth and cloud server utilization. Experiments conducted on a commercial mobile network demonstrate that our proposed adaptation techniques can significantly improve user perceived mobile gaming experience, and ensure the scalability of the CMG approach in terms of both network bandwidth and server computing needs , thereby ensuring the feasibility of the Cloud Mobile Gaming approach.

3.1 Introduction

Though the CMG approach promises to enable mobile users to play rich Internet games on any mobile device without having to download the game engines, two fundamental questions arise. Firstly, how good will the user experience be using the CMG approach, considering (a) unlike conventional PC Internet game, gaming video will have to be streamed from the cloud servers to the mobile devices through the wireless networks (as opposed to games rendered on the devices themselves), and (b) unlike conventional mobile video streaming/download, CMG is a highly interactive application, where round-trip latency will have to satisfy stringent response time requirements of gaming [WDA10][TLL07][CC06] [JVC03][QML04]. Secondly, how scalable the CMG solution can be, given that (a) compute intensive 3D rendering tasks for each concurrent gaming user will need to be performed simultaneously on the computing servers, and (b) streaming all the game video for each user to his/her mobile device may consume significant back-haul and wireless network bandwidth. In other words, there are two challenges for CMG approach: 1) communication constraint in terms of limited and fluctuating mobile network bandwidth [BLB06][LSM08], which can cause unexpected delay and packet loss, leading to undesirable increase in response time, besides adverse impact on the quality of the video streamed; 2) computation constraint reflected by the CMG server computing resource available for each gaming client, considering that the CMG cloud servers will have to host numerous clients at the same time. It is vital to address the above two constraints for the success of the CMG approach, as both of them are closely associated to the perceived gaming experience of each user and the scalability of CMG approach.

In this chapter we develop a dynamic game rendering adaptation and a Joint Rendering and Encoding Adaptation (JREA) technique, to address the communication constraint of wireless networks and the computation constraints of CMG servers. The proposed dynamic game rendering adaptation can vary the graphic rendering settings to adapt communication/computing needs of a CMG session, thereby addressing both communication and computation constraints. However, though our experiments we realized that it is not sufficient to provide acceptable gaming quality in several circumstances if only applying our proposed rendering adaptation technique. Therefore, we develop a Joint Rendering and Encoding Adaptation algorithm which integrates the proposed rendering adaptation technique and traditional video encoding adaptation technique, and simultaneously leverages both of them to address the constraints of the CMG approach, such that the aggregated Mobile Gaming User Experience (MGUE) of the CMG approach is maximized.

Section 3.2 compare our techniques with previous related work. In section 3.3, we explain the principle of rendering adaptation and provide an overview of the detailed methodology of adaptation techniques in section 3.4. The experiment results presented in section 3.5 demonstrate the success of our proposed adaptation techniques in ensuring high MGUE while addressing both the computation and communication constraints, and hence the scalability of the CMG approach. We summarize our findings and conclude our work in section 3.6.

3.2 Related Work

Several approaches have been developed to model the computation cost of graphic rendering [TML+04][WW03][RLS+02]. However they do not study the impact of rendering parameters on communication bandwidth, which is an important objective of CMG approach. There have also been several adaptive rendering techniques [SM99] [NRL+04] [MPL+J07] [MPL+C07] to address the costs associated with rendering on clients in client-server architectures. These techniques attempt to adapt the complexity of the rendering objects, depending on the bandwidth of the communication link and the computational capability of the clients. The lower complexity of the objects, the less rendering cost at the client, and also the less communication bandwidth needed to transmit the objects to the client. However, in the CMG approach, where rendering is performed by CMG servers and not clients, the computation cost can be affected by adapting the rendering tasks themselves, as proposed in our approach, with much more impact than just by adapting the complexity of graphic objects. Moreover, the communication bandwidth cost in the CMG approach is determined by the game video that needs to be streamed from the CMG servers to the clients, and not the size of the rendering objects as in the traditional client-server architectures. Hence, the above approaches [SM99][NRL+04][MPL+J07][MPL+C07] are sub-optimal to address CMG computation constraint, and not applicable to address the CMG communication constraint.

There have been many video bit rate and encoding adaptation techniques that have been developed and used for live video streaming and video conferencing applications [CV05][SMW07]. However, these video encoding adaptation techniques, depending on the application scenarios and requirements, rely on adapting two parameters, video quality and frame rate. For instance, in the case of live (real-time) video streaming, video encoding adaptation techniques [WCG+07][BSW+07] can compromise video quality as needed because users typically have some tolerance towards quality degradation. In the case of video conferencing, video adaptation techniques [CSW+11][CSH+08] mainly use frame rate adaptation to meet the network constraints as typical video conferencing scenarios involve low motion video. Unfortunately, video streaming in the case of Cloud Mobile Gaming has more challenging requirements, needing both high video quality and high frame rate, and hence the existing video adaptation techniques may not be sufficient to address network communication constraints. In our previous work [WDA10], we proposed a game aware video encoding adaptation technique, which derives the optimal video encoding settings for different types of games by analyzing the effects of different video encoding parameters, including video quality and video frame rate, on MGUE. Though we have demonstrated that using the optimal video encoding settings [WDA10] can minimize the impairment on MGUE when adapting the video bit rate, it is not able to provide acceptable MGUE under severe network constraints, as shown in experiment results (Figure 3.10(d)) in section 3.5.1. In addition, the video encoding adaptation techniques [CV05][SMW07][WCG+07][BSW+07][CSW+11][CSH+08][WDA10] do not address the problem of server computation constraint, which is another important objective of CMG approach.

There have also been several applications that are designed for desktop sharing, like windows Remote Desktop Service (RDS) and Virtual Network Computing (VNC), which can potentially be used for CMG. However, similar to video conferencing, these desktop sharing applications mainly use frame rate adaptation to meet the network constraints, as they are mainly designed for remote computer control and hence can afford to have video streamed at much lower frame rate. Consequently, such desktop sharing applications cannot be adopted for CMG which is a motion sensitive application and hence cannot compromise on video frame rate.

In summary, CMG has the most stringent requirements: need to support high video quality and high sensitivity to video frame rate, and the existing video adaptation techniques for live video streaming [WCG$^+$07][BSW$^+$07], video conferencing [CSW$^+$11][CSH$^+$08], and desktop sharing applications (RDS and VNC) are not able to address these stringent requirements to produce desired user experience. Hence, in this chapter we propose an approach to dynamically adapt the video source itself, graphic rendering which generates the gaming video, which can significantly impact the encoding bit rate of the resulting video, without compromising video frame rate or frame quality. Adapting the content source itself has not been attempted, and may not be possible to do, for live video streaming or video conferencing. We furthermore propose a joint adaptation between content source (here rendering) and video encoding, to address mobile network and server computing constraints, which again has not been attempted before to the best of our knowledge.

3.3 Overview of Application Adaptation Approach

In this section, we first give a brief introduction to the principles of the innovative rendering adaptation. Next, we provide an overview of our proposed rendering adaptation approach, including the different offline and online steps involved, and explain the novelty and feasibility of the approach.

3.3.1 Principles of Rendering Adaptation

Figure 3.1 shows the primary stages of graphics pipeline in modern GPU. All the graphic data for one frame is first cached in a display list. When a display list is executed, the data is sent from the display list as if it were sent by the application. All the graphic data (vertices, lines, and polygons) are eventually converted to vertices or treated as vertices once

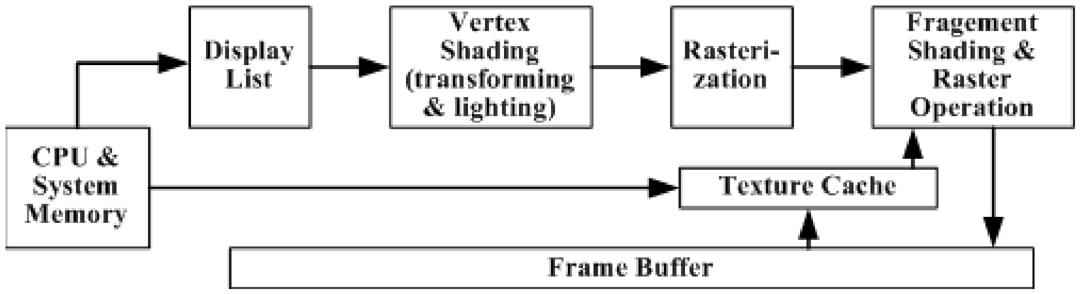

Display List
Vertex Shading (transforming & lighting)
Rasteri-zation
Fragement Shading & Raster Operation
CPU & System Memory
Texture Cache
Frame Buffer

Figure 3.1. Primary stages of graphic rendering pipeline.

the display list starts. Vertex shading (transform and lighting) is then performed on each vertex, followed by rasterization to fragments. Finally the fragments are subjected to a series of fragment shading and raster operations, after which the final pixel values are drawn into the frame buffer. To best present a geometric object, graphic rendering usually applies texture images onto the object, so as to make it look more realistic. The texture images are usually pre-calculated and stored in system memory, and loaded into GPU texture cache when needed. Each of the above rendering stage is configurable by a set of rendering parameters. The term “rendering setting” is usually used to denote a tuple of values used for these rendering parameters. We next describe the Communication Complexity (CommC) and Computation Complexity (CompC) associated with each rendering setting in CMG.

The Communication Complexity (CommC) of a rendering setting denotes the level of how much the bit rate is needed to deliver CMG video with this rendering setting. While the video bit rate is determined by the video compression rate used, it is largely affected by the video content complexity. Different rendering settings will lead to different content complexities, thereby different communication complexities. To quantitatively measure CommC, we define the value of CommC of a rendering setting as the ratio of the bit rate needs of this rendering setting to the minimum bit rate need among all the possible rendering settings, for the same game scene using the same video compression rate.

The Computation Complexity (CompC) of a rendering setting indicates the level of how much GPU computation resource needed to render the game with this rendering setting. Different rendering settings will lead to different computation complexities. The higher CompC, the richer rendering graphics, and correspondingly the higher GPU utilization that will be consumed by the game engine. Similar to quantifying CommC, we define the value of CompC of a rendering setting as the ratio of the GPU utilization using this rendering setting to the minimum GPU utilization of all possible rendering settings, under the same game scene.

We next propose two key principles how rendering adaptation can be used to affect CommC and CompC. The first principle is to reduce the number of objects in the display list, as not all objects in the display list created by game engine are necessary for playing the game. For example, in the Massively Multiplayer Online Role-Playing Game (MMORPG), a player mainly manipulates one object, his avatar, in the gaming virtual world. Many other unimportant objects (e.g. flowers, small animals, and rocks) or far way avatars will not affect the user playing the game. Removing some of these unimportant objects in display list will not only release the load of graphic computation but also reduce the video content complexity, and thereby CommC and CompC. The second key principle for rendering adaptation is related to the complexity of rendering operations. In the rendering pipeline, many operations are applied to improve the graphic reality. The complexities of these rendering operations directly affect CompC. More importantly, some of the operations also have significant impact on content complexity, and thereby CommC, such as adjusting texture detail. If we can scale these operations, we will be able to scale CommC and CompC as needed.

3.3.2 Overview of Proposed Application Adaptation Approach

We have explained the rendering adaptation principles that can change the video content complexity and GPU rendering complexity to scale the video bit rate needed and server computation needed. As will be shown in section 3.4, both the rendering adaptation principles can be affected by selecting appropriate rendering settings. In this section, we give a brief overview of our proposed rendering adaptation approach, which will dynamically select an optimal rendering setting with proper CommC and CompC to ensure video bit rate needed and server computation needed meet current available network bandwidth (communication constraint) and server computing resource (computation constraint). However, since the number of different rendering settings possible may be very large, finding the optimal rendering setting for a given communication and/or computation constraint may be time consuming. On the other hand, to be effective rendering adaptation should be performed in real time in response to rapid changes in network and server conditions. To resolve the above conflict, we propose to partition the rendering adaptation approach into two parts: offline and online steps. The offline steps will characterize and pre-determine the optimal rendering settings for different levels of CommC and CompC, thereby allowing the online steps to select and vary the rendering settings in real time in response to the fluctuations of network and server resources.

Figure 3.2 gives an overview of the proposed rendering adaptation approach which involves the above mentioned offline and online steps. Details of each of these steps will be provided in section 3.4. In the first offline step, rendering parameters are identified which can affect the communication and computation complexities of the game. Subsequently, for each possible rendering setting involving the selected parameters, CommC and CompC values are derived. This will result in a *complexity model*, which is a mapping of CommC and CompC to

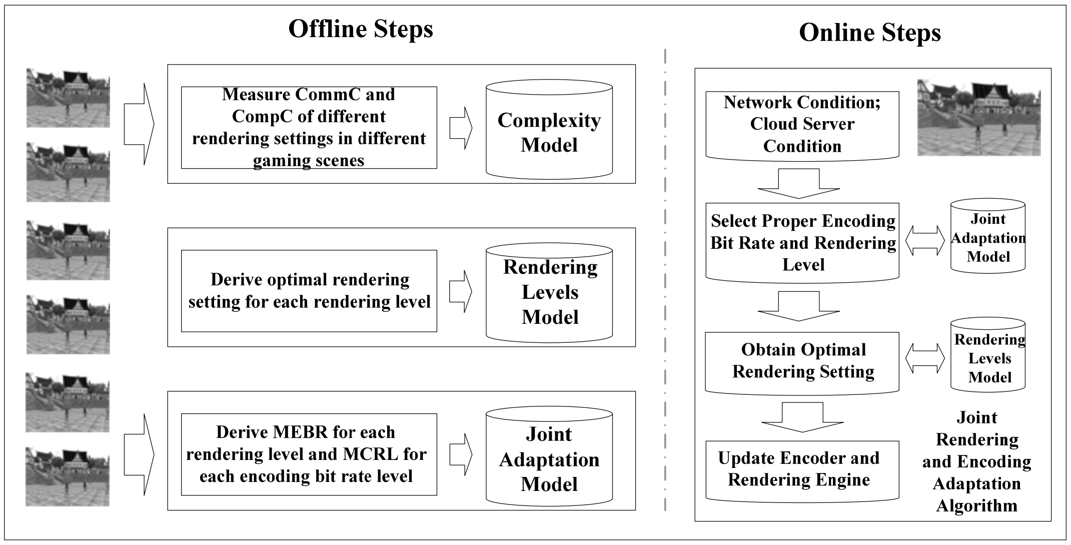

Offline Steps
Measure CommC and CompC of different rendering settings in different gaming scenes
Complexity Model
Derive optimal rendering setting for each rendering level
Rendering Levels Model
Derive MEBR for each rendering level and MCRL for each encoding bit rate level
Joint Adaptation Model
Online Steps
Network Condition; Cloud Server Condition
Select Proper Encoding Bit Rate and Rendering Level
Joint Adaptation Model
Obtain Optimal Rendering Setting
Rendering Levels Model
Update Encoder and Rendering Engine
Joint Rendering and Encoding Adaptation Algorithm

Figure 3.2. Proposed rendering adaptation methodology: offline and online steps.

different rendering settings. Next, in the second offline step, several rendering levels are selected, each of which reflects a certain CommC and a certain CompC. Then using the complexity model, optimal rendering settings are derived that meet the CommC and CompC targets of each rendering level, leading to a *rendering levels model*. During an online gaming session, our adaptation technique can select in real time a proper rendering level and the corresponding optimal rendering setting, using the rendering levels model. However, since the mobile network bandwidth can vary very frequently, use of rendering adaptation alone may lead to frequent varying of rendering levels, which is not desirable from a user experience perspective. Therefore, we develop an online Joint Rendering and Encoding Adaptation (JREA) algorithm, which addresses communication constraints by judiciously utilizing the power of changing the video source through rendering adaptation, with large impact on network bandwidth needed, together with adapting the video encoding bit rate to address relatively small but frequent network bandwidth fluctuations. Since adapting encoding bit rate cannot affect computational constraints, rendering adaptation is used by itself to address server utilization conditions.

Adapting both rendering and video encoding jointly will necessitate understanding the optimal values of encoding bit rate or rendering level that can be used when encoding or rendering is adapted respectively. More precisely, we will need to know the following: (a) for each rendering level, the Minimum Encoding Bit Rate (MEBR) below which gaming video quality will not be acceptable by game users, so the JREA algorithm can use MEBR without affecting user experience, and (b) for each encoding bit rate used, what is the Maximum CommC Rendering Level (MCRL), the rendering level which has the highest CommC while its resulting video has minimum user experience impairment. In the third offline step, a *joint adaptation model* is derived, which includes the mappings of MEBR to each rendering level and MCRL to each encoding bit rate. Also shown in Figure 3.2 are the online steps. Depending on the network and server conditions, JREA decides if the rendering level and encoding bit rate level needs to be adapted. If either of them is changed, it will check the joint adaptation model to decide if the other one needs to be changed correspondingly. If rendering level is to be changed, it will select the optimal rendering setting based on rendering levels model and update the game engine consequentially to effect the rendering level change.

3.3.3 Novelty and Feasibility of Proposed Application Adaptation Approach

The proposed rendering adaptation approach has several significant contributions. First, to the best of our knowledge, this is the first work which has proposed dynamic adaptation of rendering as an effective way to adapt the rendered video to the fluctuations in network and server utilization, opening up a completely new way of efficiently delivering

CMG video over wireless network. In addition, our work has extensively characterized how the different rendering parameters affect CMG video communication and server computation needs. The results and findings in this chapter will be very helpful for many other cloud based rendering applications, like augmented reality and telemedicine. Furthermore, this chapter has developed a joint adaptation technique which is also the first work attempting to link rendering tasks to video encoding bit rate and exploiting the dependency of video bit rate to rendering complexity.

In almost every 3D video game, game rendering parameters are designed to be configurable by a game player corresponding to the capacity of the hardware platform the player is using. Hence, in our proposed adaptive CMG approach, it should be easy to access and use these rendering parameters for any subscribed game. While the rendering parameters are set by individual game players manually and statically once for the hardware platform used/game played, the novelty and contribution of our approach is developing automated and dynamic rendering adaptation, using the same available parameters, to address dynamic changes in network and cloud server utilization conditions, thereby enabling high quality and bandwidth efficient Cloud Mobile Gaming. Also because these configurable rendering parameters are recommended by the game developers, the variations on these rendering parameters will not affect the game user to play the game though the richness of rendering graphics may vary.

It should be noted that we recommend conducting the offline modeling steps for each game individually. This is because the CommC and CompC of the same rendering parameter could be different for different games mainly due to the design changes from game to game. However, the variations of CommC and CompC in different gaming scenes of one game are marginal, as we discuss in section 3.4.2. Therefore, the offline modeling for each game will only need to be executed once, which makes our approach feasible and practical for commercial implementation.

3.4 Rendering Adaptation Technique

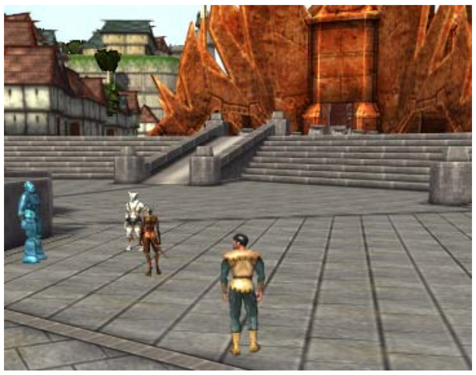
We have given a brief introduction to the principles of game rendering and an overview of our proposed rendering adaptation technique. In this section we will present the detailed methodology of how we design and enable rendering adaptation for Cloud Mobile Gaming, including three offline modeling steps and an online adaptation algorithm.

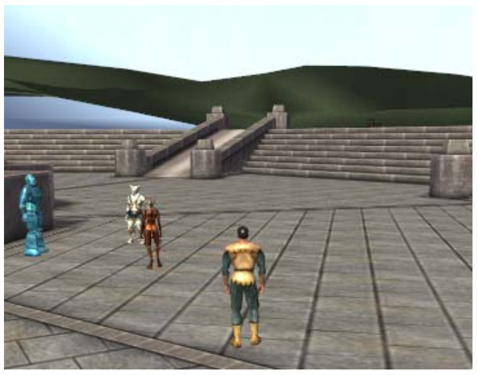
3.4.1 Adaptive Rendering Parameters and Adaptive Rendering Settings

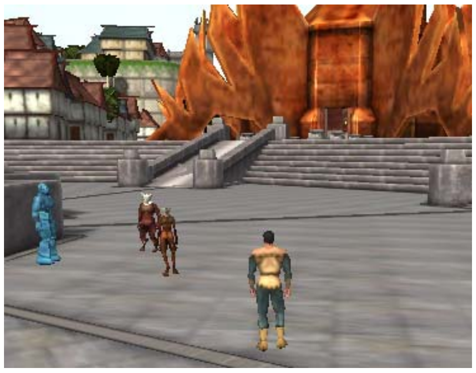
The first thing in enabling dynamic game rendering adaptation in CMG approach is to identify the adaptive rendering parameters. A game may have many different rendering parameters, but only a few of them have the obvious impacts on the CommC or the CompC. An adaptive rendering parameter *p* must be able to adapt at least one of CommC and CompC. In general, a game may have *k* different rendering parameters. We use a k-dimension tuple *s* to denote an adaptive rendering setting and use *S* to denote the set of all the possible adaptive rendering settings of a game. The elements of *s* indicate the values of the adaptive rendering parameters used in the adaptive rendering setting *s*:

$$s = (p_1, p_2, ..., p_k), \quad s \in S \qquad (3.1).$$

As discussed in section 3.3.1, reducing the number of objects in the display list or reducing the complexity of rendering operations could lead to the decreases in CommC and CompC. Based on this concept, we identify four common parameters which we believe are suitable for rendering adaptation in most 3D games:


(a) 300 meters view and high LOD


(b) 60 meters view and high LOD


(c) 300 meters view and Medium LOD

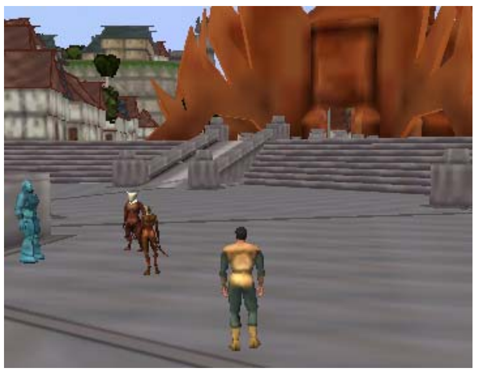

(d) 300 meters view and low LOD

Figure 3.3. Screenshots of game “PlaneShift” in different settings of view distance and texture detail (LOD).

