Towards Enhancing QoE for Software Defined Networks Based Cloud Gaming Services

by

Maryam Amiri

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the Ph.D. degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The high-profit digital gaming industry has merged with the increasing interest in transforming everything into cloud services, which leads to a novel concept called on-demand gaming. Recently, on-demand gaming, otherwise known as Cloud gaming, has become a promising paradigm for game users and providers. In this thesis, we aim to investigate the optimization of Quality of Experience (QoE) for cloud gaming systems, while considering network challenges, system constraints, and service requirements.

We aim to improve the gamer's QoE from two main network perspectives—cloud perspective, where mechanisms for QoE enhancement must be observed and controlled by the cloud to optimally allocate data center resources to user requested gaming sessions, and home gateways perspective, where the network acquires gamers' QoE parameters and uses them to maximize the overall QoE of the clients who are sharing the same gateway at home.

First, we address the issue of enhancing the client's QoE by proposing a method to optimally allocate data center resources to user requested gaming sessions. The method considers the current status of the network in terms of communication delay, available computational resources in game servers and processing delay, and the genre of the requested game to select the appropriate server in the cloud for game execution and the network path for game data transfer within the data center. Second, we propose a novel mechanism to optimize the bandwidth distribution in the player's home network to maximize gaming QoE without starving other concurrent applications. The Proposed method is a home-side Game-Aware Resource Manager (GARM) that intelligently assigns network resources to active applications based on their requirements.

Acknowledgements

First, I would like to express my deepest gratitude to my supervisors, Professor Shervin Shirmohammadi and Professor. Hussein Al Osman, for their unrelenting support, constructive guidance and encouragement throughout my graduate studies.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Peter Liu , Prof. Emil Petriu, Prof. Amiya Nayak, and Victor C.M. Leung for their insightful comments and constructive feedback.

Also, I would like to thank the Canadian federal and provincial governments for funding my research over the past couple of years, through QEII-GSST.

Moreover, I would like to thank all members of the Distributed and Collaborative Virtual Environment (DISCOVER) Laboratory, for their cooperation, support and simply being wonderful friends.

Last, but not least, I want to express my infinite appreciation and love to my husband Ashkan whose consistent encouragement, support and unconditional love have taken me through all hardships incurred during my academic journey.

Table of Contents

List of Figures

List of Tables

List of Definitions

AHP	Analytic Hierarchy Process
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
CI	Consistency Index
CR	Consistency Ratio
DCN	Datacenter Network
DP	Dynamic Programming
eCDF	Empirical Cumulative Distribution Function
ECMP	Equal Cost Multi-Path routing
FPS	First Person Shooter
FTP	File Transfer Protocol
GA	GamingAnywhere
GaaS	Game as a Service
GARM	Game Aware Resource Manager
GMC	Cloud mobile gaming
GMOS	Game Mean Opinion Score
HTTP	Hyper Text Transfer Protocol
IDC	Intra Data Center
IP	Integer Programming
ISP	Internet Service Provider
KL-divergence	Kullback-Leibler divergence
LP	Linear Programming
LR	Lagrangean Relaxation
LR-GaaS	Local Rendering GaaS
NP-hard	Non-deterministic polynomial-time hardness
OCO	Online Convex Optimization
OF	OpenFlow
OP	Overprovisioning
PDI	Preference Degree Index
PSNR	Peak Signal to Noise Ratio
Ndisc	queuing discipline

N-P	Nonlinear Programming
MCTEQ	multi-class QoS-guaranteed IDC traffic management scheme
MOSP	Modified Online Saddle Point
QoE	Quality of Experience
RPG	Role-Playing Game
RR-GaaS	Remote Rendering GaaS
RTS	real-time strategy
RTT	Round-Trip Time
SDN	Software Defined Network
SSIM	Structural Similarity Index
TC	traffic control
ToR	Top of Rack
ToS	Type of Service
VM	virtual machines
VoIP	Voice over IP
VQM	Video Quality Monitor

List of Symbols

N	Number of Gaming Sessions
i	Gaming Session Index
x_t^{ij}	Binary Variable for Selection of A DC
β^{ij}	Amount of Bandwidth Assigned to Game I
$\propto$	Fairness Coefficient Factor
$d_{r_{ij}}$	Response Delay
ω_1, ω_2	Weighting Parameters
x	Resource Allocation Vector
M	Number of Available Dcs
j	DC Index
$\delta_{i_{max}}$	Maximum Tolerable Delay for Game I
U_i	Utility Function of Gaming Session I
C_j	Maximum Bandwidth Capacity of A DC
$d_{n_{ij}}$	Transport Delay
γ, μ	Smoothing Parameters
λ , ν	Lagrangean Parameters
M	Number of Gaming Sessions
i	Gaming Session Index
j	Available Paths Index
x_{ij}	Binary Variable for Selection of a Path
β_{ij}	Amount of Bandwidth Assigned to Game I
p	Set of N Possible Disjoint Paths
P_b	Network Resource Importance Parameters for Bandwidth
d^{φ}	Set of The Inverse Mean Delay for Each Path
CI_b	Consistency Index
ρ_d	Preference Degree Index for Delay
M_{δ}^{b}	Pairwise $N \times N$ Comparison Matrices for Residual Bandwidth
j	Gaming Server Index
φ_{max}	Maximum Tolerable Delay
U_i	Utility Function of Gaming Session I

δ_{max}	Maximum Bandwidth Capacity
P_d	Network Resource Importance Parameters for Delay
b^{δ}	Set of Residual Bandwidths for Each Path
x_{ij}	Binary Variable for Selection of a Path
ρ_b	Preference Degree Index for Bandwidth
M_{φ}^{d}	Pairwise $N \times N$ Comparison Matrices for Residual Delay
φ_{ij}	Network Delay Experienced by Game Session I
N_p	Number of Available Paths
j	Gaming Server Index
Dmax	Maximum Tolerable Delay
c_{s_j}	Capacity of Server J
rb_i	Required Bandwidth of Game I
BW_{Max}	Maximum Bandwidth of Path K
x_{ij}	Binary Variable for Selection of a Game Server
z_{ijk}	Binary Variable for Selection of a Path
U(r)	Utility function
i	Gaming session index
k	Available paths index
$d_{p_{ij}}$	Processing delay of game i servers by game server j
$d_{n_{ijk}}$	Network delay of path k
α_i	Weight factor of network delay for game i
β_i	Weight factor of processing delay for game i (1- α)
N_s	number of active game servers
N_g	number of gamers
λ	Lagrangean multipliers parameter
φ^t	subgradient step size
t	Number of iterations
ρ	Number of available ports on each switch
$\upsilon_i, \sigma_i, \tau_i$	Processing delay model parameters
N_g	number of gamers
λ	Lagrangean multipliers parameter

Chapter 1. Introduction

1.1 Motivation

Recent advances in cloud computing, data center deployment and virtualization technologies bring new opportunities to the gaming industry, and enable end-users to play high-end graphics games on any low-end device [1]. Cloud gaming [2], otherwise known as on-demand gaming, has become a promising paradigm for game users and providers. Their growth is driven by recent advances in cloud computing, data center deployment, and virtualization technologies. The application of the cloud computing model to cloud gaming offers many attractive advantages, such as scalability, ubiquity, reliability and cost reduction to game users and providers [3]. Based on recent industry facts reported by the Research and Market Institute, revenue from the cloud based video games industry reached 476 million US$ by the end of 2015, and such a market will continue to thrive in the years ahead and is expected to reach 650 million US$ by the end of 2020 [4]. Generally, there are two main approaches for online gaming: online gaming based on file streaming and cloud gaming based on video streaming. Although both approaches utilize the client-server architecture, in the former, the server sends updates to the client in response to a change in game context, whereas, in the latter the game server performs game logic processing and rendering, and finally encodes the game graphics into a video that is streamed onto the client. Hence, in the video streaming approach, the game server is responsible for a significant

part of the computational operations while the computational load is considerably diminished at the client side. Consequently, with this approach, gamers can make use of thin clients. Therefore, gamers are not concerned with matching gaming devices with the specific hardware requirements of the latest games. Since the major computational work is undertaken in the cloud, realizing a hardware infrastructure that guarantees the required Quality of Service (QoS) is a key challenge for game providers. Such infrastructure conventionally makes use of overprovisioning to ensure satisfactory QoS. Hence, hardware costs can reach unmanageable levels as evidenced by OnLive's financial troubles due to their inability to maintain the servers running their services (which averaged a mere 1600 concurrent users per game server) [5].

Since public cloud infrastructures (e.g Amazon) have failed to guarantee the QoS requirements of cloud gaming systems [6], some of the existing cloud gaming vendors, like Gaikai, have invested in their own proprietary data centers [7]. Hardware requirements (e.g. processing equipment) in these centers can differ based on the game genre. Therefore, the limited computational resources in the cloud should be efficiently assigned to games while accounting for their processing requirements, to avoid over utilizing or underutilizing resources. In addition to the game genre, the type of gaming device and the experience of the gamers have a significant impact on network resource requirements and tolerable delay [8], [9]. All of these factors must be taken into account when planning and deploying a data center for cloud gaming [10].

Since the high delay experienced by players is the most challenging issue in cloud gaming, a variety of techniques have been introduced

to address it. Most of these techniques are focused on delay reduction in the game engine, or during video rendering and encoding [11]–[14].

In addition to delay, there are several QoS-related factors that contribute to the decrease in network performance and adversely impact players' QoE. These factors can stem from the cloud, client, or the network connecting them. From the client side, bandwidth usage, packet rate, and client's device affect the QoE. From the cloud side, bandwidth limitation, delay and/or delay variation (known as jitter) increase, and packet loss negatively impact QoE. The same factors apply to the network connecting the client to the cloud. However, we do not have any control over the Internet Service Provider (ISP) networks. Hence, the proposed techniques aim to principally address the QoE degradation factors within the cloud. Generally, these techniques focus on game engine enhancements related to video rendering and only a minority of works consider data center network routing improvements and virtual machines (VM) selection optimization.

1.2 Challenges and research problem

In this section, we present the challenges addressed in this thesis. For cloud gaming applications, like other real-time multimedia applications, QoE highly depends on network conditions. Generally, cloud gaming services have tight delay requirements, and the failure to meet them is one of the main reasons for losing users [8], [9], [15]. We divide the overall delay experienced by users into three main components: playout, network, and processing delay. Playout delay refers to the time it takes to decode and play a packet payload

received from the cloud. Network delay corresponds to the DC intra-delay and Internet Service Provider (ISP) delay incurred during communication. Processing delay is the time required by the game server to process the users' commands, render the corresponding video frames, and compress them. Besides, Since the rendered game scene is delivered to gamers through a compressed video stream, cloud gaming is a bandwidth demanding service. Thus, we must consider bandwidth related challenges to optimize the delivery of this service. In this regard, the most important challenge that the provider should handle is the imbalance in the distribution of game traffic among DCs. Depending on the employed allocation strategy, it is possible that some DCs are heavily overloaded, especially during peak times, while others are underutilized. Such situations may lead to server crashes and network congestion which can impair the performance of gaming applications. At the same time, other DCs may be underutilized in terms of bandwidth [16], [17], while needlessly consuming energy. This situation can be worsened by a surge of gamers, where server allocation is unwisely performed just based on the geographic proximity to DCs with the sole aim of reducing transport delay. Now, if the number of gamers located in the same geographical location increases, the closest DCs are prone to be overwhelmed [18].

Tailoring the public cloud infrastructure to the specific requirements of cloud gaming is a relatively challenging task. Also, the results in [6] indicate that public cloud providers such as Amazon are not capable of supporting cloud gaming services with proper QoS provisioning. Hence, cloud gaming service provider companies (e.g. StreamMyGame, Giaki and OTOY) are investing in

building their own infrastructure and proprietary platforms to facilitate the allocation of processing and computational resources while meeting the gaming-specific needs.

We consider the problem of network resource allocation within the cloud including intra Data Center Network (DCN), inter DCN, and home gateways to not only minimize the delay experienced by gamers, but also, maximize bandwidth utilization.

In intra DCN, firstly, we consider the problem of a cloud path selection method to reduce end-to-end delay and delay variations (jitter) within the DCN. The method adaptively disperses the game traffic load among different network paths according to their end-to-end delays. The method uses a global view of the network status as its input and hence employs a Software Defined Network (SDN) controller to direct routing decisions for the streams of gaming data within the cloud. Secondly, in addition to selecting the appropriate routing strategy to decrease the delay experienced by gamers, we consider the specific characteristics of games to select the appropriate routing strategy. Game characteristics vary between different games, and to offer an optimized experience, we must consider game specifics. Delay sensitivity often varies based on the game genre. For example, First Person Shooter (FPS) games are more sensitive to delay compared to real-time strategy (RTS) games [19]. Hence, the tolerable delay depends on game genre. Ideally, routing algorithms must resolve the best routing strategy to satisfy certain QoS requirements. In QoS-based routing schemes, routes should be chosen based on features of the transmitted data flows, such as bandwidth requirement and delay sensitivity. There are two main goals that need to be achieved by the QoS-based routing

algorithm. The first goal is to find a path that satisfies the QoS requirements [20]. The second goal is to optimize the global network resource utilization within the DCN.

In a home network, it is common for multiple heterogeneous applications that share a single access point (gateway) to the Internet to collectively experience network lag due to limited network resources. Such a lag has varying effects on different types of applications. For instance, online gaming or live conferencing are most negatively affected by latency, followed by video on demand or music streaming. Web browsing or email applications are the least affected by latency. Once network bandwidth becomes either limited or insufficient, QoS guarantees must be implemented to better support real-time streaming multimedia applications (e.g. online games). To provide such QoS guarantees, we propose a Type of Service (ToS) based priority method [21] to facilitate the specification and control of network traffic by different classes, so that certain types of traffic get precedence. For example, voice traffic, which requires a relatively uninterrupted flow of data, might get precedence over web browsing. As a result, to guarantee a desired level of QoS, network flows are assigned to resources based on their class [22].

ToS-based approaches however are mostly implemented between telecommunication companies and service providers, and, to the best of our knowledge, have never been extended to the home users' networks. Nevertheless, such approach presents an interesting strategy to provide QoS. Inspired by this strategy, we propose a home-side Game-Aware Resource Manager (GARM) that intelligently assigns network resources to active applications based

on their requirements. Thus, GARM acts as an application-aware networking router.

1.3 Methodology Approach

We summarize the approaches used for addressing the technical challenges described in Section (1.2) as follows:

To minimize the overall delay experienced by gamers within a DCN, we propose a method to optimally allocate data center resources to user requested gaming sessions. The method considers the current status of the network in terms of communication delay, available computational resources in game servers and processing delay to select the appropriate server in the cloud for game execution and the network path for game data transfer within the data center. The goal of the optimization method is to minimize the delay while improving the QoE of game users. Since the method uses a global view of the network status as its input, a Software Defined Network (SDN) controller is used to execute the method. The SDN controller is in charge of deciding on the forwarding strategy for the streams of gaming data within the data center.

Since the proposed method places high computational constraints on the SDN controller, we propose a Lagrangean Relaxation (LR) heuristic method to solve the combinatorial problem by finding a near-optimal solution in polynomial time. In addition, we model the relationship between bit rate and perceived quality, and use this model to develop our heuristic method. Such model is essential since a reduction in the bit rate can have an adverse effect on the QoE. Also, we propose a bi-objective optimization scheme that strives to jointly minimize the overall delay experienced by users and maximize bandwidth utilization within a data center by considering the requested games genres. The

proposed scheme makes use of an Analytic Hierarchy Process (AHP) based Game Aware Routing (AGAR) method to find the initial feasible solution that establishes the optimal path between the core and Top of Rack (ToR) switches in a cloud data center, while considering game-specific characteristics.

Moreover, we construct a model in which multiple DCs are connected through the network controlled by SDN. In the proposed hierarchical SDN architecture, the local SDN controllers govern the intra-DC networks, and send the network-related information of each DC to a centralized SDN controller that oversees the intra-DC communication. Due to the high computational complexity of the optimization method, we designed an algorithm based on an Online Convex Optimization (OCO) method to drastically reduce the computational complexity of the problem such that it can be practically implemented in a DC on an SDN controller.

For the home-side gateways, we model the network resource allocation as an optimization problem and use a logarithmic utility function for bandwidth management of competing flows, including game flows. Since home networks are relatively small, the computational complexity of the optimization process will not be an obstacle to the realization of the solution. The logarithmic utility function proportionally shares the bandwidth among competing flows, giving precedence to game flows over others with less stringent delay or bandwidth requirements without sacrificing fairness.

1.4 Contributions

The contributions of this thesis are as follows:

1. Proposing an optimization framework for minimizing the delay experienced by gamers by optimally allocating data center resources to user requested gaming sessions. The proposed method considers the current status of the network in terms of communication delay, available computational resources in game servers and processing delay, and the genre of the requested game to select the appropriate server in the cloud for game execution and the network path for game data transfer within the data center. To consider the different game genres, we introduce a QoE model using objective measures of video quality and QoS related metrics.
2. Proposing a bi-objective optimization scheme that strives to jointly minimize the overall delay experienced by users and maximize bandwidth utilization within a data center by considering the requested games genres. The proposed scheme makes use of an Analytic Hierarchy Process (AHP) based Game Aware Routing (AGAR) method to find the initial feasible solution that establishes the optimal path between the core and Top of Rack (ToR) switches in a cloud data center, while taking game-specific characteristics into account.
3. Proposing a hierarchical SDN model for the selection of DC for a new gaming session. We utilize a hierarchical model to consider transport delay, response delay and bandwidth status of intra and IDC traffic flows. Thereby we formulate our joint optimization problem, apply a smoothing logarithmic technique and use the Online convex method to solve the original NP-hard problem.
4. Developing an optimization method called Game Aware Resource Manager (GARM) that considers fairness and Network Utility Maximization (NUM) for the rate allocation in home networks.

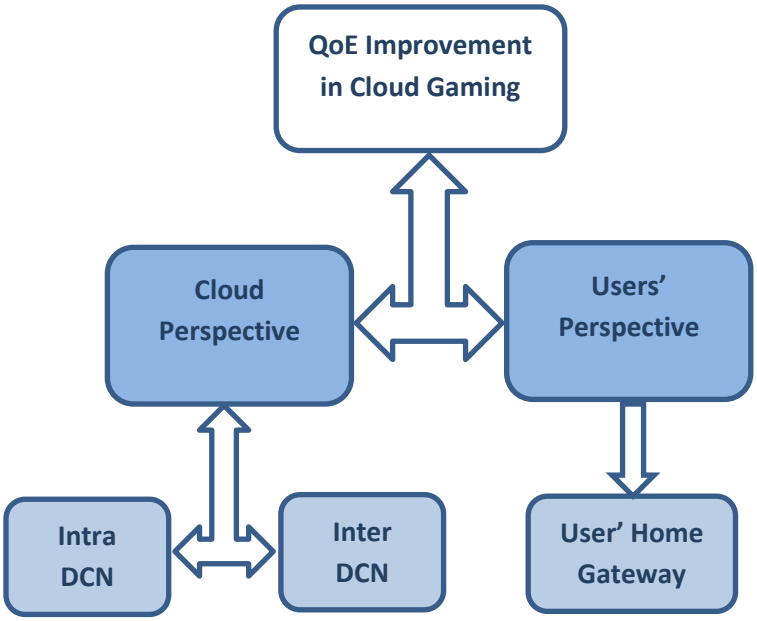


Figure 1. A flow diagram of the contributions of this thesis

As illustrated in Figure 1, in this thesis, we aim to improve the gamer's QoE from two main network perspectives—cloud perspective (Chapter 3,4 and 5), where mechanisms for QoE enhancement must be observed and controlled by the cloud to optimally allocate data center resources to user requested gaming sessions, and home gateways perspective (Chapter 6), where the network acquires gamers' QoE parameters and uses them to maximize the overall QoE of the clients that are sharing the same gateway at home.

1.5 Research Publications

In the process of completing this thesis, we have submitted or published the papers listed below.

1.5.1 Journal Articles

J1. AMIRI, M., ALOSMAN, H., SHIRMOHAMMADI, S. AND ABDOLLAH, M. Towards Delay-Efficient Game-Aware Data Centers for Cloud Gaming. ACM transaction on multimedia computing, communications, and applications (TOMM), April 2016. (The contents of this paper will appear in Chapter 3.)

J2. AMIRI, M., SOBHANI, A., ALOSMAN, H. AND SHIRMOHAMMADI. A Cloud Gaming Aware SDN Controller Using Analytic Hierarchy Process. IEEE Access, September 2017. (The contents of this paper will appear in Chapter 4.)

J3. AMIRI, M., ALOSMAN, H., AND SHIRMOHAMMADI, S. Towards Resource Optimization Through Hierarchical SDN-Enabled inter Data Center Network for Cloud Gaming. Submitter to IEEE transaction on cloud computing, (Under Revision), April 2019. (The contents of this paper will appear in Chapter 6.)

1.5.2 Refereed Conference Papers

C1. AMIRI, M., AL OSMAN, H., AND SHIRMOHAMMADI, S. 2018. Game-Aware and SDN-Assisted Bandwidth Allocation for Data Center Networks. In Proceedings of IEEE International Conference on Multimedia Information Processing and Retrieval (IEEE MIPR 2018), Miami, Florida, USA.

C2. AMIRI, M., AL OSMAN, H., AND SHIRMOHAMMADI, S. 2017. SDN-enabled Game-Aware Network Management for Residential Gateways. In Proceedings of IEEE 19th International Conference on Multimedia (IEEE ISM 2017), Taichung, Taiwan.

C3. AMIRI, M., AL OSMAN, H., AND SHIRMOHAMMADI, S. 2017. Game-Aware Bandwidth Allocation for Home Gateways. In Proceedings of ACM 8th International Conference on Multimedia Systems (MMSYS2017), Taipei, Taiwan.

C4. AMIRI, M., AL OSMAN, H., AND SHIRMOHAMMADI, S. 2016. Datacenter Traffic Shaping for Delay Reduction in Cloud Gaming. In Proceedings of the 14th Annual IEEE International Conference on Multimedia, San Jose, USA, 2016.

C5. AMIRI, M., SINGH MALIK K., AL OSMAN, H. AND SHIRMOHAMMADI, S. Game-Aware Resource Manager for Home Gateways. In Proceedings of the 14th Annual IEEE International Conference on Multimedia, San Jose, USA, 2016.

C6. AMIRI, M., AL OSMAN, H., SHIRMOHAMMADI, S. AND ABDALLAH, M. 2015. An SDN Controller for Delay and Jitter Reduction in Cloud Gaming. In Proceedings of the 23rd Annual ACM Conference on Multimedia Conference, 1043-1046.

C7. AMIRI, M., ALOSMAN, H., SHIRMOHAMMADI, S. AND ABDOLLAH, M. 2015. SDN-based Game-Aware Network Management for Cloud Gaming. In The 14th International Workshop on Network and Systems Support for Games (Netgames 2015), Croatia.

1.6 Organization of Thesis

The road map for the rest of this thesis is outlined below:

Chapter 2– Background and related works presents background information on cloud gaming; it provides an extensive review of the

existing methods aimed for delay reduction, bandwidth utilization, and fairness maximization. It also discusses QoE in the context of cloud gaming and recent paradigms in cloud networking e.g. SDN and DCN architecture.

Chapter 3– Proposed Game-Aware Optimization Architecture presents the core components of a novel method for minimizing the end-to-end latency within a cloud gaming data center. It formulates an optimization problem for reducing delay, and proposes a Lagrangean Relaxation (LR) time-efficient heuristic algorithm as a practical solution. It presents extensive experiments to show that the proposed optimization scheme leads to better system performance, such as overall latency.

Chapter 4– Proposed SDN-Enabled Game-Aware Routing for Cloud Gaming Datacenter Network provides a novel bi-objective optimization method to find an optimum path for packet transmission within a data center by minimizing delay and maximizing bandwidth utilization. We use a metaheuristic model, called analytic hierarchy process, to solve the NP-complete optimization problem. The resulting method is an analytic hierarchy process-based game aware routing (AGAR) scheme that considers requested game type and requirements in terms of delay and bandwidth to select the best routing path for a game session in a cloud gaming network. The method executes within a SDN controller, which affords it a global view of the data center with respect to communication delay and available bandwidth. The chapter presents a comparison between the proposed method and other existing methods with respect to end-to-end delay, bandwidth utilization, delay variation, and etc.

Chapter 5– Resource Optimization through Hierarchical SDN-enabled inter Data Center Network for Cloud Gaming presents the efficient central resource management for multiple distributed DCs using a SDN approach. We present a novel optimization method for assigning DCs to gaming requests that accounts for game requirements. Due to the high computational complexity of the optimization method, we designed an algorithm based on Online Convex Optimization (OCO) method to drastically reduce the computational complexity of the problem such that it can be practically implemented in a DC using the OF controller. We describe a prototype designed to demonstrate the efficiency of the proposed method.

Chapter 6– Game-Aware Bandwidth Allocation for Home Gateways presents a proposed method to optimize the bandwidth distribution in player's home network to maximize gaming QoE without starving other concurrent applications. We present a practical design methodology and implement the proposed platform. We describe a prototype designed to demonstrate the efficiency of the proposed method.

Chapter 7– Conclusion and Future Work provides a conclusion and an outlook on future works.

Chapter 2. Background and Related Works

2.1 The Architecture of Cloud Gaming Service

As a result of recent advancements in cloud computing and data centers, there has been a dramatic increase in the demand for cloud computing services in the last few years. In fact, centralizing the resources and computational processes into data centers brings about inexpensive and flexible opportunities for a multitude of existing applications. Gaming services is one of these applications that can benefit from cloud computing advancements. The ubiquitous and scalable nature of data centers leads to cost reductions for game service providers by deploying games faster through the scalable environment of the cloud and increasing their revenue by attracting more users. Furthermore, it enables users to play numerous games using a thin-client regardless of their location or what platform they use (PCs, laptops, tablets, smartphones).

Cloud gaming is considered among the most profitable cloud computing services today. Revenue from cloud gaming is growing rapidly and by 2020 is estimated to surpass 1.3 billion dollars [4], [23]. Consequently, global companies are starting to rollout cloud gaming platforms to support this expected growth. For instance, Google recently announced a new cloud gaming service called Stadia [24], [25] for streaming high-resolution games.