1) Realistic effect: Realistic effect basically includes four parameters: color depth, anti-aliasing, texture filtering, and lighting mode. Each of the four parameters only affects one stage of the graphic pipeline. Varying any one of them will only have an impact on one special-purpose processor, which may not reduce the load on bottleneck processor. Thus when we reduce/increase the realistic effect, we vary all four parameters to have a reduced/increased CompC.

2) Environment detail: Many objects and effects (grass, flowers, and weather) are applied in modern games, especially the RPG games, to make the virtual world look more realistic. However they are not really necessary for users playing the game. Therefore, we could eliminate some of these objects or effects to reduce CommC and CompC if needed.

3) Texture detail: This is also known as Level of Detail (LOD). It refers to how large and how many textures are used to present objects. The lower texture detail level, the lower resolution the textures have. As shown in Figure 3.3(a)(c)(d), the surfaces of objects get blurred as we decrease the texture detail. It is also important to be aware of that reducing texture detail has a less impact on important objects (avatars and monsters) than unimportant objects (ground, plants, and buildings), because the important objects in game engines have much more textures than unimportant objects. Thereby we could leverage this information to properly downgrade the texture detail level for less communication bit rate, while maintaining the good visual quality of important objects.

4) View distance: This parameter determines which objects in the camera view will be included in the resulting frame, and thereby should be sent to the display list for graphic rendering. Figure 3.3(a)(b) compare the visual effects in two different view distance settings (300m and 60m) in the game PlaneShift (PS), a cross-platform MMORPG game. Though shorter view distance has impairments on user perceived gaming quality, the game will be still playable if we properly control the view distance above a certain acceptable threshold. Since the view distance affects the number of objects to be rendered, it has the impact on CompC as well as the CommC.

3.4.2 Derivation of Complexity Model

Having defined adaptive rendering settings, we next introduce how to derive the complexity model. For each game, k different test scenes are used to measure the average of CommC and CompC for each rendering setting. Let SCENE denote the set of all k test scenes, and let $scene_i$ denote each test scene. As we defined in section 3.3.1, in each $scene_i$ the value of CommC and CompC of a rendering setting is the ratio of the bit rate need and computation need of this rendering setting to the minimum bit rate need and computation need among all the possible rendering settings. Therefore for each scene$_i$, we first find out the lowest adaptive rendering setting s$_i^L$, which has the minimum bit rate need and computation need among all the possible rendering settings. We use GPU utilization to quantify the computation need. Let BitRate(s,scene$_i$) denote the bit rate need of rendering setting s in test scene$_i$, and let ServUtil(s, scene$_i$) denote the server utilization of s in test scene$_i$. Then for a set of test scenes, the average CommC and CompC of rendering setting s can be calculated by Equations 3.2 and 3.3:

$$CommC(s) = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{BitRate(s, scene_i)}{BitRate(s_i^L, scene_i)}\right),\ scene_i \in SCENE \qquad (3.2),$$

$$CompC(s) = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{ServUtil(s, scene_i)}{ServUtil(s_i^L, scene_i)}\right),\ scene_i \in SCENE \qquad (3.3).$$

Having defined the CommC and CompC, given a game and its possible adaptive rendering settings, we could conduct offline experiments to measure CommC and CompC for each rendering setting to complete the complexity model for this game. We next use game PS as an example to explain our modeling approach in details, where we have elaborately studied how different rendering settings affect the CommC and CompC. Subsequently, we also have studied the impacts on CommC and CompC when video encoding setting, or video resolution, or server GPU is changed. This will help to demonstrate that the key concept that communication complexity and computation complexity can be affected by different rendering settings is broadly applicable, no matter what kind of video resolution or video encoding setting, and no matter what kind of graphic GPU is used.

Table 3.1. Adaptive rendering parameters and experiment settings.

Parameters		Experiment Values		
Realistic Effect		H(High)	M(Medium)	L(Low)
	color depth	32	32	16
	multi-sample factor	8	2	0
	texture-filter factor	16	4	0
	lighting mode	Vertex light	Lightmap	Disable
Texture Down Sample Rate (Texture Detail)		0, 2, 4		
View Distance (meter)		300, 100, 60, 40, 20		
Enabling Grass (Environment Detail)		Y(Yes), N(No)		

Four adaptive rendering parameters are selected (corresponding to the adaptive rendering parameters introduced in section 3.4.1) for game PS. Table 3.1 shows the settings used for these four rendering parameters. Then we conduct experiments to measure the CommC and CompC for every possible rendering setting s using the settings of parameters in Table 3.1. The experiments are conducted on a desktop server which integrates a NVIDIA Geforce 8300 graphic card. Video resolution used is VGA. The video codec used is X264, and its encoding method is set to Variable Bit Rate (VBR). The Quantization Parameter (QP) is 25, while the encoding frame rate is 15fps and the size of Group of Pictures (GOP) is 30. We have randomly selected 20 different gaming scenes. In each test scene, for each rendering setting *s*, we let the game avatar roam in the gaming world along the same route. We measure the average compressed video bit rate and GPU utilization in each experiment test are measured and tabulated during the test. After all the experiment tests are completed, we calculate the average CommC and CompC for each rendering setting over all the test scenes by the Equations 3.2 and 3.3, and tabulate the results for the next offline modeling step.

Figure 3.4 shows some representative data points from the experimental results. For each adaptive rendering parameter, we first present the sample results of CommC and CompC in two figures respectively. In each of these two-figures, each plot represents a rendering setting where only one of the rendering parameters is varied (marked by "X" in the associated setting), while keeping the other parameters to the fixed values shown in the rendering setting tuple. We also show the absolute error distribution and standard deviation of CommC and CompC values obtained for different test game scenes used, when only one rendering parameter is varied while the other parameters are kept to the minimum values. From Figure 3.4, we have the following observations:

1) Realistic effect has a great impact on CompC. But it affects CommC very little, because the realistic effect has little impact on content complexity of game video. The standard deviations of CommC and CompC, when only changing realistic effect among 20 test scenes, are 0.027 and 0.0936. This infers that variations of CommC and CompC among different gaming scenes are relatively marginal if only varying realistic effect.

2) The impacts of enabling environment details on CommC and CompC are limited (up to 9%), and the standard deviations of CommC and CompC are also marginal, mainly because the effect of environment details in game PS is very simple such that varying environment details only has slight impacts on frame content complexity and computation complexity.

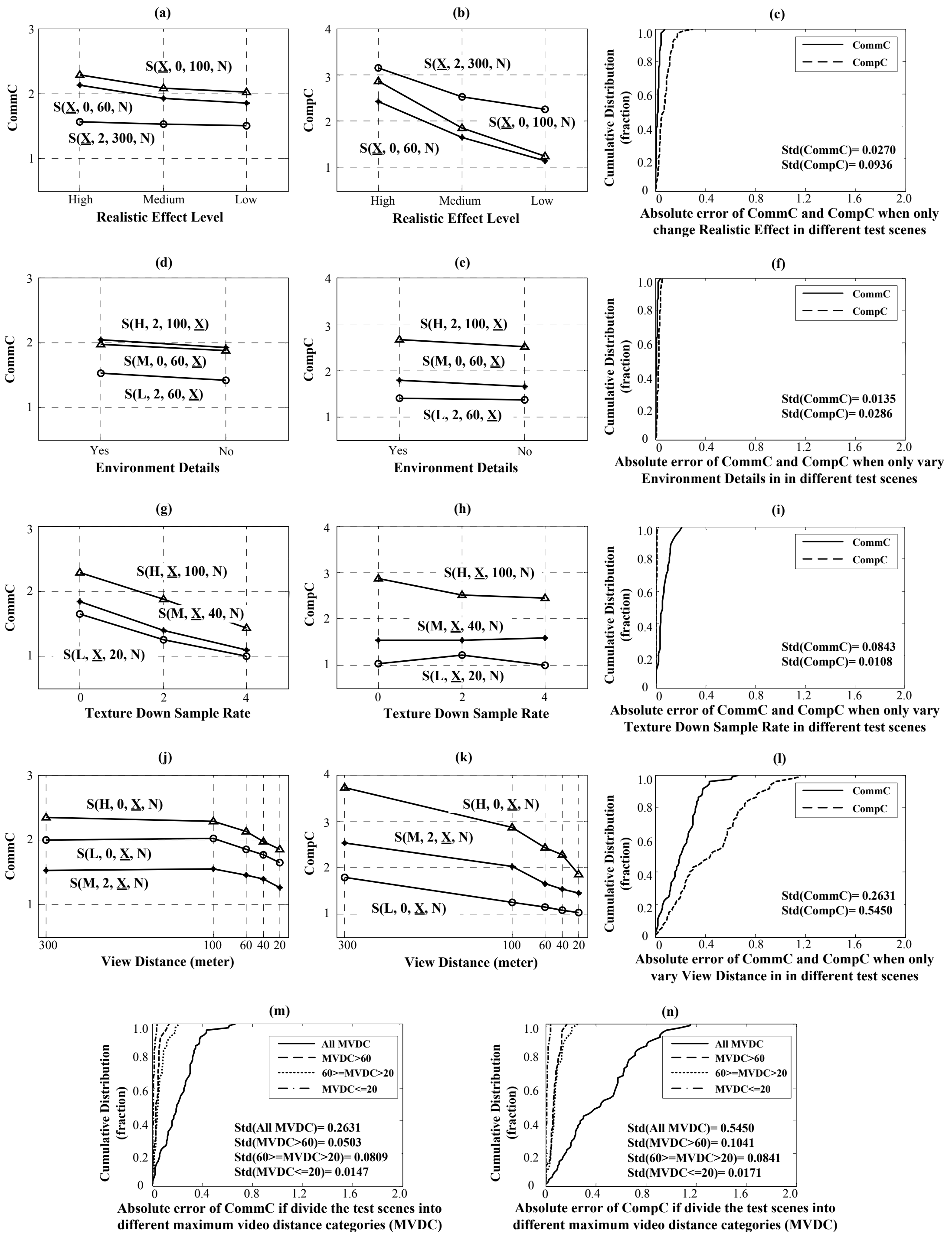


Figure 3.4. Sample data to show how CommC and CompC vary for different rendering parameters and absolute error distribution and standard deviation of CommC and CompC for each rendering parameter, for game PS.

3) Texture down sample rate significantly affects CommC. The highest CommC is about 1.6 times of the lowest CommC when we vary the texture down sample rates from 0 to 4. However, texture detail almost does not affect the CompC. This is because the reduced textures in different levels for an object are pre-calculated and loaded in the memory, so that the graphic pipeline can load the textures quickly without any additional computing cost. Similar to the realistic effect, the standard deviations of CommC and CompC when only varying texture down sample rate are not significant. They are 0.0843 and 0.0108 respectively.

4) View distance will significantly affect both CommC and CompC. While its impact on CompC is almost linear, impact on CommC becomes clear only below a certain point (100 meters). However, unlike the other rendering parameters, varying view distance in different test scenes will have a significant variation in CommC and CompC values, indicated by the higher standard deviations of CommC and CompC as 0.2631 and 0.5450 respectively. This is because the ability to reduce video content complexity by adapting view distance is limited by the maximum video distance in the real-time gaming scene. For instance, in the indoor scene where the objects rendered are all in a relatively short distance, it will become difficult to reduce the video content complexity by adapting view distance. To eliminate this constraint and improve the accuracy of our offline complexity model, we propose to divide all the test scenes into several different Maximum View Distance Categories (MVDC). Each MVDC has a certain range of the maximum view distance of the current rendered scene. The average CommC and CompC will be measured only using the test results collected in the test scenes belonging to the same scene category. Let $MVDC_j$ denote the view distance category j, and the number of test scenes in $MVDC_j$ is k_j. $MVDC_j$ is a subset of the set of all the test scenes (SCENE). Then the Equations 3.2 and 3.3 to calculate CommC and CompC need to be revised to:

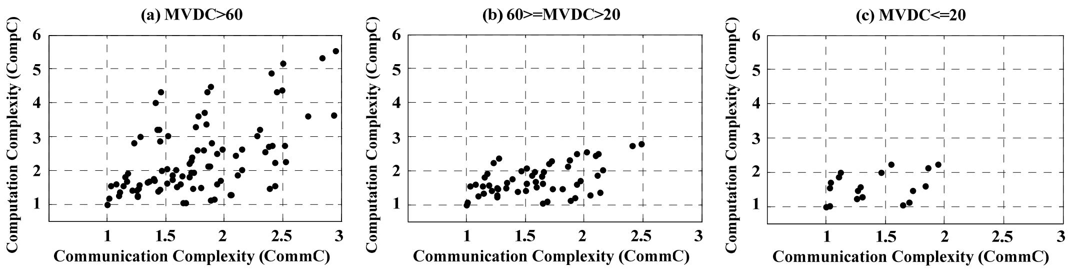


Figure 3.5. Complexity model of game PS for three different maximum view distance categories.

$$CommC(s, MVDC_j) = \frac{1}{k_j}\sum_{i=1}^{k_j}\left(\frac{BitRate(s, scene_i)}{BitRate(s_i^L, scene_i)}\right), \qquad (3.4).$$
$$scene_i \in MVDC_j, \quad MVDC_j \subset SCENE$$

$$CompC(s, MVDC_j) = \frac{1}{k_j}\sum_{i=1}^{k_j}\left(\frac{ServUtil(s, scene_i)}{ServUtil(s_i^L, scene_i)}\right), \qquad (3.5).$$
$$scene_i \in MVDC_j, \quad MVDC_j \subset SCENE$$

In order to demonstrate the efficiency of the above proposed ideas, we have divided all the 20 test scenes of game PS into three scene categories: a) maximum view distance is larger than 60 meters; b) maximum view distance is between 20 and 60 meters; c) maximum view distance is less than 20 meters. Figures 3.4(m) and 3.4(n) show the absolute error distribution and standard deviations of CommC and CompC in different MVDC. From Figures 3.4(m) and (n), we can observe that the variations of CommC and CompC in each MVDC are significantly less than the variations if we do not consider dividing the test scenes by maximum view distance. It should be noted that dividing the test scenes into different view distance categories will not increase any computation load in the offline modeling step. It

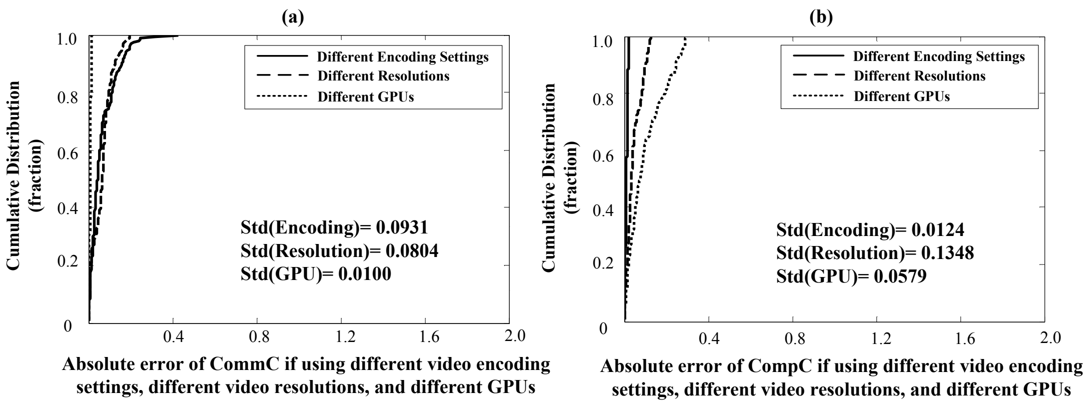


Figure 3.6. Absolute error distribution and standard deviation of measured CommC and CompC in the tests with different video encoding settings, different video resolutions, and different GPUs, for game PS.

actually has only changed the results of complexity model from one dimension to a two dimensions mapping, to make the complexity model more accurate.

The final complexity model of game PS for three different Maximum View Distance Categories is presented in Figure 3.5. Note that the complexity model will not include the rendering setting whose view distance is higher than the maximum view distance of its MVDC. For example, when MVDC is between 20 meters and 60 meters (Figure 3.5(b)), the complexity model does not have the data points for the rendering settings whose view distances are either 100 meters or 300 meters. Similarly, in Figure 3.5(c), complexity model covers the rendering settings which only have 20 meters view distance. From Figure 3.5 we can also observe that the maximum CommC and CompC becomes less when the maximum view distance becomes shorter. The results in Figure 3.5 will be subsequently used to derive rendering levels model in the section 3.4.3.

The example of complexity model we presented above was derived using a certain video encoding setting, video resolution, and GPU. We next investigate the impact of using

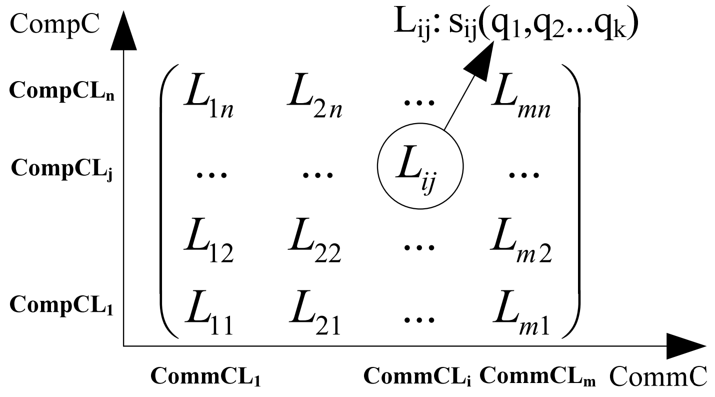

CompC
L_{ij}: $s_{ij}(q_1,q_2...q_k)$
CompCL$_n$
CompCL$_j$
CompCL$_1$
L_{1n} L_{2n} ... L_{mn}
... ... L_{ij} ...
L_{12} L_{22} ... L_{m2}
L_{11} L_{21} ... L_{m1}
CommCL$_1$
CommCL$_i$
CommCL$_m$
CommC

Figure 3.7. Rendering levels and optimal rendering setting for each rendering level.

different video encoding and resolution settings, and different GPUs, on the complexity model. We have conducted experiments and measured CommC and CompC of each rendering setting in three test cases: a) using various encoding settings (different QP and GOP settings), b) using three different resolutions (QVGA, CIF, and VGA), and c) using three different GPUs (Intel GMA4500, NVIDIA 8300, and NVIDIA GTX580). Figure 3.6 shows absolute error distribution and standard deviations of measured CommC and CompC in these test cases. From Figure 3.6, we can observe that the overall variations of CommC and CompC in these different test cases are not significant. Hence, we can conclude that the offline modeling step does not need to characterize the CommC and CompC and create different complexity models for different video resolutions, or video encoding settings, or different platforms.

3.4.3 Derivation of the Rendering Levels Model

In this section, we introduce how we leverage the complexity model to derive the rendering levels model, which is a mapping of optimal rendering setting to each rendering level. Each rendering level has two dimensions: 1) CommC rendering level, reflecting the level of network bandwidth need of that rendering level; 2) CompC rendering level, reflecting the level of computation need of that rendering level. As shown in Figure 3.7, we use L_{ij} to denote a rendering level, whose CommC and CompC rendering levels are i and j, and target complexities are $CommCL_i$ and $CompCL_j$ respectively. For each game, depending on the range of CommC and CompC values derived during the complexity modeling step, the CommC and CompC levels can be set in many ways. In our work, we evenly divide the possible ranges (from the maximum value to minimum value) of CommC and CompC into m and n levels for each MVDC:

$$CommCL_i(MVDC)= \\ 1+\frac{MaxCommC(MVDC)-MinCommC(MVDC)}{m}\times(i-1),\ \mathrm{i}\in[1,\mathrm{m}] \quad (3.6),$$

$$CompCL_j(MVDC)= \\ 1+\frac{MaxCommpC(MVDC)-MinCompC(MVDC)}{n}\times(j-1),\ \mathrm{j}\in[1,\mathrm{n}] \quad (3.7).$$

Once having target $CommCL_i$ and $CompCL_j$, given a certain MVDC, as shown in Equation 3.8 we could find a optimal rendering setting s_{ij} for each rendering level L_{ij} from all adaptive rendering settings set S, such that the root mean squared error between the target complexities (by Equations 3.6 and 3.7) of L_{ij} and measured complexities (by Equations 3.4 and 3.5) of setting s_{ij} obtained from complexity model is minimized:

$$s_{ij}(MVDC)=\{s\in S\mid \\ \sqrt{\begin{matrix}(CommCL_{\mathrm{i}}(MVDC)-CommC(s,MVDC))^2\\+(CompCL_{\mathrm{j}}(MVDC)-CompC(s,MVDC))^2\end{matrix}}\ \text{is minimized}\} \quad (3.8).$$

For instance, if we applied the above Equations 3.6, 3.7, and 3.8 with the complexity model (Figure 3.5) of game PS, we can obtain the rendering levels model of game PS, as

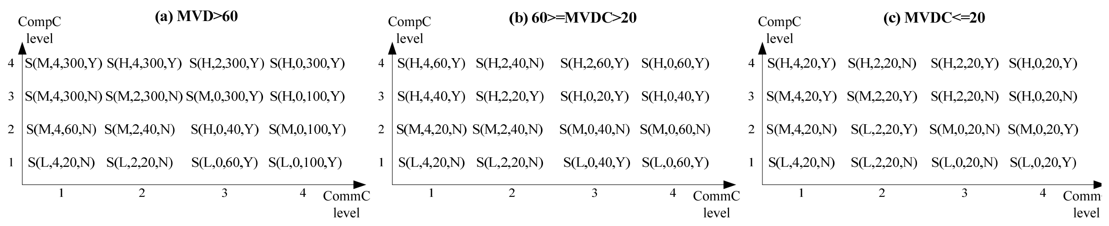

(a) MVD>60
CompC level
4 S(M,4,300,Y) S(H,4,300,Y) S(H,2,300,Y) S(H,0,300,Y)
3 S(M,4,300,N) S(M,2,300,N) S(M,0,300,Y) S(H,0,100,Y)
2 S(M,4,60,N) S(M,2,40,N) S(H,0,40,Y) S(M,0,100,Y)
1 S(L,4,20,N) S(L,2,20,N) S(L,0,60,Y) S(L,0,100,Y)
1 2 3 4 CommC level
(b) 60>=MVDC>20
CompC level
4 S(H,4,60,Y) S(H,2,40,N) S(H,2,60,Y) S(H,0,60,Y)
3 S(H,4,40,Y) S(H,2,20,Y) S(H,0,20,Y) S(H,0,40,Y)
2 S(M,4,20,N) S(M,2,40,N) S(M,0,40,N) S(M,0,60,N)
1 S(L,4,20,N) S(L,2,20,N) S(L,0,40,Y) S(L,0,60,Y)
1 2 3 4 CommC level
(c) MVDC<=20
CompC level
4 S(H,4,20,Y) S(H,2,20,N) S(H,2,20,Y) S(H,0,20,Y)
3 S(M,4,20,Y) S(M,2,20,Y) S(H,2,20,N) S(H,0,20,N)
2 S(M,4,20,N) S(L,2,20,Y) S(M,0,20,N) S(M,0,20,Y)
1 S(L,4,20,N) S(L,2,20,N) S(L,0,20,N) S(L,0,20,Y)
1 2 3 4 Comm level

Figure 3.8. Rendering levels model of game PS for three different Maximum View Distance Categories.

shown in Figure 3.8. For instance, when MVDC is higher than 60 meters, if CommC rendering level and CompC rendering level are both 2, the graphic rendering engine should use the following rendering setting: medium realistic effect, 2 for texture down sample rate, 40 meters for gaming view distance, and not enabling the environment details.

3.4.4 Derivation of the Joint Adaptation Model

Having derived rendering levels model, we can come up with an online rendering adaptation technique. During the online gaming sessions, the adaptation technique can decide a proper rendering level in real time depending on the network and server conditions, and then select the corresponding optimal rendering setting for that level. One problem of such rendering adaptation is that there can be sometimes very frequent, but limited, variations in wireless network bandwidth during a short period. If we only apply rendering adaptation to address these quick fluctuations of the network bandwidth, we may need to frequently vary the rendering levels, which probably is not desirable as game users may be sensitive if rendering levels are changed too frequently too fast during a gaming session. To address this problem, we may need the help of the traditional encoding adaptation technique, which can quickly respond to fast network bandwidth fluctuations, without game users being sensitive to the changes and without affecting the perceptual experience of the users. We have implemented a algorithm incorporating the above ideas, called Joint Rendering and Encoding Adaptation (JREA) algorithm. In JREA algorithm, rendering adaptation technique is used to address computation constraint by varying its CompC rendering level, while rendering and encoding adaptation technique will be jointly utilized to address communication constraint by varying CommC rendering level and video bit rate. However, it is imperative to know how to optimally select video bit rates and CommC rendering levels such that MGUE is maximized. In fact, for each CommC rendering level, there is a Minimum Encoding Bit Rate (MEBR) that is acceptable for the resulting video quality. And similarly, for each bit rate we use for gaming video, there is an Maximum CommC Rendering Level (MCRL) that provides the video quality which has minimum impacts on user gaming experience. We next explain how we obtain MEBR and MCRL to derive the joint adaptation model.

Minimum Encoding Bit Rate (MEBR) for each CommC Rendering Level

Encoding adaptation techniques can adapt video bit rate to the fluctuating network bandwidth to avoid network congestion. However, lowering video bit rate may lead to unacceptable gaming video quality. Therefore there is a Minimum Encoding Bit Rate (MEBR) below which gaming video quality will not be acceptable by game users. Similar to the complexity modeling, for each CommC rendering level, the offline experiment will be conducted in different test scenes. For each bit rate level, the average video PSNR of compressed game video is measured. The Minimum Encoding Bit Rate (MEBR) is the minimum bit rate which can at least provide the user minimum acceptable PSNR [WD09]. The results of MEBR for different encoding bit rate are tabulated. Thus in the online joint adaptation algorithm, when encoding bit rate used is lower than the MEBR associated with the current CommC rendering level being used, the rendering level will be adapted to a lower level to get a lower required MEBR, such that the user perceived video quality is acceptable.

Table 3.2. Minimum encoding bit rate for each CommC rendering level for game PS in VGA resolution.

Maximum View Distance Category	MVDC>60				60>=MVDC>20				MVDC <=20			
CommC Rendering level	4	3	2	1	4	3	2	1	4	3	2	1
MEBR (kbps)	300	250	200	150	400	300	250	200	500	400	300	250

Similar to complexity model and rendering levels model, MEBR also needs to divide the test scenes into different MVDC. And it should also be noted that the MEBR are dependent on the video resolution used. Table 3.2 shows the MEBR for each CommC rendering level for game PS in VGA resolution. For instance, when MVDC is higher than 60, if CommC rendering level 4 is used, the video encoder needs at least 300kbps to encode the resulting gaming video to produce the final video with acceptable gaming quality.

Maximum CommC Rendering Level (MCRL) for each Encoding Bit Rate

While encoding bit rate adaptation reduces video quality to satisfy communication bandwidth constraint, our rendering adaptation can improve video quality by lowering graphic rendering complexity. However, for a certain encoding bit rate, one should attempt to use the richest rendering possible, without having any impacts on user experienced gaming quality. Similar to the method used to derive the MEBR values for each rendering level, the following method is used to derive the MCRL for each video encoding bit rate used. For each encoding bit rate, offline experiment selects different CommC rendering levels. Then for each CommC rendering level, it measures the average PSNR of compressed video among different test

Table 3.3. Maximum CommC rendering level for each encoding bit rate for game PS in VGA resolution.

Encoding Bit Rate (kbps)	700	600	500	400	300	250	200	150
MVDC>60	4	3	3	2	2	2	1	1
60>=MVDC>20	4	3	2	2	2	1	1	1
MVDC<=20	4	3	2	2	1	1	1	1

scenes. For each encoding bit rate, the MCRL is the maximum CommC rendering level in which the resulting source video PSNR is at least higher than the excellent video quality threshold [WD09] below which user will feel the impacts due to the video quality. It is possible that in some encoding bit rates even the lowest CommC rendering level cannot meet this excellent video quality. For these bit rates, MCRL should be the lowest CommC rendering level (level 1). Table 3.3 shows the MCRL for each encoding bit rate for game PS in VGA resolution. During the gaming session, depending on the encoding bit rate used, we periodically update the CommC rendering level from Table 3.3, such that rendering setting used is maximized while user perceived video quality remains unimpaired.

3.4.5 Online JREA Algorithm

The motivation for developing an online Joint Rendering and Encoding Adaptation (JREA) algorithm is presented in the previous section 3.4.4. In this section we describe the JREA algorithm which decides when and how to switch the CommC and CompC rendering levels during a gaming session, in response to the current network conditions and server utilization.

Firstly, we need to make sure the CommC is lower than available network bandwidth. Otherwise, the network will be congested, leading to high response time and packet loss. Network delay has been widely used to indicate congestion (bandwidth constraint) [PMD$^+$03]. However, in mobile networks, network delay is not always caused by congestion. To address this problem, we make use of the observation [DJ02] that network delay keeps increasing when the network link starts to become congested, while delay in a non-congested network does not have this property. To accurately detect network congestion, in this work we decide a network link is congested, only when the Round trip Delay (RDelay) keeps increasing for a certain period and the average RDelay is bigger than a certain threshold. With the Mobile Gaming User Experience (MGUE) model and associated Game Mean Opinion Score (GMOS) described in Chapter 2, for each game, we can obtain the minimum Acceptable Round-trip Delay RD_A, which is the minimal delay threshold to achieve an acceptable MGUE (GMOS > 3.0). We thereby use this RD_A as the round-trip delay threshold to determine network congestion.

To decide on the proper CompC, we monitor the server GPU utilization. We define an upper utilization threshold, U_1, and a lower utilization threshold, U_2, and use them to decide CompC rendering level. Figure 3.9 shows the flow chart of the JREA algorithm. At short time interval λ, depending on the network conditions, network Round-trip Delay (RDelay) and packet Loss (Loss), server utilization (ServUtil), and the Maximum View Distance Category (MVDC) of the current game scene, the JREA algorithm decides to select a lower or higher CommC rendering level I, CompC rendering level J, and encoding bit rate K, such that 1) network round trip delay threshold RD_A is met, and 2) gaming video quality is maximized. The JREA algorithm consists of three steps as follows:

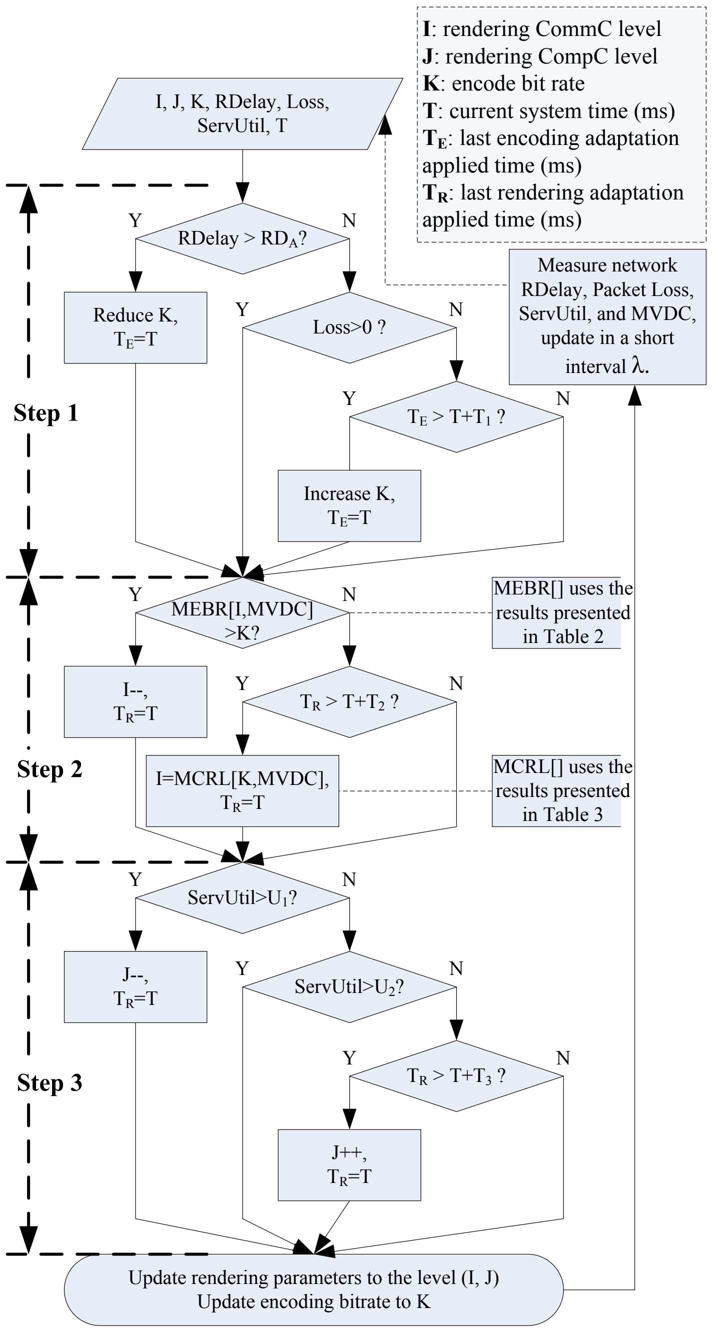

I: rendering CommC level
J: rendering CompC level
K: encode bit rate
T: current system time (ms)
T_E: last encoding adaptation applied time (ms)
T_R: last rendering adaptation applied time (ms)
I, J, K, RDelay, Loss, ServUtil, T
RDelay > RD_A?
Reduce K, T_E=T
Loss>0 ?
T_E > T+T_1 ?
Increase K, T_E=T
Measure network RDelay, Packet Loss, ServUtil, and MVDC, update in a short interval λ.
Step 1
MEBR[I,MVDC] >K?
I--, T_R=T
T_R > T+T_2 ?
I=MCRL[K,MVDC], T_R=T
MEBR[] uses the results presented in Table 2
MCRL[] uses the results presented in Table 3
Step 2
ServUtil>U_1?
J--, T_R=T
ServUtil>U_2?
T_R > T+T_3 ?
J++, T_R=T
Step 3
Update rendering parameters to the level (I, J)
Update encoding bitrate to K
Y
N

Figure 3.9. Flow chart of joint rendering and encoding adaptation algorithm.

1) The first step is to decide the encoding bit rate K used to encode the rendered video. During a certain period λ, if the network RDelay keeps increasing and its average value is greater than RD_A, we reduce encoding bit rate K. On the other hand, if for a significant time T_1, RDelay remains below RD_A and there is no packet loss, we increase the encoding bit rate.

2) The second step is to check and update CommC rendering level I. After the first step, the new encoding bit rate may be below the Minimum Encoding Bit Rate (MEBR) for the current CommC rendering level I and Maximum View Distance Category (MVDC), which will lead to an unexpected user experience as we discussed before. If this happens, we have to reduce CommC rendering level to reduce the Minimum Encoding Bit Rate. On the other hand, if the CommC rendering level has not been changed for over a certain significant period T_2, it will be updated and changed to the Maximum CommC Rendering Level (MCRL) depending on the current encoding bit rate and MVDC.

3) The third and last step is to decide on CompC rendering level J, depending on server utilization ServUtil. If ServUtil is over U_1, the lower CompC rendering level is selected. Otherwise, if CompC rendering level has not been changed for more than time T_3, and ServUtil is below U_2, we increase CompC rendering level J by 1.

Next, based on the new selected CommC and CompC rendering levels we use the optimal rendering settings from Figure 3.8 to update the game engine server, while we use new selected video bit rate to update the game streaming server.

3.5 Experiment and Results

In this section, we report on experiments conducted to verify the effectiveness of the proposed rendering adaptation technique. We first conduct experiments in a commercially available cellular mobile network, which can validate the effectiveness of our technique to satisfy wireless mobile communication constraint. The results show that it can lead to about 4 times reduction in video bit rate, while ensuring an acceptable MGUE. Next, we conduct experiments in a controlled environment which can validate the effectiveness of rendering adaptation technique in addressing computation constraint. The experiment results show that the proposed technique can ensure a high stable rendering frame rate, thus a good user experience, while the number of users on a CMG server increases by almost 5 times. The experiment results of these tests demonstrate that the proposed rendering adaptation technique can ensure MGUE and high scalability of the CMG approach in regards to both network bandwidth and server computing needs, thereby making Cloud Mobile Gaming economically feasible.

We have explained the parameters of online JREA algorithm in section 3.4.5. Generally, λ has to be a relative small period such that the JREA algorithm can quickly respond to the variations in network. The update periods (T_2 and T_3) for rendering levels should be higher than the update period (T_1) for encoding bit rate. The upper GPU utilization threshold U_1 should be much higher than lower GPU utilization threshold U_2, to avoid the adaptation from oscillating. For the experiments reported in this chapter, we use the values 3 seconds, 20 seconds, 60 seconds, 60 seconds, 90%, and 40% for the parameters λ, T_1, T_2, T_3, U_1, and U_2 as respectively. The game used is PlaneShift, its RD_A is 440ms [WD12][WD09]. The rendering levels model and joint adaptation model for game PS are pre-calculated on off-line experiments. The Maximum View Distance Category (MVDC) is measured and provided by the PS game engine during the gaming session. It should be noted that our proposed JREA algorithm can leverage any video encoding adaptation technique with our proposed rendering adaptation technique. In this chapter, for the purpose of experimental results, we use a video encoding adaptation technique [WDA10] which has been shown to produce better video quality by being aware of the gaming content.

3.5.1 Addressing Communication Constraint

To evaluate the effectiveness of our proposed technique in addressing communication constraint, we carried out multiple experiments using a 3G network and multiple test environments (locations, times) having different network conditions. In this section, we select one of the test environments to compare and discuss in details the effectiveness of our proposed rendering adaptation technique. Figure 3.10(a) shows a test environment used in the UCSD campus, where the tests are conducted starting at an indoor location A (our lab), then at location B (outdoor), C (indoor), D (indoor), then from D back to C, B. The locations are selected as they display different network conditions: network bandwidth, and signal-to-noise ratio (SINR), received by the mobile device at each location. In this case, the SINR of these four places from highest to lowest are B, A, C, D.

We first measure the maximum mobile network throughput at these locations, as presented in Figure 3.10(b). Figures 3.10(c)(d)(e) present the data collected from experiments in three scenes: without using any adaptation technique, only using game aware video encoding adaptation technique [WDA10], and using our proposed rendering adaptation technique, respectively. We provide below a summary of the key observations from our experiments:

1) In the experiment without using any adaptation technique, we stream the gaming video at the rate of 700kbps where the source video quality is good enough to avoid deteriorating user experience. Figure 3.10(c) shows the resulting round-trip delay (RDelay), Packet Loss, and GMOS score of the gaming session. At outdoor location B, as the wireless channel rate is high due to good SINR, the resulting network delay and packet loss are

relatively low, and hence the GMOS score measuring the mobile gaming user experience is mostly above 3.0 (acceptable user experience threshold). However, at indoor locations, A, C, and D, due to the bad SINR, the wireless channel rate adapts to a lower level. This causes network congestion, reflected by the high network delay and packet loss rate, leading to unacceptably poor gaming quality (GMOS scores below 3.0).

2) In the experiment only using encoding adaptation, the encoding bit rates used are adapted to the fluctuating network conditions. Figure 3.10(d) shows the resulting network delay, packet loss, the encoding bit rates used, the PSNR of the encoded video, the game Maximum View Distance Category (MVDC) and the resulting GMOS score at the different locations. As shown in Figure 3.10(d), the network congestion is almost eliminated, and as a result, network delay and packet loss are greatly improved at all the locations. However, the video quality (PSNR) is deteriorated while lowering video bit rate by encoding adaptation. When wireless network bandwidth is extremely low in bad SINR locations, like D, the user has very poor experience, reflected by poor GMOS, primarily due to the poor video quality.

3) Figure 3.10(e) shows the results of applying the proposed rendering adaptation technique, including the resulting adaptive bit rates and CommC rendering levels used. In contrast to the results shown in Figures 3.10(c)(d), Figure 3.10(e) shows that application of our rendering adaptation technique can greatly improve the network delay and packet loss rate, while maintaining a good video quality (PSNR). Consequently, the user gaming experience, which includes response time, is significantly enhanced at all the locations, reflected by the relatively high and stable GMOS, dipping below 3.0 only very occasionally when the adaptation algorithm is responding to the channel rate variations.

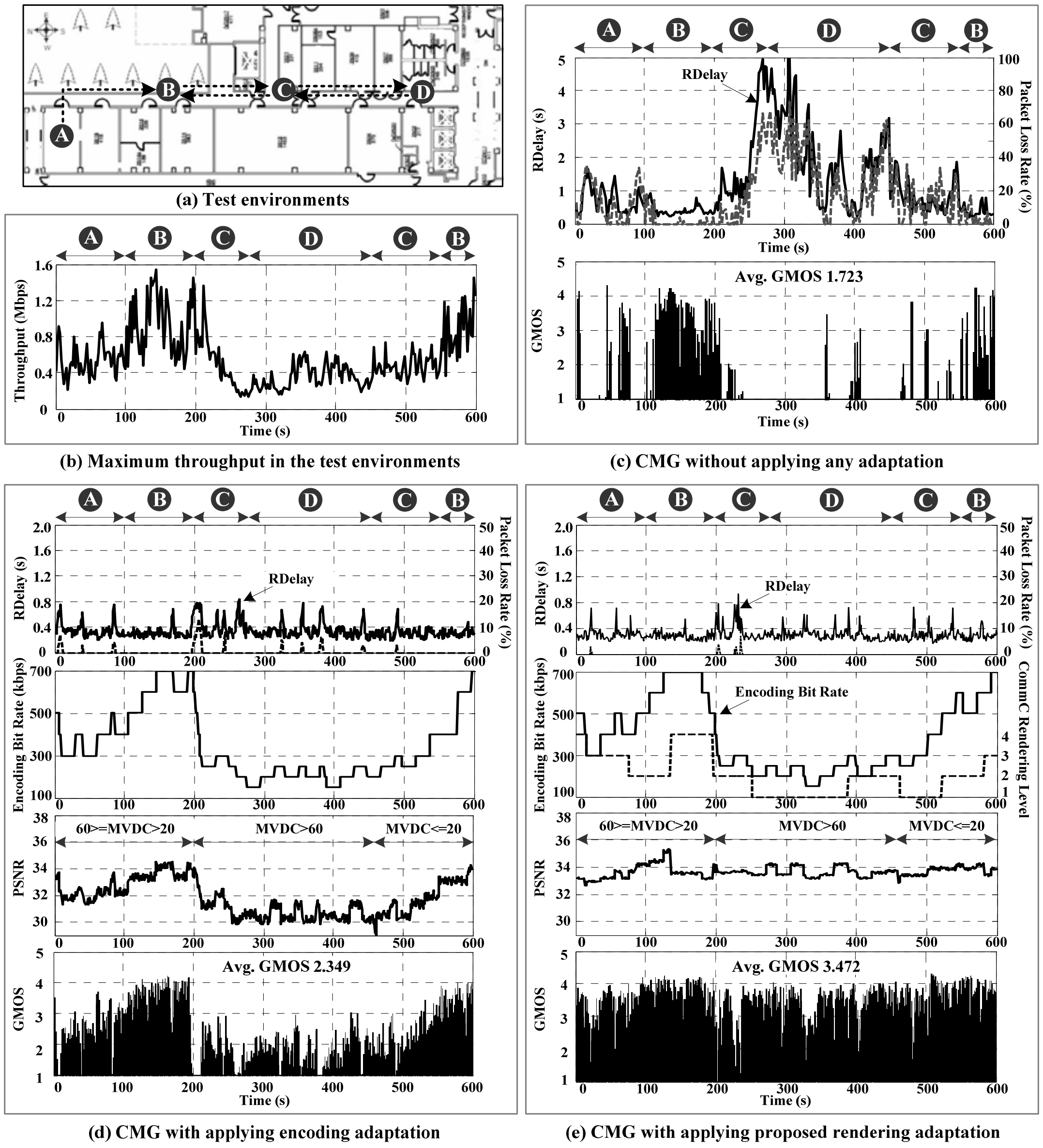


Figure 3.10. Experiment results to demonstrate the effectiveness of proposed rendering adaptation technique to address CMG communication constraint: (a) test environments; (b) network bandwidth measurement results; (c) CMG without any adaptation technique; (d) CMG with video encoding adaptation technique only; (e) CMG with our proposed rendering adaptation technique.

3.5.2 Addressing Computation Constraint

As discussed before, considering that the cloud mobile gaming (CMG) servers will need to execute concurrently a large number of compute intensive game engines (to support the concurrent gaming sessions), it is imperative to show the scalability of the CMG servers. To demonstrate the effectiveness of proposed rendering adaptation technique in addressing computation constraint so as to ensure scalability of the CMG servers, we conduct experiments where we increase server GPU load by increasing the number of concurrent game engine tasks executed. The experimental server integrates a NVIDIA GTX580 graphic card. We initialize the CMG server with one game engine for a cloud mobile gaming session. After every 60 seconds, we start a new PS game engine for each new concurrent gaming session. Each game engine is configured to render 15 frames per second, but the actual rendered frame rate produced may drop below 15 if the GPU resource is over utilized. In the latter cases, the gaming user experience will suffer, with either gaming video appearing jerky, or the response time appearing slower than expected.