The cloud gaming architecture can be divided into three main categories as discussed in [26]. The first category is known as remote rendering GaaS (RR-GaaS). In this architecture, Cloud gaming employs a client-server architecture where the game logic is executed at powerful

cloud game servers. Game scenes are encoded as compressed audiovisual signals and streamed to game users operating on heterogeneous client platforms (i.e. PC, laptop, tablet, game consoles, desktops, set-up boxes and smartphones).

Clients in this model are merely in charge of sending the game control inputs and displaying the received game video scene, as illustrated in Figure 2. Since the major part of computational operations is performed in the cloud, the game users do not need to constantly upgrade their devices to meet the specific hardware requirement of the latest games. Despite these advantages, offering cloud gaming services that match the quality of experience (QoE) of console games remains a challenge. Cloud gaming services require a minimum of a 3Mb/s bandwidth to ensure satisfactory QoE [27]. Also, cloud games introduce about 1.7 times higher latency than their console counterparts [28], which also could adversely impact the players' QoE. In fact, cloud games typically require a maximum network latency of 80 ms, a requirement that only 70% of end-users are able to meet [6].

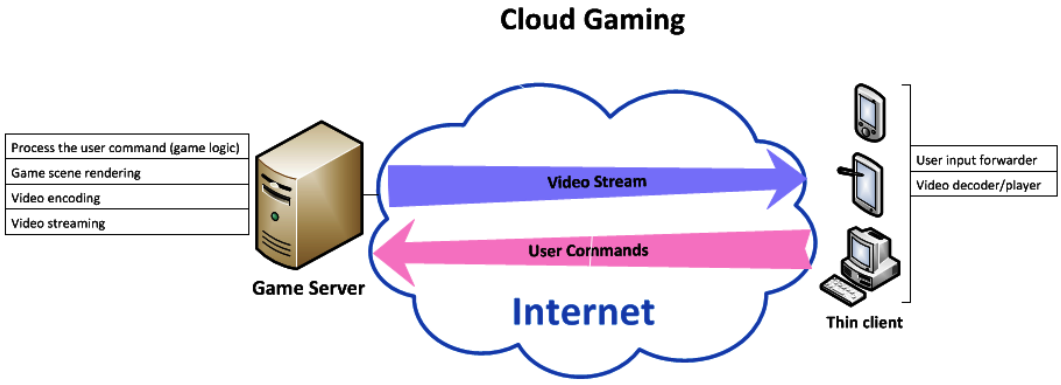


Figure 2.The architecture of cloud gaming services

In the second category, the game server is no longer responsible for performing the rendering operation such as video encoding, video streaming and video decoding. Instead, the display instruction is sent to

the game terminal by game logic, and the terminal renders and displays the game locally. This scheme is called local rendering GaaS (LR-GaaS). Although this scheme reduces the network cost for game service providers by reducing the network workload, the terminal has to meet the specific hardware requirement to be capable of rendering the game video frames locally.

In the third category, a cognitive resource allocation GaaS scheme is considered as the architecture of cloud gaming. In this structure, some of the computational operations may be performed at the server side or the client side based on the network, client, and server conditions. In other words, this architecture enables the cloud gaming system to select optimal component combinations according to the system's situation. For example, the well-known commercialized cloud gaming systems G-cluster Global Corporation[1], Gaikai[2], LiquidSky[3], OnLive[4], Playcast Media Systems[5], StreamMyGame[6] , and Google Stadia [7]utilize the RR-GaaS scheme as the cloud gaming architecture.

In the RR-GaaS scheme, games are run in the cloud; i.e., gamers' commands are sent to the cloud server and appropriate actions are taken by the game engine so that it produces the corresponding video frames, rendered by a GPU and encoded by a video codec. Afterwards, the compressed frames are sent to players through the network, and finally decoded and played out on thin-client devices. All of the above steps can impose additional latency on cloud gaming systems. A wide range of

1 http://www.g-cluster.com/en/index.html
2 http://www.gaikai.com/
3 https://liquidsky.com/
4 http://www.onlive.com/
5 https://www.gamefly.com/streaming
6 http://streammygame.com/smg/index.php
7 https://en.wikipedia.org/wiki/Google_Stadia

techniques have been proposed to deal with these latency issues. Most of these techniques focus on the processing of the game and the communication of game data within the cloud, and can be divided to two main categories:

1. Computational methods to optimize the processing of games in data center servers (e.g. gaming logic processing, video rendering, video encoding, and video streaming).
2. Network methods to improve the distribution of streamed video within the cloud.

In the next section, we look at the background and related work in four areas: A) cloud gaming delay, and delay reduction techniques, B) relation between network resources and game genres, and C) DCNs and SDN.

2.2 Cloud gaming delay, and delay reduction techniques

Delay is the most important problem affecting cloud gaming today. Cloud gaming users experience higher delays in comparison to users of online and console games. It is well-accepted that the maximum tolerable delay by cloud gaming users depends on various factors, such as the pace of game, experience of gamers, game genres and type of gaming device (e.g. desktop, laptop or smartphone). For example, First Person Shooter (FPS) games are more sensitive to delay compared to real-time strategy (RTS) games [9]. Also, a delay of less than 100 ms is highly desirable for high action paced games, whereas 150 ms is the delay threshold for slow paced games [10]. In general, previous works have found that a delay of more than 100 ms becomes noticeable by gamers [29] and has a negative impact on the QoE. Therefore, ideally,

the network delay threshold should be set at 80 ms to account for the additional computational processing delay on game servers [6].

In addition to latency, the amount of latency variation, known as jitter, also has detrimental effects on gamers' QoE [28]. In [30], the authors show that cloud games suffer from a much higher level of jitter compared to console games. In some cases, jitter has been found to be even larger than 100 ms in cloud games. However, a jitter of more than 70 ms has a significant adverse effect on players' QoE [29].

According to [31], the total cloud gaming end-to-end service delay experienced by users consists of three components: 1) playout, 2) network, and 3) game server delay. Playout delay is the time it takes the client to play the compressed video after receiving it from the cloud. Networks delay consists of data center intra-delay and ISP delay. Server delay is the time spent by the game server to process the users' commands, render the corresponding video frames, and compress them.

Since the overall delay experienced by users stems from different sources, the proposed techniques to reduce delay can be divided into two main categories. The first category includes methods that improve the game engine and video codec components through techniques of image warping for motion estimation, view compensation, and object information extraction to facilitate video compression [32]. The second category includes mechanisms for network resource provisioning, VM placement, and server selection. Although there is a large body of work that covers the first category of methods, the schemes pertaining to the second category are closer to our work. Therefore, in this section, we will further explore the latter methods.

Several works proposed to mitigate the delay through VM migration and/or dynamic game server provisioning. Jalaparti et al. proposed a

framework in which the end-to-end delay of each game session is being constantly monitored. Once it reaches a critical threshold, the corresponding game server is moved to a new optimum location to lower the end-to-end delay [33]. Similarly, [34] presented a VM placement solution that aims to maximize the net profit for the service provider while providing users with satisfactory QoE. In this regard, a server selection method was proposed by [29] which aims to minimize the end-to-end latency by assigning optimal servers to a group of gamers within the same geographic area.

In regards to the efficient utilization of DCNs, the results of the experiments conducted by [35] on a DCN comprising 150 inter-switches and 1500 servers, show that despite the existence of many network links, 15% of congestions remained for more than 100 seconds.

2.3 Network Resources and Game Genres

Online games in general have been remarked as delay sensitive applications; however, the maximum tolerable delay varies with the game genre. As described in [9], for First Person Shooter (FPS) games, the users will notice a delay of more than 80 milliseconds while for Real-Time Strategy (RTS) games, the delay of more than 150 ms has a detrimental effect on users' QoE.

It is well established that game design significantly affects network traffic characteristics [27], [36], [37]. Although the fundamentals of game design are similar for most video games, there are some design aspects that differ from one genre to another (e.g. camera perspective). Moller et al. [38] proposed a generic evaluation framework that identifies the influence factors that are relevant for gaming and have an

impact on QoE. They identified the following three layers: 1) QoS influence factors; 2) User and system interaction performance aspects; 3) QoE features related to the end user quality perception and judgement processes. Claypool [39] showed that game design can dictate the required computational and network resources and affect the users' perceived quality score. Moreover, he introduced two indices that are directly relevant to QoS parameters: the average of Intra-coded Block Size (IBS) and percentage of Forward/backward or Intra-coded Macroblocks (PFIM) to measure the scene complexity and the amount of motion for different game designs. Suznjevic et al. [40] demonstrated the relation between game genre and network characteristics (e.g. bandwidth usage and packet rate). Jarschel et al. [10] showed the impact of game genre on perceived QoE under the same network conditions.

In addition to game genre, several other factors contribute to the maximum tolerable delay [10]. For example, experienced users are more sensitive to delay compared to novice users. In general, for a high QoE, the network delay must be below 100 ms [41].

2.4 DCNs and SDN

A cloud data center can host many delay sensitive applications. A study of a DCN consisting of a 6000 production server clusters has shown that it is incapable of handling network flows with tight completion deadlines as a result of the limited capacity of its switches and links [42]. Most DCNs today adopt a multi-tier network architecture that consists of distinct layers of network elements (e.g. core, aggregation and access) as depicted in Figure 3. Typically, a higher

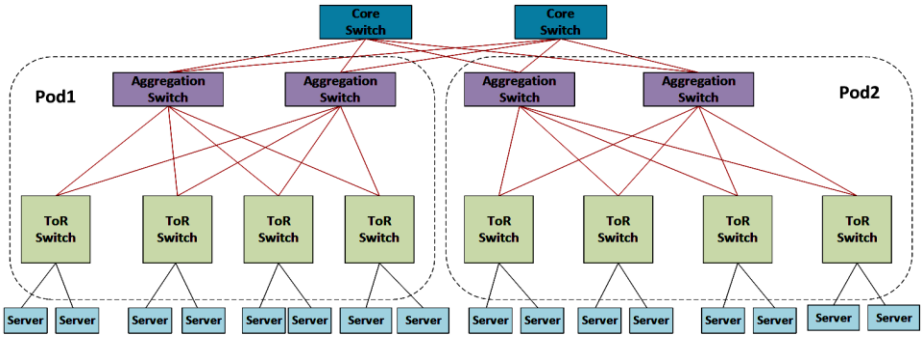


Figure 3. Typical Datacenter 3-tier Model[43]

number of layers is expected to result in greater network availability. In a cloud gaming data center,

a long packets downstream is transmitted as compressed video. The game flows are delay sensitive and should meet certain delay requirements to provide the user with satisfactory QoE. Moreover, the game flows are also throughput sensitive, bounded by the overall long flow completion time. Given the characteristics of the network flows transmitting within the cloud gaming data center, the congestion in data center switches is inevitable. Congestion control solutions through overprovisioning (OP) are commonly used to reduce the effects of congestion in the DCN. However, OP solutions can be expensive as they require the deployment of additional resources in the DCN. Conversely, current data centers use multipath routing along with traffic engineering to improve efficiency and reliability. Modern networks employ multipath routing consisting of the shortest and alternate paths to avoid interruption of service. Even though several approaches exist to determine the optimal paths in the network, they usually select the best routes based on the link metrics provided at setup time, and they often fail to take into account the current status of links, such as current

available bandwidth, delay and packet loss. Also, they do not consider the requirements of data flows passing through the network. In traditional networks, despite the fact that a router advertises its routing table to other routers, each router autonomously controls its own routing table using the incomplete network information it possesses. This distributed control is not efficient with increasing traffic. So, SDNs present a new approach for traffic forwarding through a centralized network controller. In short, SDN separates the network controls and the forwarding plane to optimize each layer separately. Also, SDN brings the application layer and the network layer closer together, allows the network to be more adaptable to different conditions and assists it in responding to application requests. SDN allows the network to be more adaptable to different conditions and assists the network to respond to application requests in real time. This agility in control can be effectively exploited by cloud gaming systems. However, current SDN-enabled switches have very narrow capacity limits[44].

The OpenFlow (OF) protocol is the most commonly used control protocol in SDN-enabled networks. It is used by an SDN controller to dictate forwarding rules to switches and routers. The flow table is the principal element of OF. It enables OF switches to cache the network control policies and rules. Due to hardware limitations, most of the current OF switches can store fewer than a thousand entries in their flow tables. Hence, flow table capacity is a resource that must be taken into account during flow control.

Network resource allocation techniques using SDN can be divided into two main types: 1) user-centric methods where end-users' benefits are maximized and 2) service provider-centric approaches where service providers' profit is maximized.

In terms of user-centric methods, Xu and Li presented a proportional fairness optimization scheme for the DC selection problem for cloud services [45]. They adopted a dual decomposition approach to develop a distributed algorithm based on the sub-gradient method. Wu et al. [46] proposed a cloud-based computing resource allocation technique, and formulated a task scheduling problem to minimize the workflow end-to-end delay under a user-specified financial constraint. In terms of service provider-centric approaches, [47] proposed Decentralized Server Selection for cloud services (DONAR's) algorithms to jointly consider client performance and server load in replica-selection to minimize the network costs. Boloor et al. [48] proposed a distributed dynamic rank-based request allocation and a Geographical Indication_FIFO (gi-FIFO) scheduling scheme that aims to maximize the total profit charged by the cloud service provider.

In terms of SDN-enabled wide-area networks, Hong et al. [49] presented SWAN, a framework that frequently reconfigures the data plane to minimize congestion while employing a limited number of forwarding rules to accommodate the restricted forwarding table capacity of commodity switches. Similarly, [50] presents an SDN-based traffic distribution algorithm for IDC WANs to fairly allocate bandwidth among all flow groups while maximizing average bandwidth utilization.

Although most relevant research focuses on optimizations that either benefit the end-user or service provider, few studies jointly consider both. Li et al. [51] proposed an SDN-enabled joint optimization model that maximizes bandwidth utilization to benefit the provider and minimize delay to benefit end-users. However, the authors proposed a single SDN controller to make forwarding decisions. The controller represents a single point of failure. In addition, it might be unable to

resolve congestion once the number of users abruptly increases. Similarly, Wang et al. [52] presented a multi-class QoS-guaranteed IDC traffic management scheme (MCTEQ) to demonstrate the benefits of deploying a traffic engineering scheme to prioritize the interactive traffic flows while maximizing overall bandwidth utilization. The traffic engineering technique is generally cumbersome and might not be practical in a real-time context [53], [54]. Also, Ghosh et al. [55] presented a multi-class bandwidth allocation model for IDC traffic management. They proposed a weighted objective function that consists of throughput and delay requirements to minimize the average end-to-end delay for interactive traffic.

2.5 The Appropriateness Of SDN For Cloud Gaming Applications

Although several methods have been proposed to minimize the computational and network cost within a data center, these methods usually take into consideration either the network status or the computational resource status of servers [56]. Moreover, they determine the optimal solution based on static network metrics, and often fail to consider the current condition of the network, such as current available bandwidth, delay, packet loss etc., or the requirements of data flows passing through the network.

The adaptability of SDN-based routing systems is highly desirable for delay sensitive applications like cloud gaming, especially when the number of requests is abruptly increased. In traditional networks, the routing protocols often rely on predefined static routes and do not take into account dynamic parameters like bandwidth utilization, packet loss, delay etc. SDN provides a means for controlling the network in a

centralized manner to not only accommodate current network conditions, but also optimize QoS parameters like throughput, bandwidth availability, and delay.

2.6 Quality of experience (QoE) in Cloud Gaming

Determining and maintaining an acceptable QoE for cloud gaming services is not a trivial task. As explained in [57], cloud gaming is considered as a highly interactive, multimedia-intensive entertainment service. Since the game video scene has to be generated and transmitted to the players at low latency and high throughput to ensure a good gaming experience, cloud gaming is one of the most complex multimedia services for both providers and users. Furthermore, QoE assessment, and measurement in multimedia applications (e.g. cloud gaming) is a challenging task as the evaluation involves a number of subjective factors (e.g., the mood of the user, or the responsiveness of the system), as opposed to the classical QoS assessment platforms, which are mostly network centered [58].

Suznjevic et al. [38] proposed a generic evaluation framework that identifies the influence factors that are relevant for gaming and have an impact on QoE. They recognize the following three layers: 1) QoS influence factors; 2) User and system interaction performance aspects; 3) QoE features related to the end user quality perception and judgement processes.

Jarschel et al. [10][59] researched the perceived QoE of users under different network conditions (e.g. network delay, and packet loss) and contexts, such as game genres and gamer skills in cloud gaming. They concluded that in fast-paced games, the delay component becomes the

dominant metric affecting the QoE. Unfortunately, the base delay of the system was not measured in the study. This is why the latency limits found cannot be directly used to set boundaries to acceptable end-to-end latencies.

In [60], the authors assessed the impact of different factors (e.g. the type of game, the display size, and network delay) on the perceived QoE of cloud gaming. The results of this study show that display size has a considerable impact on both overall quality. For delay, the impact is more heterogeneous than screen size, as it depends on how network latency impacts game rules and visual rendering.

In another study [61], the authors observed the impact of different network (e.g. delay and loss conditions) on QoE for in-home cloud gaming. They used GamingAnyWhere [27] as the cloud gaming platform, and the results of their subjective test showed that while delay and packet loss have a statistically significant impact on gaming QoE, the diminution of QoE caused by delay and loss, even when these network degradations are high, are not severe.

Huang et al. [62] carried out a user study to evaluate the user satisfaction in mobile and desktop cloud gaming. Their experiments revealed important facts about gamers' QoE. First, gamers are more satisfied with the graphics quality on mobile devices, while they are more satisfied with the control quality on desktops, second, the bitrate, frame rate, and network delay significantly affect the graphics and smoothness quality, and third, the control quality only depends on the client type (mobile versus desktop).

The authors in [63] evaluated the factors affecting users' QoE for cloud mobile gaming in a cellular network. They introduced the Game Mean Opinion Score (GMOS) which is a function of game genre, frame

rate, rendering delay, encoding delay, network latency, and packet loss to model the QoE.

Chapter 3. Proposed Game-Aware Optimization Architecture

3.1 Introduction

Since the high delay experienced by players is the most challenging issue in cloud gaming, a variety of techniques have been introduced to address it. Most of these techniques are focused on delay reduction in the game engine, or during video rendering and encoding. Only a handful of works consider network routing and/or resource management as discussed in section 2.2. In this chapter, we propose a method to optimally allocate data center resources to user-requested gaming sessions. The method considers the current status of the network in terms of communication delay, available computational resources in game servers and processing delay, and the genre of the requested game to select the appropriate server in the cloud for game execution as well as the network path for game data transfer within the data center. The goal of the optimization method is to minimize the delay while improving the QoE of game users. Since the method uses a global view of the network status as its input, a Software-Defined Network (SDN) controller is used to execute the method. The SDN controller is in charge of deciding on the forwarding strategy for the streams of gaming data within the data center. In addition, we model the relationship between bit rate and perceived quality, and use this model in our heuristic method. This model is essential since a reduction in the bit rate can have an adverse effect on the QoE.

3.2 Proposed Framework

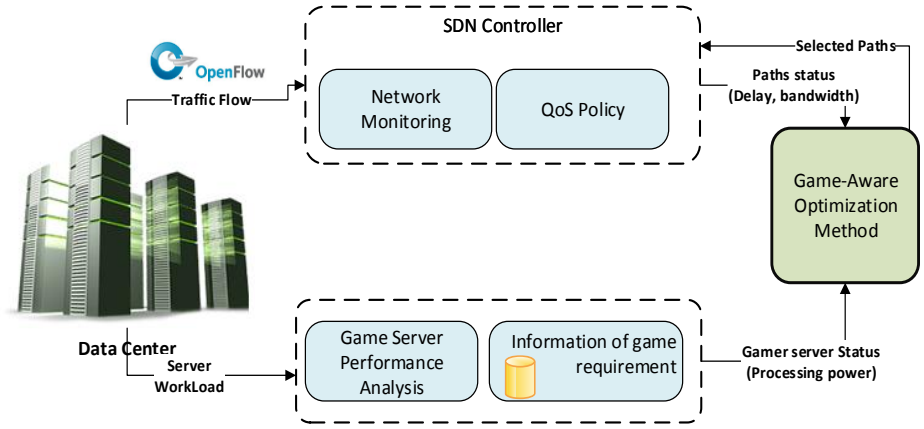


Figure 4.Game-Aware Optimization Method

In Figure 4, we present the high-level architecture of the system that supports our method. Our goal is to near-optimally assign games to servers in the cloud and select the best communication path within the data center for a game session's data streams. Hence, to minimize the end-to-end overall delay within the data center, the SDN controller periodically monitors the latency and available bandwidth on each link using the OF protocol. The game server performance analysis module monitors and analyzes the performance of the game servers in terms of available processing resources. Also, the data requirement for each game is stored in an auxiliary database. Having network-related information, server-related information, and games requirements, the optimization method makes decisions on near-optimally assigning game

servers to gaming sessions and selecting the best communication path within a cloud gaming datacenter.

3.3 User utility and processing delay models

Cloud games possess diverse requirements when it comes to the effect of delay on QoE. Hence, in the next two sections, we will:

- Conduct a measurement study to derive a utility model of the delay sensitivity of three cloud games that we will use in our evaluation, and
- Define a parametric model to describe the processing delay of a game based on its processing requirements.

3.3.1 Utility Model

Game delay sensitivity is dependent on various factors, one being the game genre [9]. Game genre refers to the common style or characteristics of a set of games, e.g. perspective, gameplay, interaction, objective, etc. For example, First Person Shooter (FPS) games are more sensitive to delay compared to Real Time Strategy (RTS) ones. Hence, we must take this property into account to optimally assign gaming sessions to servers and communication paths.

Towards this goal, we conduct a study on the impact of video bit rate on the quality perceived by the gamer for four representative games: Assault Cube, Madden, Warcraft3 and Magicka. These games belong to four popular game genres: FPS, sport/Role-Playing Game (RPG), RTS, and ARPG respectively. We derive utility functions that model the relationship between bit rate and two objective QoE metrics: Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) index.

The utility models obtained from this study apply only to the measured games. However, since the motion and scene complexity characteristics tend to be similar among games within a genre[39], the same model derivation method can be applied to other games in these measured games' genres.

We perform the study using GamingAnywhere (GA) [27], an open source cloud gaming system. We have set up the GA server and client on two machines using windows 7/enterprise that are equipped with Intel 3.4 GHz and 8 GB RAM. To perform our measurements, we capture the video output of five minutes of game play for each game using FRAPS [64]. We calculate PSNR and SSIM index as objective measures of the normalized gamers' quality perception. The results of the measurements are plotted in Figure 5. We tried many candidate functions such as $x^2, \log x, 1/x, \sqrt{x}, x^3$ to model the data, and have found that a sigmoid function can best fit our results[65], [66]. In fact, the sigmodal (s-shaped) function is the most commonly used single criteria utility function for real-time applications such as Video Streaming, Teleconferencing, and Voice over IP (VoIP) [66], [67]. We report the mean square error between the fitted curve and collected data for each game in Table 1.

Therefore, we can derive a parametric utility model, U(r), corresponding to the sigmoid function fitted to the data set of the three games plotted in Figure 5, where l and k are model parameters derived through regression and r is the bit rate (see equation 1). Also, we denote (α_i) as a priority factor obtained from the derivative of the utility function U(r) (see equation 2). Hence, α_i represents the rate of change of the utility curve, or how quickly the PSNR and SSIM deteriorates or improves when the bit rate is decreased or increased respectively for a

particular game. All the notations that are used in the equations of the remaining sections of this chapter are described in Table 2.

$$U(r) = l/(1 + e^{-kr}) \tag{3.1}$$

$$\alpha_i = \left[\frac{dU(r)}{dr}\right]_{r=rb_i} = [lke^{-kr}/(1 + e^{-kr})^2]_{r=rb_i} \tag{3.2}$$

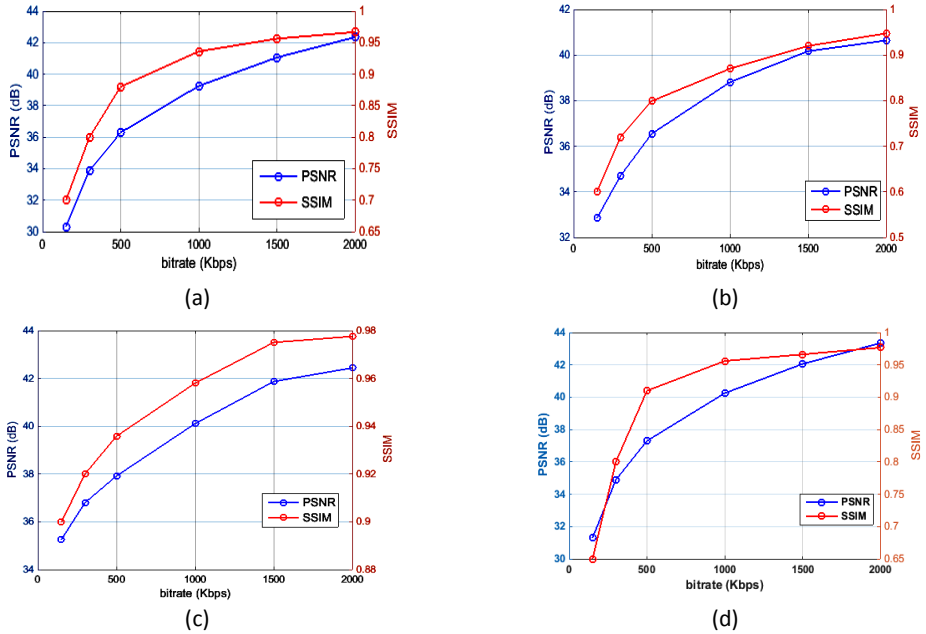


Figure 5.PSNR (dB) and SSIM values versus bitrate (Kbps) for three games (a) Assault Cube, (b)Madden, (c) Warcraft, (d) Magicka

Table 1.Mean Squared error values of different games

	Assault Cube	Madden	Warcraft	Magicka
GENRE	FPS	RPG	RTS	ARPG
PERSPECTIVE	1st	3rd(linear)	Omnipresent	3rd(isometric)
PSNR	.9764	.7649	.7585	.9511
SSIM	.9258	.8362	.6758	.6025

3.3.2 Processing Delay Model

Processing delay is the difference between the time the server receives a user's command and the time the server responds with a corresponding rendered frame.

We define processing delay as $d_{p_{ij}}(rp_i)$, where i refers to a particular gaming session executing in the cloud, j refers to a gaming server, and rp_i is the processing requirement of gaming session i. rp_i is considered to be any type of server resource e.g. CPU, GPU, bandwidth, Number of VMs etc.