Figure 3.11 presents the effects of increasing the number of concurrent gaming sessions on a CMG server, and the resulting server GPU utilization and the rendering quality achieved (rendering frames per second) for one sample gaming session, without and with applying rendering adaptation technique (Figures 3.11(a) and (b) respectively). It should be noted that when the server utilization goes above the upper utilization threshold U_1, the adaptation technique will adapt the CompC rendering levels on all the gaming engines executed. Also showed in Figure 3.11(b) is the MVDC used for the sample gaming session during the test. We summarize below the following observations:

1) As shown in Figure 3.11(a), without the use of our rendering adaptation technique, the CMG server can support only 6 CMG sessions with good quality (expected rendering rate

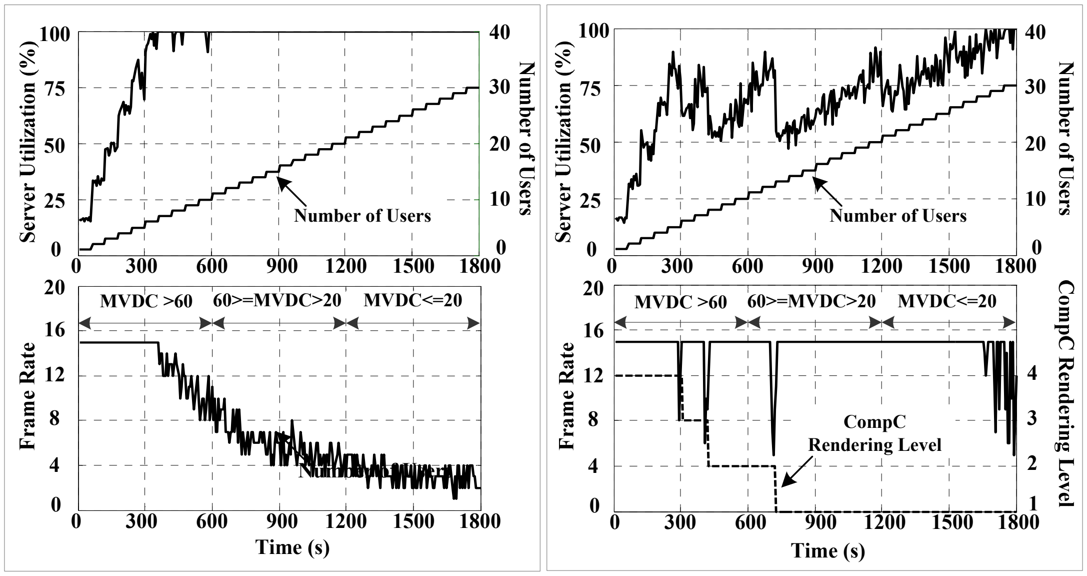


Figure 3.11. Experiment results to demonstrate the effectiveness of proposed rendering adaptation technique in addressing CMG computation constraint.

of 15fps), as the GPU utilization reaches 100% when we add the 7th gaming session and the rendered frame rate drops to as low as 12. With the addition of each new game engine task, the rendering frame rate keeps going down, to as low as 1 frame per second when the CMG server has to execute 30 concurrent game engines.

2) In contrast, as shown in Figure 3.11(b), the CMG server can support up to 27 clients when using our proposed rendering adaptation technique, without deterioration in rendered quality (rendered frame rate). With appropriate adaptation of the CompC rendering levels used, rendering adaptation technique is able to ensure that the GPU is able to deliver the expected 15 fps for each gaming session, dipping a little below 15 only very occasionally when the adaptation algorithm is responding to the changes in-server GPU loading. The above experimental results demonstrate that our proposed rendering adaptation technique can address computation constraint of CMG server by properly adapting the computation need of

each mobile gaming session, thereby increasing by almost 5x the scalability of the CMG application.

3.6 Conclusions

To ensure the feasibility of the CMG approach, it is vital to provision high Mobile Gaming User Experience (MGUE) for each CMG user, and ensure the scalability both in terms of communication and computation needs. To achieve our proposed objectives, in this chapter, we have developed a dynamic game rendering adaptation technique which can simultaneously vary the richness and complexity of graphic rendering to adapt the communication and computing needs of each CMG session in responding to the dynamic conditions of the wireless mobile networks and CMG server. The experiments conducted on the commercial cellular mobile network and CMG server demonstrate the effectiveness of the proposed adaptation techniques to make the CMG approach feasible: ensuring protection against wireless network conditions and thus ensuring acceptable mobile gaming user experience and scalability in regards to the availability of the mobile network and CMG server.

The text of this chapter, in part or in full, is based on material that has been published in IEEE Transactions on Multimedia (TMM) (S. Wang, S. Dey, " Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications," IEEE Transactions on Multimedia, Jun. 2013) and material published in IEEE Global Communications Conference (Globecom) (S. Wang, S. Dey, "Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming," IEEE Globecom, Miami, Dec. 2010).

The dissertation author was the primary researcher and author in the publications, and the coauthors listed supervised the research that forms the basis of this chapter.

Chapter 4

Mobile Cloud Scheduling: Scheduling Heterogeneous Network and Cloud Resources to Enable Scalable Cloud Mobile Gaming

In this chapter, we address the problem of making Cloud Mobile Gaming scalable and economically feasible by proposing a novel Mobile Cloud Scheduling (MCS) solution that increases the allowed number of concurrent CMG sessions to be scheduled. The method is implemented while satisfying the Quality of Service (QoS) requirements for each scheduled user using the available wireless network resources while minimizing the cloud cost incurred by the CMG provider. We consider heterogeneous cloud and network resources with different capacities and delays, including cloud resources in the mobile network supplementing traditional Internet clouds and heterogeneous access networks, including macro cells, small cells, and WiFi. Unlike conventional mobile network or cloud schedulers, MCS considers simultaneously the constraints of the wireless access networks that may be available to each CMG user, as well as the cost of available cloud resources, while scheduling the most optimal wireless access link and cloud server for each CMG session. To further enhance the performance of MCS, we also propose a joint scheduling-adaptation approach that can systematically leverage application adaptation techniques proposed in Chapter 3 to adapt the communication bit rate needs of in-service users if the available wireless network bandwidth is not sufficient for a new user, or adapt the computational needs if the reserved cloud resource is over utilized. Our simulation results demonstrate that the use of MCS and the joint scheduling-adaptation approach, can significantly improve the performance of the CMG method , increasing the number of simultaneous CMG sessions which can be supported, while maximizing aggregate Mobile Gaming User Experience (MGUE) and minimizing the average cloud cost.

Section 4.1 introduces the overview and objectives of mobile cloud scheduling, and compares our contributions with related work. In section 4.2, we identify and formulate mobile cloud scheduling problems. The Mobile Cloud Scheduling (MCS) approach and Joint Scheduling-Adaptation (JSA) approach are proposed and studied in sections 4.3 and 4.4. In section 4.5, we perform a set of simulation experiments to compare and characterize the performance of our proposed approaches and the performance of the original CMG approach where MCS and JSA approaches are not applied. The simulation results show that a) our proposed MCS can help the CMG approach to minimize the number of blocked users and cloud cost while ensuring the user perceived MGUE when comparing with the original CMG approach under the same conditions; b) the proposed JSA approach can significant increase (by about 2.5 times) the system capacity while ensuring the minimum acceptable MGUE for each assigned user. Section 4.6 summaries our findings in this chapter.

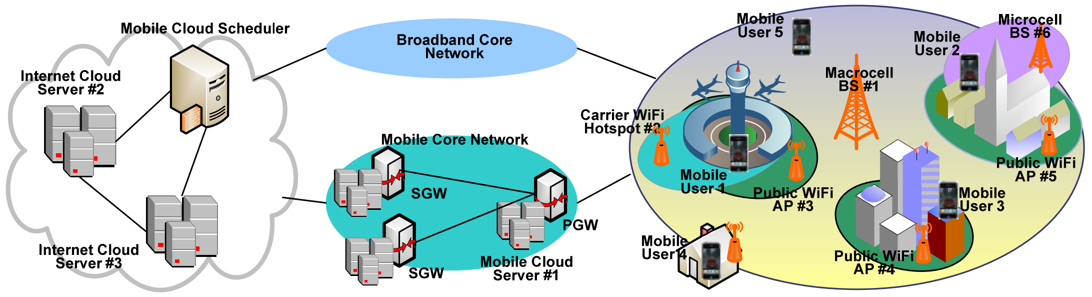
4.1 Mobile Cloud Scheduling: Overview, Objectives, and Related Work

In this section, we first provide an overview of Mobile Cloud Scheduling (MCS) and summarize the objectives of the proposed MCS approach. While the problem of mobile cloud scheduling has not been addressed before, we summarize previous work in cloud and network scheduling, and compare with our contributions.

4.1.1 Overview of Mobile Cloud Scheduling Problem

Figure 4.1 shows the architecture and eco-system of a heterogeneous mobile cloud computing system, consisting of Heterogeneous Network (HetNet) access nodes, cloud computing servers in both the mobile network as well as traditional Internet cloud servers, and a Mobile Cloud Scheduler (MCS) proposed in this dissertation. To deal with tremendous growth in mobile wireless data demands, mobile network operators are increasingly adopting the Heterogeneous Network (HetNet) architecture [ARC11][PQ11]. The heterogeneous network can consist of different cell sizes which range from macro to micro, pico and even femto cells, potentially sharing the same spectrum. Nodes can deploy different access technologies such as LTE and WiFi, over both licensed and unlicensed bands. While the macro cells provide coverage, small cells (micro, pico and femto) complement macro cells to provide high capacity and better indoor coverage.

In Figure 4.1, we show an example scenario consisting of four types of wireless access nodes: macrocell Base Stations (BS), microcell BSs, carrier WiFi Hotspots, and public WiFi Access Points (AP). The application data from/to mobile device has to be routed via a



Available Options For Mobile User 1	Mobile Cloud Server #1 (Available Capacity $a^S_1 = 90$) (Probability $p^S_1 = 20\%$)	Internet Cloud Server #2 (Available Capacity $a^S_2 = 10$) (Probability $p^S_2 = 60\%$)	Internet Cloud Server #3 (Available Capacity $a^S_3 = 30$) (Probability $p^S_3 = 20\%$)
Mobile Macro Base Station #1 (Available Capacity $a^N_1 = 20$, Probability $p^N_1 = 30\%$)	Choice A: 83ms (std:21ms)	Choice B: 135ms (std:47ms)	Choice C: 171ms (std:53ms)
Carrier WiFi Hotspot #2 (Available Capacity $a^N_3 = 10$, Probability $p^N_3 = 45\%$)	Choice D: 74ms (std:18ms)	Choice E: 127ms (std:43ms)	Choice F: 167ms (std:48ms)
Public WiFi AP #3 (Available Capacity $a^N_4 = 90$, Probability $p^N_4 = 25\%$)	Choice G: NA	Choice H: 93ms (std:23ms)	Choice I: 145ms (std:32ms)

Figure 4.1. Heterogeneous mobile cloud computing system, and an instance of the mobile cloud scheduling problem.

service provider's core network - in this case, the macrocell and microcell BSs and carrier WiFi Hotspots connect to the Internet through mobile provider's core network, while public WiFi APs uses the broadband core network. Most of mobile users (like mobile users 1, 2, and 3 in Figure 4.1) can access using more than one wireless node; for example, user 1 can access through macrocell BS #1, carrier WiFi Hotspot #2, and public WiFi AP #3; user 2 can access through macrocell BS #1, public WiFi AP #5, and microcell BS #6; and user 3 can access through macrocell BS #1 and public WiFi AP #6. However, due to the wireless coverage and device constraints, at any given time, some users may only be able to connect only one node, like mobile users 4 and 5 in Figure 4.1.

To enable rich mobile applications on mobile devices, CMG applications can utilize the cloud servers in the Internet Cloud. However, a critical challenge for CMG applications is to satisfy the round-trip latency requirement between the mobile device and the cloud servers. Moreover, the transmission of large amount of content between cloud servers and mobile devices poses a major concern for the capacity of the networks to enable CMG applications.

To address these concerns, we propose a new architecture which supplements Internet cloud servers with cloud servers located in the mobile carrier core network, so that the content processing and retrieval can be performed at the edge of the mobile networks, as opposed to in Internet clouds, thereby reducing round-trip latency, as well as reducing the traffic load between the Internet cloud and mobile core network. Figure 4.1 shows an example with the service gateways (SGW) and packet gateways (PGW) of a mobile core network supplemented by cloud servers.

We assume CMG provider reserves Internet cloud servers and mobile network cloud servers from Internet cloud providers and mobile network operators respectively, and reserves certain amount of access capacity from mobile network operators and WiFi service providers. While a certain number of cloud servers are reserved by the CMG provider, at any time any excess computing demand may be met using the pay-as-you-go model for cloud computing, though the latter is more expensive than the reservation cost. On the other hand, we assume the mobile cloud service provider is constrained by the wireless access capacity reserved by it, and the pay-as-you-go model is not available for wireless access capacity.

The mobile cloud scheduling, which is operated by the CMG provider, will keep monitoring the network availabilities and server utilization from the cloud server and access network reservations made by the mobile cloud service provider. When a new mobile cloud user requests service, it will schedule the proper access network and cloud server for this user. At any given time, there usually might be multiple choices available to serve a new requesting mobile cloud user. For instance, the table in Figure 4.1 shows all the possible scheduling choices for the mobile user 1, including the possible networks and the bandwidth utilization for each network, the possible cloud servers and the resource utilization for each server. In addition, we have shown a usage probability for each network and cloud server, which indicates the probability that a requesting user will use this network or cloud server; the usage probability can be maintained based on usage statistics. For each pair of access network and cloud server, we have provided the average and standard deviation of network round-trip delay obtained from thirty measurements conducted using commercial wireless networks and cloud servers. The networks used in the measurements are Verizon 3G (macrocell BS #1), AT&T WiFi (carrier WiFi Hotspot #2), and Time Warner Cable WiFi (public WiFi AP #3). Note the round-trip delays with respect to AT&T Packet Gateway (column 2) are significantly lower than when using either the Amazon server located in Southern California (column 3) or Amazon server located in Eastern US (column 4), justifying the inclusion of mobile cloud servers in our heterogeneous architecture.

The mobile cloud scheduling decision from the multiple choices available for a new mobile user request can be complex because the available choices may force a tradeoff between available capacity and usage probability of access network, available capacity and usage probability of cloud server, and round-trip delay (which affects user experience). For example, if minimizing the number of *Blocked Users (BUsers)*, users that cannot be scheduled due to the network bandwidth constraint, is the primary objective, we should select the network with the highest available capacity and the lowest usage probability, like public WiFi AP #3 in Figure 4.1. If preventing over-utilizing cloud resources is the primary objective, we should select the cloud server with the highest available capacity and the lowest usage probability, like mobile cloud server #1 in Figure 4.1. However, the mobile users connected to public WiFi AP #3 may not be able to access the mobile cloud server #1 due to the policy issue. Therefore, choice G is not available for mobile user 1. Besides choice G, if MCS wants to select the mobile cloud server #1, the cloud server which has the highest available capacity and the lowest usage probability, choice A is better than choice D in terms of available capacity and usage probability of access network. Similarly, if MCS wants to select public WiFi AP #3, the access network with the highest available capacity and the lowest usage probability, choice I is better than choice H in terms of the cloud server capacity and usage probability. However, it is still difficult to decide which one is better between choice A and choice I: a) if choosing choice A, the number of blocked users due to the subsequent requests on mobile BS #1 will increase; b) if choosing choice I, the chance of over-utilizing reserved cloud resource on Internet cloud server #3 for the future requests will increase.

In this chapter, we propose a mobile cloud scheduling approach which takes into account the available capacity and usage probability of each heterogeneous access network and cloud server choice for each mobile user, and the corresponding network round-trip delays, to maximize the number of concurrent mobile cloud users that can be served and minimizing the average cloud cost, while satisfying their QoS requirements, including response time.

4.1.2 Mobile Cloud Scheduling Objectives

In this section, we describe in details the objectives for mobile cloud scheduling. Due to the limited network resource reserved by CMG provider, it is possible that a user request cannot be scheduled due to the insufficient network available capacity. Hence, one of the objectives of mobile cloud scheduling should be to minimize the number of blocked users (*BUsers*) given the available (reserved) access network capacity. A challenge for mobile cloud scheduling is that when a new user needs to be scheduled, the already live (in-service) users cannot be rescheduled, that is, a new optimal schedule cannot be obtained for all concurrent users with all available network and cloud resources. This is because rescheduling the live users to either different access networks or cloud servers may introduce unacceptable interruption and delay in the mobile cloud sessions. Because it is not feasible to reschedule the resources for the live cloud users every time a new user request comes, the scheduling decision for any requesting user will affect the availability and hence scheduling decision of future users. For instance, in Figure 4.1, scheduling a requesting user to a highly utilized network resource among several available choices (like carrier WiFi Hotspot #2) may increase the number of blocked users (due to the subsequent requests on carrier WiFi Hotspot #2) in the long run. Therefore, scheduling decision for a new user has to be performed in a way that minimizes blocking of future users/requests, so as to minimize the value of *BUsers*.

Second, unlike enterprise applications, the CMG application is extremely compute intensive and network bandwidth demanding, leading to a high cloud cost. Besides being expensive, the cloud costs are different across different cloud instance types. As we discussed in Chapter 1, Amazon Elastic Compute Cloud (EC2) [EC2] offers two types of cloud instances: on-demand instances and reserved instances. On-demand instances let customers pay for compute capacity by the hour with no long-term commitments, while reserved instances give customers the option to make a low, one-time payment for each instance that customers want to reserve and in turn receive a significant discount on the hourly charge for that instance. Table 1.1 has shown the costs for each world of warcraft CMG session using two different AWS EC2 instances. From Table 1.1, we can observe that cloud cost of using on-demand cloud instances is much higher (more than 2x) than the cloud cost of using reserved cloud instances, due to higher computing price of on-demand cloud instances. Hence, in this chapter, we assume that the mobile cloud service provider reserves a certain set of Internet and mobile cloud servers, which forms the cloud resource constraint, above which the provider will have to pay for Additional computing Cost (*ACost*) which is charged on-demand. Similar to the previous discussion for network constraint, because mobile cloud scheduling cannot reschedule the in-service users, the scheduling decision for a new user will impact cloud server availability for future users, and may impact the *ACost* that mobile cloud service provider may pay for the future users. Therefore, for scheduling decision the MCS needs to consider the cloud server conditions and try to find out the optimal cloud server for the current request such that the future *ACost* will be minimized.

Third, the proposed mobile cloud scheduling technique needs to satisfy Quality of Service (QoS) requirements of an CMG application, which can be characterized by: acceptable Round-trip Delay $RDelay_A$ (above which user cannot tolerate the service any more), application bit rate needed $CMG_{DataRate}$, and computation needed CMG_{Comp}.

In summary, for the success of Cloud Mobile Gaming approach it is vital for mobile cloud scheduling technique to achieve the following three objectives: 1) minimizing the number of Blocked Users (*BUsers*), 2) minimizing the Additional computing Cost (*ACost*), 3) and satisfying QoS requirements to meet acceptable user experience for each scheduled user.

4.1.3 Related Work

The problems of cloud computation scheduling and network communication scheduling have been studied extensively but mostly considered in isolation. In [AAB+10] [Gho+11] [LCK11] [BVB10] [Bel+10] [NKG10] [CGF+10], problems of assigning computing resources to a cloud user over a set of cloud servers are studied, to achieve any or multiple of the following objectives: increasing resource utilization [AAB+10], ensuring fairness among requesting tasks [Gho+11] [LCK11], reducing cloud cost [BVB10], reducing energy consumption [Bel+10], and alleviating the impacts of cloud resource fluctuation on service quality a cloud user perceives [NKG10] [CGF+10]. On the other hand, scheduling and allocation of wireless network resources has been widely studied and incorporated in current wireless networks. For example, [VDG05] [SKG+11] propose various types of wireless network schedulers to allocate network bandwidth to users for different goals, including maximizing the aggregate throughput [SKG+11] [CKX+08] [EMC+07], achieving network fairness [VDG05] [CKX$^+$08] and maintaining quality of service [EMC$^+$07]. However, the above techniques do not consider the problem of scheduling considering simultaneously both cloud computing and wireless network resources, which is the goal of our work. There has been one prior research [MSD07] which has considered scheduling computing tasks on mobile devices to remote processors, considering both the availability of the computing resources as well as the bandwidth availability of the mobile access networks. However, the above technique needs knowledge of all the tasks that need to be scheduled and all the available resources to make a scheduling decision, and hence is not suitable for the MCS problem we address in this chapter, where the set of CMG users can change dynamically, and rescheduling the entire set of users every time a new user requests service would be too computationally expensive. Moreover, the approach in [MSD07] considers fixed computing resources, while MCS needs to consider elastic cloud resources, where the prime consideration is cloud cost and not the constraint on availability of computing resources.

Next, we will give a problem formulation for Mobile Cloud Scheduling which will meet all its objectives. Subsequently, we will propose a MCS approach which is designed to achieve the three objectives discussed in section 4.1.2. Consequentially, we will propose a Joint Scheduling-Adaptation (JSA) approach which can leverage the application adaptation techniques to ensure the scalability of CMG applications.

4.2 Problem Formulation for Mobile Cloud Scheduling

In this section, we develop a formal definition of the mobile cloud scheduling problem. We first present our system model and assumptions, and then present a formal definition of the problem based on this model.

4.2.1 Mobile Cloud System Model

Our system consists of cloud servers located in different locations with different reserved computing capacities, heterogeneous network (HetNet) offering multiple wireless access methods, and mobile clients who can enter or leave the system at random times. A mobile client can be connected to multiple wireless access networks, for example a cellular macrocell, a cellular microcell, a carrier WiFi network, and a public WiFi network, as shown in Figure 4.1.

As we described in section 4.1, we assume that the CMG provider reserves certain amount of access capacity from mobile network operators and WiFi service providers. The capacity (in terms of number of users that can be simultaneously served) for each access network is determined by the total bandwidth of the network reserved by CMG provider. Different types of CMG applications may have different network bandwidth needs. In our system, we assume the scheduler estimates the bandwidth need of each CMG session as the average bandwidth need of the different types of CMG applications being supported. Based on the above assumptions, we use a term a^N_i to denote the available network capacity for any access network i at a given time, which is the number of concurrent users that can be additionally severed by this network, taking into account the number of users it is currently serving.

Similarly, the computation need of a CMG session also depends on the application type. We assume the scheduler estimates the computation need of each session as the average computation need among all types of CMG applications. As we mentioned before (section 4.1), to reduce the cloud cost, CMG provider will reserve a certain amount of computing units on each cloud server. Given the available reserved computing units and the average computation need of the CMG applications, the number of concurrent users that can be severed on any server *j* can be calculated and denoted as a^S_j.

In addition, according to the statistical history of the CMG system, we can obtain the maximum concurrent number of users in the system (denoted as *MAX*), and the probability distribution of all the possible concurrent numbers of users in the system at any given time (denoted as *P(x)*, where *x* is the concurrent number of users in the system). Also can be obtained from statistic history are the access probability of any network *i* (p^N_i), the probability that any new requesting user may use the network *i*, and the access probability of any server *j* (p^S_j), the probability that any new requesting user may use server *j*. As we mentioned in section 4.1.2, the MCS has to satisfy QoS requirements (including minimum acceptable round trip delay $RDelay_A$, application bit rate need $CMG_{DataRate}$, and computation need CMG_{Comp}) to meet acceptable user experience for each scheduled user. Because cloud resource is elastic, CMG_{Comp} can be always satisfied. To meet $CMG_{DataRate}$, MCS needs to guarantee the available capacity of network *i* is higher than 1 (i.e. a^N_i>1). We use $RDelay_{ij}$ to denote the achievable network round trip delay if using network *i* and cloud server *j*. Let $RDelay_A^{\mu}$ denote the minimum acceptable round trip delay for the new mobile user μ. To provision a user acceptable network delay, MCS needs to make sure $RDelay_{ij}$ is lower than $RDelay_A$.

While MCS will periodically monitor the system status including the network delays and resource capacities, it will only be invoked for scheduling when mobile clients enter or leave the system. When the mobile user μ starts a CMG session, the MCS will check all the available assignments of access networks and cloud servers of user μ (denoted as R_{μ}), and assign an appropriate wireless network *i* and a proper cloud server *j* (denoted as assignment r_{ij}), such that the QoS requirements of the requested CMG session will be met. Note that it is possible that MCS cannot find any appropriate assignment for user μ, due to the QoS

Table 4.1. Summary of symbols for Chapter 4.

Symbol	Explanation
n	Number of all the current users in the system, including scheduled and queued users
MAX	Number of maximum concurrent users in the system
p^N_i	Usage probability that a requesting user will use access network i
p^S_j	Usage probability that a requesting user will use server j
a^N_i	Available capacity for access network i
a^S_j	Available capacity for server j
r_{ij}	An assignment using access network i and server j
R_μ	The set of all the available assignments for user μ
$RDelay_{ij}$	Achievable network round trip delay if using assignment r_{ij}
$RDelay_A^\mu$	Minimum acceptable round trip delay for user μ
$P(x)$	Probability distribution of number of concurrent users in the system

requirements or resource limitation. When this happens, the user μ will be queued into a waiting list and his/her achieved QoS is 0. Whenever a client leaves the system, MCS will be invoked, checking the waiting list in the order of the user arrival time to see if any queued user can be scheduled. We use n to denote the number of all current users in the system, including both scheduled and queued users. For convenience, Table 4.1 summarizes all the symbols discussed above.

4.2.2 Definition of Mobile Cloud Scheduling Problem

Based on the mobile cloud system model described above, we now present a formal definition of the mobile cloud scheduling problem. As discussed before (in section 4.1.2), when a new user requests service, MCS allocates network and cloud computing resources only for that user, and does not re-schedule in-service users. Hence, different scheduling decisions for a current requesting user may affect the scheduling of future requesting users, resulting in different numbers of blocked users and cloud costs in the long run. Therefore, the mobile cloud scheduling problem is different than the traditional network and processor scheduling problems. Given a certain system state condition, while the traditional network and processor scheduling problems try to maximize the number of users/tasks that can be scheduled, MCS will try to minimize the impact on *BUsers* and *ACost* for future requests.

We first define an objective criteria (*D*) for mobile cloud scheduling, which can be formulated as the sum of *BUsers* and *ACost* with two different weights, W_N and W_S:

$$D{=}W_N \times BUsers + W_S \times ACompCost,\ where\ W_N + W_S = 1 \qquad (4.1).$$

The objective for MCS is to minimize *D*. The weights W_N and W_S in Equation 4.1 are used to represent system-specific interests. A CMG provider can adjust these two weights to achieve different primary and secondary objectives, for example, minimizing *BUsers* as the primary objective with minimizing *ACost* as the secondary objective, and vice verse. We use *C* to denote the system state condition, including the number of current users, and current available capacities and usage probabilities of all the access networks and cloud servers. Given the current system condition *C*, we use $E[D^C]$ to denote the value of *D* that can be expected in the long run. When we schedule a requesting user to network and server resources, the available capacities of the networks and servers will be reduced. This will lead to an increase in $E[D^C]$.

We use $\delta_{ij}{}^{\mu}(E[D^C])$ to denote the change on $E[D^C]$ due to scheduling user μ to the network i and the server j. To minimize $E[D^C]$, we need to minimize the increase of $E[D^C]$ whenever we schedule a new requesting user. Therefore $\delta_{ij}{}^{\mu}(E[D^C])$ is the scheduling criteria for MCS for any new requesting user μ. Then the Mobile Cloud Scheduling problem for minimizing the expected number of *BUsers* and the expected *ACost* by allocating optimal assignment r_{ij} (network i and server j) for user μ is formulated as:

$$\begin{aligned} &Min_{r_{ij}}\{\delta_{ij}{}^{\mu}(E[D^C])\}, \text{ where } r_{ij} \in R_{\mu}, \\ &\text{subject to: (C1) } \mathrm{a}_i^N \geq 1; \text{(C2) } RDelay_{ij} \leq RDelay_A{}^{\mu} \end{aligned} \qquad (4.2).$$

Constraint 1 in Equation 4.2 is the network bandwidth constraint to provision the sufficient network bandwidth for the requesting user. Constraint 2 ensures that each (non-blocked) user will receive an acceptable network round trip delay via the scheduled network i and server j. It should be noted that the computation need of any request can be always meet because of the elastic nature of the cloud computing resources, though any requirements additional to the reserved CUs will incur additional cost *ACost*. Thus, there is no additional constraint for computation need in Equation 4.2, and minimizing *ACost* is incorporated in the objective function in Equation 4.2.

4.3 Mobile Cloud Scheduling Approach

In this section, we introduce our mobile cloud scheduling approach, which can simultaneously find the optimal access network and cloud server resources for the new requesting user, to solve the mobile cloud scheduling problem formulated in section 4.2. We first develop a MCS approach which can optimally solve the MCS problem in Equation 4.2, followed by the run time analysis for this approach. In order to reduce the run time of MCS approach, we have proposed a heuristic approach which has been validated to be able to significantly reduce the run time thereby ensuring the feasibility of applying the MCS approach to practical CMG systems.

4.3.1 Solution for Mobile Cloud Scheduling Problem

The key to solve the MCS problem is to be able to calculate the value of $\delta_{ij}^{\mu}(E[D^C])$. In this section, we first develop a solution which can quantitatively calculate $\delta_{ij}^{\mu}(E[D^C])$ for each possible choice. Next we propose a MCS method which will use the proposed solution to calculate $\delta_{ij}^{\mu}(E[D^C])$ for all the possible choices for a new requesting user and decide on the optimal choice for this user.

As we explained in section 4.2.2, $\delta_{ij}^{\mu}(E[D^C])$ denotes the change of $E[D^C]$ due to scheduling user μ to access network i and cloud server j. After this scheduling, the conditions of all the networks and servers have not changed, except network i and server j. Hence, any change on $E[D^C]$ after scheduling user μ will be only caused by changes in availabilities of network i and server j. For instance, as shown in Figure 4.1, if we schedule mobile user 1 to choice A, then only the capacities on mobile cloud server #1 and macrocell BS #1 will be reduced by 1, while the conditions of all other networks and servers remain the same. To calculate $\delta_{ij}^{\mu}(E[D^C])$, we only need to consider the expectation change of *BUsers* due to the change in availability of macrocell BS #1 and the expectation change of *ACost* due to change in availability of mobile cloud server #1. Therefore, the MCS objective (Equation 4.2) can be simplified as below:

$$\delta_{ij}^{\mu}(E[D^C]) = W_N \times F_N(a_i^N, p_i^N, n) + W_S \times F_S(a_j^S, p_j^S, n) \quad (4.3).$$

$F_N(a^N_i,p^N_i,n)$ is the network scheduling factor for network *i*, which denotes the changes on the expected number of *BUsers* if scheduling the current requesting user μ to the network *i*. Similarly, *$F_S(a^S_j,p^S_j,n)$* is the server scheduling factor for server *j*, which denotes the changes on the expected *ACost* when CMG system if scheduling the current requesting user μ to the server *j*. We use *k* to denote the possible number of increased concurrent users in the system, and use $\delta_i^{\mu}(E[BUsers(k)]\|_{a_i^N}^{p_i^N})$ to denote the expected value of F_N for each possible *k*. *P(x)* is the probability distribution of all the possible number of concurrent users in the system. Then F_N can be calculated by Equation 4.4:

$$F_N(a_i^N, p_i^N, n) = \sum_{k=0}^{MAX-n} (P(k+n) \times \delta_i^{\mu}(E[BUsers(k)]\|_{a_i^N}^{p_i^N})),\ a_i^N > 0 \qquad (4.4).$$

Similarly, we use $\delta_j^{\mu}(E[ACost(k)]\|_{a_j^S}^{p_j^S})$ to denote the expected value of F_S for each *k*. It should be noted that Equation 4.4 assumes a^N_i is nonzero value. This is always true, as the available capacity of any possible access network *i* should be more than 1. However, for cloud server, it is possible that the reserved cloud resource has completely used up (i.e. a^S_j =0). If this is the case, we determine the server expectation change factor always as the maximum value 1. Thus the equation to calculate *$F_S(a^S_j,p^S_j,n)$* is as below:

$$F_S(a_j^S, p_j^S, n) = \begin{cases} \sum_{k=0}^{MAX-n} P(k+n), & if\ a_j^S = 0 \\ \sum_{k=0}^{MAX-n} (P(k+n) \times \delta_j^{\mu}(E[ACost(k)]\|_{a_j^S}^{p_j^S})), & if\ a_j^S > 0 \end{cases} \qquad (4.5).$$

To calculate network and server scheduling factors in Equations 4.4 and 4.5, we next discuss and simplify the calculations for $\delta_i^{\mu}(E[BUsers(k)]\|_{a_i^N}^{p_i^N})$ and $\delta_j^{\mu}(E[ACost(k)]\|_{a_j^S}^{p_j^S})$. The probability of any new user using access network *i* is p^N_i, while it has the probability of 1- p^N_i

to use other networks. This is a well known Bernoulli Trial [Haz] process. Therefore, given the capacity (a^N_i) and probability (p^N_i) of access network i, the expected number of *BUsers* when CMG system has k increased users can be calculated by the following equation:

$$E[BUsers(k)]\Big|_{a_i^N}^{p_i^N} = \begin{cases} \sum_{m=a_i^N+1}^{k} ((m-a_i^N)\binom{k}{m} \times (p_i^N)^m (1-p_i^N)^{k-m}),\ if\ k > a_i^N \\ 0,\ otherwise \end{cases} \quad (4.6).$$

When we schedule user μ to the network i, only the capacity of network i will be reduced by 1 while other conditions do not change. Therefore the network expectation change factor can be calculated as below:

$$\delta_i^{\mu}(E[BUsers(k)]\Big|_{a_i^N}^{p_i^N}) = \begin{cases} E[BUsers(k)]\Big|_{a_i^N-1}^{p_i^N} - E[BUsers(k)]\Big|_{a_i^N}^{p_i^N},\ if\ k \ge a_i^N \\ 0,\ otherwise \end{cases} \quad (4.7).$$

Equation 4.7 can be further derived to Equation 4.8. The latter part of Equation 4.8 is a well known binomial cumulative distribution (binocdf) function [Haz]. Therefore, the final solution to calculate network expectation change factor $\delta_i^{\mu}(E[BUsers(k)]\Big|_{a_i^N}^{p_i^N})$ can be expressed to Equation 4.9.

$$\begin{aligned} & E[BUsers(k)]\Big|_{a_i^N-1}^{p_i^N} - E[BUsers(k)]\Big|_{a_i^N}^{p_i^N} \\ & = \sum_{m=a_i^N}^{k} ((m-a_i^N+1)\binom{k}{m}(p_i^N)^m (1-p_i^N)^{n-m}) - \sum_{m=a_i^N+1}^{k} ((m-a_i^N)\binom{k}{m}(p_i^N)^m (1-p_i^N)^{n-m}) \quad (4.8). \\ & = \sum_{m=a_i^N}^{k} (\binom{k}{m}(p_i^N)^m (1-p_i^N)^{k-m}) = 1 - \sum_{m=0}^{a_i^N-1} (\binom{k}{m}(p_i^N)^m (1-p_i^N)^{k-m}) \end{aligned}$$

$$\delta_i^{\mu}(E[BUsers(k)]\Big|_{a_i^N}^{p_i^N}) = \begin{cases} 1 - binocdf(a_i^N - 1, k, p_i^N),\ if\ k \ge a_i^N \\ 0,\ otherwise \end{cases} \quad (4.9).$$

Similarly, we can express the function to calculate $\delta_j^{\mu}(E[ACost(k)]\Big|_{a_j^S}^{p_j^S})$ as below:

$$\delta_j{}^{\mu}(E[ACost(k)]|_{a_j^S}^{p_j^S}) = \begin{cases} 1 - binocdf(a_j^S - 1, k, p_j^S), \ if \ k \geq a_j^S \\ 0, \ otherwise \end{cases} \quad (4.10).$$

There are many existing models [MAT] which can calculate the binocdf function efficiently. Equations 4.9 and 4.10 together with Equations 4.3, 4.4, and 4.5, give a simple and feasible way to calculate the scheduling criteria $\delta_{ij}{}^{\mu}(E[D^C])$ in MCS problem (Equation 4.2).

Given the equations to quantitatively calculate $\delta_{ij}{}^{\mu}(E[D^C])$, we next propose a Mobile Cloud Scheduling (MCS) method, which will help to choose the optimal choice for a requesting user using the equations provided above.

The MCS method will simultaneously monitor the available resources of each network and cloud server, so the values $a^N{}_i$ and $a^S{}_j$, can be always updated. Note that according to our CMG system model described in section 4.2.1, the initial values of $a^N{}_i$ and $a^S{}_j$ are the capacities of the individual access networks and cloud servers provisioned by the CMG provider. Besides the capacities, the number of maximum number of concurrent users in the CMG system (*MAX*), the access probabilities $p^N{}_i$ and $p^S{}_j$, and the probability distribution of concurrent number of users *P(x)*, are all obtained from the statistic history of CMG system off-line. As shown in Figure 4.2, given R_{μ}, the set of all the available assignments for the requesting user μ, the MCS method first decides the possible solution set *S*. Each element r_{ij} in S indicates the choice of network *i* and server *j*, which satisfies the two constraints defined in MCS problem (Equation 4.2): 1) available network capacity $a^N{}_i$ is greater than 0; 2) $RDelay_{ij}$ is less than acceptable delay threshold for user μ, $RDelayA^{\mu}$. Then MCS method selects the optimal choice r_{xy} (network *x* and server *y*) from *S*, which can maximize the value of the scheduling criteria $\delta_{ij}{}^{\mu}(E[D^C])$. Note that the requesting user μ may not be able to be scheduled, if we cannot find any qualified network and server to satisfy its requirements.

```
Initial set S = $\Phi$;
For each $r_{ij} \in R$   (network $i$ and server $j$)
    If $RDelay_{ij} <= RDelay_A^{\mu}$, and $a^N_i > 1$
    Then S = S ∪ {$r_{ij}$}; Endif
Endfor
If S equal to $\Phi$
  Then user $\mu$ cannot be scheduled; Exit;
Else
  Calculate $\delta_{ij}^{\mu}(E[D^C])$ for each $r_{ij}$ in S using equations 4.3, 4.4, 4.5, 4.9, and 4.10.
  Select x, y, where $r_{xy} \in S$, such that
  $\delta_{ij}^{\mu}(E[D^C])$ is minimized Exit;
Endif
```

Figure 4.2. Mobile Cloud Scheduling (MCS) methodology.

For example, as shown in Figure 4.1, available assignment set R_{μ} for mobile user 1 includes all the choices listed in the table except choice G. Assuming the minimum Round-trip Delay requirement for user 1 is 120ms, then the possible solution set S will only include choices A, D, and H. Assuming the weights W_N and W_S are 0.5 and 0.5, and assuming *MAX* is 300 and there are 50 users in the system, then for each choice in S we can calculate $\delta_{ij}^{\mu}(E[D^C])$ by Equations 4.3, 4.4, 4.5, 4.9, and 4.10. After compared the resulting values of $\delta_{ij}^{\mu}(E[D^C])$ of all three choices, we realize choice A is the best choice for mobile user 1, as it results in the least value of $\delta_{ij}^{\mu}(E[D^C])$ among the three possible choices.

We next analyze the run time of the MCS method. To determine the optimal solution for user μ, MCS has to calculate $\delta_{ij}^{\mu}(E[D^C])$ for each possible solution r_{ij}, which might consume huge amount of time. From Equation 4.3, the run time to calculate $\delta_{ij}^{\mu}(E[D^C])$ mainly depends on the run time to calculate F_N and F_S, which depend on the upper limit in Equations

4.4 and 4.5. The maximum upper limit in Equations 4.4 and 4.5 is the maximum concurrent number of users in the system *MAX* (when n is 0). If MCS has to support a high number of concurrent users, the time to calculate F_N and F_S can be high, leading to a high run time to compute $\delta_{ij}^{\mu}(E[D^C])$.

For example, we have tested the run time of calculating $\delta_{ij}^{\mu}(E[D^C])$ on a platform using Intel i7 CPU. The capacity and probability of access network i given is 200 and 0.2, and the capacity and probability of cloud server j given is 2000 and 0.1. If the max number of concurrent users *MAX* is 1000, the run time to calculate $\delta_{ij}^{\mu}(E[D^C])$ is about 2.2 seconds, but if *MAX* is one million, the run time to calculate $\delta_{ij}^{\mu}(E[D^C])$ will be as high as 21362 seconds. Obviously, the latter will be an unacceptably long time for scheduling a new mobile user.

To obtain scalability in terms of number of concurrent users that can be served by MCS, there can be two approaches. The first is to use multiple MCS servers, each server addressing a separate geographical area and serving a maximum number of concurrent users limited to say 1000. The second alternative, explored in the next section, is to introduce a heuristic approach which can significantly reduce the run time of calculating $\delta_{ij}^{\mu}(E[D^C])$ by relaxing the accuracy when computing F_N and F_S.

4.3.2 Heuristic Approach to Calculate F_N and F_S for MCS Method

The way to reduce the run time of calculating $\delta_{ij}^{\mu}(E[D^C])$ is to relax accuracy of computing the network and server scheduling factors F_N and F_S. As shown in Equations 4.4 and 4.5, to obtain F_N and F_S, MCS has to calculate the network and server expectation change factors for all the possible n. We have conducted experiments to simulate the network expectation change factor $\delta_i^{\mu}(E[BUsers(k)]\big|_{a_i^N}^{p_i^N})$. As shown in Figure 4.3, three access networks with different network capacities $a^N{}_i$ (200, 800, and 2000) and different usage probabilities $p^N{}_i$

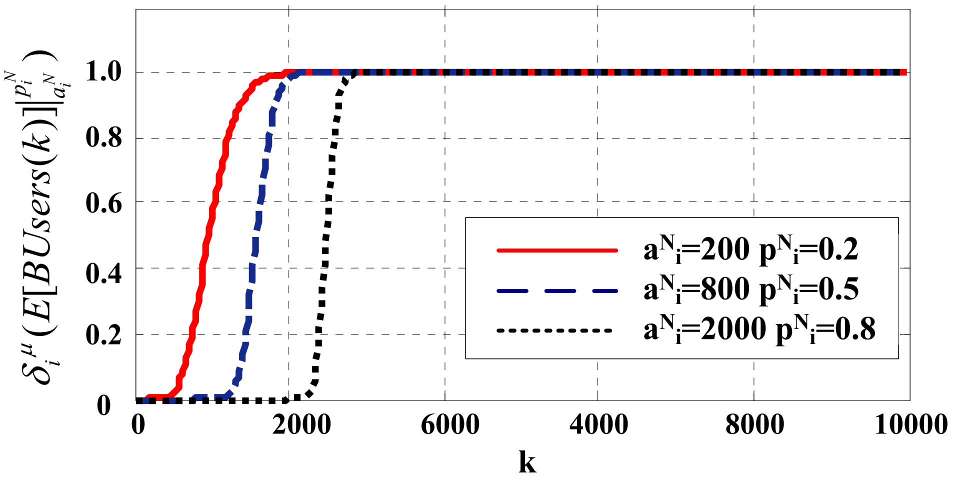


Figure 4.3. Simulation results showing how network expectation change factor varies when adding *k* users in the system.

(0.2, 0.5, and 0.8) are simulated, and the maximum number of users (*MAX*) is set to 10000. From Figure 4.3 we can observe that the value of network expectation change factor is monotonically increasing, and it is either 0 or 1 for most input *k*. We make a similar observation for server expectation change factor. Therefore, when computing the F_N and F_S (by Equations 4.4 and 4.5), the MCS approach does not need to calculate the network and server expectation change factors for all the possible *k*.