Since measurement studies in [34] show that the processing delay of a gaming session can be modeled using a sigmoid function that accounts for the amount of resources available on the j^{th} server allocated to run the i^{th} gaming session with processing requirement of rp_i, hence $d_{p_{ij}}(rp_i)$ can be modeled as follows, where the processing resource requirement of game i and υ_i, σ_i and τ_i are model parameters that are derived from regression.

$$d_{p_{ij}}(rp_i) = \upsilon_i/(1 + \sigma_i e^{-\tau_i rp_i}) \quad (3.3)$$

Table 2.Table of Notations

Notation	Description	Notation	Description
U(r)	Utility function	N_p	number of available paths
i	Gaming session index	j	Gaming server index
k	Available paths index	Dmax	Maximum tolerable delay
$d_{p_{ij}}$	Processing delay of game i servers by game server j	c_{s_j}	Capacity of server j
$d_{n_{ijk}}$	Network delay of path k	rb_i	Required bandwidth of game i
α_i	Weight factor of network delay for game i	BW_{Max}	Maximum bandwidth of path k
β_i	Weight factor of processing delay for game i (1- α)	x_{ij}	binary variable for selection of a game server
N_s	number of active game servers	z_{ijk}	binary variable for selection of a path
N_g	number of gamers	Z_{LR}	Lagrange relaxation of O_{total}

λ	Lagrangean multipliers parameter	δ_{ijk}	$z_{ijk} - x_{ij}$
φ^t	subgradient step size	γ_t	Subgradient scalar (between 0 to 2)
t	Number of iterations	z^*_{LR}	lower bound value
ρ	Number of available ports on each switch	rp_i	processing resource requirement
υ_i, σ_i, τ_i	Processing delay model parameters	k, l	Utility function model parameters
O_{total}	weighted average of the total network and processing delay		

3.4 Optimization Problem Formulation

As discussed in Section (3.1), we aim to propose an optimization method that considers the processing delay, network delay, and game genres to allocate the best game servers and connecting path to each game session. We refer to this approach as the Game Routing Optimization (GRO) method. We base our model on a fat-tree network architecture for data center. We consider the data center network architecture as a graph where game servers and switches (core, aggregation and access) are represented as nodes and connecting paths as links in the graph.

We define an objective function to determine which paths and servers can minimize the overall delay associated with all gaming sessions running on the datacenter. The objective function consists of the weighted average of the total network and processing delay. Our goal is to minimize (3.4).

$$O_{total} : \sum_i^{N_g} \sum_j^{N_s} ((\beta_i \, d_{p_{ij}} x_{ij} + (\alpha_i) \sum_k^{N_p} d_{n_{ijk}} \, z_{ijk})) \qquad (3.4)$$

Subject to:

ai. $\sum_i^{N_g} rp_i x_{ij} \leq C_{s_j} \quad \forall\, i \in \{1, \dots, N_g\}, \forall\, j \in \{1, \dots, N_s\}$

aii. $\sum_i^{N_g} z_{ijk} = 1 \quad \forall\, j \in \{1, \dots, N_s\}, \forall\, k \in \{1, \dots, N_p\}$

aiii. $\sum_{j}^{N_s} x_{ij} \leq 1 \quad \forall i \in \{1, \dots, N_g\}$

aiv. $\sum_{i}^{N_g} \sum_{k}^{N_p} rb_i z_{ijk} \leq BW_{Max} \quad \forall j \in \{1, \dots, N_s\}$

av. $z_{ijk} \leq x_{ij} \; \forall i \in \{1, \dots, N_g\}, \forall j \in \{1, \dots, N_s\}, \forall k \in \{1, \dots, N_p\}$

avi. $\sum_{j}^{N_s} d_{p_{ij}} x_{ij} + \sum_{k}^{N_p} d_{n_{ijk}} z_{ijk} < D_{Max} \quad \forall i \in \{1, \dots, N_g\}$

avii. $z_{ijk} \in \{0,1\} \quad \forall i \in \{1, \dots, N_g\}, \forall j \in \{1, \dots, N_s\}, \forall k \in \{1, \dots, N_p\}$

aviii. $x_{ij} \in \{0,1\} \quad \forall i \in \{1, \dots, N_g\}, \forall j \in \{1, \dots, N_s\}$

In (3.4), we define two binary decision variable x_{ij} and z_{ijk}. Binary variable z_{ijk} is equal to 1 if the k^{th} path is chosen among the N_p available paths and 0 otherwise. Also, binary variable x_{ij} will take value 1 if j^{th} server is selected to host the i^{th} gaming session, and 0 otherwise.

Constraint (ai) ensures that the total allocated processing resource requests on a server cannot exceed its maximum processing capacity. Constraint (aii) ensures that each gaming session is routed exactly through a single path, and constraint (aiii) indicates that each gaming session is hosted at most on a single server. Constraint (aiv) guarantees that maximum path bandwidth capacity is not exceeded. Constraint (av) ensures that there exists one or more paths to transfer the data of gaming session i on game server j. Constraint (avi) restricts the total amount of delay (processing and network delay) to the maximum tolerable delay (Dmax) within the datacenter.

In sections (3.2.1) and (3.2.2), we provided a model for the calculation of delay processing ($d_{p_{ij}}(rp_i)$). Also, we provided the utility function ($U(r)$) to measure the priority factor (α_i) for individual games.

For the network delay, we proposed the following method to compute the network delay ($d_{n_{ijk}}$) using an SDN controller. Since the controller sends a message to each of the two switches directly connected to each

other by that link and measures the RTT pertaining to each switch, i.e. RTT_{jk} and RTT_{jk+1}. Having RTT_{jk+1} and RTT_{jk}, d_{jk} can then be calculated using equation (3.5).

$$d_{jk} = t_a - t_s - 1/2(RTT_{jk+1} - RTT_{jk}) \quad (3.5)$$

where t_a and t_s denote arrival and send time respectively.

The objective function of (3.4) presents two splittable flow problems. Hence, the resolution of (3.4) is an NP hard problem [68]. To address the computational complexity issue of (3.4), we will propose a time-efficient heuristic method for the calculation of O_{total} in Section (3.5).

3.5 Proposed heuristic method

A significant number of large-scale optimization problems can be solved more easily if the complex constraints are removed. One of the methods that are commonly used for the elimination of these constraints is the Lagrangean Relaxation (LR) method, which eliminates the set of constraints that impose computational complexity on general integer problems. Once these constraints are eliminated, the Lagrange multiplier is introduced to the objective function. The Lagrange multiplier is used to penalize violations of the constraints. The process of updating the penalty parameters will continue until convergence is reached [69]. In practice, the new problem resulting from LR is simpler than the original objective function through the construction of the lower bounds [70].

In our problem formulation, constraint set (av) represents the relation between two variables x and z. By removing constraint set (av), the problem can be converted to the form of known problems; therefore, the solution can be easier to educe. To do so, in section 3.5.1, we propose the Game Routing Heuristic (GRH) method based on the Lagrangean

Heuristic approach. We dualize the constraint set (av) to generate the lower bounds. In addition, the subgradiant optimization method can be employed to get the upper bounds.

3.5.1 Lagrange relaxation of O_{total}

By dualizing constraint (av) with the Lagrange multiplier set λ, our objective function can be expressed as follows:

$$Z_{LR}(\lambda) = \min(\sum_i^{N_g} \sum_j^{N_s} \sum_k^{N_p} (\alpha_i d_{n_{ijk}} + \lambda_{ijk}) z_{ijk} + \sum_i^{N_g} \sum_j^{N_s} (\beta_i d_{p_{ij}} - \sum_k^{N_p} \lambda_{ijk})\, x_{ij}) \quad (3.6)$$

Subject to:

bi. $\sum_i^{N_g} rp_i x_{ij} \leq C_{s_j} \quad \forall j \in \{1, \dots, N_s\}$

bii. $\sum_i^{N_g} z_{ijk} = 1 \quad \forall j \in \{1, \dots, N_s\}, \forall k \in \{1, \dots, N_p\}$

biii. $\sum_j^{N_s} x_{ij} \leq 1 \quad \forall i \in \{1, \dots, N_g\}$

biv. $\sum_i^{N_g} \sum_k^{N_p} rb_i z_{ijk} \leq BW_{Max} \quad \forall j \in \{1, \dots, N_s\}$

bv. $\sum_j^{N_s} d_{p_{ij}} x_{ij} + \sum_k^{N_p} d_{n_{ijk}} z_{ijk} < D_{Max} \quad \forall i \in \{1, \dots, N_g\}$

bvi. $z_{ijk} \in \{0,1\} \quad \forall i \in \{1, \dots, N_g\}, \forall j \in \{1, \dots, N_s\}, \forall k \in \{1, \dots, N_p\}$

bvii. $x_{ij} \in \{0,1\} \quad \forall i \in \{1, \dots, N_g\}, \forall j \in \{1, \dots, N_s\}$

Given that the Lagrange relaxation problem consists of two separate terms depending on x_{ij} and Z_{ijk}, we decompose $Z_{LR}(\lambda)$ into two separate problems for the purpose of simplification.

The first sub-problem, which only considers binary variable Z_{ijk}, can be expressed as follows:

$$Z_{LR1}(\lambda) = min \sum_i^{N_g} \sum_j^{N_s} \sum_k^{N_p} (\alpha_i d_{n_{ijk}} + \lambda_{ijk}) z_{ijk} = \sum_j^{N_s} \sum_k^{N_p} \{min \sum_i^{N_g} (\alpha_i d_{n_{ijk}} + \lambda_{ijk})\, z_{ijk}\} \quad (3.7)$$

Subject to:

ci. $\sum_j^{N_s} \sum_k^{N_p} z_{ijk}=1 \quad \forall\, i \in \{1, \dots, N_g\}$

cii. $\sum_i^{N_g} \sum_k^{N_p} rb_i z_{ijk} \;\leq BW_{Max} \quad \forall\, j \in \{1, \dots, N_s\}$

ciii. $z_{ijk} \in \{0,1\} \quad \forall\, i \in \{1, \dots, N_g\}, \forall\, j \in \{1, \dots, N_s\}, \forall\, k \in \{1, \dots, N_p\}$

Constraint (ci) can be considered as a Generalized Upper Bound (GUB) constraint such that the $j^{th}k^{th}$ Lagrangean problem consists of N_sN_p multiple choice problems and can be solved in a time proportional to N_sN_p where $i = \text{argmin}\{\alpha_i d_{n_{ijk}} + \lambda_{ijk}\}$ for $j = 1, \dots, N_s$, $k = 1, \dots, N_p$ and $z_{ijk} = 1$ [71]. The second sub-problem is obtained by dualizing constraint (v), which consists of the term x_{ij}, and is given as follows:

$$Z_{LR2}(\lambda) = min \sum_i^{N_g} \sum_j^{N_s} (\beta d_{p_{ij}} - \sum_k^{N_p} \lambda_{ijk})\, x_{ij} \tag{3.8}$$

Subject to:

di. $\sum_i^{N_g} rp_i x_{ij} \leq C_{s_j} \quad \forall\, j \in \{1, \dots, N_s\}$

dii. $\sum_j^{N_s} x_{ij}=1 \quad \forall\, i \in \{1, \dots, N_g\}$

diii. $x_{ij} \in \{0,1\} \quad \forall\, i \in \{1, \dots, N_g\}, \forall\, j \in \{1, \dots, N_s\}$

As sub-problem $Z_{LR2}(\lambda)$ does not satisfy constraint (ii), the optimal solution may include a game request that is served by a game server without using a path. This is obviously an infeasible solution for this problem. This constraint dissatisfaction can be removed by adding constraint set (av) into sub problem $Z_{LR2}(\lambda)$.

$$z_{ijk \leq x_{ij}} \begin{cases} \forall\, i \in \{1, \dots, N_g\} \\ \forall\, j \in \{1, \dots, N_s\} \\ \forall\, k \in \{1, \dots, N_p\} \end{cases} \longrightarrow \sum_i^{N_g} z_{ijk} \leq \sum_i^{N_g} x_{ij} \longrightarrow \sum_i^{N_g} x_{ij} \geq 1 \quad , \forall\, j \in \{1, \dots, N_s\}$$

Having constraint (v), sub-problem $Z_{LR2}(\lambda)$ can be revised as follows:

$$Z_{LR2}(\lambda) = min \sum_i^{N_g} \sum_j^{N_s} (\beta d_{p_{ij}} - \sum_k^{N_p} \lambda_{ijk})\, x_{ij} \tag{3.9}$$

Subject to:

ei. $\sum_{i}^{N_g} rp_i x_{ij} \leq C_{s_j} \quad \forall j \in \{1, \dots, N_s\}$

eii. $\sum_{j}^{N_s} x_{ij}=1 \quad \forall i \in \{1, \dots, N_g\}$

eiii. $x_{ij} \in \{0,1\} \quad \forall i \in \{1, \dots, N_g\}, \forall j \in \{1, \dots, N_s\}$

eiv. $\sum_{i}^{N_g} x_{ij} \leq 1 \ \forall i \in \{1, \dots, N_g\}$

ev. $\sum_{i}^{N_g} x_{ij} \geq 1 \quad \forall j \in \{1, \dots, N_s\}$

By considering constraint (ei), sub-problem $Z_{LR2}(\lambda)$ can be considered as a 0-1 Knapsack problem with a slight difference in coefficients. As explained in [72], it can be solved in time proportional to $N_s \sum_{j}^{N_s} C_{s_j}$.

3.5.1.1 DETERMINING λ

A subgradient optimization algorithm is used to iteratively adjust the value of the Lagrange multiplier λ [73] to find the best or near best lower bound. In fact, the subgradient optimization method is particularly appropriate to large-scale or non-differentiate problems with decomposition techniques. Since we consider our objective function as a piece-wise linear case, we apply the subgradient method to maximize the value of the lower bound in our problem, which is the dual objective function (Z_{LR}). Our proposed algorithm is detailed in Algorithm 3.1.

ALGORITHM 3.1: Subgradient Optimization Algorithm

1: Inputs:
λ_0: initial value for Lagrangean multipliers
t: number of Iterations
z^*_{LB}: lower bound value
V_{UB}: upper bound value
γ_0: scalar parameter
έ: error index

2: Initialize $\lambda_0 = 0$,t = 0, $t_{max} = 100$, $z^*_{LB} = -\infty$, $V_{UB} = +\infty$, $\gamma_0 = 2$, έ=+∞,

3: while($t \leq t_{max}$ && $\acute{\varepsilon} \geq 0.0001$) do
// computing lower bound

4: *Solve* $Z_{LR1}(\lambda_t)$, $Z_{LR2}(\lambda_t)$, *set* $\overline{X}_t \leftarrow$ *output* $Z_{LR2}(\lambda_t)$ *// feasible solution* $\overline{X}_t \in \{x_{ij} | i \in \{1, \dots, N_g\}, \forall j \in \{1, \dots, N_s\}\}$
Compute $Z_{LR}(\lambda_t) = Z_{LR1}(\lambda_t) + Z_{LR2}(\lambda_t)$
Set $z^*_{LB} = max\{ z^*_{LB}, Z_{LR}(\lambda_t)\}$
// computing upper bound

5: *Solve* O_{total} $\forall x_{ij} \in \overline{X}_t$, *set* $\overline{Z}_t \leftarrow$ *output* O_{total} *// the optimal value of* $\overline{Z}_t$ *can be found by setting* $\overline{Z}_{ijk}$*=1 where* $i = argmin\{\alpha_i d_{n_{ijk}}\}$ *for* $j = 1, \dots, N_s$, $k = 1, \dots, N_p$.
Compute $\overline{O}_{total} = O_{total}(\overline{X}_t, \overline{Z}_t)$
Set $V_{UB} = min\{ V_{UB}, \overline{O}_{total}\}$

6: $\|\delta_{ijk}\| = z_{ijk} - x_{ij}$ *// positive scalar called step size in the* t^{th} *iteration based on Fisher's formula[Fisher. 1985]*
Calculate $\varphi^t = \frac{\gamma_t \ (V_{UB} - z^*_{LB})}{(\sum_i^{N_g} \sum_j^{N_s} \sum_k^{N_k} \delta_{ijk})^2}$
$\lambda^{t+1}_{ijk} = max\left\{\lambda^t_{ijk} + \varphi^t \delta^t_{ijk}, 0\right\}$ // $\forall i \in \{1, \dots, N_g\}, \forall j \in \{1, \dots, N_s\}, \forall k \in \{1, \dots, N_p\}$
$\acute{\varepsilon} = | V_{UB} - z^*_{LB}|$
$t = t + 1$

7: End while;

3.6 Experimental setup and evaluation

3.6.1 Experimental set up

In this section, we evaluate the performance of the proposed method (i.e. GRH) and compare it to the optimization method of Section (3.4) (i.e. GRO). The simulations are run on a 3.4 GHz Intel workstation with 8 GB of RAM.

The proposed algorithm is coded in Matlab using the convex optimization package (CVX) [74]. To simulate the data center network architecture, we ran our experiments on an Ubuntu version 14.4 box and a Mininet emulator on the Oracle virtual box version 4.3. The Mininet emulator enabled us to create a realistic network experiment with OpenFlow and SDN. We implemented a fat-tree based architecture as a data center network. A collection of switches and servers are built within a pod. It is worth noting that in the fat-tree architecture, the numbers of required switches and servers in each pod are dependent on the number of ports in each switch. For example, if each switch has ρ ports, the Data Center Network (DCN) consists of ρ pods and each pod has $\rho/2$ access and aggregation switches, and $\rho^2/4$ core switches and servers. In our experiment, the DCN is controlled by an OpenFlow SDN controller deployed using the POX controller libraries [75]. The application running on the controller manages network flows and informs the open virtual switches (vSwitches) where to send the packets.

In the first experiment, we aim to evaluate the performance of the heuristic algorithm. Hence, we consider two different scale problem instances as follow:

Class I problems, where each vSwitch has 4 available ports so that each pod consists of 4 game servers (i.e. 16 game servers in total) and 4 core switches. While the number of game servers is fixed, we consider three different scenarios in terms of the number of gaming sessions, as shown in Table 3.

Class II problems, where each vSwitch has 10 available ports so that each pod consists of 25 game servers (i.e. 250 game servers in total) and

110 switches (core, aggregation and ToR switches). The number of gaming sessions is shown in Table 3.

In each scenario, users are playing two different games. The first half of the users are playing Assault Cube and the second half of the users are playing Warcraft. These games belong to different genres, FPS and RTS, respectively, with noticeable delay sensitivity. The details of the network traffic parameters for these games are presented in

Table 4 and the initial values of λ_0, t, γ_0 and t are mentioned in Algorithm 3.1.

Table 3.Details of three different scenarios

	Scenario		Number of Gaming Sessions	Gaming Session Ratio to Gaming Server
Class I Problem	1	Small Size	40	2.5
	2	Medium Size	80	5
	3	Large Size	160	10
Class II Problem	1	Small Size	625	2.5
	2	Medium Size	1250	5
	3	Large Size	2500	10

Table 4.Network Traffic Parameters of two games

Games	Genre	Upstream traffic		Downstream traffic	
		Mean Packet Size (bytes)	Mean Inter-Departure time (ms)	Mean Packet Size (bytes)	Mean of Inter-Departure time (ms)
assault cube	FPS	35.03	2.4	1,108.59	1.6
Warcraft	RTS	59.83	1.5	674.38	2.1

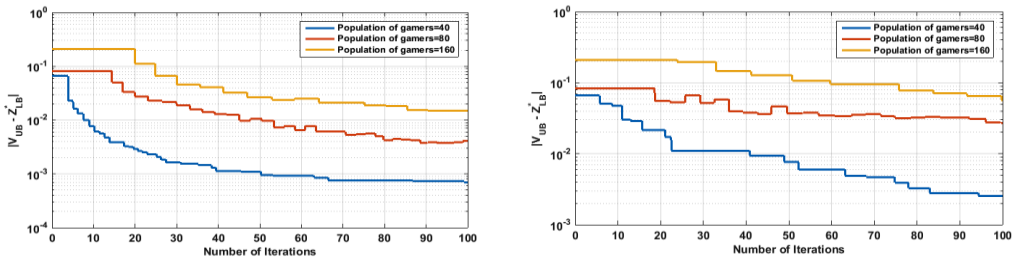
3.6.2 PERFORMANCE EVALUATION

In Tables 5, 6, and 7, we summarize the results of the Class I and Class II problems for the small, medium, and large scenarios respectively. We compare the algorithm presented in Section (3.3) (i.e. GRO) with the one proposed in Section (3.5) (i.e. GRH). For the GRO method, we measure the time required to reach the optimal solution for each problem instance. For GRH method, we specify the iteration

number that yielded the best solution, the average computational time per iteration, and the time required to reach the best solution. These results highlight the advantage of relieving some of the constraints of O_{total} using the proposed heuristic method. The best solution is reached using the GRH method far faster than the optimal solution is obtained using the GRO approach.

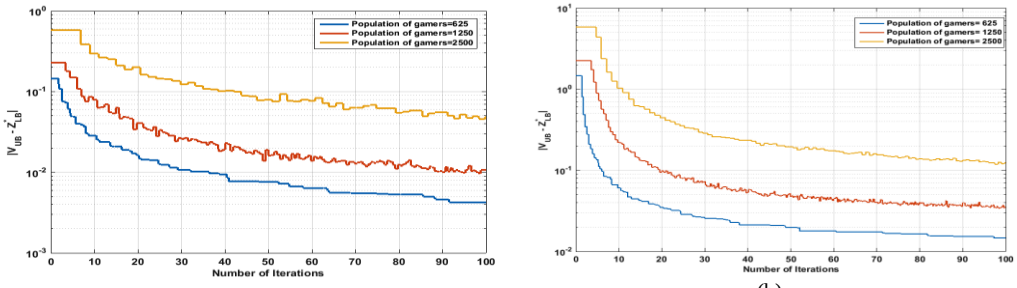
The computational results for the Class II problems instances, summarized in Table 5, Table 6 and Table 7 indicate that the GRH method is capable of dealing with large-scale problems in reasonable computational time.

Furthermore, for the GRH method, Figure. 6a and Figure. 6b show the semi-logarithmic convergence of the lower bound to a feasible sub-optimal solution with an adaptive and fixed step size for the Class I problems. The feasible solution is obtained after 17, 19, 24 iterations for scenarios 1, 2, and 3. However, more accurate solutions are obtained as the number of iterations increase.



(a) (b)

Figure. 6. The value of $|V_{UB} - z^*_{LB}|$ versus the number of iterations (t) for small, medium and large population of gamers, (a) Adaptive step size, (b) Fixed step size (1/t)



(a) (b)

Figure 7. The value of $|V_{UB} - z^*_{LB}|$ versus the number of iterations (t) for small, medium and large population of gamers, (a) Adaptive step size, (b) Fixed step size (1/t)

Table 5.Computational results for the small size Class I and II problems

	GRH			GRO
	Best solution iteration number	Total CPU time	CPU time per iteration	Total CPU time
Class I Problems	17	0.23	0.01	0.63
Class II Problems	31	3.19	0.102	8.67

Table 6.Computational results for the medium size Class I and II problems

	GRH			GRO
	Best solution iteration number	Total CPU time	CPU time per iteration	Total CPU time
Class I Problems	19	0.34	0.02	1.52
Class II Problems	38	7.25	0.1907	32.49

Table 7.Computational results for the large size Class I and II problems

	GRH			GRO
	Best solution iteration number	Total CPU time	CPU time per iteration	Total CPU time
Class I Problems	24	1.50	0.07	11.08
Class II Problems	47	17.64	0.371	130.34

In addition, the accuracy of the solutions is completely dependent on the size of the population of gamers. For example, in our experiment with $t_{max} = 100$, the problem with the smallest size (scenario 1) reached the highest accuracy solution in comparison with scenario 2 and scenario 3. By comparing Figures 6a and 6b, we can conclude that V_{UB} will converge faster to z^*_{LB} with adaptive step size. Moreover, the adaptive step size method outperforms the fixed step size method in terms of accuracy.

Figure 7a and Figure 7b show the convergence results for the Class II problems. The adaptive step size method outperforms the fixed step size method by achieving a higher accuracy with a fixed number of iterations. However, the fixed step size method's curves decrease faster than those of the adaptive step size method. Moreover, it is shown that

the increase in the number of players drastically impacts the accuracy of the solution.

We also measure the overall delay (network delay + processing delay) and delay variation (jitter) for Class I and II problems. Hence, we consider the following three typical resource allocation strategies in data centers:

1) Server-centric, in which the game server with the lowest processing delay is selected. The server centric approach focuses on minimizing processing delay. For this strategy, the weighted factor α is set to 0.
2) Network-centric, in which the game server with the lowest network delay for its connecting path would be selected, strives to minimize network delay. For the network-centric approach where the SDN controller has a centralized view of the current network conditions, α is set to 1.

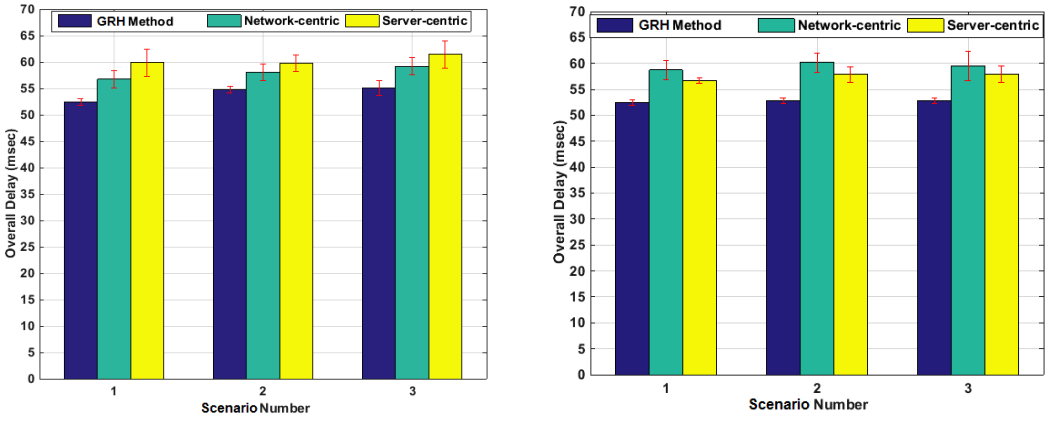


Figure 8.Overall delay and Jitter for Class I problem instances a)Assault Cube b)Warcraft

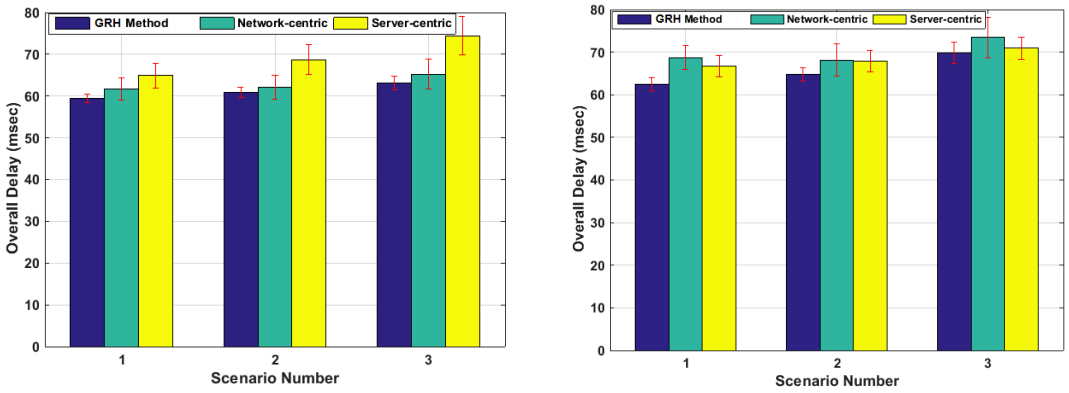


Figure 9.Overall delay and Jitter for Class II problem instances a)Assault Cube b)Warcraft

3) Proposed Method, where we calculate the weighted factor α for the two games in our experiment based on equation (2). The value of α was calculated as 0.67 and 0.253 for Assault Cube and Warcraft, respectively.

For the Class I problem, Figure 8a shows the average overall delay experienced by players playing Assault Cube. The proposed method results in almost 10% and 12% lower delay compared to the network-centric and server-centric methods, respectively. Also, the average delay variation experienced by players using our proposed method is almost 13% and 14% less than that experienced in the network-centric and server-centric approaches.

Moreover, Figure 8b shows that the average delay experienced by gamers who are playing Warcraft using our proposed method is almost 12% and 9% less than the overall delay experienced in the network-centric and server-centric methods, respectively. In addition to the significant improvement of the overall delay, Figure 8b shows that the proposed method brings about around 16% and 11% reduction in the delay variation compared to those of the network-centric and server-centric methods, respectively.

The results for the Class II problems are shown in Figures 9a and 9b. The proposed method still outperforms the server-centric and network-centric methods by producing mostly lower average delay and delay variation. Although the delay results for the network centric approach in scenario 2 when the users are playing Assault Cube are comparable with the proposed method, the average delay variation experienced by players using the proposed method is almost one third of that of the network-centric method.

Since we defined the utility model to prioritize processing and network resource assignment for each user to minimize the overall delay, there is a concern about fairness in resources allocation. Hence, for the final experiment, we investigate the fairness of the proposed method compared to the server-centric and network-centric methods. We focus on evaluating the fairness related to QoS parameters e.g. network delay, delay variation, etc. It is well understood that the different levels of delay between gamers and game servers can negatively impact the fairness [76]. We use the fairness index proposed in [76] to assess the fairness among multiple gamers competing for higher resources using our system.

Equation (9) shows the kill rate experienced by gamers where d_i and p_i are the overall delay and packet loss rate experience by i^{th} gamer. The unfairness between two groups of gamers is calculated as the difference between their kill rate indices.

$$\mu(d_{overall}) = \frac{\sum_1^{N_g} p_i d_i}{N_g} \qquad (10)$$

We measured the unfairness index $\mu(d_{overall})$ for the three scenarios of table 3 (Class I problem) while applying server-centric, network-centric and the proposed methods for server and/or network path selection.

A lower difference between the unfairness index of games shows that there is a smaller difference between the loss rates of gamers in different groups of games; hence the resources are better allocated to gamers. The results in Figure 10 show that gamers in different groups have a smaller difference in kill rate when our proposed method (GRH method) is used.

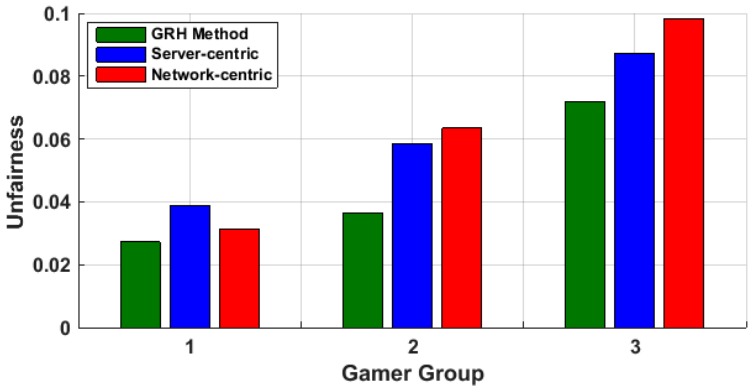


Figure 10.Unfairness for three different of gamers

3.7 SUMMARY

As the major computational part of game processing is performed on the DCs, a properly designed DC can provide high quality games to end-users and reduce costs. In this section, we focused on the DC's network resource management in a centralized fashion using SDN. We presented a novel optimization-based method for near-optimally assigning game servers to gaming sessions and selecting the best communication path within a cloud gaming datacenter. We proposed an optimization model that considers the type of requested games, current server loads, and current path delays to make a decision on which game server and communication path will minimize the delay within the DC. Due to the high computational complexity of the optimization method, we proposed an LR heuristic method to significantly reduce the computational complexity of the problem such that it can be practically implemented in the real data center using an OF controller. We evaluated the proposed method using three scenarios and the results

indicate that it can provide close to optimal solutions to the game server and path selection problem. Moreover, experimental results showed that the proposed method, on average, outperforms conventional algorithms (server-centric and network-centric) by, first, minimizing the overall delay and delay variation (jitter) within data centers and, second, having better performance compared to other existing solutions in terms of fair resource allocation among multiple competing players.

CHAPTER 4. SDN-enabled Game-aware Routing for Cloud Gaming Datacenter Network

4.1 Introduction

Tailoring the public cloud infrastructure to the specific requirements of cloud gaming is a relatively challenging task. Also, public cloud providers such as Amazon are not capable of supporting cloud gaming services with proper QoS provisioning. Hence, cloud gaming service provider companies (e.g. StreamMyGame, Giaki and OTOY) are investing in building their own infrastructure and proprietary platforms to facilitate the allocation of processing and computational resources while meeting the gamingspecific needs. In our previous chapter, we proposed a cloud path selection method to reduce end-to-end delay and delay variations (jitter). The method adaptively disperses the game traffic load among different network paths according to their end-to-end delays. The method uses a global view of the network status as its input and hence employs a Software Defined Network (SDN) controller to direct routing decisions for the streams of gaming data within the cloud. Although this method showed promising results by decreasing the delay experienced by users, it did not consider the specific characteristics of games to select the appropriate routing strategy. Game characteristics vary between different games, and to offer an optimized experience, a solution must consider game specifics. Delay sensitivity often varies based on the game genre. For example, First Person Shooter (FPS) games are more sensitive to delay compared to real-time strategy (RTS) games. So, the tolerable delay depends on game genre. Ideally, routing algorithms must resolve the best routing

strategy to satisfy certain Quality of Service (QoS) requirements. In QoS-based routing schemes, routes should be chosen based on features of the transmitted data flows, such as bandwidth requirement and delay sensitivity. There are two main goals that need to be achieved by the QoS-based routing algorithm. The first goal is to find a path that satisfies the QoS requirements. The second goal is to optimize the global network resource utilization. In this chapter, we propose a bi-objective optimization scheme that strives to jointly minimize the overall delay experienced by users and maximize bandwidth utilization within a data center by considering the requested games genres. The proposed scheme makes use of an Analytic Hierarchy Process (AHP) based Game Aware Routing (AGAR) method to find the initial feasible solution that establishes the optimal path between the core and Top of Rack (ToR) switches in a cloud data center, while taking game-specific characteristics into account.

4.2 Proposed Architecture Overview

We propose AGAR, a bi-objective optimization method that considers network status and game characteristics to select the appropriate network path between core and ToR switches for a gaming session request (see Figure 11). The proposed optimization scheme, as we will demonstrate in Section (4.2), cannot be solved in polynomial time for large instances of the problem. Therefore, we employ an iterative approach to relax the original problem and find the optimal solution. We use the Analytical Hierarchy Process (AHP) metaheuristic method to find the initial solution that we feed into the optimization scheme to select the best routing path in the cloud network. The goal of AGAR is to find the candidate paths that reduce the end-to-end delay experienced by gamers and improve their QoE based on game flow characteristics.

Since the optimization scheme uses a global view of the network status as its input, an SDN controller is used to decide on the forwarding strategy for the streams of gaming data within the data center.

Conventional SDN controllers usually find the optimal path in terms of different network criteria (e.g., shortest delay, least hop-count etc.) [77]–[79]. Conversely, in our proposed method, first, all possible paths from the requesting core switch to the ToR switches are identified by the controller. While this is an NP-problem, the graphs constructed based on the current architectures of data centers (e.g. Fat-tree [80] and VL2[81]) are simple, and the source (i.e. core switch) is fixed for all flows. Therefore, the problem of finding all possible paths can be solved using the algorithm described in [77]. Afterwards, the AHP method is utilized to find the candidate solution that will feed into the optimization scheme to find the optimal routing path for a gaming session request.

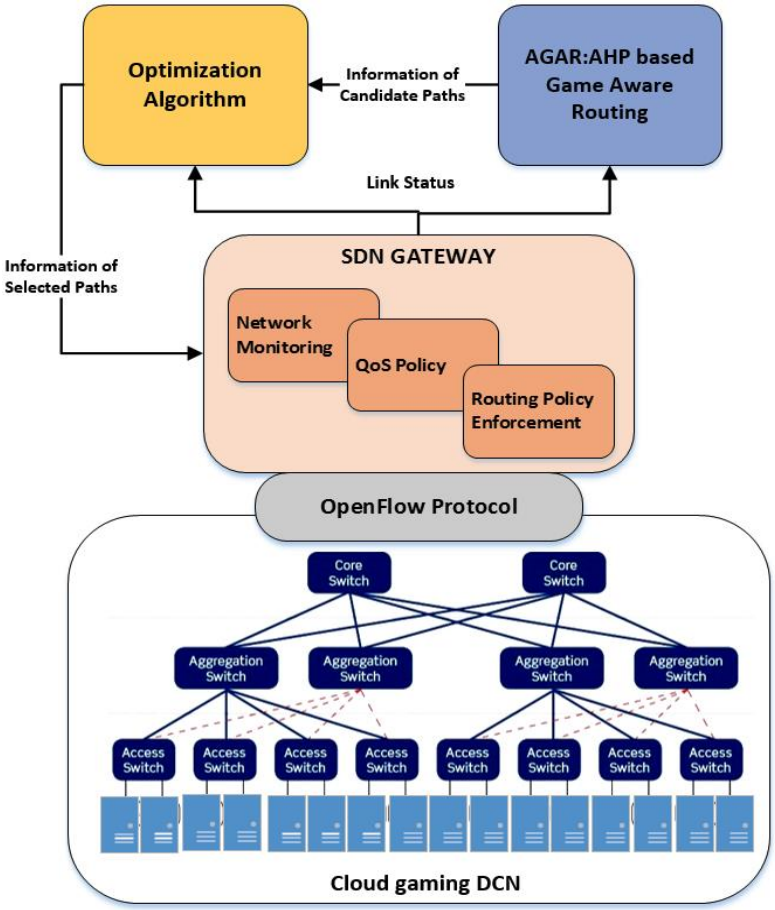


Figure 11. AHP based Game aware Routing Framework (AGAR)

4.3 Problem Formulation

The network topology of a data center can be modeled by a graph G(V,P), in which V indicates the set of nodes mapped to network (core, aggregation and access) switches, and P denotes the set of disjoint paths linking core switches to the game servers. Provided that there are M active gaming sessions in the data center, the index i ($1 \leq i \leq M$) represents i^{th} gaming session. Let j be the number of paths between the core switch and ToR where $1 \leq j \leq N$. Let φ_{ij} denote the network delay experienced by game session i if its traffic is routed through path j. the first objective

function aims to find the network paths that minimize the network delay experienced by the gaming session, so we define f_1 as follows:

$$f_1 = \sum_{i=1}^{M} \sum_{j=1}^{N} \varphi_{ij} x_{ij} \quad (4.1)$$

Subject To:

i. $\sum_{j=1}^{N} x_{ij} = 1, \forall i \in \{1, \dots, M\}$

ii. $\sum_{j=1}^{N} \varphi_{ij} x_{ij} \leq \varphi_{max}, \forall i \in \{1, \dots, M\}$

iii. $x_{ij} \in \{0,1\} \quad \forall i \in \{1, \dots, M\}, \forall j \in \{1, \dots, N\}$

Table 8.TABLE OF NOTATIONS

Notation	Description	Notation	Description
M	number of gaming sessions	N	number of available paths
i	Gaming session index	j	Gaming server index
j	Available paths index	φ_{max}	Maximum tolerable delay
x_{ij}	binary variable for selection of a path	U_i	utility function of gaming session i
β_{ij}	Amount of bandwidth assigned to game i	δ_{max}	maximum bandwidth capacity
p	Set of n possible disjoint paths	P_d	Network resource importance parameters for delay
P_b	Network resource importance parameters for bandwidth	b^{δ}	Set of residual bandwidth for each path
d^{φ}	Set of the inverse mean delay for each path	x_{ij}	binary variable for selection of a path
CI_b	Consistency Index	ρ_b	Preference Degree Index for bandwidth
ρ_d	Preference Degree Index for delay	M_{φ}^{d}	pairwise N × N comparison matrices for residual delay
M_{δ}^{b}	pairwise N × N comparison matrices for residual bandwidth	φ_{ij}	network delay experienced by game session i

In (4.1), we define a binary decision variable x_{ij} that is equal to 1 if game session i is served by the connecting path j, and 0 otherwise (Constraint iii). Only, a single path is assigned to each gaming session, and it is guaranteed by Constraint i, and the total amount of network delay is confined by constraint ii to a predefined delay (φ_{max}), i.e. a maximum tolerable delay within the data center as explained in [19].

φ_{max} depends on the genres of games executing within the data center. In addition to the network delay experienced by gaming sessions, we define the function f_2 to express the bandwidth resource utilization within the data center network.

Given that traffic flow of gaming session i is routed through the path j, the fraction of available bandwidth in path j dedicated to gaming session i is represented by β_{ij}. Hence, the bandwidth allocated to gaming session i using path j can be denoted as $\beta_{ij}x_{ij}$ where x_{ij} is a binary variable, as explained above.

Table 8 presents all notations used throughout this article. For each gaming session, a utility function (denoted by U_i for i^{th} gamer) is defined to capture the level of satisfaction of i^{th} gamer from receiving the bandwidth β_{ij}. It should be noted that this satisfaction corresponds to the quality of gaming perceived by the gamer.

In Section (3.2.1), we derived the utility functions of four popular game genres: FPS, sport/Role-Playing Game (RPG), RTS, and ARPG. These utility functions model the relationship between bit rate and objective QoE measured by two metrics: Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) index.

Therefore, the utility function of the proposed optimization problem can be obtained by summing up all of gamers' utility functions as follows:

$$f_2 = \sum_{i=1}^{M} U_i\left(\sum_{j=1}^{N} \beta_{ij}x_{ij}\right) \tag{4.2}$$

Subject To:

i. $\sum_{i=1}^{M} \beta_{ij}x_{ij} \leq \delta_{max} \quad \forall\, j \in \{1, \dots, N\}$

ii. $\sum_{j=1}^{N} x_{ij} = 1, \forall\, i \in \{1, \dots, M\}$

iii. $x_{ij} \in \{0,1\}, \forall\, j \in \{1, \dots, N\}, \forall\, i \in \{1, \dots, M\}$

The constraints ii and iii were described previously. Constraint i ensures that the total amount of bandwidth of the gaming sessions on a path j cannot exceed its maximum bandwidth capacity δ_{max}.

In addition to maximizing the overall bandwidth utilization, to prevent any game session from being subjected to starvation, we incorporate some level of fairness among the sessions by applying the logarithmic utility function. So we rewrite the objective function (4.2) as:

$$f_2 = \sum_{i=1}^{M} \log(\sum_{j=1}^{N} \beta_{ij} x_{ij}) \quad (4.3)$$

In this route selection problem within the data center, two objectives are simultaneously pursued, (4.1) to minimize the network delay experienced by the gaming session, and (4.2) to maximize the bandwidth utilization. Since the objective functions, f_1 and f_2, are not completely compatible, we introduce the optimization problem Z as follows:

$$Z: \text{Min}\{f_1\}, \text{Max}\{f_2\} \equiv \text{Min}\{f_1, -f_2\} \quad (4.4)$$

Subject To:

i. $\sum_{j=1}^{N} x_{ij} = 1, \forall\, i \in \{1, \dots, M\}$

ii. $\sum_{i=1}^{M} \delta_{ij} x_{ij} \leq \delta_{max}, \forall\, j \in \{1, \dots, N\}$

iii. $\sum_{j=1}^{N} \varphi_{ij} x_{ij} \leq \varphi_{max}, \forall\, i \in \{1, \dots, M\}$

iv. $x_{ij} \in \{0,1\} \quad \forall\, i \in \{1, \dots, M\}, \forall\, j \in \{1, \dots, N\}$

Decision variable x_{ij} turns our problem into an Integer Programming (IP) model. These problems are harder to solve compared to Linear Programming (LP) models because of their intrinsically combinatorial nature. In addition, large problem sizes make IP-based models computationally complex to solve. However, given constraints ii and iii, we can consider that Z consists of two separate generalized assignment problems that can be solved in proportional time. We use a standard branch-and-bound algorithm to solve for Z which requires the establishment of the lower bound solution. Although the lower bound

can be obtained by the relaxed model of Z, the two sub problems f_1 and f_2 contain binary variables that still makes them computationally costly. Therefore, for the problem Z, the computational complexity of iterative methods (e.g. subgradient) may be excessively high. More importantly, the tuning process to determine the sequence of step-lengths to update successive iterations is not a trivial task. Hence, we use the AHP metaheuristic to define the initial lower bound solution. A branch and bound method is used to ascend from the lower bound to an optimal solution of the generalized problem. In the next section we demonstrate the details of this approach.

4.3.1 Analytic Hierarchy Process (AHP)

The AHP is a structured method to deal with multi-criteria decisions. While a mix of qualitative, quantitative, and sometimes conflicting factors are taken into account to find the best solution among several alternatives, AHP enables the decision maker to find one that best meets problem requirements. Although various multi- criteria models, such as TOPSIS, grey rational analysis, and PROMETHEE II have been proposed to assess, prioritize, rank, and evaluate decision choices, most of these methods are inherently complex when factors with diverse and conflicting priorities are considered. The main difference between AHP and the other decision-making methods is that the factors are weighted to indicate that they are of different importance.

AHP splits a complex problem into a multi-leveled hierarchical process. The number of levels in the hierarchy is used to derive ratio-scaled measures for decision alternatives[82]. AHP generally includes four steps: First, define the problem and state the goal or objective. Second, define the criteria or factors that influence the goal and structure these factors into levels and sublevels. Third, perform paired

comparisons between factors using a comparison matrix containing calculated weights, ranked eigenvalues, and consistency measures. Finally, synthesize the ranks of alternatives until the final choice is made. In the following paragraphs, we detail how we applied these steps in the formulation of our proposed AHP method.

Step 1. Problem Statement

As it can be seen in Figure 11, the DCN is controlled by the SDN, and the OF protocol is used to make our proposed method adaptable to a variety of networks. In our method, the importance of network resources for a particular game genre is specified in the AHP quantitatively. Having these quantized values of network resources, the best path between alternative paths is selected and the results are fed into the optimization scheme of Section (4.2). We adopt the fat-tree 3-tier DCN architecture as our network topology.

To describe the algorithmic details of the proposed method, we first introduce some notations:

1. $G(V, E)$: Network topology digraph of a data center where V denotes the set of switches (including core, aggregation and ToR switches) and E denotes a set of links in the network.
2. $p = \{p_1, p_2, \dots, p_n\}$: Set of n possible disjoint paths from V_s: Source Vertex to V_d: Destination Vertex
3. P_d and P_b: Network resource importance parameters for a game genre corresponding to delay sensitivity and bandwidth requirement respectively.
4. $b^{\delta} = \{b_1^{\delta}, b_2^{\delta}, \dots, b_n^{\delta}\}$: Set of residual bandwidth for each path. We define residual bandwidth b_{i}^{δ} as the lowest available bandwidth on any of the links that belong to p_i.
5. $d^{\varphi} = \{d_1^{\varphi}, d_2^{\varphi}, \dots, d_n^{\varphi}\}$: Set of the inverse mean delay for each path (i.e., d_{i}^{φ} corresponds to the inverse of the mean delay on p_i).

Step 2. Define the criteria or factors that influence the goal

To build the routing method, we have to specify the most important factors that influence network resource management. In this work, we categorize these factors into network and application factors.

For the network factors, we consider residual bandwidth and delay on the network links. For the application factors, we consider the network resource requirements for different game genres. Game genres define games in terms of having a common style or set of characteristics, e.g. as defined in terms of perspective, gameplay, interaction, objective, etc. Authors in [39] show that the extents of motion and scene complexity vary noticeably among different game genres. A complex scene demands a higher amount of bandwidth compared to a scene with lower complexity. On the other hand, games with higher amounts of motion are more sensitive to delay variation. Accordingly, we summarize the network resource importance for various game genres in Table 9.

We then design Table 9, which maps the network resource importance qualifications used in Table 10 to discrete numeric values that can be employed in the AHP based decision making.

Step 3. Pairwise Comparison Matrix (PCM)

1. In AHP, the Pairwise Comparison Matrix (PCM) is used to calculate the priority vector of alternatives[82]. Having two main criteria, bandwidth and delay, we build two comparison matrices for the set of N candidate paths. Therefore, we define M_{δ}^{b} and M_{φ}^{d} as the pairwise $N \times N$ comparison matrices for residual bandwidth and inverse mean delay. The matrices are obtained as follows:

$$(M_{\delta}^{b})_{jk} = b_j^{\delta}/b_k^{\delta}\ ,\ (M_{\varphi}^{d})_{jk} = d_j^{\varphi}/d_k^{\varphi} \tag{4.5}$$

$$M_{\delta}^{b} = \begin{pmatrix} 1 & b_1^{\delta}/b_2^{\delta} & b_1^{\delta}/b_3^{\delta} & \dots & b_1^{\delta}/b_N^{\delta} \\ b_2^{\delta}/b_1^{\delta} & 1 & b_2^{\delta}/b_3^{\delta} & \dots & b_2^{\delta}/b_N^{\delta} \\ b_3^{\delta}/b_1^{\delta} & b_3^{\delta}/b_2^{\delta} & 1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_N^{\delta}/b_1^{\delta} & b_N^{\delta}/b_2^{\delta} & b_N^{\delta}/b_3^{\delta} & \dots & 1 \end{pmatrix} \quad (4.6)$$

$$M_{\mu}^{d} = \begin{pmatrix} 1 & d_1^{\varphi}/d_2^{\varphi} & d_1^{\varphi}/d_3^{\varphi} & \dots & d_1^{\varphi}/d_N^{\delta\varphi} \\ d_2^{\varphi}/d_1^{\varphi} & 1 & d_2^{\varphi}/d_3^{\varphi} & \dots & d_2^{\varphi}/d_N^{\varphi} \\ d_3^{\delta}/d_1^{\delta} & d_3^{\delta}/d_2^{\delta} & 1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_N^{\varphi}/d_1^{\varphi} & d_N^{\varphi}/d_2^{\varphi} & d_N^{\varphi}/d_3^{\varphi} & \dots & 1 \end{pmatrix} \quad (4.7)$$

Table 9.Game Network Resource Importance

GAME GENRE	PERSPECTIVE	NETWORK RESOURCE IMPORTANCE	
		DELAY SENSITIVITY (ρ_d)	BANDWIDTH REQUIREMENT (ρ_b)
First Person Shooter (FPS)/ Action Role-Playing Game/Action-adventure	First	High	Low
Action-Adventure/Action Role-Playing Game (ARPG)	Isometric	Medium-High	Low-Medium
Third Person Shooter/Role-Playing Game (RPG)	Linear	Low-Medium	Low-Medium
Trading Card Game/Real Time Strategy (RTS)	Omnipresent	Medium	Medium-High

We calculate the priority vectors of matrices M_{δ}^{b} and M_{φ}^{d} which correspond to the normalized Eigenvector of each one of them. We define ω_b and ω_d as the normalized Eigenvectors and λ_b, λ_d as the Eigenvalues of M_{δ}^{b} and M_{μ}^{d} respectively.

Table 10.RATING SCALE FOR IMPORTANCE INDEX

DEGREE OF IMPORTANCE	DEFINITION
1	Very High
0.8	High
0.6	Medium
0.4	Low
0.2	Very Low
0.1,0.3,0.5,0.7	Intermediate Values between the two adjacent degrees

2. We check the consistency of the pairwise comparison matrices using the test if $\lambda_{b\_max} = N$, then the pairwise comparison matrix M_{δ}^{b} is perfectly consistent. We calculate the Consistency Index (CI) and the Consistency Ratio (CR) to measure the consistency degree for two comparison metrics as follows [82]:

$$CI_b = \frac{\lambda_{b\_max} - N}{N-1}, CI_d = \frac{\lambda_{d\_max} - N}{N-1} \quad (4.8)$$

$$CR_b = \frac{CI_b}{RI}, CR_d = \frac{CI_d}{RI} \quad (4.9)$$

where RI is a Random Index scale. In this work, we use the RI scaled proposed by [83] as displayed in Table 11. The pairwise comparison metrics are accepted if the value of CR is smaller or equal to 0.1.