The above observation gives us the opportunity to significantly reduce the run time to calculate F_N and F_S. As shown in Figure 4.4, we propose a heuristic approach to calculate F_N (same approach can be applied to calculate F_S also; it just needs to change $a^N{}_i$ and $p^N{}_i$ *to* $a^S{}_j$ and $p^S{}_j$.). Given an acceptable calculation error, *err*, we first find two bounds among all the possible *k*. One is the Greatest Lower Bound (*GLB*), at which the value of expectation change factor is just smaller than *err*, and the other one is the Least Upper Bound (*LUB*), at which the value of expectation change factor is just larger than 1-*err*. Because the expectation change

```
Given $err$, Find $GLB$ and $LUB \in [0, MAX\text{-}n]$,
Such that
    $\delta_i^{\mu}(E[BlockedUser(GLB)]\big|_{a_i^N}^{p_i^N}) < err$, and $\delta_i^{\mu}(E[BlockedUser(GLB+1)]\big|_{a_i^N}^{p_i^N}) >= err$
    $\delta_i^{\mu}(E[BlockedUser(LUB)]\big|_{a_i^N}^{p_i^N}) > 1\text{-}err$, and $\delta_i^{\mu}(E[BlockedUser(LUB\text{-}1)]\big|_{a_i^N}^{p_i^N}) <= 1\text{-}err$
if $k < a^N_i$ or $k < GLB$ then $\delta_i^{\mu}(E[BlockedUser(k)]\big|_{a_i^N}^{p_i^N}) = 0$
else if $k > LUB$ then $\delta_i^{\mu}(E[BlockedUser(k)]\big|_{a_i^N}^{p_i^N}) = 1$
else $\delta_i^{\mu}(E[BlockedUser(k)]\big|_{a_i^N}^{p_i^N}) = 1\text{-binocdf}(a^N_i, k, p^N_i)$ end if
$F_N(a_i^N, p_i^N, n) = \sum_{k=0}^{MAX-n} (P(k+n) \times \delta_i^t(E[BlockedUser(k)]\big|_{a_i^N}^{p_i^N}))$
```

Figure 4.4. Heuristic approach to calculate F_N.

factor is monotonically increasing, the problem to find these two bounds will be as simple as to search a element with a given value in a sorted list, which leads to a complexity of O(log_2MAX). Once we have these two bounds, we can set expectation change factor to 0 if k is less than a^N_i or GLB, and set it to 1 if k is larger than LUB. Therefore, we only need to calculate the value of expectation change factor for k between these two bounds. This optimization will significantly reduce run time to compute F_N. For instance, using the same example as above (network capacity and probability are 200 and 0.2) and giving the *err* as 10e-3, the run time to calculate the network scheduling factor F_N is only about 0.7 second even if *MAX* is as big as one million.

In addition, when the maximum concurrent number of users *MAX* is very big, the small variations of a^N_i and a^S_j will almost not change the values of resulting network and server scheduling factors, unless the values of a^N_i and a^S_j are very small. For instance, if *P(x)* is an even distribution and *MAX* is one million, the changes of network and server scheduling factors will be less than 10e-3 if the variation of a^N_i or a^S_j is less than 1000. Due to this property, the MCS server does not need to calculate the network and server scheduling factors F_N and F_S every time when MCS has a new request. Instead, MCS just needs to periodically update the values of F_N and F_S, to account for big variations in a^N_i and a^S_j. With this periodical updating improvement, the average run time of MCS approach to schedule each new request is less than 1ms, even if the maximum number of users in the system is as big as one million. As shown later in the experiment section 4.5, we have conducted experiments to investigate the loss of accuracy by applying the proposed heuristic approach. The results show that the resulting objective criteria *D* if applying our proposed heuristic approach is only 0.2% higher than the result without applying the heuristic approach. This demonstrates that our proposed heuristic approach can achieve a good accuracy in minimizing objective criteria *D* while significantly reducing the run time for MCS approach.

As a conclusion, with the heuristic approach to calculate F_N and F_S proposed in this section, the MCS approach can be made very run-time efficient and scalable, taking only milliseconds for each scheduling decision even while supporting a very large numbers of concurrent users. In the next section, we will further propose a Joint Scheduling-Adaptation (JSA) approach, which can adapt the QoS requirements of CMG users to increase capacity while trading off with user experienced Quality of Experience (QoE).

4.4 Joint Scheduling-Adaptation Approach

As discussed in section 4.1, CMG applications may require significant data and computation needs. The provisioning for the peak demands can be significantly more than normal or average demands, thus very expensive. On the other hand, under provisioning can lead to insufficient network bandwidth for new CMG users (when the reserved networks get over utilized) or large additional computing cost (when the reserved cloud servers are over utilized). One way to address the above challenge is to make the CMG applications scalable and adaptive to the available capacities of the network and server, such that during busy or peak periods MCS can leverage the scalable CMG applications to reduce their QoS requirements to satisfy more concurrent users and reduce the cloud cost.

Fortunately, for most CMG applications, it is possible to adapt the application such that the QoS requirements in terms of bit rate need and computation need can be reduced. For instance, the bit rate requirement of a video application based on Scalable Video Coding (SVC) [WCG$^+$07] [KSG04] [CSW$^+$11] can be adapted by scaling the spatial and temporal quality of the video content. The above may come at the cost of impairment [WDA10] [WD09] on user perceived Quality of Experience (QoE). For rendering based cloud mobile applications, both bit rate need and computation need can be adapted (as we shown in Chapter 3), each resulting in an impairment [WD12] [WD09] [LWD12] on the user perceived QoE. These adaptation techniques provide a great opportunity to significantly minimize the MCS objective criteria D (Equation 4.1), i.e. minimizing the number of blocked users and minimizing addition cloud cost.

In this section, to ensure the scalability for Cloud Mobile Gaming, we introduce a Joint Scheduling-Adaptation (JSA) approach. Our approach is to increase the number of users that can be assigned in CMG system and reduce the cloud cost by leveraging computation and bit rate adaptation techniques to reduce the resource needs of the CMG users while performing scheduling. For each application adaptation, we give an associated adaptation level from K to 1, each of which reflects a certain communication bit rate need and a certain computation need. Level 1 has the least computation and communication needs, and level K has the highest needs which are same as the needs when adaptation techniques are not applied. Due to application adaptation, the capacity on each resource will be significantly increased. In fact, for each resource there is a Maximum Capacity (*MaxCap*), which denotes the number of users that can be assigned to this resource assuming the adaptation levels of all the assigned users are the lowest level 1.

It should be noted that the objective of MCS problem defined in section 4.2 is to minimize scheduling criteria *D* while ensuring the user QoS requirements without applying any adaptation (i.e. the QoS requirements at adaptation level *K*). However, in JSA it is targeted to minimizing *D* while ensuring the user minimum QoS requirements (i.e. the QoS requirements at adaptation level 1). Therefore, the first objective of JSA approach is:

JSA objective 1: minimize D (Equation 4.1) while ensuring the minimum QoS requirements of each assigned user.

To achieve this objective, JSA can utilize the MCS approach but using the Maximum Capacity (MaxCap) of the resources to calculate the current maximum available capacity of a resource. When a new CMG user comes, JSA will first let MCS find the optimal assignment for this user, considering MaxCap capacity. However, the current available capacities on the assigned resources may not be sufficient because the users of these resources may not be at the lowest adaptation level. If the new CMG user cannot be scheduled due to lack of currently available capacity on the assigned network resource, bit rate adaptation will be used to lower the user QoS requirements (but ensuring the minimum acceptable QoS requirement) of the currently assigned users, albeit at a possible impairment on the user perceived QoE. This will provide additional network capacity, thereby allowing MCS to successfully schedule the new CMG user, such that the value of *BUsers* is minimized. Similarly, when the assigned cloud resource has been completely utilized, computation adaptation can be invoked to lower the applications' QoS requirements on the cloud resource to minimize *ACost* (the additional cloud cost charged for on-demand cloud resource), such that the aggregate cloud cost is minimized.

Though JSA approach is promising to minimize *BUsers* and *ACost*, it also has to minimize the impairments on user QoE whenever an application adaptation occurs. Therefore, the second objective of JSA approach is:

JSA objective 2: minimize the aggregate QoE impairments (due to the application adaptation) of all the assigned users.

Objective 2 is a new problem which has not been studied before. In the reminder of this section, we will first give a formal definition of objective 2. Next, we propose a level-determination algorithm which can find out the optimal adaptation level for each assigned user given the resource capacity, such that objective 2 is addressed. Finally, we will present a Joint Scheduling-Adaptation approach which can leverage the application adaptations while performing the scheduling instructed by MCS approach, such that the objectives 1 and 2 of JSA approach are both satisfied.

4.4.1 Problem Formulation for JSA Objective 2

As explained earlier, when an assigned network or cloud resource is not sufficient to provide the highest quality to the new scheduled user, application adaptation will be invoked. The JSA approach needs to simultaneously decide the optimal adaptation level for each user assigned to the resource, such that objective 2 is satisfied. For any resource (either an access network or a cloud server), let α_l denote the number of users using this resource whose applications are running at adaptation level l. Let I_l denote the impairment on user experience due to using adaptation level l. As discussed before, application adaptation has K adaptation levels. Level 1 has the least resource need, and level K has the highest resource need which is same as the need when adaptation techniques are not applied. Therefore, adaptation level 1 will have the highest impairment (I_1) on user experience, while the impairment at adaptation level K (I_K) is 0. Let Req_l denote the resource requirement at adaptation level l. The value of resource requirement at the lowest level 1 (i.e. Req_1) is equal to 1. The values of other Req_l indicate the resource needed at level l is how many times the resource needed at level 1, Req_1. For instance, if bit rate requirement at level 1 is 200kbps and is 600kbps at level K, then the value of Req_K is 3. The number of current users which are using this resource is *NumCurUser*. Given the above terms, JSA objective 2 can be stated as determining values of α_l such that the aggregate QoE impairments (due to application adaptation) of all the assigned users is minimized, and can be formalized as below:

$$\begin{aligned} &JSA\ objective\ 2: Min\sum_{l=1}^{K}(\alpha_l \times I_l), \text{ where } \alpha_l > 0, I_l > I_{l+1}, \forall l; \\ &s.t.: (\text{C1}) \sum_{l=1}^{K}(\alpha_l \times Req_l) \le MaxCap, where\ Req_{l+1} > Req_l, \forall l \\ &\quad (\text{C2}) \sum_{l=1}^{K}\alpha_l = NumCurUser \end{aligned} \qquad (4.11)$$

Constraint 1 in Equation 4.11 is the resource capacity constraint. The higher adaptation level, the higher QoS requirements needed. The total resource requirement for all the assigned users on a certain resource should be no more than *MaxCap* of this resource. Constraint 2 indicates that the total number of users at different adaptation levels should be equal to *NumCurUser*. We next develop a solution to solve JSA objective 2 defined in Equation 4.11.

4.4.2 Solution for JSA objective 2

Next, we develop the solution to solve the problem defined in Equation 4.11. From constraint C2 of Equation 4.11, we can have:

$$\alpha_K = NumCurUser - \sum_{l=1}^{K-1}\alpha_l \qquad (4.12).$$

Using Equation 4.12, we can derive Equation 4.11 to:

$$\begin{aligned}&JSA\ objective\ 2: Min\sum_{l=1}^{K-1}(\alpha_l \times I_l),\ \text{where } \alpha_l > 0,\ I_l > I_{l+1},\ \forall l\\ &s.t.: \sum_{l=1}^{K-1}(\alpha_l \times (Req_K - Req_l)) \geq NumCurUser \times Req_K\\ &\qquad - MaxCap, where\ Req_{l+1} > Req_l,\ \forall l\end{aligned} \tag{4.13}$$

We next define several symbols to simplify Equation 4.13:

$$y_l = \alpha_l \times I_l \tag{4.14}$$

$$\beta_l = (Req_K - Req_l) / I_l \tag{4.15}$$

$$S = NumCurUser \times Req_K - MaxCap \tag{4.16}$$

Then Equation 4.13 can be simplified to:

$$\begin{aligned}&JSA\ objective\ 2: Min\sum_{l=1}^{K-1} y_l,\ \text{where } y_l > 0,\ \forall l\\ &s.t.: \sum_{l=1}^{K-1}\beta_l \times y_l \geq S, where\ \beta_l > 0,\ \forall l\end{aligned} \tag{4.17}$$

If S is a negative value (*MaxCap* is big enough to support all the assigned users at level K), the solution to Equation 4.17 is: $\{y_K = NumCurUser, y_l = 0$, for all l from level 1 to K-1$\}$. This indicates all the assigned users can be scheduled to the highest adaptation level K. If S is a positive value, the key to solve the above problem is to find out the maximum value of β_l from level 1 to K-1. Without the loss of generality, assuming level m has the biggest value of β among all the levels from 1 to K-1, then the solution for Equation 4.17 is:

$$\begin{cases} y_m = S / \beta_m \\ y_l = 0,\ \forall l \in [1, K-1]\ except\ m \end{cases},\ if\ S > 0 \tag{4.18}$$

Based on Equations 4.12, 4.14, 4.15 and 4.18, we can have the final solution for Equation 4.11:

$$\begin{cases} \alpha_m = S / (Req_K - Req_m) & (4.19.a) \\ \alpha_K = NumCurUser - \alpha_m & (4.19.b),\ if\ S > 0 \\ \alpha_l = 0,\ \forall l \in [1, K-1]\ except\ m & (4.19.c) \end{cases} \tag{4.19}$$

```
Calculate $\beta_l$ for each $l:1 \rightarrow K-1$
Sort $\beta_l$, resulting to a list $L[e]$, where $\beta_{L[e]} >= \beta_{L[e+1]}$;
Init $e$=1; $\alpha_{L[e]}$=0, $\forall e \in [1, K-1]$;
LOOP:
  $\alpha_{L[e]} = Round(S/(Req_K - Req_{L[e]}))$;
  If $\alpha_{L[e]} > NumCurUser$ Then $e$++; Goto LOOP;
  Else
     $NumCurUser = NumCurUser - \alpha_{L[e]}$;
     $MaxCap = MaxCap - \alpha_{L[e]} \times Req_{L[e]}$;
     $S = NumCurUser \times \beta_K - MaxCap$;
     If $NumCurUser > 0$ Then
        $e$++;
        If $e==K$ Then $\alpha_K = NumCurUser$; EXIT;
        Else Goto LOOP;
     Else EXIT;
```

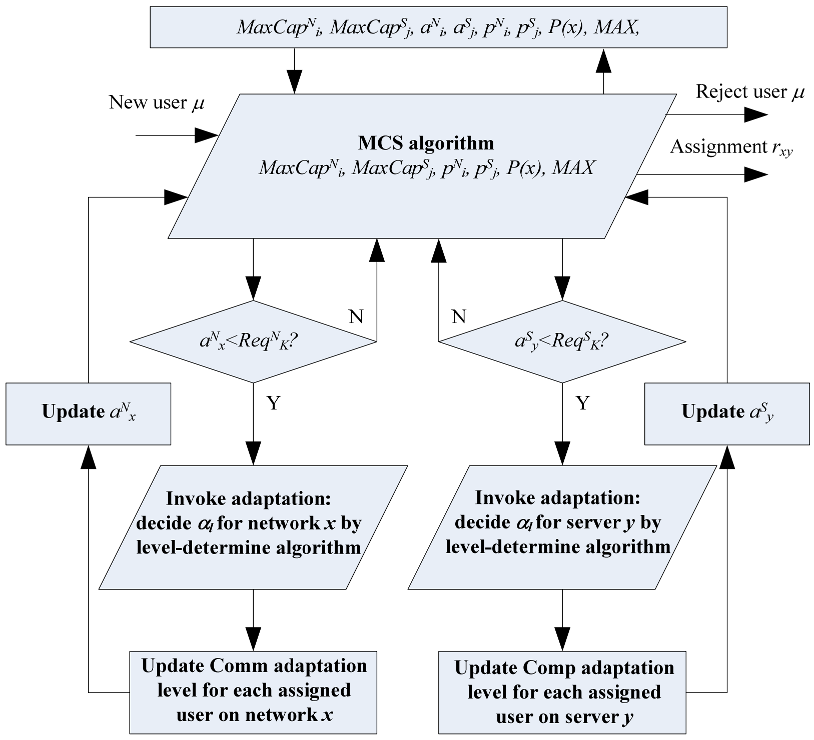
Figure 4.5. The level-determining algorithm to decide all the α_l for the users using an access network or a cloud server resource.

However, the results of α_m calculated by Equation 4.19.a may not be an integer. In this case, we cannot round down the value of α_m from results of Equation 4.19.a and calculate α_K by Equation 4.19.b, because the values of α_m and α_K will not meet the constraint C2. It is also not an optimal solution to round up α_m, because it may not achieve the minimum aggregate impairments.

To address the above situation, we propose a level-determining algorithm to find the optimal values of α_l for the problem defined in Equation 4.11. As shown in Figure 4.5, we first sort β_l from level 1 to *K-1*, resulting to a level list *L[e]*. Each element in list *L* indicates an adaptation level. The list *L* is sorted in descending order in terms of the value of β (i.e. $\beta_{L[e]}$ >= $\beta_{L[e+1]}$). Starting with *L[1]* which has the biggest value of β, we calculate $\alpha_{L[1]}$ by Equation 4.19.a and give the round-down value to $\alpha_{L[1]}$. After this, we update *NumCurUser* (deducted by $\alpha_{L[e]}$) and *MaxCap* (deducted by $\alpha_{L[e]} \times Req_{L[e]}$) leading to a new value of *S*. The level-determining algorithm will repeat the above step for each *L[e]*, from *L[2]* to *L[K-1]*, until *NumCurUser* is 0, indicating the adaptation levels for all the users have been determined.

4.4.3 Joint Scheduling-Adaptation Algorithm

Having derived the solution for JSA objective 2, we next propose a Joint Scheduling-Adaptation (JSA) algorithm which can simultaneously leverage the MCS method proposed in section 4.3 and application adaptation with adaptation levels determined by the level-determining algorithm (Figure 4.5), such that the objectives 1 and 2 of JSA problem can be both achieved. The mechanism of JSA algorithm is shown in Figure 4.6, and is described below. For a requesting user μ, given the set of all the available assignments R_μ, the Joint Scheduling-Adaptation algorithm will first find the optimal assignment r_{xy} by the MCS method proposed in section 4.3. As explained at the beginning of section 4.4, to minimize *D* (objective 1 of JSA problem), JSA approach uses the *MaxCap* (the maximum capacity when all users are assigned the lowest adaptation level) to calculate the maximum available capacity of the resource. The maximum available capacity of a resource is equal to *MaxCap-n*, where *n* is the number of concurrent users of this resource. MCS will use this maximum available capacity instead of available capacity (i.e. a^N_i and a^S_j) to make the scheduling decision. It is possible that MCS cannot find any appropriate assignment for user μ, due to the QoS requirements or resource limitation. When this happens, the request from user μ will be rejected.



$MaxCap^N_i$, $MaxCap^S_j$, a^N_i, a^S_j, p^N_i, p^S_j, $P(x)$, MAX,
New user μ
Reject user μ
Assignment r_{xy}
MCS algorithm
$MaxCap^N_i$, $MaxCap^S_j$, p^N_i, p^S_j, $P(x)$, MAX
$a^N_x < Req^N_K$?
$a^S_y < Req^S_K$?
N
N
Y
Y
Update a^N_x
Update a^S_y
Invoke adaptation: decide α_l for network x by level-determine algorithm
Invoke adaptation: decide α_l for server y by level-determine algorithm
Update Comm adaptation level for each assigned user on network x
Update Comp adaptation level for each assigned user on server y

Figure 4.6. Joint Scheduling-Adaptation algorithm.

Once the assignment r_{xy} is determined, JSA will check if the current available capacity of access network x (i.e. a^N_x) can be sufficient enough to meet the QoS requirement at the highest network adaptation level K (i.e. Req^N_K) for user μ. If a^N_x is able to meet requirement Req^N_K, user μ will be scheduled at the highest adaptation level K. Otherwise, application adaptation will be invoked. To minimize the aggregate QoE impairments of all the assigned users on network x (JSA objective 2), JSA will utilize the level-determining algorithm (Figure 4.5) to decide all the α_l for network x. Similarly, JSA will also check if the current available capacity of cloud server y (i.e. a^S_y) can meet the highest server resource requirement Req^S_K. If not, JSA will execute the level-determining algorithm to decide all the α_l for server y. Once all the α_l for access network x and cloud server y are determined, JSA will update the application adaptation levels for all assigned users on access network x and cloud server y. Finally, JSA will adjust and update MCS the available capacity of network x (a^N_x) and cloud server y (a^S_y).

The run time for JSA algorithm is mainly decided by two parts: MCS and level-determining algorithm. In section 4.3.2, we have proposed a heuristic approach which can reduce the average running time of MCS approach to under 1ms per scheduling decision. The running time of level-determining algorithm consists of time to sort K β_l and time to calculate $\alpha_{L[e]}$ (up to K loops). Hence, the run time complexity of the level-determining algorithm is $O(K\log_2 K)+O(K)$. It should be fine to assume K is less than 100, as it will be unusual for an application to have more than 100 adaptation levels. For example, the scalable CMG application introduced in [WD10] has only 4 network adaptation levels and 4 computing adaptation levels. We have simulated the level-determining algorithm on a platform which uses an Intel i7 CPU. With K equaling 100, the total run time of level-determine algorithm is only 0.8ms. The above analysis shows the run time feasibility of using the proposed JSA algorithm.

4.5 Experimental Evaluation

We have developed a simulation framework to validate the effectiveness of our proposed MCS and JSA approaches for CMG applications. With this simulation framework, we evaluate our proposed approaches in two parts. In the first part, we characterize the performance of our proposed MCS approach using different combinations of scheduling weights W_N and Ws (defined in Equation 4.1). We compare the performance of the MCS approach with two scheduling approaches, a Random Scheduling (RS) approach where the access network and cloud server used are selected randomly, and a Maximizing QoE

Scheduling (MQS) approach where the scheduling decision is targeted to maximize the QoE that can be achieved for each assigned user. In the second part, we apply our proposed JSA approach and compare its performance with the results of MCS approach without applying adaptation techniques.

4.5.1 Simulation Framework and Test Scenario

Our MATLAB-based simulation framework consists of several geographical regions (reflecting each mobile macrocell) with heterogeneous access network coverage with different bandwidth and delay characteristics, and a set of Internet and mobile cloud servers with different reserved cloud capacities beyond which CMG provider has to pay for additional computing cost. For users arriving and leaving the system, the simulation framework can use any kind of user arrival and departure model. Depending on the types of the CMG applications, the framework allows the generation of CMG user requests with different QoS requirements, including acceptable Round-trip Delay $RDelay_A$, application bit rate needed, and computation needed.

During the simulation, the incoming user (under a certain arrival model) is randomly added into one of the geographical regions. Though the user's mobile device can support multiple communication modes in each region, it may be bound by one of the communication modes depending on which area he/she is in and the network availability in that area. Our simulation framework allows the generation of CMG users with different types of communication modes even if these users are in the same geographical region.