Table 11.Random Consistency values RI(N)

N	3	4	5	…	13	14	15
RI	0.5245	.0.8815	1.1086		1.5551	1.5713	1.5838

3. We define the game network resource importance indices ρ_b and ρ_d as the weights corresponding to the importance of bandwidth and delay, respectively, for a particular game genre. The importance of bandwidth and delay for a game genre is obtained from Table 9 and Table 10. The values of ρ_b, ρ_d should satisfy the following constraint:

$$\rho_b + \rho_d = 1 \quad (4.10)$$

where:

$$0 \leq \rho_b \leq 1, 0 \leq \rho_d \leq 1 \tag{4.11}$$

Using the priority vectors for the main network resource criteria, ω_b, ω_d and the game network resource importance indices ρ_b, ρ_d, we calculate the Preference Degree Index (PDI) for a candidate path as follows:

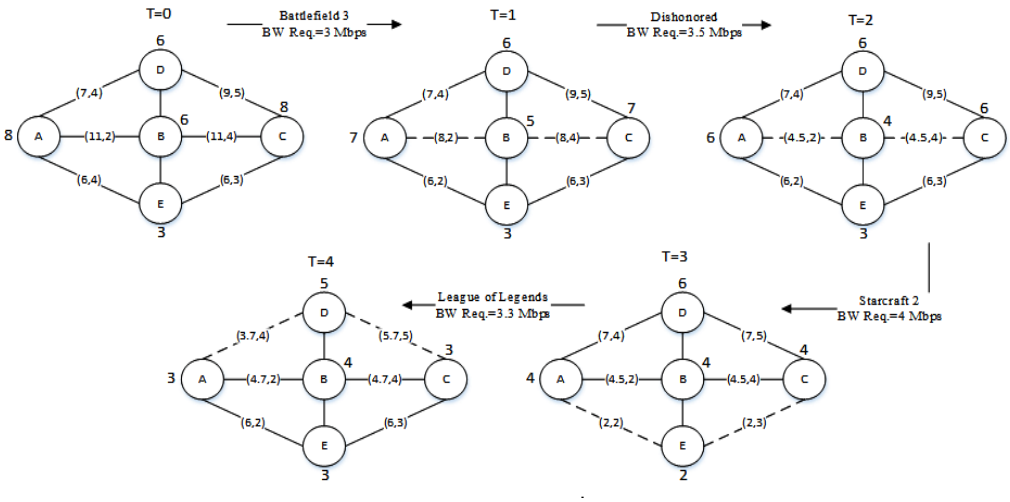


Figure 12.Toy Network Example

$$PDI_i = \rho_b * \omega_b + \rho_d * \omega_d \tag{4.12}$$

The PDI is a vector with N elements, each one corresponding to the preference of a candidate path. The path with the highest preference value is chosen as a routing path for the requested game flow. The entire process of AHP is summarized in Algorithm 4.1.

To further illustrate the inner workings of AHP, we present an example that details the process of route selection. Figure 12 shows a toy data center consisting of five OF-enabled switches. The flow entry capacity of each switch is marked next to it. Also, the pair of bandwidth and delay for each connecting link is presented using the format (b, d), where b corresponds to bandwidth and d corresponds

to delay. We assume that there are four game flow requests from games belonging to diverse genres, namely FPS, RPG, ARPG, and RTS, and three available routing paths with different characteristics. Note the game flow requests arrive in the same order as they are stated in the latter statement. Using Table 9 , and 10 the constraints of Equations (4.9) and (4.10), we set the game network resource importance indices ρ_b, ρ_d for FPS, ARPG, RPG, and RTS games as (0.147,0.853), (0.643,0.357), (0.29,0.71) and (0.65,0.345), respectively.

Table 12.Route Selection Results

Game Genre	Tested game	Bandwidth Requirement	PDI (Path1)	PDI (Path2)	PDI (Path3)	Selected Route
FPS	Battlefield 3	3	0.28	0.373	0.34	ABC
RPG	Dishonored	3.5	0.316	0.367	0.292	ABC
ARPG	Starcraft 2	4	0.312	0.334	0.353	AEC
RTS	League of Legends	3.3	0.433	0.342	0.224	ADC

The route section results are presented in Table 12, where we can see that the flow pertaining to the game with high delay sensivity is routed through the path with the lower delay. Moreover, the game with the high bandwidth requirement is more likely to be assigned to the path with more residual bandwidth.

ALGORITHM 4.1: ANALYTICAL HIERARCHY PROCESS

1:
Inputs:
$G(V, E)$: Data Center Topology Digraph
v_s: Source Vertex
v_d: Destination Vertex
Output: **Routing path from v_s and v_d**

2: $p = (G, W, v_s, v_d)$

3: Build the PCMs M_{δ}^{b}, M_{φ}^{d}

4: Calculate the normalized Eigenvectors and Eigenvalues for each matrix
Set ω_b, ω_d// as Eigenvectors

5: Measure the consistency ratio (CR) of the PCMs

6: If $CR > 0.1$ then
go to step 3

8: Establish the game network resource importance indices P_b, P_d for the requested game genere

7: Calculate the PDI for each candidate path i
$PDI_i = P_b * \omega_b + P_d * \omega_d$

9: Set the route with maximum preference value in the PDI vector as the routing path

4.4 Performance Evaluation

4.4.1 Experimental setup

In this section, we evaluate the performance of the proposed routing scheme. Our simulations are run on an Ubuntu version 14.4 (3.4 GHz Intel workstation) with 8 GB of RAM and the coding was developed using MATLAB version 8.4. To simulate the DCN architecture, we use a Mininet emulator on the Oracle virtual machine version 4.3. The Mininet emulator enables us to create a realistic network experiment with OF and SDN. We implement a fat-tree [80] based architecture where a collection of switches and servers are built within a pod. In the fat-tree architecture, the numbers of required switches as well as servers in each pod are dependent on the number of ports in each switch. For example, if each switch has k ports, the DCN consists of k pods and

each pod has $k/2$ access and aggregation switches, and $k^2/4$ core switches and servers. The DCN is controlled by an OF SDN controller deployed using the POX controller libraries [75]. The application running on the controller manages network flows and informs the open virtual switches (vSwitches) where to send the packets. In the experiments, we consider a network topology that consists of 20 vSwitches and 600 links. The OF table capacity of each vSwitch is 1000 and the maximum bandwidth of each link is 20 Mbps. While the number of switches and links are fixed, we study two scenarios with varying number of game flows running within the DCN: Scenario 1 with 100 game flows and Scenario 2 with 200 games flows. The game flows in both scenarios belong to four games: Battlefield 3, Dishonored, Starcraft 2 and League of Legends. These games belong to the following game genres respectively: FPS, Action-adventure, RTS, and Multiplayer online battle arena and have varying delay and bandwidth requirements. The details of the network traffic parameters for these games are presented in Table VI [37]. The flow requests arrive at random intervals (normal distribution) and are then assigned to a route by the SDN controller. In our simulation, we compare AGAR with the Equal Cost Multi-Path routing (ECMP) [84], [85], Hedera [86] and the Delay-Based Dijkstra (DBD) multi-path routing methods in data centers. For ECMP, the switch identifies routing path locally using a hash algorithm. In fact, flows distribute with an equal probability across all paths. On the other hand, for DBD, we employ the algorithm proposed in [77] in which routes with the least delay can be obtained by applying Dijkstra's algorithm and using delay on the links as the cost. And finally, for Hedera, a centralized scheduler collects flow information from the edge switches, performs scheduling every few seconds, and distributes flows.

4.4.2 Comparison method

To compare the proposed algorithm with other candidate methods, we employ the Kullback-Leibler divergence (KL-divergence) method so-called related entropy [87]. The KL divergence emanates from the field of information theory, and it is now accepted widely as a good measure of the difference between two probability distributions as follows [88], [89]:

$$D_{KL}(P||Q) = \sum_i P(i) \log P(i)/Q(i) \quad (4.13)$$

Where D_{KL} presents the logarithmic difference between the probabilities P and Q in which P represents the true uniform distribution of results, and Q is the observed data during the experiments and defined as follows:

$$P(c_i) = C(i)/\sum_i^n C(i), 1 \leq i \leq n, \quad (4.14)$$

n indicates number of OF switches

$$P(b_i) = b(i)/\sum_i^n b(i), 1 \leq i \leq n, \quad (4.15)$$

n indicates available links

Equations (4.14) and (4.15) represent the probability of residual flow table capacity and residual bandwidth respectively.

4.4.3 Performance Evaluation

We employ KL-divergence as a comparison method between the three evaluated approaches. Hence, in equation (4.14), P represents a true uniform distribution of network resources across all paths and Q represents the actual distribution achieved by the routing method. Figure 13 presents the Cumulative Distribution Function (CDF) for the

residual flow table capacity and residual bandwidth measured for simulation A (100 game flows). Similarly, Figure 14 shows the CDF of the network resources for simulation B (200 game flows). Furthermore, Table 13 in terms of bandwidth and flow entry capacity for the evaluated methods.

Table 13.Results of KL- divergence calculation

		KL-Divergence	ECMP	DBD	AGAR
A	100 game Flows	Residual of Bandwidth	0.6677	0.6345	0.5719
		Residual of Flow Capacity	0.7707	0.5344	0.3967
B	200 game Flows	Residual of Bandwidth	0.6626	0.6259	0.6062
		Residual of Flow Capacity	0.4144	0.375	0.3425

It can be seen that the KL-divergence value for AGAR is lower compared to ECMP and DBD, and much closer to a uniform distribution of flows across paths. This shows that the game traffic is fairly distributed throughout the network which results in diminishing network congestion.

The average delay experienced by the game flows belonging to the different games is depicted in Figure 15a and Figure 15b. For Scenario 1, the average delay experienced by players is 1.9%, 5% and 3.7% lower when we use AGAR compared to DBD, ECMP and Hedera, respectively. For Scenario 2, the average delay experienced by players is 2.4%, 2.7% and 9.5% lower when we use AGAR compared to DBD, ECMP and Hedera, respectively.

Figure 16a and 16b show that the average delay variation experienced by players is 14% and 8% lower when AGAR is used compared to ECMP for Scenarios 1 and 2, respectively. As for DBD, the results are almost comparable with AGAR. On average, AGAR

decreases the delay variation experienced by users by almost 7% and 2% with respect to DBD for Scenarios 1 and 2 respectively. Also, the average delay variation experienced by players using the proposed method is almost 6% and 4% less compared to Hedera for Scenarios 1 and 2, respectively.

Although AGAR renders lower overall delay variations on the network, there are some cases where DBD performs better for high delay sensitive games as evidenced by the League of Legends and Dishonored results. DBD assigns the path with the minimum delay to a game flow request without accounting for bandwidth. Therefore, this approach might, under limited circumstances, reduce the delay variations experienced by the flows of a high delay sensitive game, at the expense of other games in the network. Aside from the improvements in experienced delay, we compare the Packet Loss Rate (PLR) for different schemes. The results are illustrated in Figure 17a and 17b. For Scenario 1, the proposed method achieves almost 14%, 23% and 27% lower packet loss than the DBD, ECMP and Hedera methods, respectively. For Scenario 2, the proposed method continues to outperform DBD, ECMP and Hedera in terms of PLR by 19%, 17% and 14%, respectively.

This improvement stems from the fact that the proposed scheme monitors bandwidth usage of network paths, and dynamically regulates traffic load accordingly.

The bandwidth utilized by all network layers (i.e. core, aggregation and access) in scenarios 1 and 2 is illustrated in Figure 18a and 18b respectively. Our proposed method yields improvement in bandwidth utilization compared to ECMP and Hedera by 28% and 35%, respectively. The effective bandwidth utilization can enhance network fairness and contribute to traffic congestion avoidance. In scenario 2, access link bottlenecks occur, hence, the proposed approach

maximizes the utilization of all network links. Our proposed method better utilizes available bandwidth and outperforms the ECMP and Hedera schemes by 5% and 7% respectively.

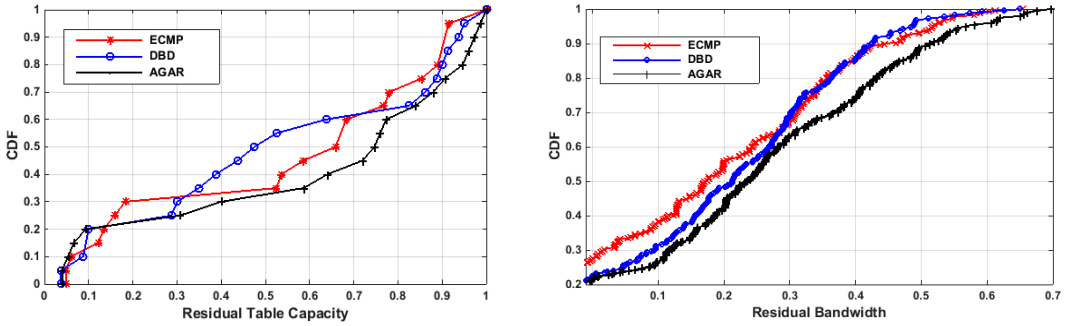


Figure 13.CDF of Residual Flow Table Capacity and Residual Bandwidth, 100 Game flows

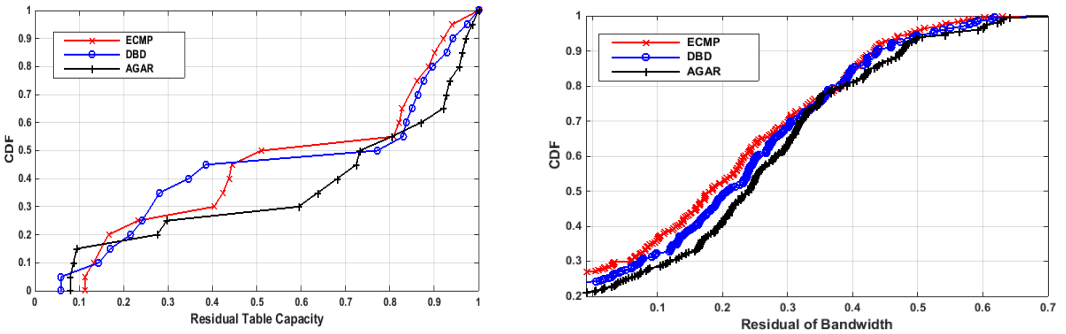


Figure 14.CDF of Residual Flow Capacity and Bandwidth, 200 Game flows

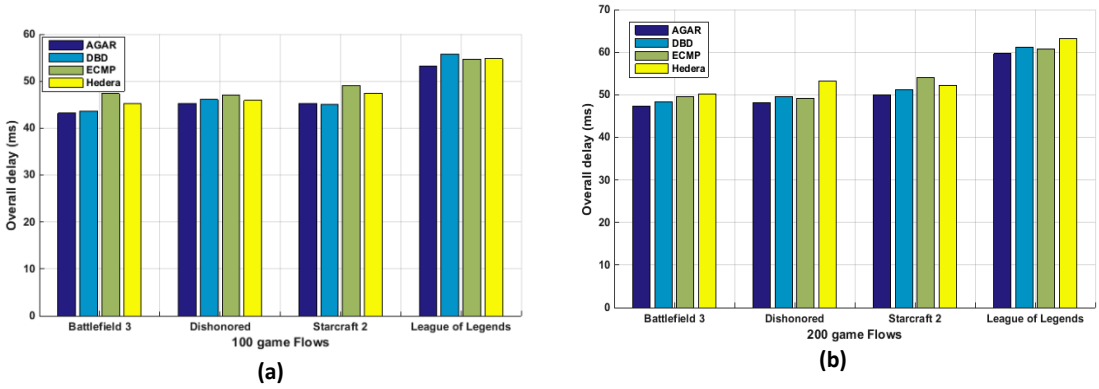
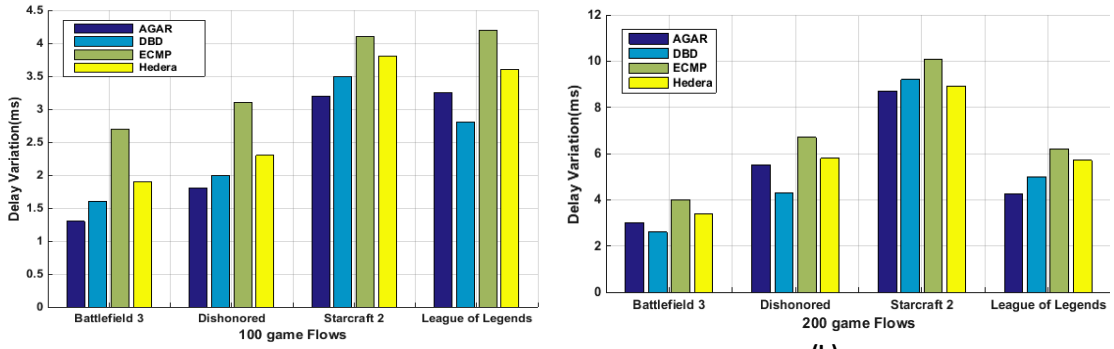
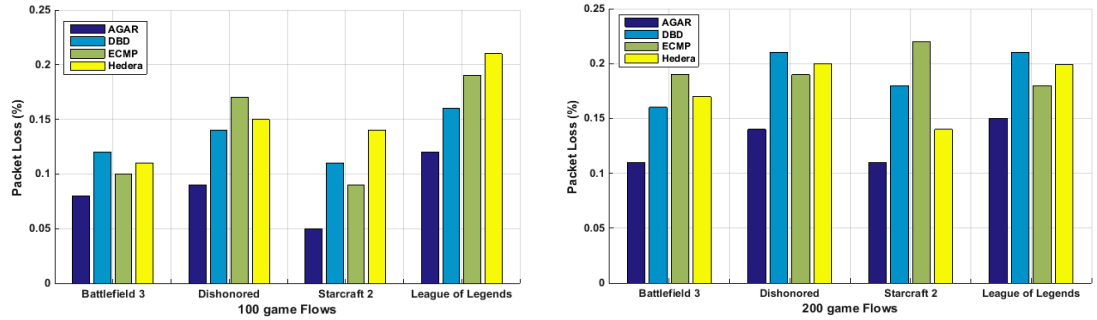


Figure 15.The Overall delay Experienced by Users a) Number of users=100 b) Number of users=200



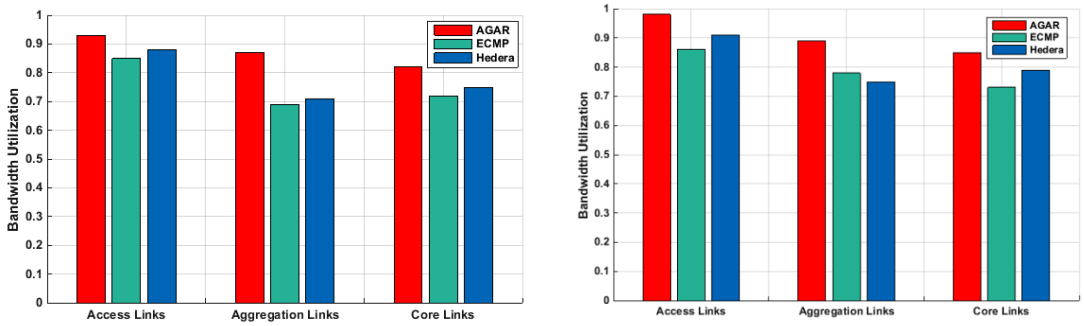
(a) (b)

Figure 16.The Average of delay variation Experienced by Users a) Number of users=100 b) Number of users=200



(a) (b)

Figure 17.Average of Packet Loss rate Experienced by Users a) Number of users=100 b) Number of users=200



(a) (b)

Figure 18.The bandwidth utilization of layers for a) Number of users=100 b) Number of users=200

4.5 Summary

We focused on DCN's resource management using SDN. We presented an optimization-based routing scheme that efficiently assigns gaming flows to communication paths while jointly minimizing the overall delay and maximizing bandwidth utilization. Unlike the

proposed method, existing cloud gaming DCN routing schemes do not consider the QoS characteristics of game genres for path assignment. In our evaluation, we compared the proposed method to three other representative approaches: the ECMP, DBD and Hedera. We conducted two simulations involving 100 and 200 game flows in a fat-tree DCN. Our results indicate that the proposed method results in the reduction of delay within the DCN compared to ECMP, DBD and Hedera by an average of 2.7%, 2.2% and 6.8%, respectively. We also note that there was no additional packet loss when we reduce the delay and jitter; and therefore the QoE perceived by a cloud gamer is being improved considerably. Also, our results show that the proposed optimization scheme produces a more balanced assignment of game flows across the DCN paths compared to the other three methods. This can potentially help prevent routing failures.

CHAPTER 5. Resource Optimization Through Hierarchical SDN-Enabled Inter Data Center Network For Cloud Gaming

5.1 The Hierarchy SDN-Enabled Data Centers Model

Conventionally, content distribution service providers (e.g. Akamai) use DNS servers as a mean to manage traffic across their DCs. They dynamically redirect client requests to appropriate content servers to optimize network utilization and improve the performance of service delivery [90]. However, recently, a new emerging network architecture approach, namely SDN, has enabled network operators to centrally control the entire network regardless of the underlying network technology. In SDN, a central controller is used to collect all network related information such as bandwidth utilization and congestion level to optimize resource allocation strategies. In this respect, we construct a model in which multiple DCs are connected through the network controlled by SDN as shown in Figure19. In the proposed hierarchical SDN architecture, the local SDN controllers govern the intra-DC networks, and send the network-related information of each DC to a centralized SDN controller that oversees the IDC communication. To avoid having a single point of failure, we applied an SDN controller master-slave scheme. Hence, each DC is controlled by a master SDN controller with three backup controllers (slaves) on standby to achieve fault tolerance [91].

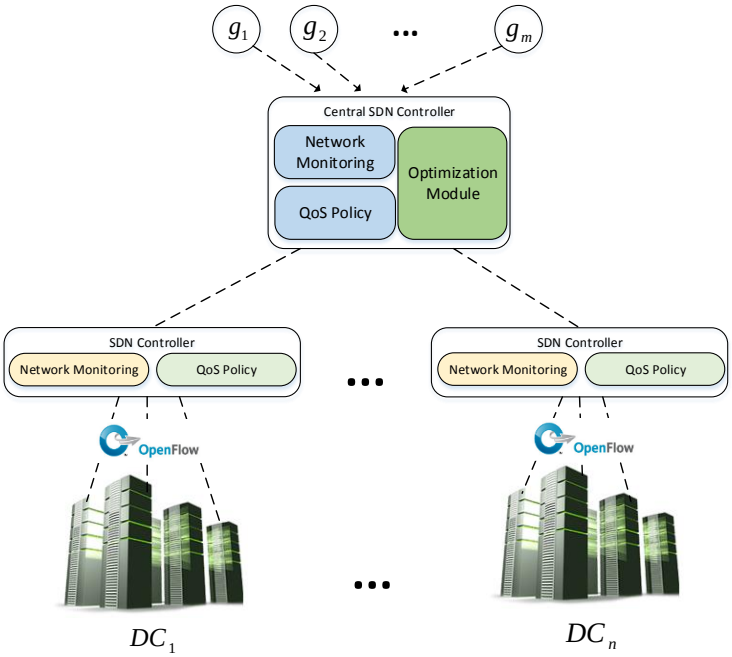


Figure19. Hierarchical SDN Framework for IDC Network

5.2 Problem Formulation

In this model, cloud servers hosted by a set of geographically distributed DCs can run game engines, instantly encoding and stream video games to any gaming device through the internet. The IDC network can be modeled by a complete and directed graph $\Gamma = \langle V, E \rangle$, whose vertices (V) represent the $|V| = M$ DCs. $G = \{g_1, g_2, \dots, g_N\}$ represents the set of gamers interacting with the cloud system. A gaming session executing on a server is created for each gamer. All DCs host the same collection of games, and multiple instances of a supported game can execute within a DC. It is highly unlikely that all gamers require assignment to a DC all at once as they join and leave the games at different times. Hence offline optimization approaches that implicitly suppose that all players join simultaneously when the system begins accepting requests do not reflect real world scenarios. To account for the dynamicity of the players' behaviors in terms of joining and leaving

games, we apply an Online Convex Optimization (OCO) approach [92]to find the best DC to serve gamers' requests over time.

5.2.1 Maximize bandwidth utilization for cloud gaming service providers

Let us assume that each gamer can only be served by a single DC at a time. Accordingly, we define a binary variable, x_t^{ij}, to reflect this property such that if j^{th} DC is assigned to serve the i^{th} gamer at time t, x_t^{ij} is 1, and x_t^{ij} is 0 otherwise. Also, the amount of bandwidth allocated to ith gamer by the jth DC is represented by β^{ij}.

To characterize the satisfaction of the i^{th} gamer when receiving β^{ij} bandwidth, we define a utility function, U_i which can capture the quality of the gaming service as perceived by the gamer. The problem can be formulated as a constrained utility optimization problem where the objective is to maximize the overall utility, defined as the sum of individuals' utilities, and constrained by the DCs' bandwidth capacities (see function (5.1)).

$$P_1: \text{Maximize } f(x_t) = \sum_{i=1}^{N} U_i\left(\sum_{j=1}^{M} \beta^{ij} x_t^{ij}\right) \quad (5.1)$$

Subject to:

i. $\sum_{i=1}^{N} \beta^{ij} x_t^{ij} \leq C_j \quad \forall\, j \in \{1, \dots, M\}$

ii. $\sum_{j=1}^{M} x_t^{ij} = 1, \forall\, i \in \{1, \dots, N\}$

iii. $x_t^{ij} \in \{0,1\}, \forall\, j \in \{1, \dots, M\}, \forall\, i \in \{1, \dots, N\}$

Constraint (i) guarantees that the bandwidth aggregation of all game sessions using a DC does not surpass the DC's capacity denoted by C_j. Constraints (ii) and (iii) ensure that exactly one DC serves a gamer and the corresponding decision variable, i.e. x_t^{ij}, can only be set to binary values.

In addition to maximizing the bandwidth utilization, it is also important to achieve a level of fairness among the gamers sharing the available bandwidth. For instance, the algorithm refrains from allocating near zero bandwidth to a gaming session to avoid resource starvation. Similarly, it does not allocate most of the available bandwidth to a gaming session at the expense of others. In this respect, the authors of [93], [94] propose using α-fair function, defined in equation (5.2), as a general utility function. Depending on the value of α, the α-fair function can capture different fairness notions. α can take a value in the range of $(0, \infty)$. For instance, if α is set to one, it corresponds to proportional fairness, and for large values of α, it converges to max-min fairness.

$$\alpha - \text{function}(\mathrm{x}_t) = \begin{cases} \log(\mathrm{x}_t), \text{if } \alpha = 1 \\ (1-\alpha)^{-1}(\mathrm{x})^{1-\alpha}, \text{if } \alpha \geq 0 \text{ and } \alpha \neq 1 \end{cases} \quad (5.2)$$

According to the measurement study in [43], the utility function of gaming traffic is typically logarithmic. Since the α-fair function when $\alpha = 1$ is also logarithmic, it is able to properly model the gamers' QoE. Hence, the utility function for bandwidth can be defined as (5.3).

$$\mathrm{U_i}\left(\sum_{\mathrm{j}=1}^{\mathrm{M}} \beta^{ij} x_t^{ij}\right) = \log\left(\sum_{\mathrm{j}=1}^{\mathrm{M}} \beta^{ij} x_t^{ij}\right), \text{for } \alpha = 1 \quad (5.3)$$

So, the objective function of bandwidth utilization maximization ($\mathrm{f_1}(\mathrm{x}_t)$) can be rewritten as (5.4):

$$\mathrm{P_1}\text{: Maximize } f_1(\mathrm{x}_t) = \sum_{i=1}^{N} \log\left(\sum_{j=1}^{M} \beta^{ij} x_t^{ij}\right) \quad (5.4)$$

Subject to:

i. $\sum_{i=1}^{N} \beta^{ij} x_t^{ij} \leq C_j \quad \forall\, j \in \{1, \dots, M\}$

ii. $\sum_{j=1}^{M} x_t^{ij} = 1, \forall\, i \in \{1, \dots, N\}$

iii. $x_t^{ij} \in \{0,1\}, \forall\, j \in \{1, \dots, M\}, \forall\, i \in \{1, \dots, N\}$

Since $x_{t_{ij}}$ is a binary variable, the objective function $f(x_t)$ represents a combinatorial problem that is difficult to solve for a high number of gamers. However, cloud service providers often own a relatively small number of DCs in different geographic locations, such that the problem $f_1(x_t)$ can be computed in a polynomial time.

5.2.2 Minimize delay experienced by the gamers

According to studies [95], [96], network delay and its variations (jitter) have a dramatic influence on the network performance, and consequently on the quality of the cloud gaming service. Low setup and response time is crucial for providing a satisfactory cloud gaming service. The service provider should be able to react promptly to abrupt changes in demand so that the required resources are provisioned, and the corresponding gaming sessions are set up in a reasonable amount of time. Due to the interactive nature of gaming, the quality perceived by the gamer is very susceptible to the performance of the underlying network. For instance, larger network delay negatively impacts interactivity which might frustrate the gamer and prompt them to abandon the game. The widely adopted approach to alleviate network delay is to send the incoming game requests to the closest DC. However, this approach may result in overwhelming some DCs, particularly when the distribution of gamers is not uniform; assigning a number of requests beyond a DC capacity might result in an exponential increase in response time. Meanwhile, other DCs that we can use to mitigate the problem might be underutilized. Accordingly, as opposed to the conventional method, in this section, we propose an optimization mechanism that minimizes the total network delay with respect to constrains imposed by both service provider and the acceptable user's QoE. The total end to end delay between a gamer and the assigned

server can be divided into two components, transport delay and response delay. The latency between the i^{th}gamer and the corresponding j^{th} DC is referred to as transport delay and denoted by $d_{n_{ij}}$. The latency within a DC is called response delay and indicated by $d_{r_{ij}}$. We define the objective function ($f_2(x_t)$) as the sum of transport and response delay for all gamers. So, the problem of assigning DCs to gamers can be formulated as a binary optimization problem as shown in equation (5.5) to minimize the total network delay.

$$P_2: \text{Minimize } f_2(x_t) = \sum_{i=1}^{N} \sum_{j=1}^{M} (d_{n_{ij}} + d_{r_{ij}}) x_t^{ij} \qquad (5.5)$$

Subject to (i, ii, iii) and:

$$\sum_{j=1}^{M} (d_{n_{ij}} + d_{r_{ij}}) x_t^{ij} \leq \delta_{i_{max}} \quad \forall i \in \{1, \dots, N\}$$

x_t^{ij} is a binary decision variable. It is used to assign a serving DC to each gamer. In addition to constraints (i, ii and iii) which are identical the P1 constraints defined in (4), constraint (iv) ensures that the end to end delay (response and network delay) of each gamer does not exceed a predefined threshold i.e. the maximum delay ($\delta_{i_{max}}$) tolerable for a gaming application.

5.2.3 Joint Optimization Model

We have formulated two optimization problems, P_1 and P_2, for distributing gamers' requests among the DCs while maximizing resource utilization to benefit the service provider and minimizing delay to increase the QoE of gamers. However, we should note that the optimal solutions obtained from these problems are not necessarily compatible, and in some cases conflict with each other. To tackle this issue, we introduce a bi-objective optimization problem P_3, equation (5.6), with the aim of jointly minimizing the overall delay experienced by gamers and

maximizing the overall bandwidth utilization among all DCs.

$$P_3: \max\{f_1\}, \min\{f_2\} \equiv \min\{-f_1, f_2\} \quad (5.6)$$

Subject to:

i. $\sum_{i=1}^{N} \beta^{ij} x_t^{ij} \leq C_j \quad \forall \, j \in \{1, \dots, M\}$

ii. $\sum_{j=1}^{M} x_t^{ij} = 1 \quad \forall \, i \in \{1, \dots, N\}$

iii. $\sum_{j=1}^{M} (d_{n_{ij}} + d_{r_{ij}}) x_t^{ij} \leq \delta_{i_{max}} \quad \forall \, i \in \{1, \dots, N\}$

iv. $x_t^{ij} \in \{0,1\} \quad \forall \, i \in \{1, \dots, N\}, \forall \, j \in \{1, \dots, M\}$

As we can see in (5.6), the maximization problem of P_1 is converted to a minimization problem to achieve an overall minimization problem, and the union constraints sets of P_1 and P_2 constitute the constraints set for P_3. As previously pointed out, since the decision variable is a binary variable, P3 is regarded as a 0-1 Knapsack problem with a minor difference in coefficients. In this context, we can use Dynamic Programming (DP) to solve this optimization problem. Although DP provides a generic approach to attach to such problems, according to the DP formulation, the principal drawback of dynamic programming is its high dimensionality due to the large number of variables required to track intermediate solutions. For example, if we have 6 DCs and 1,000,000 gamers, the DP algorithm requires 6 million partial solutions to be examined concurrently. Hence, DP cannot handle massive numbers of gamers. To realize a scalable algorithm that solves P3 with a manageable computational complexity, we further simplify and relax the problem by replacing the discrete decision variables with continuous ones. Consequently, we add an extra non-negative penalty term to the objective function to force the algorithm to pick values close to either 0 or 1. In this respect, equation (5.7) defines the penalty function, g(x), to

convert the binary variable x_{ij} into the continuous variable $\tilde{x}_{ij}$ so that $0 \leq \tilde{x}_{ij} \leq 1$.

$$g\left(\widetilde{x_t^{ij}}\right) = \sum_{i=1}^{N} \sum_{j=1}^{M} \widetilde{x_t^{ij}}(1 - \widetilde{x_t^{ij}}) \tag{5.7}$$

We can see from this definition that g(.) has its lowest value, i.e. 0, when $\tilde{x}_{ij}$ is either zero or one.

Adding a concave penalty function to a minimization problem may create local minima to almost most of the feasible points [97]. This means that the solver can likely get trapped in those minima. This phenomenon is known as premature convergence. However, reducing the number of local minima can increase the chance of reaching the global optimum. To this end, we can utilize a method that transforms the problem into an analogous one with a smoother objective function. Accordingly, we utilize the logarithmic smoothing technique [97], to eliminate the local optima. This method adds a term, the so-called logarithmic barrier function, defined in equation (5.8), to the objective function of the problem to eliminate the inequality constrains of $0 \leq \tilde{x}_{ij} \leq 1$. The main feature of this smoothing technique is that $S(\tilde{x}_{ij})$ is strikly convex, and if the original problem is convex, the transformed problem remains convex.

$$S\left(\widetilde{x_t^{ij}}\right) = \sum_{i=1}^{N} \sum_{j=1}^{M} \log(\widetilde{x_t^{ij}}(1 - \widetilde{\widetilde{x_t^{ij}}})) \tag{5.8}$$

Thus, we will rewrite the problem P3 with the penalty function and the smoothing logarithmic term as follows:

$$P_4: \min\{-f_1, f_2, \gamma . (g\left(\widetilde{x_t^{ij}}\right)) - \mu . S\left(\widetilde{x_t^{ij}}\right)\} \tag{5.9}$$

Where $\gamma > 0$ is a penalty parameter and $\mu > 0$ is the smoothing parameter.

The optimization problem P_4 contains two conflicting objectives, maximizing the overall utility function for all DCs (f_1) and minimizing the delay experienced by gamers (f_2) simultaneously. In such problems where multiple contradicting objectives are present, several optimal solutions, namely Pareto optimal solutions, exist depending on the various levels of importance given to each objective. However, obtaining all Pareto solutions is an enormously computationally complex exercise, and consequently cannot be performed in real-time. Therefore, our goal is to find a single preferred point on the Pareto frontier instead of calculating the entire frontier. Accordingly, we use the weighted-sum method to convert the multi-objective optimization problem into one with a single objective optimization through the summation of weighted objectives. Thereby, weights are defined for objective functions to reflect their corresponding significance in the final objective problem. By applying the weighted sum method to the objectives, f_1 and f_2, in P_4, the multi-objective model (5.9) can be restated as the following single objective problem:

$$P_5: \min\{-\omega_1 \mathrm{f}_1 + \omega_2 \mathrm{f}_2 + \gamma \,.\, g\left(\widetilde{x_t^{ij}}\right) - \mu \,.\, S\left(\widetilde{x_t^{ij}}\right)\} \tag{5.10}$$

Subject to:

ai. $\sum_{\mathrm{i}=1}^{\mathrm{N}} \beta^{ij} \widetilde{x_t^{ij}} \leq \mathrm{C_j} \quad \forall\, \mathrm{j} \in \{1, \dots, \mathrm{M}\}$

aii. $\sum_{\mathrm{j}=1}^{\mathrm{M}} (\mathrm{d_{n_{ij}}} + d_{r_{ij}}) \widetilde{x_t^{ij}} \leq \delta_{\mathrm{i}_{max}} \quad \forall\, \mathrm{i} \in \{1, \dots, \mathrm{N}\}$

aiii. $\sum_{\mathrm{n}=1}^{2} \omega_{\mathrm{n}} = 1\,,\ \omega_{\mathrm{n}} > 0\ \forall \mathrm{n} \in \{1,2\}$

Where ω_1 and ω_2 are weighting coefficients, and correspond to the relative importance of the objective f_1 and f_2 respectively. Accordingly, we have eliminated $0 \leq \tilde{\mathrm{x}}_{\mathrm{ij}} \leq 1$ from the constraint set.

Even though the weighted sum method provides an effective approach

to handle the multi objective problem, assigning proper weight coefficients to objectives is not a trivial task since small changes in weights may lead to significant changes in the optimal solution. Hence, even having known a priori satisfactory solution does not help determine weight coefficients that are general enough to accommodate most preferences. To address this challenge, we propose assigning weights to objectives based on games' specific characteristics such as their network bandwidth requirements. Due to the substantial number of existing games, it is not feasible to determine a weight for each one. However, games can be categorized into game genres. Accordingly, weights can be determined based on game genres. A game genre defines a group of games that have common characteristics, e.g. perspective, gameplay, level of interaction, objective, etc. As shown in [39], the amount of scene and motion complexity tend to depend on the game genre. More complex game scenes require a higher bandwidth compared to a less complex ones. In addition, games with more motion are more susceptible to delay variation. Hence, for Real Time Strategy (RTS) games, we assign a larger weight to f_2 compared to f_1, whereas for First Person Shooter (FPS) games, we perform the opposite. In general, we define three different sets of weights for game genres including RTS, Role-Playing game (RPG) and FPS. The weights associated with a specific game genre implicitly reflect the importance given to the corresponding objective for that game genre.

5.2.4 Online Lagrange dual problem (Dual Problem)

To the relax the optimization problem P_5, i.e. constrained Nonlinear Programming (NLP), we employ the Lagrange multiplier heuristic to eliminate the constraints by introducing new corresponding penalty terms in the cost function. In this way, we can utilize an iterative

gradient algorithm to achieve a nearly optimum solution in an acceptable time. We note that although the Lagrange multiplier heuristic approach bears some similarities to the penalty function method, the Lagrange multiplier provides some advantages that make it more efficient than the penalty function method. In the Lagrange heuristic approach, the multipliers are considered to be decision variables and treated as optimization variables. So, the function converges much faster to the near optimal solution compared to the penalty function. Accordingly, the constraint sets (ai) and (aiii) in P_5, are dualized with the Lagrange multiplier sets, λ and ν respectively, and the Lagrange dual function can be expressed as follows:

$$P_L(\beta^{ij}, \lambda, \nu) = (-\omega_1 f_1 + \omega_2 f_2) + \sum_{j=1}^{M} \lambda_j^T \left(\sum_{i=1}^{N} \beta^{ij} x_t^{ij} - C_j\right) + \sum_{i=1}^{N} \nu_i^T \left(\sum_{j=1}^{M} (d_{n_{ij}} + d_{r_{ij}}) x_t^{ij} - \delta_{i_{max}}\right) \quad (5.11)$$

It is important to realize that the Lagrange dual function yields a lower bound for the optimal solution of P_5 (5.10) for any combination of λ and ν with $\lambda \geq 0$ and $\nu \geq 0$. Hence, finding the maximum value for the Lagrange dual function corresponds to finding the closer lower bound for the optimum solution of the primal problem. Respectively, the problem of finding the maximum value of the Lagrange dual function can be formulated as shown in equation (5.12), and referred to as the Lagrange dual problem.

$$P: max\ P_L(\lambda, \nu) \quad (5.12)$$

Subject to:

$$\lambda_j \geq 0, \nu_i \geq 0, \forall\, i \in \{1, \dots, N\}, \forall\, j \in \{1, \dots, M\}$$

5.2.5 Modified Online Saddle-Point Algorithm

Finally, the characteristics of the problem in equation (5.12) allows us to efficiently obtain a suboptimal solution through a less

computationally complex process. We solve the online Lagrangean problem (5.11) using the Modified Online Saddle Point (MOSP) approach as it can provide a sub-linear dynamic regret under some mild assumptions [98], [99] (see Algorithm 5.1). The MOSP approach takes a modified descent step in the primal domain, and a dual ascent step at each time slot t. Specifically, given the previous primal iterate $x_{t-1}$and the current dual iterate λ^T and ν^T at each slot t, the current decision x_t is the minimizer of the following optimization problem.

$$\min_{x} \nabla^T P_5^{t-1}(x_{t-1})(x - x_{t-1}) + \sum_{j=1}^{M} \lambda_j^T \left(\sum_{i=1}^{N} \beta^{ij} x_{t-1}^{ij} - C_j\right) + \sum_{i=1}^{N} \nu_i^T \left(\sum_{j=1}^{M} (d_{n_{ij}} + d_{r_{ij}}) x_{t-1}^{ij} - \delta_{i_{max}}\right) + {\|x - x_{t-1}\|}/{2\alpha} \quad (5.13)$$

In (5.13), we have defined $\nabla^T P_5^{t-1}(x_{t-1})$ as a gradient of objective P_5 (5.10) at $x = x_{t-1}$, and α a gradient step size. Given that the current decision x_t is made, $P_5^t(x)$ is observed, and the dual update takes the form of the online Lagrangean (5.11).

ALGORITHM 5.1: Modified online saddle-point (MOSP) method

1: Inputs:

$\boldsymbol{\lambda_0}$: initial value for Lagrangean multipliers

$\boldsymbol{\nu_0}$: initial value for Lagrangean multipliers

K: number of Iterations

$\mathbf{K_{max}}$: Maximum number of Iterations

$\boldsymbol{\alpha}$:step size

2: Initialize:

$\boldsymbol{\lambda_i^k = \lambda_0}$, $\boldsymbol{\nu_{ij}^k = \nu_0}$, $\mathbf{k = 1}$, $\mathbf{K_{max} = 100}$

3: While ($\acute{\varepsilon} \geq .001$) || $K \leq K_{max}$

$\xi_k \leftarrow$ Calculate the gradient at current solution

4: Update primal variable x_t by solving (13)

5: Observe the current cost $P_5^t(x_t)$

6: Update Lagrangean multipliers

$$\lambda_i^{k+1} = \lambda_i^k + \alpha^k \frac{\xi^k}{|\xi^k|} \text{ , } \forall\, i \in \{1, \dots, L\}$$

$$\nu_{ij}^{k+1} = \nu_{ij}^k + \alpha^k \frac{\xi^k}{|\xi^k|}, \forall\, i \in \{1, \dots, L\}, \forall\, j \in \{1, \dots, k_i\}$$

$$\acute{\varepsilon} = \left| P_L(\lambda^{k+1}, \nu^{k+1}) - P_L(\lambda^k, \nu^k) \right|$$

6: $\alpha^{k+1} = {}^{1}/_{k}$

$k = K + 1$

7: Go to step 3

End while;

5.3 Experimental Evaluation

This section describes our simulation setup, presents the experimental scenario, and briefly introduces existing algorithms we used in our comparisons. We also provide the performance evaluation of the proposed optimization method and discuss its effectiveness for both gaming service providers and gamers.

5.3.1 Simulation Setup

To evaluate the proposed method, we defined a scenario in which the gaming service provider owns a private cloud network consisting of six geographically distributed DCs. The bandwidth capacity of inter-connecting links are randomly set to values uniformly distributed within the range [1 - 10] Gbps, which is a common WAN bandwidth range for many companies [8]. We presumed that 500 gamers are playing four games belonging to different genres. They are being served by the gaming service provider such that each DC can serve up to 100 gamers. The data stream pertaining to each gamer was generated using a throughput and bandwidth performance tool, Iperf [100]. The data traffic of each game conforms to the associated game genre. To measure the characteristics of different games, we performed a measurement study based on a methodology proposed by [37]. Table 14provides the characteristics of the data traffic of different game genres.

We consider a three-tier based network topology for the DCs (8-pod fat-tree [80]) with 8 core OF-enabled switches. We use the Mininet emulator to create a realistic network experiment with OpenFlow and SDN. We employ the POX 1.0 controller [75] running on a PC with an Intel i7-6700 processor. The OF-enabled switches run Openflow v1.3, and the forwarding rules are installed and updated via POX's API, and each game server connects to the OF-switch though a 1 Gbps link. We implemented Algorithm 5.1 in Matlab using the convex optimization package (CVX) [74]. Table 15 shows the weighting coefficients we used for different objectives. We applied the technique described in [101] to find the corresponding weighting coefficients for the games.

Table 14. Network Traffic Parameters of Games

Games	Genre	Upstream traffic		Downstream traffic	
		Mean Packet Size (bytes)	Mean Inter-Departure time (ms)	Mean Packet Size (bytes)	Mean of Inter-Departure time (ms)
Half-life	First Person Shooter (FPS)	38.03	2.6	1,118.59	1.5
Diablo	Action-adventure	62.14	3.3	752.58	1.8
Starcraft	Real Time Strategy (RTS)	34.01	2.6	925.65	2. 8
Age of Empire [102]	Multiplayer online battle arena(MOBA)	59.74	1.2	634.38	2.1

Table 15.weighting coefficients for each game

GAME	Genre	ω_1	ω_2
Half-life	First Person Shooter (FPS)	0.147	0.853
Diablo	Action-adventure	0.643	0.357
Starcraft	Real Time Strategy (RTS)	0.29	0.71
Age of Empire	Multiplayer online battle arena	0.65	0.345

To measure the impact of applying the OCO algorithm for the resource allocation of cloud gaming services, we compared the performance of the MOSP (Algorithm 1) to the offline Lagrangean algorithm. We assumed that 150 players were arriving at random times within a 400 seconds period are requesting the games. Figure 20 shows the Average Cost Ratio for both approaches with respect to time. We can observe that the MOSP converges to a smaller time-average cost compared to the offline Lagrangean solution.

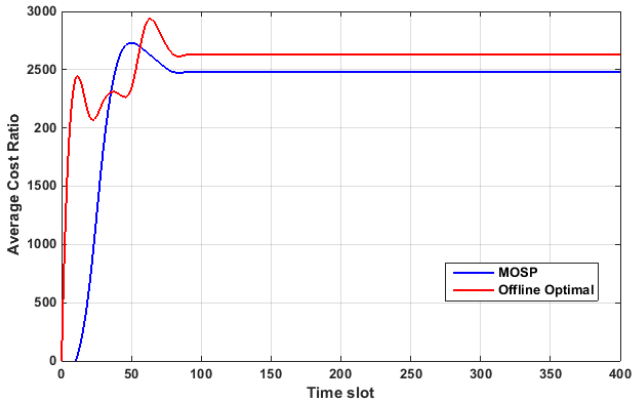


Figure 20.Comparison between the OCO approach and offline Lagrangean algorithm

To evaluate the performance of the proposed method, we compare our results to DDCCast [103], Amoeba [104], and Closestnode [105], as these approaches represent the behavior of conventional methods that concentrate on only one objective, either bandwidth utilization or network delay. We summarize these existing methods as follows:

i. DDCCast [103] employs a centralized point to deliver the content from a source DC to the required destination DCs. This method utilizes the forwarding tree algorithm to minimize bandwidth usage and balance the load across all links.

ii. Amoeba [104] aims at improving network utilization by leveraging a TE technique for IDC communication while meeting the required content deadline. Hence, they considered a slotted timeline where the transmission rate of senders is constant during each timeslot but can vary from one timeslot to the next.

iii. Closestnode [105] minimizes the network delay experienced by a user by applying local policies in which a request is assigned to the closest DC.

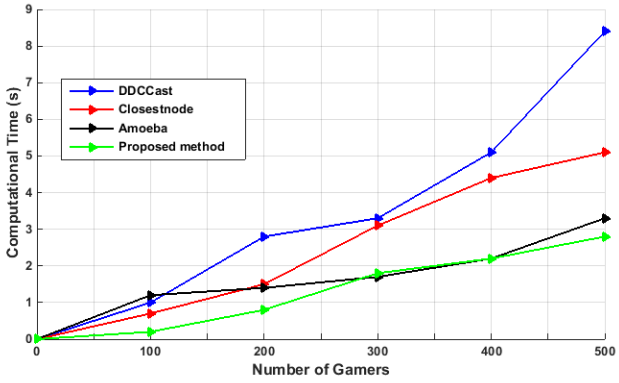


Figure 21.The Execution time of Algorithm 1(Number of DCs=5)

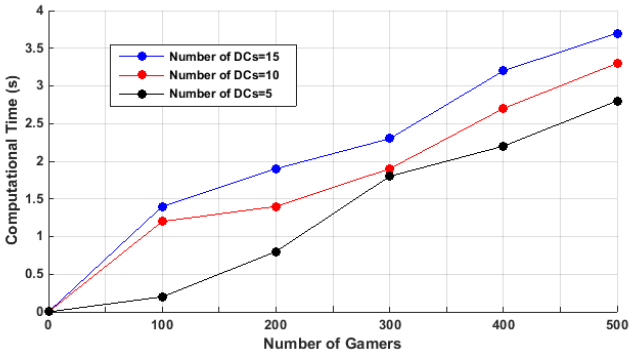


Figure22. The Execution time of Algorithm 1 for different number of DCs

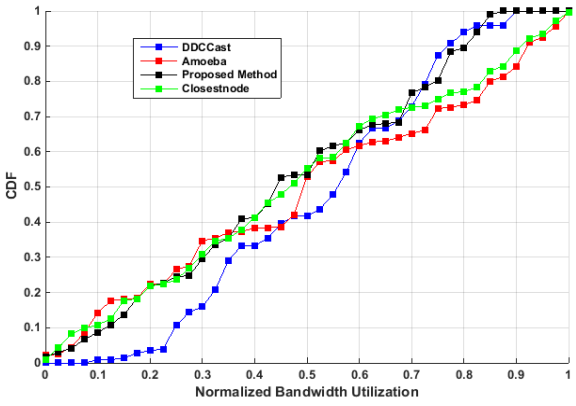


Figure 23.The CDF of normalized Bandwidth Utilization

5.3.2 Evaluation Results

5.3.2.1 The Performance of the Proposed Algorithm

To investigate the scalability of the proposed algorithm, we conducted experiments where we varied the numbers of gamers and DCs, and measured the corresponding execution time. Figure22 shows the average measured execution time for 15 distinct runs for different algorithms. Figure22 shows that the execution time proportionally increases as the number of games increases for different numbers of DCs. This demonstrates the linear behavior of the proposed algorithm. In addition, we can see in Figure22 that when the number of gamers is 500, the running time is less than 3.5 seconds, an acceptable processing time [106]. The results prove that our method converges to a near optimal solution in a reasonable amount of time. Moreover, since in practice the number of DCs is usually small, we can effectively apply the proposed method to real-world scenarios.

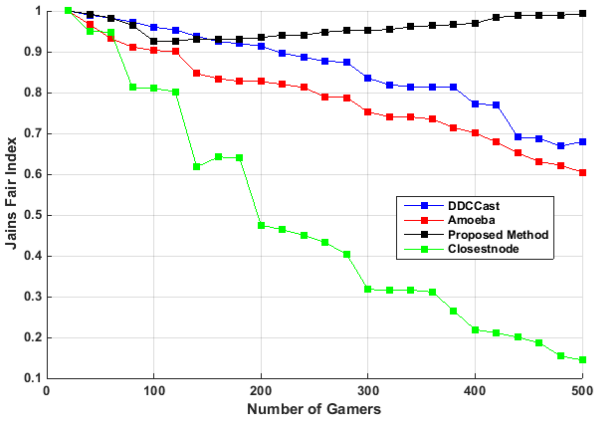


Figure 24.Jain's Fairness Index for different Number of Gamers

5.3.2.2 Bandwidth Utilization

One of our primary goals is to maximize bandwidth utilization to benefit gaming service providers. Hence, we evaluate the effectiveness of the proposed method in that regard.