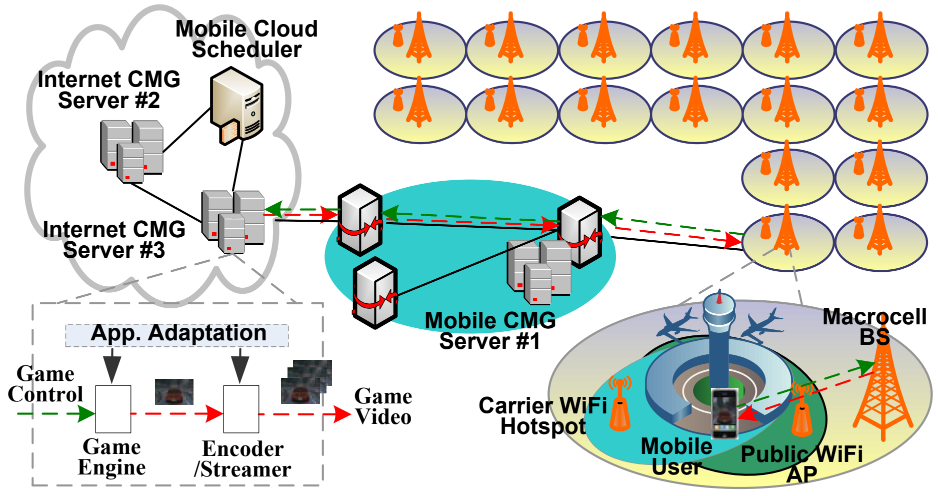


Figure 4.7. Test scenario simulated.

While we have tested our proposed scheduling approaches with multiple scenarios, Figure 4.7 shows one such scenario based on which we present and discuss the experimental results in this chapter. Note that our observations of the experimental results are valid for other test scenarios we have simulated. In the test scenario shown in Figure 4.7, we use three CMG servers. Like in Figure 4.1, we assume one of the three cloud servers is located in a mobile core network (Mobile CMG Server #1) while the other two are Internet CMG Servers #1 and #2. The total reserved Cloud Compute Units (CU) [EC2] for these servers are configured as 2800, 2100, and 1750 respectively. We simulate 16 geographic regions. In each geographic region, we assume there are three types of networks available, a mobile 3G/4G network (via mobile macrocell BS), a mobile carrier WiFi network (via carrier WiFi Hotspot), and a public WiFi network (via public WiFi AP). The maximum data rates for the three networks are configured as 60Mbps, 90Mbps, and 200Mbps respectively. In each geographic region, we assume 60% incoming users have both the mobile cellular connection and a WiFi network

Table 4.2. Different types of user requests used in simulation.

User Requests	WoW in XGA resolution	PlaneShift in XGA resolution	WoW in VGA resolution	PlaneShift in VGA resolution
Computing Need	2 CU	1.6 CU	1.2 CU	1 CU
Communication Need	1.2 Mbps	0.9 Mbps	0.7 Mbps	0.6 Mbps

Table 4.3. Capacities for access network and cloud server, and average network delay/standard deviation for each pair of network and cloud server.

	Mobile Cloud Server #1 (Capacity a^S_1=1920)	Internet Cloud Server #2 (Capacity a^S_2=1440)	Internet Cloud Server #3 (Capacity a^S_3=1200)
Mobile macrocell BS (Capacity a^N_1=71)	83ms/21ms	135ms/47ms	171ms/53ms
Carrier WiFi Hotspot (Capacity a^N_2=106)	74ms/18ms	127ms/43ms	167ms/48ms
Public WiFi AP (Capacity a^N_3=261)	N/A	93ms/23ms	145ms/32ms

connection, while 20% incoming users only have a WiFi network connection, and the rest 20% have only the mobile cellular connection. For each pair of network and cloud server in Figure 4.7, we use the same average and standard deviation of network round-trip delay measured in the real experiments conducted in the commercial wireless networks and cloud servers as described in section 4.1 (Figure 4.1).

To measure the user perceived Quality of Experience (QoE) for CMG, we use our Mobile Gaming User Experience (MGUE) model described in Chapter 2, which can quantitatively measure *GMOS* in a real-time gaming session.

We have used two different games in the simulation: World of Warcraft (WoW) and Planshift. Two resolutions are used for each game: VGA(800x600) and XGA(1024x768).

During the simulation, we generate user requests randomly from four different types of profiles as shown in Table 4.2. The average computation need of all user requests is 1.45CU, while the average communication need is 0.85Mbps. Given the above user request types, we could calculate the capacity of network and cloud servers in terms of the number of users. The complete setup of cloud servers and networks for each geographic region are shown in Table 4.3, including the capacity, and the average network round-trip delay and standard deviation for each pair of possible choice.

While the number of users in the system will go up and down during a day, we simulate peak periods when the user inter-arrival time is smaller than the user inter-departure time. In our simulation, the user arrival and departure model used is birth-death process [LR99]. The user mean inter-arrival time is 1 second, while the user mean inter-departure time is 5 seconds.

4.5.2 Experimental Results Using MCS

We first compare our proposed Mobile Cloud Scheduling (MCS) with two scheduling approaches, namely Random Scheduling (RS) and Maximizing QoE Scheduling (MQS). RS will select the wireless access network and cloud server randomly as long as they can meet QoS requirement. MQS will choose the access network and server for each requesting user such that his/her QoE is maximized. To characterize our proposed MCS approach with different weights W_N and W_S (in Equation 4.1), we have used three different settings. As we mentioned before, the weights W_N and W_S are used to represent system-specific interests. The first simulated MCS approach assumes the weights are W_N=1, W_S=0, termed as MCS-1-0. MCS-1-0 is aimed to help cloud provider to minimize the number of blocked users (that is, maximize capacity in terms of number of concurrently scheduled users). In the second MCS simulation (termed as MCS-0.5-0.5) it assumes W_N and W_S are both 0.5. MCS-0.5-0.5 gives equal weights to the dual objectives of maximizing capacity (minimizing number of blocked users) and minimizing additional cloud cost. The third MCS simulation (termed as MCS-0-1) assumes the weights are W_N=0, W_S=1, which is only targeted to minimize the additional cloud service cost.

The experiments are conducted using the simulation framework and setup introduced above (section 4.5.1). The number of maximum concurrent users in the system, *MAX*, is set to 10000. We use a Gaussian distribution for *P(x)* (as defined in Table 4.1). Its mean is 5000, and standard deviation is 1000. We first use the user arrival/departure model described in section 4.5.1 to train the CMG system to obtain the profiles of the history of the resource usage for deriving the access probability of each network (p^N_i) and each server (p^S_j). Then we simulate the experiments with five different scheduling approaches: MCS-1-0, MCS-0.5-0.5, MCS-0-1, RS, and MQS. During the simulations of using each approach, we measure and calculate the number of blocked users (*BUsers*), the number of users who need additional computing cost (*ACost*), the objective criterion *D* (as defined in Equation 4.1), the average cloud cost per scheduled user, the average *GMOS* per user, and the statistical average cloud cost and *GMOS* by applying the probability distribution of concurrent number of users *P(x)*. Figure 4.8 shows the experimental results.

Figure 4.8(a) presents the relationship between *BUsers* and the number of users in the entire CMG system, *n*. From Figure 4.8(a), we note that MCS may not be able to schedule some incoming users when *n* keeps increasing. This is mainly because these incoming users have entered into the regions where network bandwidth is fully utilized. It can be also observed from Figure 4.8(a) that the performance of MCS-1-0 and MCS-0.5-0.5 are very close and always much better than other three scheduling approaches. This is because MCS-1-0 and

MCS-0.5-0.5 have considered the utilization and accessing probability of networks during each scheduling decision so as to avoid over-utilization.

Figure 4.8(b) shows the results of *ACost* while *n* keeps increasing. As we expected, MCS-0-1 has the best performance in reducing the *ACost* because it fully considers the reserved cloud server resource during the scheduling. Besides MCS-0-1, we noticed that MCS-0.5-0.5 also performs much better than the other three approaches. The results in Figures 4.8(a) and (b) demonstrate that the proposed MCS approach of minimizing *BUsers* and *ACost* for future users is effective.

To demonstrate the efficiency of our proposed MCS approach in minimizing the MCS objective criteria *D* defined in section 4.2, we also calculate the statistical average of *D* during the simulation. For each *n*, we first calculate *D* by Equation 4.1 using the results shown in Figures 4.8(a)(b), and then we calculate the statistical average of *D* by applying the *P(x)*, the weights of all possible values that *n* can take on. Figure 4.8(c) compares the objective criteria *D* between our proposed MCS and the other two scheduling approaches, RS and MQS. Note that for scheduling weights {W_N=1, Ws=0}, WCS-1-0 is used. Similarly, for scheduling weights {W_N=0.5, Ws=0.5} and {W_N=0, Ws=1}, MCS-0.5-0.5 and MCS-0-1 are used respectively. From Figure 4.8(c), we can observe that for different scheduling weights, the proposed MCS approach always performs the best among the tested scheduling approaches. For example, for the first pair of scheduling weights {W_N=1, Ws=0}, the statistical average of *D* by using MCS-1-0 is about 200, which is at least 2 times less than the other two approaches. The same trend and observation can be found in the test results for the other two pairs of scheduling weights. The above experiment and results have demonstrated that our proposed MCS approach can achieve the best performance in terms of minimizing MCS objective criteria *D*, given any scheduling weights W_N and W_S.

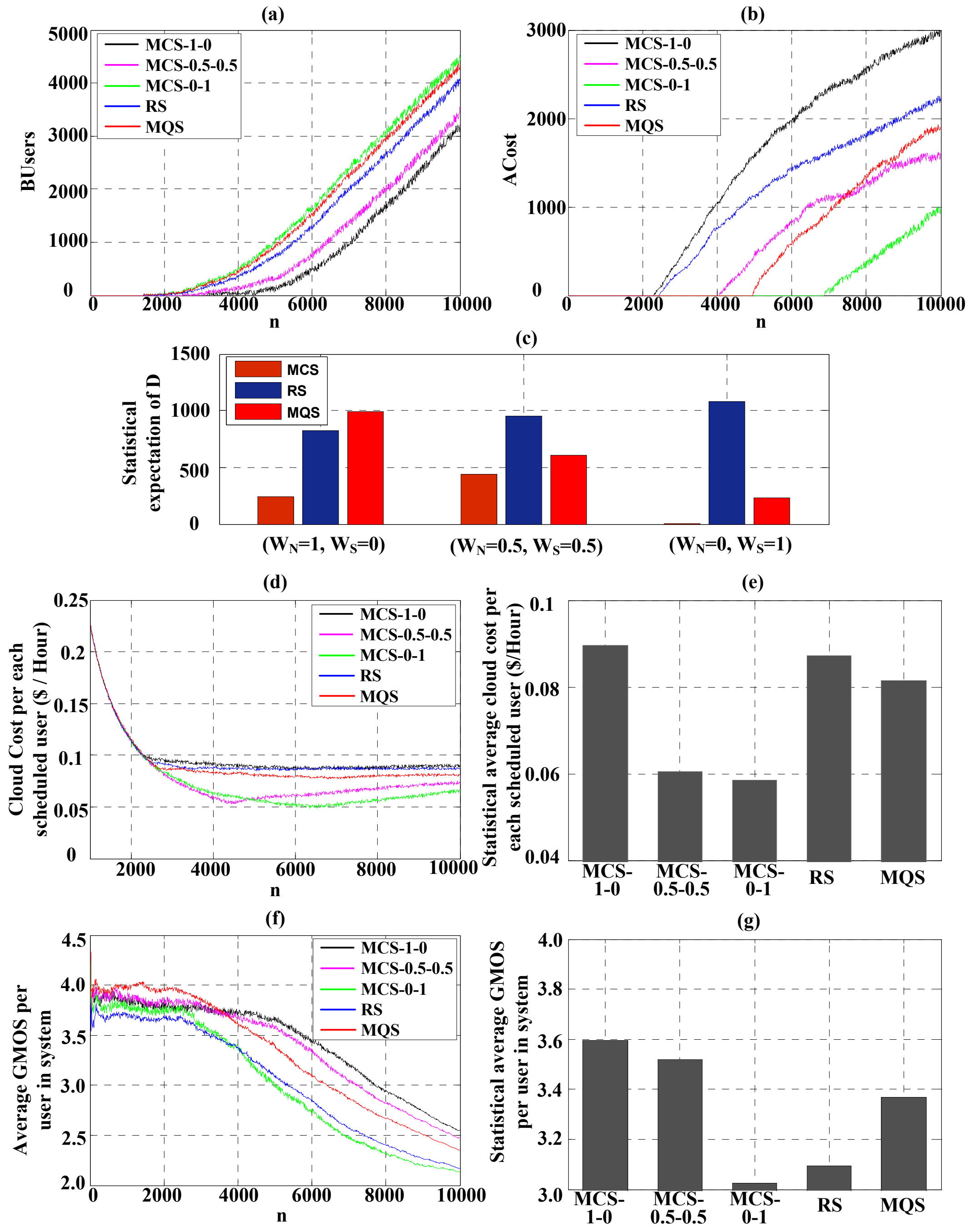


Figure 4.8. Results of simulation experiments: scheduling using the proposed MCS approach, and RS and MQS approaches.

It should be noted that the results obtained in this section has used our proposed heuristic approach (section 4.3.2) with *err* setting of 10e-3. To investigate the loss of accuracy by applying the proposed heuristic approach, we have conducted experiments to compare the objective criteria *D* achieved between the scheduling without applying heuristic approach and the scheduling with applying heuristic approach. For the pair of scheduling weights {W_N=0.5, W_S=0.5}, the resulting statistical average value of *D* is 439.9 if without applying heuristic approach, while it is 440.8 (about 0.2% higher) with applying heuristic approach. This result demonstrates that our proposed heuristic approach can achieve a good accuracy in minimizing objective criteria *D* while significantly reducing the run time for the MCS approach.

We have measured the average cloud cost per scheduled user, including the costs paid for reserved and on-demand computing resource and the costs for on-demand usage of cloud network bandwidth and storage as shown in Figure 4.1. Figure 4.8(d) shows how the average cloud cost changes given different *n*. From Figure 4.8(d) we notice that for each scheduling approach the average cloud cost will keep decreasing while number of concurrent users *n* is increasing until the CMG provider starts to pay the *ACost* to the cloud provider. MCS-0.5-0.5 performs best among all five approaches when *n* is less than 5000, because it can achieve good performance in both minimizing *BUsers* and *ACost*, MCS-0-1 has a little higher average cloud cost when *n* is small, but it will eventually drop lower than MCS-0.5-0.5 when *n* is less than 5000 where the *ACost* spent becomes significant for MCS-0.5-0.5. To better evaluate and compare the performances of different scheduling approaches in reducing average cloud cost, we have calculated the statistical average cloud cost by applying *P(x),* the weights of all possible values that *n* can take on. The results are shown in Figure 4.8(e). From Figure 4.8(e), the benefits of using MCS-0.5-0.5 and MCS-0-1 are obvious. The statistical average cloud cost per each user of using these two approaches are only about \$0.06 per hour, which is at least \$0.02 less than the results of using other approaches.

We have also measured the *GMOS* of each user in the system (including blocked users), and calculated the average *GMOS* and statistical average *GMOS*. When there is no blocked user, MQS will result in the best average *GMOS* among all five approaches, as shown in Figure 4.8(f). However, when *n* increases, some requesting users may be blocked due to the network resource constraint (as shown in Figure 4.8(a)). Because the *GMOS* of a blocked user is 0, the average *GMOS* will decrease. The approach which has less blocking rate, like MCS-0.5-0.5 and MCS-1-0, will eventually have better average *GMOS*. For any given number of user requests *n*, Figure 4.8(f) has given the corresponding value of average *GMOS*. We next calculate the statistical average GMOS by applying *P(x)*, the weights of all possible values that *n* can take on. The results of statistical average GMOS for each scheduling approach is shown in Figure 4.8(g). Based on Figure 4.8(g), we can observe that approaches MCS-0.5-0.5 and MCS-1-0 have similar performance and both of them can achieve much higher average GMOS than the other three approaches.

Based on the above results (Figures 4.8(e)(g)) and discussions, $\{W_N=0.5,\ W_S=0.5\}$ may be an effective setting, because MCS-0.5-0.5 can achieve a low average cloud cost as well as a high average GMOS, which also indicates its capability of achieving a low number of blocked users and hence high capacity. Note that while the experimental results reported in this dissertation compare three pairs of W_N and W_S settings, the proposed MCS method allows CMG providers to explore different values of W_N and W_S, and select the appropriate setting depending on the tradeoff between capacity, cloud cost, and aggregate user experience they want to achieve.

Table 4.4. QoS Requirements (Req_l) / QoE Impairment (I_l) for each level in bit rate and computation adaptation.

Adaptation levels	1	2	3	4
Bit rate adaptation	1 / 22	1.5 / 7	2 / 4	2.5 / 0
Computation adaptation	1 / 18	2 / 12	3 / 8	4 / 0

4.5.3 Experiment Results Using the Proposed Joint Scheduling-Adaptation

To demonstrate the efficiency of the proposed Joint Scheduling-Adaptation (JSA) approach, we use similar simulation framework and setup as described in sections 4.5.1 and 4.5.2. The only difference is that we have used a larger number (30000) of maximum concurrent users in the system, *MAX*, considering JSA is expected to increase the system capacity. Correspondingly, we set the mean and the standard deviation of *P(x)* to 15000 and 3000.

The application adaptation used is a joint rendering and encoding adaptation technique, proposed in [WD13], which can adapt the bit rate and computation needs of each CMG session to the dynamic conditions of the wireless network and cloud server simultaneously. In [WD13], we have defined 4 bit rate adaptation levels and 4 computation adaptation levels. The QoS requirements in terms of bit rate and computation needs for each level are defined in [WD13]. The QoE impairments for each level have been studied in [WD12][LWD12]. Note that the range of impairment is from 0 (no impairment) to 100. It can be converted to GMOS by a non-linear function [WD12]. Table 4.4 presents the values of QoS requirements (Req_l) QoE impairments (I_l) for each bit rate and computation adaptation level. From Table 4.4, we can observe the bit rate need without any adaptation (level 4) is 2.5 times of the bit rate need at level *1*. This indicates that with the bit rate adaptation technique the maximum capacity of each access network will become 2.5 times of the capacities (Table 4.4) without application adaptation. Similarly, from Table 4.4 we can observe that the computing requirement without any adaptation (level 4) is 4 times of the requirement at level 1. Thus, we can multiply the reserved cloud server capacities (Table 4.4) by 4 to get the maximum capacity of reserved cloud server.

We simulate JSA approach with three pairs of scheduling weights as used for MCS in section 4.5.2: JSA-1-0 where $\{W_N=1, W_S=0\}$, JSA-0.5-0.5 where $\{W_N=0.5, W_S=0.5\}$, and JSA-0-1 $\{W_N=0, W_S=1\}$. Similar to the experiment conducted in section 4.5.2, during the simulation, we measure and calculate the number of *BUsers*, the number of users who need the *ACost*, the statistical average of objective criteria *D*, the statistical average of cloud cost per each scheduled user, the statistical average of achieved GMOS per each user in the entire CMG system. We also compare the performance of JSA with the performance of MCS with settings $\{W_N=0.5, W_S=0.5\}$, the latter performing the best amongst other settings and approaches tested in section 4.5.2. The results of the simulation experiments are shown in Figure 4.9.

From Figure 4.9(a) we can observe that given the same number of concurrent users *n*, the number of *BUsers* by applying JSA is much lower than the number of *BUsers* using MCS-0.5-0.5. It should be noted that the number of scheduled users can be calculated by deducting the number of *BUsers* from *n*. For instance, from the results presented in Figure 4.9(a), we see that when the number of concurrent user requests is 24000, the number of scheduled users is only about 7000 if using MCS with settings 0.5-0.5 while it is about 20000 (about 3x improved) if using JSA with any of three scheduling weights settings.

Figure 4.9(b) shows the *ACost* while *n* keeps increasing. With the application adaptation techniques, the computation need of cloud mobile gaming will be greatly reduced.

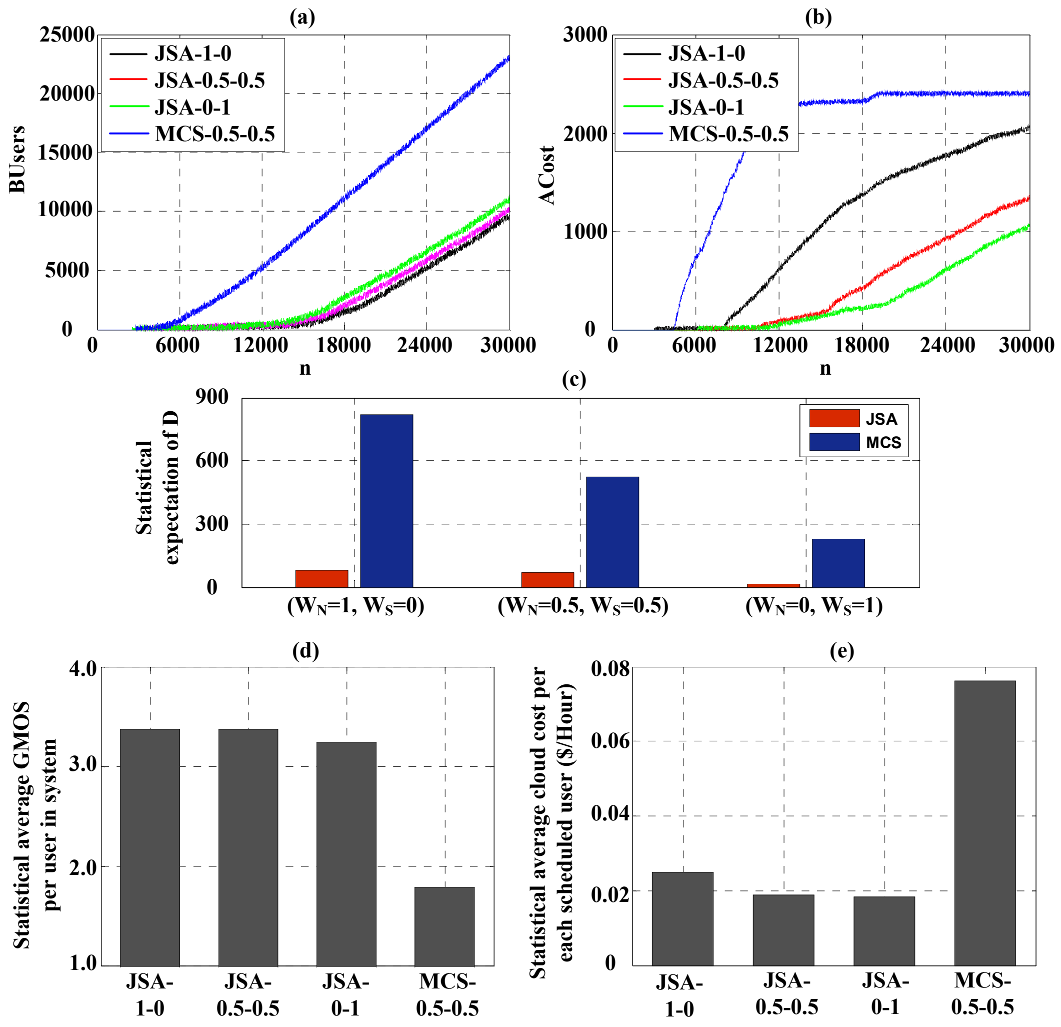


Figure 4.9. Results of simulation experiments applying the proposed joint scheduling-adaptation approach.