We conducted 40 runs for the proposed and conventional (i.e. DDCCast [103], Amoeba [104], and Closestnode [105]) methods to measure the average bandwidth utilization across DCs. If we consider the average bandwidth utilization to be a realization of a random process, then the set of these 40 sample realizations allows us to construct the empirical cumulative distribution function (eCDF) of the bandwidth utilization. As shown in Figure 23, our method outperforms Amoeba and Closestnode, and is comparable to DDCCast, which directly aims at maximizing the bandwidth utilization without considering the network delay constraints. In addition to bandwidth utilization, fairly spreading gamers over the DCs is also crucial to avoid overloading a particular DC. Accordingly, we use Jain's fairness index [107] as an indication of how well the gaming traffic is distributed across available DCs. As shown in Figure 24, Jain's fairness index results show that the proposed method significantly improves fairness compared to the other three methods, with improvements of at least 8- percent. The reason is that our method uses a logarithmic function when it attempts to maximize utility. This means that it does not allocate more gamers to a DC that has reached the upper end of the logarithmic function in terms of allocated bandwidth since doing so does not result in considerable improvement in the corresponding utility.

5.3.2.3 Delay

To investigate the performance of the proposed method from the gamers' perspective, we consider a critical metric: the number of gamers that meet the maximum tolerable delay deadline without affecting QoE (i.e. $\delta_{i_{max}} = 80$ ms [31]). As shown in Figure 25, it is evident that conventional approaches are not able to meet the delay deadline when the number of game requests increases. Although Amoeba aims to

guarantee the IDC deadline, it does not consider the nature and requirements of traffic transferred over the network. Moreover, the improvement on this metric closely follows the improvement in bandwidth utilization. This result also confirms our observations for the bandwidth utilization: more requests can be handled when the available bandwidth is efficiently utilized.

Aside from the improvement in the number of game requests satisfying the delay deadline, the proposed algorithm can also reduce the delay experienced by gamers. Figure 26 shows the overall delay for each game. The proposed method reduces the overall delay for most of the games in comparison with the other approaches. In addition, it is important to note that on average, the gamers playing delay sensitive games (e.g. Half-life) experience more delay reduction compared to the others. This implies that not only the proposed approach reduces the average delay for all games, but it also accounts for the delay-sensitive nature of some games.

Figure 27 plots the average delay variation (jitter) for each game. As we can see, the jitter experienced by gamers when we use the proposed method is almost 40%, 46% and 53% less than what is experienced by players when we employ the DDCCAST, Amoeba and Closetnode approaches respectively. To further investigate the robustness of the proposed method, we show the packet loss rate in Figure 28. In this scenario, the proposed method achieves almost 45% lower packet loss than the conventional methods. Based on these results, we can hypothesise that the proposed method decreases jitter without degrading packet delivery rate.

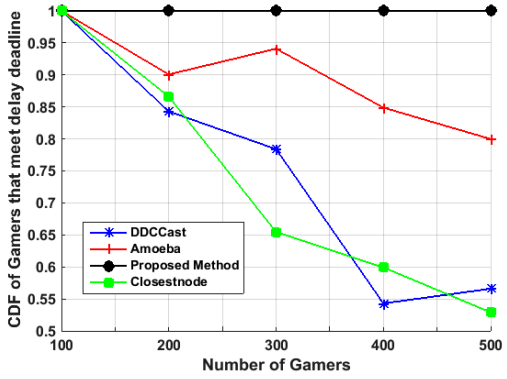


Figure 25.CDF Of Gamers that meet maximum tolerable delay ($\delta_{max} = 80\ ms$)

Moreover, we can see that the impact of the proposed controller differs based on the type of the game. For instance, as shown in Figure 27, the jitter reduction resulting from the proposed method for Half-life (FPS genre) and Starcraft (RTS) is approximately two times that of Diablo. The games with higher scene complexity and higher interaction level produce larger packet sizes and shorter departure-intervals, which means that these games produce higher data rates. This results in more congestion on the network. The proposed controller manages the congestion by distributing network flows among the available paths based on the overall current network condition.

Conversely, conventional methods disregard such conditions. Therefore, our method supports games with higher data rates better than conventional methods.

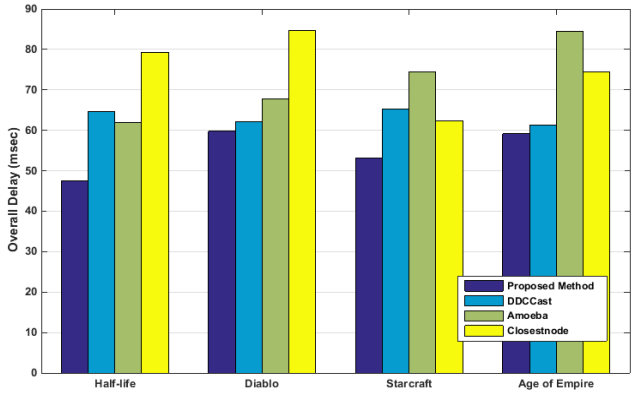


Figure 26. Average of Overall Delay Experienced by Gamers

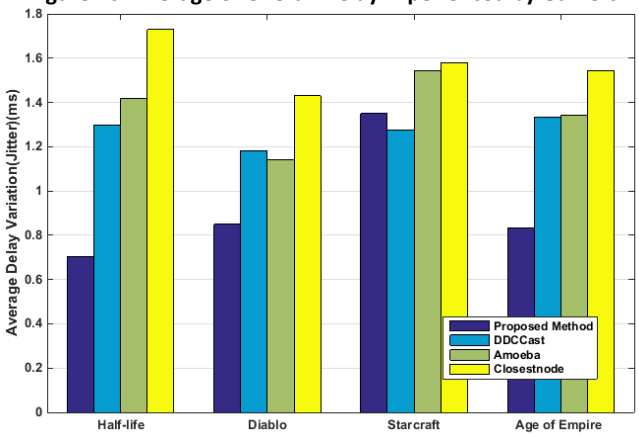


Figure 27.Average Delay Variation

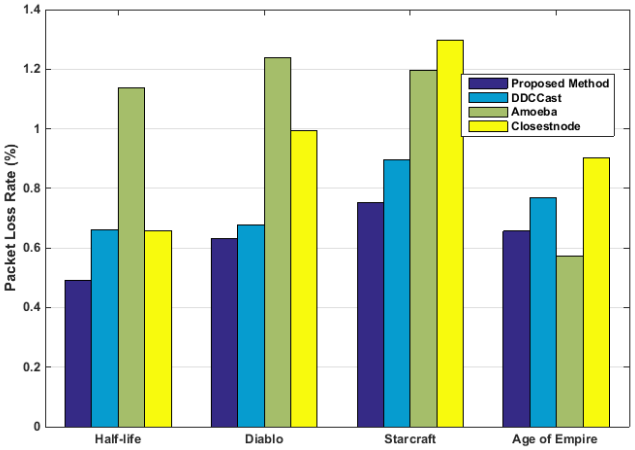


Figure 28.Packet Loss Rate

5.4 Summary

We focused on the efficient central resource management for multiple distributed DCs using a SDN approach. We presented a novel optimization method for assigning DCs to gaming requests that accounts for game requirements. Due to the high computational complexity of the optimization method, we designed an algorithm based on the Online Convex Optimization (OCO) method to drastically reduce the complexity of the problem such that it can be practically implemented in a DC using the OF controller. The evaluation of the proposed method indicated that it can converge to a near optimal solution in a reasonable amount of time. Moreover, experimental results showed that the proposed method, on average, outperformed existing algorithms by maximizing the bandwidth utilization, reducing delay, and achieving fair resource allocation among multiple competing gamers.

Chapter 6. Game-Aware Bandwidth Allocation for Home Gateways

6.1 Proposed Game-Aware Resource Manager

6.1.1 Introduction

In this section, we propose an optimization scheme we call GARM, to fairly allocate bandwidth to competing applications transmitting and receiving data via a shared gateway. The GARM-enabled home gateway understands each application's QoS requirements, monitors traffic patterns of active applications, and constantly assesses available bandwidth to optimally assign network resources to specific applications. The proposed system attempts to enhance the gamers' QoE without starving background applications. Since the size of home networks is relatively small, the computational complexity of the optimization scheme will not be an obstacle to its practical realization.

As illustrated in Figure 29, GARM is composed of three components: network monitoring and traffic classification, resource optimization, and resource policy enforcement. The network monitoring and traffic classification component assesses the available bandwidth, and the number and nature of applications transmitting data. It is in charge of monitoring the traffic travelling through a common gateway to understand the properties of the various implicated flows. This is conducted in pass-through mode so that the traffic can be seen and classified without an intervention from a network administrator. We use IP header's three bits in its Type of Service (ToS) field to classify the incoming traffic into six classes representing different priorities, as shown in table 16. Each class is assigned a priority and an outgoing bandwidth depending on the application's needs. The six classes will be

handled such that delay-sensitive applications get priority while background applications can still function. The traffic classification module generates a report of its findings and feeds it to the resource optimization module which in turn optimally assigns bandwidth amongst the applications competing for resources. The resource optimization module ensures that game traffic takes precedence over other applications. Finally, the resource policy enforcement module implements the traffic shaping and scheduling based on the output generated by the resource optimization component. It is responsible for traffic shaping, scheduling, and prioritization based on the output of the resource optimization component. It maintains queues and forwards packets while taking into account their priority.

6.2 Resource Optimization

The GARM resource optimization module assigns bandwidth to game flows. We have employed an optimization technique which considers the current estimate of available bandwidth and information about active applications to optimally assign the bandwidth while maximizing long-term fairness.

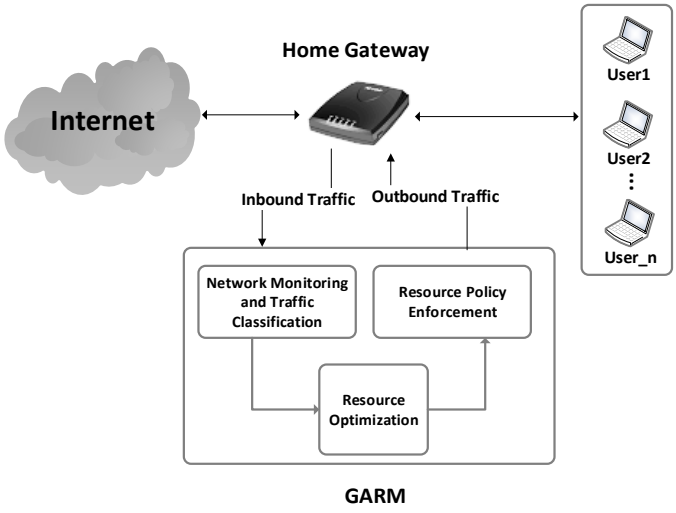


Figure 29.Proposed Game-Aware Optimization Method

6.2.1 Optimization Problem Formulation

The goal of the proposed bandwidth allocation scheme is to maximize the aggregate utility of all applications competing for bandwidth. Hence, we employ a weighted sum rate maximization technique. Let $i = \{1, \dots, N\}$ be a set of active applications sharing a single bandwidth bottleneck with other concurrent applications via a residential home gateway. Let $w_i \in (R^+)^N$ be the weight that represents the priority of application i. We define U_i as the utility function of application i when it receives bandwidth resource x_i. The utility function reflects the level of benefit an application i receives as the value of x_i increases.

Utility is an economic term that represents worth and preference for resources. Since the capacity of a link is fixed, the optimization problem is formally described as a maximization of an aggregate utility of all the applications subject to total available bandwidth capacity:

$$Maximize\ f(x) = \sum_{i=1}^{N} U_i(x_i) \qquad (6.1)$$

Subject to

i. $\sum_{i=1}^{N} x_i \leq C_{Max}$

ii. $x_i \geq 0$

Constraint (i) ensures that the aggregate of the bandwidth rate assigned to all of the applications would not exceed the bandwidth capacity C_{Max}. Constraint (ii) guarantees that the rate of x_i of any application is non-negative. We aim not only to maximize the allocated throughput of each active application, but also to obtain some level of fairness among them. More importantly, the allocated bandwidth to an application flow should neither be zero nor too large. To do so, we employ the logarithmic utility function to assign bandwidth to gaming sessions within the permissible range. So we rewrite the objective function (6.1) as follows:

$$\text{Maximize } f(x) = \sum_{i=1}^{N} \log x_i \quad (6.2)$$

Subject to

i. $\sum_{i=1}^{N} x_i \leq C_{Max}$
ii. $x_i \geq 0$

In addition, the utility (weighted) proportional fairness function prioritizes game traffic and real-time applications over others. By proportional fairness we aim to maximize total bandwidth utilization while allowing all users at least a minimal level of service. Hence, by assigning a weight w_i to Equation (6.2), we achieve weighted proportional fairness:

$$Maximize\ f(x) = \sum_{i=1}^{N} w_i log\ x_i \quad (6.3)$$

i. $\sum_{i=1}^{N} x_i \leq C_{Max}$
ii. $x_i \geq 0$

where w_i is the weight factor for application i, obtained by equation (6.4).

$$w_i = {P_i}\Big/{\sum_{i=1}^{N} P_i}, \forall i \in \{1, .., N\} \quad (6.4)$$

Table 16.Priority Level

Priority Level (P)	Application Class
6 (highest)	Operations,Administration,and Maintenance (OAM) Protocols
5	Interactive Gaming
4	VoIP/ Real-Time Interactive
3	Multimedia Conferencing
2	Multimedia Streaming
1 (lowest)	Flows that need no bandwidth assurance

To solve the optimization problem of Equation (6.3), we use the standard technique based on the Lagrangean Relaxation (LR) method, which eliminates the set of constraints that impose computational complexity on general integer problems. Once these constraints are eliminated, the Lagrange multiplier is introduced to the objective function. Lagrange multipliers are used to penalize violations of constraints. The process of updating the penalty parameters continues until convergence is reached. In practice, the new problem resulting from the LR is simpler than the original one due to the determination of the lower bound [70].

The problem of (6.3) is difficult to solve due to presence of coupling constraint (i), which involves x_i. By removing constraint (i), the problem can be converted to the form of known problems; therefore the solution can be easier to educe. We dualize constraint (i) by introducing the Lagrange multiplier set λ corresponding to capacity constraint of (i), hence the problem of Equation (6.3) can be re-written as follows:

$$l(X,\lambda) = \sum_{i=1}^{N} w_i log\, x_i - \lambda_i\left(\sum_{i=1}^{N} x_i - C_{Max}\right) \quad (6.5)$$

Where the Lagrange multiplier sets are $\lambda \geq 0$.

The problem of Equation (6.3) is convex with linear constraints (i) and (ii), and holds strong duality conditions, such that given a set of non-negative Lagrange multipliers λ ($\lambda \geq 0$), by setting $\frac{\partial l}{\partial x} = 0$ for each application flow i, the optimum value can be obtained as follows:

$$\frac{\partial l}{\partial x} = \frac{w_i}{x_{ij}} - \lambda_j w_i \quad (6.6)$$

Therefore:

$$x_i = {}^{1}/_{\lambda_i} \quad (6.7)$$

To solve the primal optimization problem (6.3) using the LR technique, we solve the dual problem to find the optimal values of λ. To do so, we adopt the gradient decent algorithm to iteratively find the Lagrangean multipliers as detailed in algorithm 6.1.

Algorithm 6.1. Gradient Optimization Algorithm

1: **Inputs:**
λ_0: initial value for Lagrangean multipliers
K: number of Iterations
K_{max}: Maximum number of Iterations
α :step size
έ: error index

2: **Initialize:**
$\lambda_i^k = \lambda_0$,k = 1, $K_{max} = 50$

3: While (έ ≥ .001) || K ≤ K_{max}
$\xi_k \leftarrow$ Calculate the gradient at current solution

4: Update Lagrangean multipliers
$\lambda_i^{k+1} = \lambda_i^k + \alpha^k \frac{\xi^k}{|\xi^k|}$
$\acute{\varepsilon} = |l(\lambda_{k+1}) - l(\lambda_k)|$

5: $\alpha^{k+1} = {}^{1}/_{k}$
k ←k+1

6: **Go to step 3**
End while;

6.3 Implementation and Performance evaluation

6.3.1 GARM Implementation

We implemented GARM on Mikrotik RB450G [108], featuring AR7161 CPU running at 680 MHz with 256 MB of RAM. The gateway is connected to the Internet via Ethernet with a theoretical bandwidth of 100 Mbit/s. To emulate a limited connection to the Internet, we restrict the downlink from the Internet to 5Mbit/s. As shown in Figure

30, we have four different clients who are running different applications. The clients were using Ubuntu 14.04 desktop with Intel 1.80 GHz i5 processor acting as a server and Ubuntu 14.04 desktop with 3.40 Ghz i7 processor. They were connected to a gateway through a LAN with the maximum available network bandwidth of 100 Mbps. while one of the clients is playing the game, there are three clients running YouTube, file uploading processes, and a remote connection. This is done to create competition for resources among different applications. Creation of the competition scenario allowed for minimum link sharing amongst different classes, thus allowing for a better performance analysis of the effect of GARM on game traffic. The cloud gaming traffic was generated using the in-house-streaming option available on Steam [109]. The video stream received on the client side was captured and analysed to assess QoE. The game selected for the experiment was Time of Dragons which is available on Steam. The video was captured using Avconv tool [110] available in Ubuntu. All packets entering the network interface are classified into six classes depending on their ToS, as shown in Table 16. In the resource policy enforcement module, each category is assigned a certain priority and a certain outgoing bandwidth, depending on the application's needs, and it can also be tailored to prioritize certain traffic over other traffic and reallocate bandwidth to specific application/category. In our case, we have chosen to prioritize the game-traffic and allocate more outgoing bandwidth to game packets. This is done to improve the Quality of Service (QoS) and Quality of Experience (QoE) of the gamer. The Optimization module is coded in Python which extensively makes use of the cvx's [111] base library to achieve proportional fairness amongst applications. To classify incoming packets, we use the Linux Traffic Control tool (tc) which helps to set the mechanism by which packets are received and transmitted on a network

interface. Resource policy enforcement directly interacts with the network adapter; therefore, it can shape the network traffic. It can be used to define a queuing discipline for the packets waiting to be dequeued at the network driver. The queuing discipline (qdisc) supported by Linux enables the traffic control (tc) which is vital to adjusting the QoS for competing flows. Packets that arrive at the network driver are filtered to one of the said six classes. Classification is done using the tc filter option and iptables, which is a generic table structure for the definition of rulesets [112]. It is used to set up and maintain the tables of Ipv4 packet filter rules. It verifies the packet header and classifies packets based on their IP address, destination port and source port. This feature has been used in GARM for the classification of packets; i.e., setting their ToS field. It is imperative to know the type of packet, destination port, source IP, destination IP of the packet to be able to perform QoS on the running applications.

6.3.2 Performance evaluation

We studied network delay, encoding delay, and decoding delay experienced by a user with and without GARM for two scenarios: 1) Single player and 2) multiplayer. In the single player scenario, one client is running the game while the other three clients are running YouTube, file uploading and remote connection. In the multiplayer scenario, three client are playing the game and the forth client is using YouTube. To evaluate the quality of the received video frames on the client side, we use the MSU Video Quality Measurement Tool [113] to record relevant QoE metrics, namely PSNR, SSIM, and the Video Quality Monitor (VQM) metric for the single player scenario.

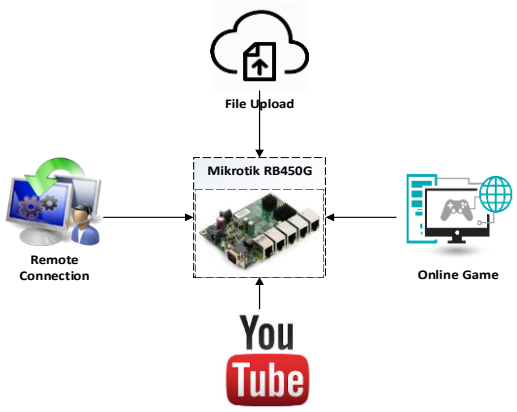


Figure 30.Experiment Setup

For the single player scenario, a client has to only communicate with the game server; however, for the multiplayer scenario, complex game mechanics are employed and the game state has to be rendered more frequently. Hence, it is extremely important in the case of multiplayer games for the packets to leave the client's network interface even faster. A small delay could give an undue advantage to the opponent, thus affecting player's QoE considerably.

Figure 31 presents the delay experienced by the gamer for scenarios 1 and 2. We can see that for single player, the network delay that a gamer would experience with GARM is nearly half of that without GARM. As FIFO is used by the router as the default queueing strategy, we compared GARM with the FIFO strategy. In the multiplayer scenario, network delay was reduced by 40% when GARM was used. These improvements stem from the fact that GARM ensures that game packets get a minimum amount of bandwidth and can still borrow more if required. The fixed minimum allocation of bandwidth allows the user to experience less delay; however when GARM is not running there is no fixed minimum allocation of bandwidth. This can lead to packet loss if background applications are more resource-demanding, thus affecting the player's QoE.

Traffic shaping and network prioritization also affected the decoding delay. When GARM was turned on, decoding delay was reduced by 21% and encoding delay was reduced by 36% as shown in Figure 32a and b. The reduction in decoding delays is likely caused by the reduction in video frame inter arrival time, causing the frame to complete faster. Since the receiver cannot display the frame until the last packet of that frame arrives, reducing the inter-arrival time of packets has an impact on decoding delay.

Beside delay, video streaming quality directly affects QoE; therefore we compared the quality of the frames with and without GARM. The game was recorded in both cases and external parameters (game surroundings, character etc.) were kept constant throughout the experiment. Recorded videos of the game running with and without GARM were compared against the server-side recorded game, which served as the original reference for comparison purposes. The PSNR value in our experiment was observed to be 8% higher when GARM is applied (Figure 33). Furthermore, the SSIM values also improved by 2% for the game running with GARM (Figure 34). Finally, VQM was also improved by 5% when running with GARM (Figure 35).

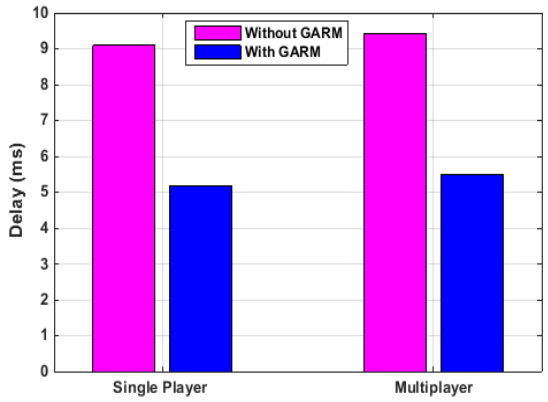


Figure 31.Delay experienced by the gamer

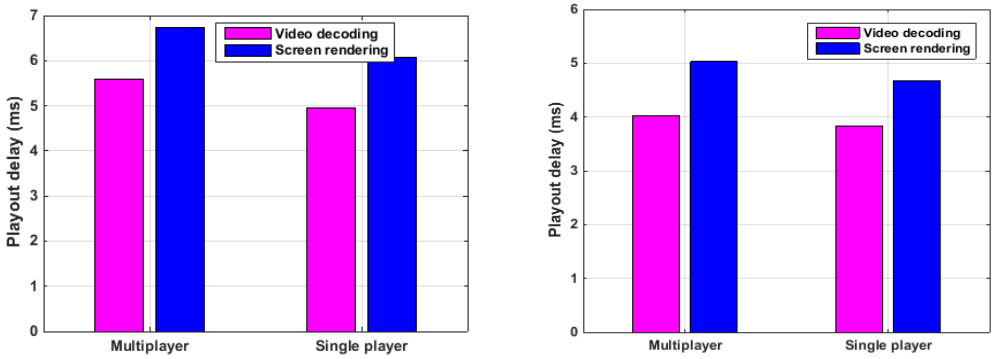


Figure 32.Processing delay Experienced by gamer a) without GARM and b) with GARM

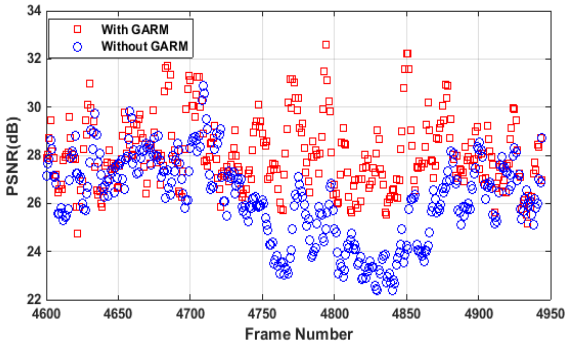


Figure 33. PSNR versus frame number

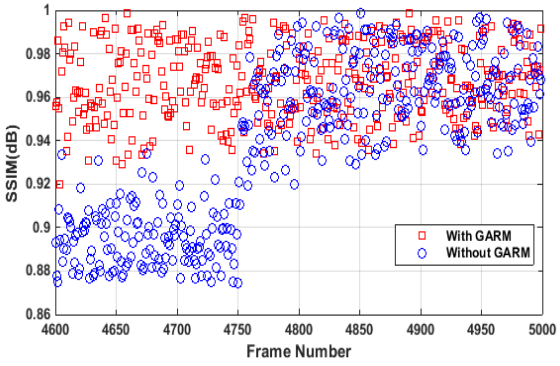


Figure 34.SSIM versus frame number

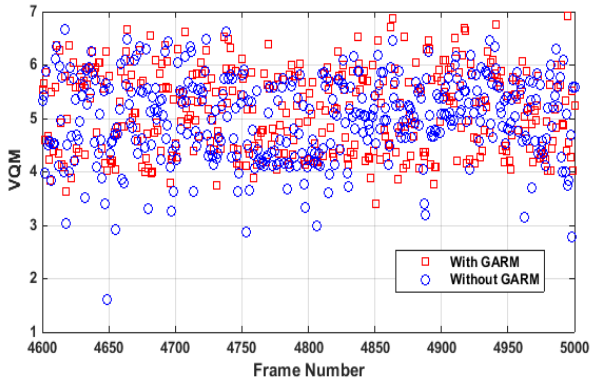


Figure 35.VQM versus frame number

6.4 Summary

We presented an optimization-based method to fairly allocate bandwidth to competing applications sharing a single home gateway. More specifically, we present the logarithmic-based utility function that attempts first, sharing the bandwidth resource according to the utility proportional fairness policy and second, maximizing the overall bandwidth utilization. The proposed GARM monitors network flows to understand the nature of applications, assesses available bandwidth, and intelligently assigns home network resources to each specific application by giving precedence to game flows over the other flows without sacrificing fairness. We have shown that the proposed GARM method, on average, outperforms existing conventional strategy by reducing the delay experienced by user, and by improving the user's QoE through improvements to video quality.