This will let each cloud server be able to support much higher number of users within the reserved computing cloud resource. Due to this reason, as shown in Figure 4.9(b), the *ACost* starts to increase at about 4000 user requests if using MCS-0.5-0.5, while with use of the JSA, the *ACost* starts to increase at about 8000 user requests.

Base on the results in Figures 4.9(a)(b), we calculate the statistical average objective criteria *D* using three different pairs of scheduling weights. The results are presented in Figure 4.9(c). From the Figure 4.9(c), we can observe that JSA in all three scheduling weights settings can achieve much less value of *D* than MCS-0.5-0.5.

We have also calculated the statistical average cloud cost and statistical average GMOS. The results are present in the Figure 4.9(d) and Figure 4.9(e). Based on the results shown in these two figures, we can conclude that compared to MCS, the JSA with any of the three settings can significantly reduce the average cloud cost (by about 3.5x), and can significantly increase the average GMOS achieved (by more than 2x). The above results demonstrate that the proposed joint scheduling-adaptation approach is efficient in achieving the JSA objectives defined in section 4.4: minimizing the number of *BUsers* and the *ACost* while maintaining a high average GMOS, thus providing the desired scalability to the CMG approach.

Besides the above improvements, the other test results and conclusions are very similar as what we have described in section 4.5.2, that JSA-1-0 has the best performance in minimizing the *BUsers*; JSA-0-1 has the best performance in minimizing *ACost*; and JSA-0.5-0.5 provides the best setting among the three JSA settings simulated as it can achieve a good statistical expectation of average cloud cost and average *GMOS*.

4.6 Conclusions

In this chapter, we proposed a mobile cloud scheduler for cloud mobile gaming, which can simultaneously schedule the wireless network and cloud server resources to requesting mobile cloud users in a dynamically changing and heterogeneous CMG environment. We first developed a Mobile Cloud Scheduling (MCS) approach, with can allocate resources to meet user Quality of Service (QoS) requirements while maximizing number of users that can be scheduled concurrently and minimizing cloud cost. We have conducted a set of simulation experiments using the CMG application to compare the performance between CMG sessions without applying MCS and CMG session with applying MCS. Our simulation results demonstrate that our proposed MCS approach can help the CMG provider to achieve higher capacity and lower cloud cost than the original CMG approach without MCS. In order to further enhance scalability, we introduce a Joint Scheduling-Adaptation (JAS) approach where the communication and computation requirements of CMG application can be adaptively adjusted according to the dynamic conditions of the wireless access network and cloud server at anytime. Simulation results show that the proposed JAS approach can significantly increase the number of concurrent CMG users and reduce cloud cost while meeting user minimum acceptable QoS requirements.

The text of this chapter, in part or in full, is based on material that has been published in IEEE International Conference on Communications (ICC) (S. Wang, Y. Liu, S. Dey, “Wireless Network Aware Cloud Scheduler for Scalable Cloud Mobile Gaming”, IEEE ICC, Jun. 2012) and material submitted to IEEE Transactions on Networking (S. Wang, Y. Liu, S. Dey, “Mobile Cloud Scheduling: Scheduling Heterogeneous Network and Cloud Resources to Enable Scalable Mobile Cloud Computing”). The dissertation author was the primary researcher and author in the publications, and the coauthors listed supervised the research that forms the basis of this chapter.

Chapter 5

Conclusions and Future Directions

This chapter concludes the dissertation with a summary of our principle contributions and some thoughts about the possible future research work.

In Chapter 2, we developed a Mobile Gaming User Experience (MGUE) model, which can quantitatively measure user perceived mobile gaming experience. Consequently, we develop a MGUE prototype measurement tool to enable in-network monitoring of MGUE. The MGUE model developed in this dissertation can be potentially used by researchers to assess the performance of new CMG techniques, and by future mobile gaming service providers, including network operators, to better plan and optimize their CMG services, as well as monitor in-network the user experience of their mobile gaming subscribers. We also believe that the approach used to develop the MGUE model outlined in this work can be useful for developing QoE models for other new cloud server based interactive multimedia applications, such as virtual reality and augmented reality.

To address the communication constraint imposed by the fluctuating bandwidth and mobile network and computation constraint due to the cost and availability of cloud servers, in Chapter 3, we developed a dynamic game rendering adaptation technique which can simultaneously vary the richness and complexity of graphic rendering to adapt the communication and computing needs of each CMG session in responding to the dynamic conditions of the wireless mobile networks and CMG server. Adapting the content source has not been studied, and may not be a possibility for other types of video applications, like live video streaming or video conferencing. We furthermore, proposed a joint adaptation between content source (here rendering) and video encoding to address mobile network and server computing constraints, which again has not been previously attempted to the best of our knowledge.

In Chapter 4, we proposed a mobile cloud scheduler for Cloud Mobile Gaming, which can simultaneously schedule the wireless network and cloud server resources to requesting mobile cloud users in a dynamically changing and heterogeneous CMG environment. We first developed a Mobile Cloud Scheduling (MCS) approach, which can allocate resources to meet user Quality of Service (QoS) requirements while maximizing number of users which can be scheduled concurrently and minimizing cloud cost. We demonstrate that the proposed MCS approach can assist the CMG provider to achieve a higher capacity and a lower cloud cost than the original CMG approach without MCS. In order to further enhance scalability, we introduce a Joint Scheduling-Adaptation (JAS) approach where the communication and computation requirements of CMG applications can be adaptively adjusted according to the dynamic conditions of the wireless access network and cloud server at anytime.

In the future, we envision two ideas to be the initial steps in the direction of the new techniques specialized for the CMG system. The first is to enhance our proposed mobile cloud scheduling approach because the current proposed MCS approach works best for stationary and nomadic users, and does not address mobility explicitly. Thus, if the access network coverage for a user has changed, MCS will consider this user a new requesting client and may reschedule the access network resource and possibly even the cloud server for this user. This may lead to some interruption and delay in his/her CMG session, especially when the cloud server which performs all computing tasks must be rescheduled. In the future, mobile cloud scheduling techniques will need to be developed with consideration to user mobility and access network handover issues which may occur even for a stationary user in a heterogeneous wireless access network environment.

Secondly, while in our proposed Cloud Mobile Gaming architecture the entire game engine is executed on cloud servers, it may be valuable to investigate whether it is possible to partition the game engine into different tasks. If this is possible, both mobile devices and cloud servers can be involved/designed to process part of gaming engine tasks. Depending on the types of game engine tasks and the costs of network communication and computing on different platforms, the game engine tasks will be dynamically scheduled to the appropriate hardware platforms. For instance, the mobile devices will mainly process the tasks which need less computing resource but demanding quick response time, while the cloud servers will take over the tasks that are not critical for the gaming response time. We envision that appropriately partitioning the game engine will eventually be adopted by the future cloud mobile gaming system, as it is very promising for reducing the gaming response time as well as the cloud cost.

Bibliography

[3DG] 3D game system requirements, http://www.game-debate.com.

[AAB$^+$10] P. Armstrong, A. Agarwal, A. Bishop, et al, “Cloud Scheduler: a resource manager for distributed compute clouds,” arXiv:1007.0050v1 [cs.DC], 2010.

[ADH12] H. Ahlehagh, S. Dey, “Hierarchical Video Caching in Wireless Cloud: Approaches and Algorithms”, *Workshop on Realizing Advanced Video Optimized Wireless Networks (ICC'12 WS – ViOpt),* in *Proc. of IEEE ICC,* Ottawa, Jun. 2012.

[ADV12] H. Ahlehagh and S.Dey “Video Caching in Radio Access Network*”,* in *Proc. of IEEE WCNC,* Paris, Apr. 2012.

[AM13] Analysis Mason, http://www.analysysmason.com/About-Us/News/Insight /Enterprise-cloud-services-revenue-Feb2013/#.UYKEzsqOrdw, Jan. 2013.

[ANA] Anandtech iphone performance reviews, http://www.anandtech.com.

[ARC11] ARCchart, “The Mobile Cloud: Market Analysis and Forecasts”, Jun. 2011.

[ARC12] ARCchart, “HetNet Market Summary & Forecasts: Macro Cells, Small Cells & Wi-Fi Offload,” Retrieved 17, Nov. 2012.

[AT10] Asia Times, “Game still on at Tencent”, http://www.atimes.com/atimes/China_Business/LC24Cb01.html, Mar. 2010.

[Bel$^+$11] A.Beloglazov, et al, “Energy Efficient Resource Management in Virtualized Cloud Data Centers”, *in Proceedings of 2010 IEEE/ACM Conference on Cluster, Cloud and Grid Computing,* Jul. 2010.

[BF10] M. Bredel, M. Fidler, “A Measurement Study regarding Quality of Service and its Impact on Multiplayer Online Games”, In ACM SIGCOMM workshop on Network and System Support for Games, (NetGames), 2010.

[BLB06] N. Bhusha, C. Lott, P. Black, et al., “CDMA200 1xEV-DO Revision A: A physical layer and MAC layer overview,” *IEEE Comm. Magazine*, 44(2), Feb. 2006.

[BSW$^+$07] P. Baccichet, T. Schierl, T. Wiegand, and B. Girod, “Low-delay Peer-to-Peer Streaming Using Scalable Video Coding,” in *Packet Video 2007*, Nov. 2007, pp. 173–181.

[BVB10] R. Bossche, K. Vanmechelen, J. Broeckhove, “Cost-optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workload”, *in Proc. of IEEE International Conference on Cloud Computing,* Miami, Jul. 2010.

[CAN12] Canalys, "Smart phones overtake client PCs in 2011", http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011 ", Feb. 2012.

[CC06] M. Claypool and K. Claypool, "Latency and Player Actions in Online Games," *Communications of the ACM,* 49(11), 2006.

[CGF+10] T. Cucinotta, D. Giani, D. Faggioli, F. Checconi. "Providing performance guarantees to virtual machines using realtime scheduling", in *in Proceedings of the 5th Workshop on Virtualization and High-Performance Cloud Computing,* Italy, Aug. 2010.

[CISCO12] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2011–2016", May 30, 2012.

[CKX+08] P. Chaporkar, K. Kar, L. Xiang, S. Sarkar, "Throughput and fairness guarantees through maximal scheduling in wireless networks," in *IEEE Transactions on Information Theory*, 2008.

[COM12] ITU-T Contribution COM12-D13-E, "Proposal on Basic Concepts of a Multimedia Quality Assessment Model," Jan. 2005.

[CSH+08] H. L. Cycon, T. C. Schmidt, G. Hege, et al, "Peer-to-Peer Videoconferencing with H.264 Software Codec for Mobiles," in *WoWMoM08 – WS on Mobile Video Delivery (MoViD)*, IEEE Press, Jun. 2008, pp. 1–6.

[CSW+11] H. L. Cycon, T. C. Schmidt, M. Wahlisch, et al, "A Temporally Scalable Video Codec and its Applications to a Video Conferencing System with Dynamic Network Adaption for Mobiles", *IEEE Trans. on Consumer Electronics*, vol. 57, no. 3, pp. 1408-1415, Aug. 2011.

[CV05] S.-F. Chang and A. Vetro, "Video Adaptation: Concepts, Technologies, and Open Issues," *Proc. IEEE*, vol. 93, no. 1, pp. 148–158, Jan. 2005.

[Dey12] S. Dey, "Cloud Mobile Media: Opportunities, challenges, and directions", *in Proceedings of IEEE International Conference on Computing, Networking and Communications (ICNC)*, Jan. 2012.

[DJ02] C. Dovrolis, and M. Jain, ``End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput''. *In Proceedings of ACM SIGCOMM 2002*, Aug. 2002.

[DLN+11] H.T. Dinh, C. Lee, D. Niyato, P. Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications and Approaches," *Wireless Communications and Mobile Computing, Wiley Journals*, Oct. 2011.

[DWW05] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games", In ACM SIGCOMM workshop on Network and System Support for Games, (NetGames), 2005.

[EC2] Amazon EC2 Pricing, http://aws.amazon.com/ec2/pricing/.

[EMC+07] J. Elias, F. Martignon, A. Capone, G. Pujolle, "A new approach to dynamic bandwidth allocation in Quality of Service networks: Performance and bounds", in *International Journal of Computer and Telecommunications Networking*, 2007.

[FCF+05] W.-C. Feng, F. Chang, W.-C.Feng, and J. Walpole, "A Traffic Characterization of Popular On-line Games", IEEE/ACM Trasactions on Networking, vol 13, no.3, pp.588-500, Jun. 2005.

[FLR12] N. Fernando, S.W. Loke, W. Rahayu. "Mobile cloud computing: A survey," *Future Generation Computer Systems*, Jun. 2012.

[FRS05] T. Fritsch, H. Ritter and J. Schiller, "The Effect of Latency and Network Limitations on MMORPGs - (A Field Study of Everquest2)", In ACM SIGCOMM workshop on Network and System Support for Games, (NetGames), 2005.

[G107] ITU-T Recommendation G.107, "The E-model, a Computational Model for Use in Transmission Planning," Mar. 2005.

[G1070] ITU-T Recommendation G.1070, "Opinion Model for Videotelephony," Apr. 2007.

[Gho+11] A. Ghodsi, et al, "Dominant Resource Fairness: Fair Allocation of Multiple Resource Types", *Berkerley Technical Report,* Mar. 2011.

[Han04] D. S. Hands, "A Basic Multimedia Quality Model," IEEE Transactions on Multimedia, vol. 6, no. 6, pp. 806-816, Dec. 2004.

[Haz] M. Hazewinkel, "Bernoulli trials", *Encyclopedia of Mathematics, Springer*, ISBN 978-1-55608-010-4.

[HK09] I-H. Hou and P.R. Kumar. "Admission control and scheduling for QoS guarantees for variable-bit-rate applications on wireless channels," *In Proc. of ACM MobiHoc*, pages 175-184, May. 2009.

[IDC11] IDC, "More Mobile Internet Users Than Wireline Users in the U.S. by 2015", http://www.idc.com/getdoc.jsp?containerId=prUS23028711 , Sep. 2011.

[J144] ITU-T J.144, "Objective Perceptual Video Quality Measurement Techniques for Digital Cable Television in the Presence of a Full Reference," Mar. 2004.

[JFE+09] A. Jurgelionis, P. Fechteler, P. Eisert, et al., "Platform for Distributed 3D Gaming", International Journal of Computer Games Technology, Volume 2009, Article ID 231863.

[JUN11] Juniper Research, "Mobile Cloud: Smart Device Strategies for Enterprise & Consumer Markets 2011-2016", http://juniperresearch.com/ , Jul. 2011.

[LCC11] Y-T. Lee, K. Chen, Y. Cheng, C. Lei., "World of Warcraft Avatar History Dataset," In *Proc. Of ACM Multimedia Systems*, Feb. 2011.

[LCK11] G. Lee, B. Chun, R. Katz, "Heterogeneity-aware Resource Allocation and Scheduling in the Cloud", *in Proc. of 3rd USENIX Workshop on Hot Topics in Cloud Computing*, Jun. 2011.

[JVC+03] T. Jehaes, D. De Vleeschauwer, T. Coppens, et al., "Access Network Delay In Networked Games," *in Proc. of the Second Workshop on Network and System Support for Games,* 2003.

[KSG04] M. Kalman, E. Steinbach, and B. Girod, "Adaptive Media Playout for Low–delay Video Streaming over Error–prone Channels," *IEEE Trans. On Circuits*

Syst. Video Technol., 2004.

[LR99] G. Latouche, V. Ramaswami. "Chapter 1: Quasi-Birth-and-Death Processes," *Introduction to Matrix Analytic Methods in Stochastic Modelling, 1st edition*, ASA SIAM, 1999.

[LSM+08] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang, "Experiences in a 3G Network: Interplay between the Wireless Channel and Applications," in *Proc. ACM MobiCom 2008*, Sep. 2008.

[LWD12] Y. Liu, S. Wang, S. Dey, "Modeling, characterizing, and enhancing user experience in Cloud Mobile Rendering", *in Proc. of IEEE ICNC, Maui,* Jan. 2012.

[MAM10] MarketsAndMarkets, "World Mobile Applications Market - Advanced Technologies, Global Forecast (2010-2015)", http://www.marketsandmarkets.com/, Aug. 2010.

[MAT] MATLAB, http://www.mathworks.com

[MPL+C07] F. Morán, M. Preda, G. Lafruit, et al., "Adaptive 3D Content for Multi-Platform On-Line Games," *IEEE International Conference on Cyberworlds*, 2007.

[MPL+J07] F. Morán, M. Preda, G. Lafruit, et al., "3D Game Content Distributed Adaptation in Heterogeneous Environments", in *EURASIP Journal on Advances in Signal Processing,* 2007.

[MSD07] S. Mukhopadhyay, C. Schurgers, S. Dey, "Joint Computation and Communication Scheduling to Enable Rich Mobile Applications," in *Proc. of IEEE GLOBECOM,* Washington D.C., Nov. 2007.

[NDS+08] I. Nave, H. David, A. Shani, A. Laikari, P. Eisert, and P.Fechteler, "Games@Large Graphics Streaming Architecture," Proceedings of the 12th Annual IEEE International Symposium on Consumer Electronics, pp. 1–4, Algarve, Portugal, Apr. 2008.

[NKG10] R. Nathuji, A. Kansal, A. Ghaffarkhah, "Q-clouds: managing performance interference effects for QoS-aware clouds", *in Proc. 5th European conference on computer systems,* pp. 237–250. ACM, New York, 2010.

[NRL+04] N. Pham Ngoc, W. Van Raemdonck, G. Lafruit, et al., "A QoS Framework for Interactive 3D Applications", *in Proc. of the 9th international conference on 3D Web technology*, 2004.

[OneAPI] https://gsma.securespsite.com/access/Access%20API%20Wiki/Home.aspx

[ONL] "Onlive", http://www.onlive.com.

[P800] ITU-T Recommendation ITU-T P.800, "Methods for Subjective Determination of Transmission Quality," 1996.

[PMD+03] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," *IEEE Network*, vol. 17, pp. 27–35, Nov. 2003.

[PQ11] Y. Peng, F. Qin, "Exploring Het-Net in LTE-Advanced System: Interference

Mitigation and Performance Improvement in Macro-Pico Scenario", *in Proc. of IEEE ICC,* Kyoto, Jun. 2011.

[QML+04] Peter Quax, Patrick Monsieurs, Wim Lamotte, et al., "Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game," *in Proc. of the Third Workshop on Network and System Support for Games*, 2004.

[RLS+02] W. Van Raemdonck, G. Lafruit, E.F.M. Steffens, et al., "Scalable 3D Graphics Processing in Consumer Terminals", *IEEE International Conference on Multimedia and Expo*, 2002.

[RNR08] M. Ries, O. Nemethova, M. Rupp, "Video Quality Estimation for Mobile H.264/AVC Video Streaming," Journal of Communications, vol. 3, pp. 41 - 50, 2008.

[PS] Planeshift, http://www.planeshift.it/.

[SKG+11] A. Shieh, , S. Kandula, A. Greenberg, et al., "Sharing the data center network", in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, 2011.

[SM99] B.-O. Schneider, and I. Martin, "An Adaptive Framework for 3D Graphics over Networks", *in Computers and Graphics*, 23, 867-874, 1999.

[SMW07] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. On Circuits System and Video Technology.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[TLL07] W. Tan, F. Lam, and W. Lau, "An Empirical Study on 3G Network Capacity and Performance," *in Proc. of IEEE INFOCOM*, pages 1514–1522, 2007.

[TML+04] N. Tack, F. Morán, G. Lafruit, R. Lauwereins, "3D Graphics Rendering Time Modeling and Control for Mobile Terminals," in *Proc. of Int. Conf. on 3D Web technology* , 2004.

[TW07] S. Tasaka and Y. Watanabe, "Real-Time Estimation of User-Level QoS in Audio-Video IP Transmission by Using Temporal and Spatial Quality," IEEE GLOBECOM, Nov. 2007.

[TYK04] A. Takahashi, H. Yoshino, and N. Kitawaki, "Perceptual QoS Assessment Technologies for VoIP," IEEE Communications Magazine, pp. 28- 34, Jul. 2004.

[VDG+05] N. Vaidya, A. Dugar, S. Gupta, P. Bahl, "Distributed Fair Scheduling in a wireless LAN," in *IEEE Transaction on Mobile Computing*, vol. 4, 2005.

[WBD] WoW Basic Demographics, http://www.nickyee.com/daedalus/archives-/001365.php .

[WCG+07] M. Wien, R. Cazoulat, A. Graffunder, A. Hutter, and P. Amon, "Realtime System for Adaptive Video Streaming Based on SVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, Sep. 2007.

[WD09] S. Wang, S. Dey, "Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach," in IEEE GLOBECOM, Nov. 2009.

[WD10] S. Wang, S. Dey, “Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming,” in IEEE GLOBECOM, Dec. 2010.

[WD12] S. Wang, S. Dey, “Cloud Mobile Gaming: Modeling and Measuring User Experience in Mobile Wireless Networks,” *ACM SIGMOBILE MC2R*, vol. 16, issue 1, Jan. 2012, pp. 10-21.

[WD13] S. Wang, S. Dey, “Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications,” *IEEE Transactions on Multimedia*, vol. 15, no. 4, Jun. 2013.

[WDA10] S. Wang, S. Dey, "Addressing Response Time and Video Quality in Remote Server Based Internet Mobile Gaming," in *Proc. of IEEE WCNC*, Sydney, Mar. 2010.

[WLD12] S. Wang, Y. Liu, S. Dey, “Wireless Network Aware Cloud Scheduler for Scalable Cloud Mobile Gaming”, *in Proc. of IEEE ICC,* Ottawa, Jun. 2012.

[WW03] M. Wimmer and P. Wonka, “Rendering Time Estimation for Real-Time Rendering,” in *Eurographics Symposium on Rendering,* 2003.

[YH06] K. Yamagishi and T. Hayashi, “Opinion Model for Estimating Video Quality of Videophone Services,” IEEE GLOBECOM, Nov. 2006.

[YII05] T. Yasui, Y. Ishibashi, T. Ikedo, “Influences of Network Latency and Packet Loss on Consistency in Networked Racing Games,” In ACM SIGCOMM workshop on Network and System Support for Games, (NetGames), 2005.