Chapter 7. Conclusion and Future Work

In this chapter, we conclude the work presented in this thesis and propose future research directions.

7.2 Conclusion

Gaming on demand is an emerging service that combines techniques from Cloud Computing and Online Gaming. This new paradigm is garnering prominence in the gaming industry and leading to a new "anywhere and anytime" online gaming model. Despite its advantages, cloud gaming's QoE is challenged by high and varying end-to-end communication delay. Since the significant part of the computational processing, including game rendering and video compression, is performed on the cloud, properly allocating game requests to the geographically distributed DCs can lead to QoE improvements resulting from lower delays.

To this end, the overall focus of this thesis was on improving the gamer's QoE from two main network perspectives—cloud perspective, where mechanisms for QoE enhancement must be observed and controlled by the cloud to optimally allocate data center resources to user requested gaming sessions, and home gateways perspective, where the network acquires gamers' QoE parameters and uses them to maximize the overall QoE of the clients that are sharing the gateway at home.

The extensive study of the current state of the art documented in Chapter 2 helped us identify the properties of cloud gaming architecture, obtain insights about the existing shortcomings of the current streaming, and delivery solutions, and understand the factors

influencing the perceived quality including network, device, game design, etc.

We investigated a novel method for minimizing the end-to-end latency within a cloud gaming data center. We formulated an optimization problem for reducing delay, and proposed a Lagrangean Relaxation (LR) time-efficient heuristic algorithm as a practical solution in Chapter 3. The proposed optimization model takes into consideration the type of requested games, current server loads, and current path delays to make decisions on which game server and communication path will minimize the delay within a data center. The proposed method was evaluated with three different scenarios; the results indicate that it can provide a close-to-optimal solution. Moreover, experimental results showed that the proposed method, on average, outperformed conventional algorithms (server-centric and network-centric) by minimizing the overall delay and delay variation (jitter) within data centers and exhibiting better performance compared to the conventional solutions in terms of fair resource allocation among gamers.

To consider the game specific characteristics in resource allocation, we proposed a bi-objective optimization method in Chapter 4 to find an optimum path for packet transmission within a data center by minimizing delay and maximizing bandwidth utilization. We used a metaheuristic model, called AHP, to solve the NP-complete optimization problem. The resulting method is an AHP-based game aware routing (AGAR) scheme that considers requested game type and requirements in terms of delay and bandwidth to select the best routing path for a game session in a cloud gaming network.

In Chapter 5, to complement the intra-DCN routing solutions for cloud gaming that we presented in Chapter 3 and 4, we introduced the allocation strategy for inter-DCN backbone that take into account both

requirements of a service (i.e. gaming service) and limitations and geographical distribution of DCs. In this regard, we proposed an optimized allocation mechanism that assigns incoming requests to geographically distributed data centers. The multi-objective optimization jointly enhanced bandwidth utilizations and minimized delay. The proposed allocation mechanism enables the gaming service provider to increase the overall bandwidth utilization by offloading the workload from saturated data centers onto ones with low bandwidth utilization. Moreover, although it is possible that the candidate DCs are farther from the requesting gamer and require more transport delay, this strategy may reduce the overall delays experienced by gamers. This can be achieved when the reduction in response delay achieved from the allocation of the new gaming session to the less burdened DC is comparable to the increase in transport delay imposed by the longer physical distance. However, applying such optimization algorithm requires having a central entity that can collect information from the underlying network and control network elements. Hence, we introduced a hierarchical SDN model for the selection of a DC for a new gaming session. We utilized the hierarchical model to consider transport delay, response delay and bandwidth status for the intra and inter-DCN traffic flows. In this configuration, each DCN is associated with a local SDN controller that monitors its delay, congestion, and bandwidth utilization. These local SDN controllers feed their information to the central SDN controller who will make decisions regarding the allocation of gaming sessions to the various data centers within the cloud gaming system.

Beyond intra and inter DCN routing, residential gateways play a key role in providing internet access to home consumers, and users in the same home with heterogeneous applications sharing a common

gateway. As such, the gateway becomes the bandwidth bottleneck, leading to impairments and negatively affecting users' QoE. In the case of delay sensitive applications like video streaming and online gaming, this impairment becomes more crucial. Therefore, in Chapter 6, we discussed an SDN-enabled optimization-based scheme for optimally sharing the bandwidth among network flows within a residential gateway. We targeted online game flows and attempted to optimize the QoE of the gamers while avoiding the starvation of other traffic flows. Our optimization model considers the nature of network flows, and aims to maximize the bandwidth utilization.

7.2 Topics for future works

In this section, we discuss interesting possibilities of extension of the current work.

In connection with the area of resource allocation in cloud gaming, there are a number of extensions that can further enhance the performance of the current proposed approaches. We can enhance our online convex method by introducing a heuristic method to dynamically handle newly arriving and leaving gaming flows. We can apply machine learning techniques to automatically understand and predict both fluctuating workloads (requests) and changing environments. This rather provides opportunities for network operators to adapt their scheduling strategies to new data, and to learn how to efficiently deal with scheduling problems such as consolidating co-located workloads, and guaranteeing applications' QoS.

Given that the relative importance of QoS parameters is related to a game genre, and different game genres have different characteristics, investigating the impact of developing a distinct utility function for each

game genre, rather than applying one function for all, in optimization problems can potentially further ameliorate the current work.

In chapter 4, we applied a standard branch-and-bound algorithm to solve the generalized assignment problems in proportional time. However, there are other decomposition methods in which the problem is translated into new subproblems that are easier to solve, and as a result, the computational complexity can be further reduced more.

In the context of the Network Utility Maximization (NUM) framework proposed for the rate allocation in home gateways networks so-called GARM, there are lots of opportunities that can enhance the current work. We used Jain's index as our measure of fairness to construct the function, while there are various mathematical formulations of fairness based on different points of views, and different fairness notions. As satisfying all of them at the same time is shown to be impossible, investigating other fairness measures by incorporating them into the objective function could be an interesting research avenue. Furthermore, finding a good measure of fairness which results in a convex optimization problem for NUM, would allow us to inexpensively solve the NUM problem.

Energy consumption is an important factor for gamers using mobile devices. For instance, a gamer may prefer to play longer sessions with average quality than shorter periods with high-quality. Hence, we can define an objective to maximize the battery life duration. In this regard, the level of battery and its rate of depletion can also be reported as feedback along with other QoE related parameters to the central optimizer. Hence, depending on the battery level, the optimization process can maximize the quality of gaming session or conserve energy.

References

[1] S. Shirmohammadi, M. Abdalla, D. T. Ahmed, K.-T. Chen a.k.a. ShengWei Chen, Y. Lu, and A. Snyatkov, “Introduction to the Special Section on Visual Computing in the Cloud: Cloud Gaming and Virtualization,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1955–1959, Dec. 2015.

[2] R. Shea, Jiangchuan Liu, E. C.-H. Ngai, and Yong Cui, “Cloud gaming: architecture and performance,” *IEEE Netw.*, vol. 27, no. 4, pp. 16–21, 2013.

[3] A. Ojala and P. Tyrvainen, “Developing Cloud Business Models: A Case Study on Cloud Gaming,” *IEEE Softw.*, vol. 28, no. 4, pp. 42–47, Jul. 2011.

[4] “Research and Markets: Global Cloud Gaming Market 2015-2020 - Gaming Audience, Devices & Technology Analysis | Business Wire.” [Online]. Available: http://www.businesswire.com/news/home/20150708005534/en/Research-Markets-Global-Cloud-Gaming-Market-2015-2020. [Accessed: 27-Apr-2017].

[5] “Sony buys streaming games service OnLive only to shut it down - The Verge.” [Online]. Available: https://www.theverge.com/2015/4/2/8337955/sony-buys-onlive-only-to-shut-it-down. [Accessed: 16-Feb-2018].

[6] S. Choy, B. Wong, G. Simon, and C. Rosenberg, “A hybrid edge-cloud architecture for reducing on-demand gaming latency,” *Multimed. Syst.*, vol. 20, no. 5, pp. 503–519, 2014.

[7] “Gaikai will be fee-free, utilize 300 data centers in the US.” [Online]. Available: https://www.engadget.com/2010/03/11/gaikai-will-be-fee-free-utilize-300-data-centers-in-the-us/. [Accessed: 27-Apr-2017].

[8] M. Claypool and K. Claypool, “Latency can kill: precision and deadline in online games,” *In Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, pp. 215-222. ACM, 2010.

[9] M. Claypool and K. Claypool, “Latency and player actions in online games,” *Commun. ACM*, vol. 49, no. 11, p. 40, Nov. 2006.

[10] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, “An Evaluation of QoE in Cloud Gaming Based on Subjective Tests,” in *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2011, pp. 330–335.

[11] S. Shi, C.-H. Hsu, K. Nahrstedt, and R. Campbell, “Using graphics rendering contexts to

enhance the real-time video coding for mobile cloud gaming," in *Proceedings of the 19th ACM international conference on Multimedia - MM '11*, 2011, p. 103.

[12] N. Tizon, C. Moreno, M. Cernea, and M. Preda, "MPEG-4-based adaptive remote rendering for video games," in *Proceedings of the 16th International Conference on 3D Web Technology - Web3D '11*, 2011, p. 45.

[13] M. Semsarzadeh, A. Yassine, and S. Shirmohammadi, "Video Encoding Acceleration in Cloud Gaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1975–1987, Dec. 2015.

[14] S. Xiang, L. Cai, and J. Pan, "Adaptive scalable video streaming in wireless networks," in *Proceedings of the 3rd Multimedia Systems Conference on - MMSys '12*, 2012, p. 167.

[15] M. Claypool, R. Eg, and K. Raaen, "The Effects of Delay on Game Actions," in *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts - CHI PLAY Companion '16*, 2016, pp. 117–123.

[16] M. Armbrust *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 28(13), p.2009.*

[17] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected Cloud Computing Environments," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 1–47, May 2014.

[18] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 2861–2865.

[19] M. Claypool and K. Claypool, "Latency and player actions in online games," *Commun. ACM*, vol. 49, no. 11, p. 40, 2006.

[20] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Commun. Mag.*, vol. 40, no. 12, pp. 50–55, Dec. 2002.

[21] S. Blake, T. Networking, D. Black, and C. EMC, "RFC 2475 - An Architecture for Differentiated Services," *IETF - Network Working Group*, 1998. [Online]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc2475.txt.pdf.

[22] V. P. Kumar, T. V. Lakshman, and D. Stiliadis, "Beyond best effort: Router architectures

for the differentiated services of tomorrow's internet," *IEEE Commun. Mag.*, vol. 36, no. 5, pp. 152–164, 1998.

[23] "Global Cloud Gaming Market – Trends & Forecast, 2015-2020 | | Infoholic Research." [Online]. Available: https://www.infoholicresearch.com/global-cloud-gaming-market-trends-forecast-2015-2020/. [Accessed: 25-Jan-2018].

[24] M. Farokhmanesh, "Doom Eternal is coming to Google's cloud gaming service, Stadia," The Verge, [Online]. Available: https://www.msn.com/en-ca/entertainment/gaming/doom-eternal-is-coming-to-googles-cloud-gaming-service-stadia/ar-BBUYlOw. [Accessed: 25-Jan-2018].

[25] H. Techno, "Google's Stadia Gaming Platform," HighlyTechno, [Online]. Available: hhttps://www.theverge.com/2019/3/20/18273977/google-stadia-cloud-game-streaming-service-report. [Accessed: 25-May-2019].

[26] W. Cai, M. Chen, and V. C. M. Leung, "Toward Gaming as a Service," *IEEE Internet Comput.*, vol. 18, no. 3, pp. 12–18, May 2014.

[27] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "GamingAnywhere," in *Proceedings of the 4th ACM Multimedia Systems Conference on - MMSys '13*, 2013, pp. 36–47.

[28] W. Cai, Z. Hong, X. Wang, H. C. B. Chan, and V. C. M. Leung, "Quality-of-Experience Optimization for a Cloud Gaming System With <italic>Ad Hoc</italic> Cloudlet Assistance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 2092–2104, Dec. 2015.

[29] P. B. Beskow, P. Halvorsen, and C. Griwodz, "Latency Reduction in Massively Multi-player Online Games by Partial Migration of Game State," in *2nd Int Conf on Internet Technologies and Applications*, 2007, pp. 153–163.

[30] U. Lampe, Q. Wu, S. Dargutev, R. Hans, A. Miede, and R. Steinmetz, "Assessing Latency in Cloud Gaming," Springer, Cham, 2014, pp. 52–68.

[31] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Annual Workshop on Network and Systems Support for Games*, 2012.

[32] S. Shi, C.-H. Hsu, K. Nahrstedt, and R. Campbell, "Using graphics rendering contexts to

enhance the real-time video coding for mobile cloud gaming," in *Proceedings of the 19th ACM international conference on Multimedia - MM '11*, 2011, p. 103.

[33] V. Jalaparti, M. Caesar, S. Lee, J. Pang, and J. Van Der Merwe, "SMOG: A Cloud Platform for Seamless Wide Area Migration of Online Games.", In Proceedings of the 11th Annual Workshop on Network and Systems Support for Games, p. 9. IEEE Press, 2012.

[34] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Placing Virtual Machines to Optimize Cloud Gaming Experience," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 42–53, Jan. 2015.

[35] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference - IMC '09*, 2009, p. 202.

[36] M. Claypool, D. Finkel, A. Grant, and M. Solano, "Thin to win? Network performance analysis of the OnLive thin client game system," in *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2012, pp. 1–6.

[37] M. Manzano, M. Urueña, M. Sužnjević, E. Calle, J. A. Hernández, and M. Matijasevic, "Dissecting the protocol and network traffic of the OnLive cloud gaming platform," *Multimed. Syst.*, vol. 20, no. 5, pp. 451–470, Oct. 2014.

[38] M. Suznjevic, L. Skorin-Kapov, and M. Matijasevic, "The Impact of User, System, and Context factors on Gaming QoE: a Case Study Involving MMORPGs.", In Proceedings of Annual Workshop on Network and Systems Support for Games, pp. 1-6. IEEE Press, 2013.

[39] M. Claypool and Mark, "Motion and scene complexity for streaming video games," in *Proceedings of the 4th International Conference on Foundations of Digital Games - FDG '09*, 2009, p. 34.

[40] M. Suznjevic, J. Beyer, L. Skorin-Kapov, S. Moller, and N. Sorsa, "Towards understanding the relationship between game type and network traffic for cloud gaming," in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2014, pp. 1–6.

[41] A. Beznosyk, P. Quax, K. Coninx, and W. Lamotte, "Influence of network delay and jitter on cooperation in multiplayer games," in *Proceedings of the 10th International*

Conference on Virtual Reality Continuum and Its Applications in Industry - VRCAI '11, 2011, p. 351.

[42] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM '10*, 2010, vol. 40, no. 4, p. 63.

[43] M. Amiri, H. Al Osman, S. Shirmohammadi, and M. Abdallah, "Toward Delay-Efficient Game-Aware Data Centers for Cloud Gaming," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 12, no. 5s, pp. 1–19, Sep. 2016.

[44] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter, "PAST," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies - CoNEXT '12*, 2012, p. 49.

[45] H. Xu and B. Li, "A General and Practical Datacenter Selection Framework for Cloud Services," in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 9–16.

[46] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, "End-to-End Delay Minimization for Scientific Workflows in Clouds under Budget Constraint," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 169–181, Apr. 2015.

[47] P. Wendell *et al.*, "DONAR," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, p. 231, Aug. 2010.

[48] K. Boloor, R. Chirkova, Y. Viniotis, and T. Salo, "Dynamic Request Allocation and Scheduling for Context Aware Applications Subject to a Percentile Response Time SLA in a Distributed Cloud," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 2010, pp. 464–472.

[49] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM - SIGCOMM '13*, 2013, vol. 43, no. 4, p. 15.

[50] S. Jain *et al.*, "B4," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM - SIGCOMM '13*, 2013, vol. 43, no. 4, p. 3.

[51] W. Li, H. Qi, K. Li, I. Stojmenovic, and J. Lan, "Joint Optimization of Bandwidth for Provider and Delay for User in Software Defined Data Centers," *IEEE Trans. Cloud Comput.*, vol. 5, no. 2, pp. 331–343, Apr. 2017.

[52] J. M. Wang, Y. Wang, X. Dai, and B. Bensaou, “SDN-based Multi-Class QoS Guarantee in Inter-Data Center Communications,” *IEEE Trans. Cloud Comput.*, pp. 1–1, 2016.

[53] T. Benson, A. Anand, A. Akella, and M. Zhang, “MicroTE,” in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies on - CoNEXT ’11*, 2011, pp. 1–12.

[54] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in SDN-OpenFlow networks,” *Comput. Networks*, vol. 71, pp. 1–30, Oct. 2014.

[55] A. Ghosh, S. Ha, E. Crabbe, and J. Rexford, “Scalable Multi-Class Traffic Management in Data Center Backbone Networks,” *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2673–2684, Dec. 2013.

[56] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, “The cost of a cloud,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, p. 68, Dec. 2008.

[57] T. Hobfeld, R. Schatz, M. Varela, and C. Timmerer, “Challenges of QoE management for cloud applications,” *IEEE Commun. Mag.*, vol. 50, no. 4, pp. 28–36, Apr. 2012.

[58] R. Serral-Gracì, E. Cerqueira, M. Curado, M. Yannuzzi, E. Monteiro, and X. Masip-Bruin, “An Overview of Quality of Experience Measurement Challenges for Video Applications in IP Networks.”

[59] M.Jarschel, D. Schlosser, S.Scheuring, & Hoßfeld, “Gaming in the clouds: QoE and the users’ perspective.”, *Mathematical and Computer Modelling 57, no. 11-12 (2013): 2883-2894*

[60] J. Beyer and S. Möller, “Assessing the Impact of Game Type, Display Size and Network Delay on Mobile Gaming QoE.”, *PIK-Praxis der Informationsverarbeitung und Kommunikation 37, no. 4 (2014): 287-295.*

[61] I. Slivar, M. Suznjevic, L. Skorin-Kapov, and M. Matijasevic, “Empirical QoE study of in-home streaming of online games,” in *13th Annual Workshop on Network and Systems Support for Games*, 2014, pp. 1–6.

[62] C.-Y. Huang, C.-H. Hsu, D.-Y. Chen, and K.-T. Chen, “Quantifying User Satisfaction in Mobile Cloud Games,” *Proc. Work. Mob. Video Deliv.*, p. 4, 2014.

[63] S. Wang and S. Dey, “Cloud mobile gaming: modeling and measuring user experience in mobile wireless networks,” *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 16, no.

1, p. 10, Jul. 2012.

[64] "FRAPS game capture video recorder fps viewer." [Online]. Available: http://www.fraps.com/. [Accessed: 12-Feb-2018].

[65] J. Fox, *Applied regression analysis and generalized linear models*. Sage Publications, 2015.

[66] "Measuring user quality of experience for a streaming media service," U.S. Patent 9,479,562, issued October 25, 2016.

[67] C.-Y. Huang, P.-H. Chen, Y.-L. Huang, K.-T. Chen, and C.-H. Hsu, "Measuring the client performance and energy consumption in mobile cloud gaming," *Proceedings of the 13th Annual Workshop on Network and Systems Support for Games*. IEEE Press, p. 5, 2014.

[68] G. Baier, E. Köhler, and M. Skutella, "On the k-Splittable Flow Problem," Springer, Berlin, Heidelberg, 2002, pp. 101–113.

[69] C. A. Floudas and P. M. (Panos M. . Pardalos, *Encyclopedia of optimization*. *Vol. 1. Springer Science & Business Media, 2001*

[70] A. M. Geoffrion, "Lagrangean relaxation for integer programming," *Approaches to Integer Programming*, vol. 2. pp. 82–114, 1974.

[71] M. Guignard, "Lagrangean relaxation," *Top*, vol. 11, no. 2, pp. 151–200, Dec. 2003.

[72] L. Chalmet and L. Gelders, "Langrangean relaxations for a generalised assignments-type problem." pp. 103–109, 01-Jan-1976.

[73] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Math. Program.*, vol. 6, no. 1, pp. 62–88, Dec. 1974.

[74] "CVX: Matlab Software for Disciplined Convex Programming | CVX Research, Inc." [Online]. Available: http://cvxr.com/cvx/. [Accessed: 14-Feb-2017].

[75] "POX Wiki - Open Networking Lab - Confluence." [Online]. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki. [Accessed: 27-Apr-2017].

[76] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l*, 2005, pp. 250–257.

[77] M. Amiri, H. Al Osman, S. Shirmohammadi, and M. Abdallah, “An SDN Controller for Delay and Jitter Reduction in Cloud Gaming,” in *Proceedings of the 23rd ACM international conference on Multimedia - MM ’15*, 2015, pp. 1043–1046.

[78] R. Wang, D. Butnariu, and J. Rexford, “OpenFlow-Based Server Load Balancing Gone *Wild.”, Hot-ICE 11 (2011): 12-12.*

[79] C. . Asia-Pacific Signal and Information Processing Association. Annual Summit and Conference (2012 : Hollywood and Asia-Pacific Signal and Information Processing Association. 2012 Annual Summit and Conference International Organizing Committee., “OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over,” 2012.

[80] M. Al-Fares, A. Loukissas, A. Vahdat, M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication - SIGCOMM ’08*, 2008, vol. 38, no. 4, p. 63.

[81] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, “The cost of a cloud,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, p. 68, Dec. 2008.

[82] T. L. Saaty, “How to make a decision: The analytic hierarchy process,” *European journal of operational research48, no. 1 (1990): 9-26.*

[83] J. A. ALONSO and M. T. LAMATA, “CONSISTENCY IN THE ANALYTIC HIERARCHY PROCESS: A NEW APPROACH,” *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 14, no. 04, pp. 445–459, Aug. 2006.

[84] A. Greenberg *et al.*, “VL2,” in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication - SIGCOMM ’09*, 2009, vol. 39, no. 4, p. 51.

[85] He, Keqiang, Eric Rozner, Kanak Agarwal, Wes Felter, John Carter, and Aditya Akella. "Presto: Edge-based load balancing for fast datacenter networks." *ACM SIGCOMM* Computer Communication Review 45, no. 4 (2015): 465-478.

[86] Al-Fares, Mohammad, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. "Hedera: dynamic flow scheduling for data center networks." In Nsdi, vol. 10, no. 2010.

[87] S. Kullback and R. A. Leibler, “On Information and Sufficiency,” *Ann. Math. Stat.*, vol.

22, no. 1, pp. 79–86, Mar. 1951.

[88] T. Homem-de-Mello, “A Study on the Cross-Entropy Method for Rare-Event Probability Estimation,” *INFORMS J. Comput.*, vol. 19, no. 3, pp. 381–394, Aug. 2007.

[89] R. Y. Rubinstein and R. Y., “Cross-entropy and rare events for maximal cut and partition problems,” *ACM Trans. Model. Comput. Simul.*, vol. 12, no. 1, pp. 27–53, Jan. 2002.

[90] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, “Making middleboxes someone else’s problem,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, p. 13, Sep. 2012.

[91] A. Dixit *et al.*, “Towards an elastic distributed SDN controller,” in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN ’13*, 2013, vol. 43, no. 4, p. 7.

[92] S. Shalev-Shwartz, “Online Learning and Online Convex Optimization,” *Found. Trends® Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[93] S. G. Shakkottai and R. (Rayadurgam) Srikant, *Network optimization and control*. Now Publishers, 2008.

[94] R. Srikant, The mathematics of Internet congestion control. Springer Science & Business Media; 2012 Dec 6.

[95] M. Claypool, “The effect of latency on user performance in Real-Time Strategy games,” *Comput. Networks*, vol. 49, no. 1, pp. 52–70, Sep. 2005.

[96] L. Pantel and L. C. Wolf, “On the impact of delay on real-time multiplayer games,” in *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video - NOSSDAV ’02*, 2002, p. 23.

[97] W. Murray and K.-M. Ng, “An algorithm for nonlinear optimization problems with binary variables,” *Comput. Optim. Appl.*, vol. 47, no. 2, pp. 257–288, Oct. 2010.

[98] T. Chen, S. Member, Q. Ling, S. Member, and G. B. Giannakis, “An Online Convex Optimization Approach to Dynamic Network Resource Allocation.”

[99] Chen, Tianyi, Qing Ling, and Georgios B. Giannakis. "An online convex optimization approach to proactive network resource allocation." *IEEE Transactions on Signal Processing*, no. 24 (2017): 6350-6364.

[100] “iPerf - The TCP, UDP and SCTP network bandwidth measurement tool.” [Online]. Available: https://iperf.fr/. [Accessed: 03-May-2017].

[101] M. Amiri, A. Sobhani, H. Al Osman, and S. Shirmohammadi, “SDN-Enabled Game-Aware Routing for Cloud Gaming Datacenter Network,” *IEEE Access*, vol. 5, pp. 18633–18645, 2017.

[102] “Steam Workshop :: 6 Player Jungle MOBA v1.1.” [Online]. Available: https://steamcommunity.com/sharedfiles/filedetails/?l=czech&id=708909245. [Accessed: 21-Jan-2019].

[103] M. Noormohammadpour and C. S. Raghavendra, “DDCCast: Meeting Point to Multipoint Transfer Deadlines Across Datacenters using ALAP Scheduling Policy,” Jul. 2017.

[104] H. Zhang *et al.*, “Guaranteeing Deadlines for Inter-Data Center Transfers,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 579–595, Feb. 2017.

[105] B. Wong and E. G. Sirer, “ClosestNode.com,” *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, p. 62, Jan. 2006.

[106] Djokic and B., “Execution and processing time in computer performance measurements,” *Proc. 17th Conf. ACM Annu. Comput. Sci. Conf.*, pp. 435–435, 1989.

[107] Jain, Rajendra K., Dah-Ming W. Chiu, and William R. Hawe. "A quantitative measure of fairness and discrimination." Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA (1984).

[108] “RouterBoard.com : RB450G.” [Online]. Available: https://routerboard.com/rb450g. [Accessed: 23-Jan-2017].

[109] “Steam In-Home Streaming.” [Online]. Available: http://store.steampowered.com/streaming/. [Accessed: 13-Jan-2017].

[110] “Avconv tool.” [Online]. Available: https://git.libav.org/?p=libav.git. [Accessed: 13-Jan-2017].

[111] “CVX Research, Inc. | Software for Disciplined Convex Programming.” [Online]. Available: http://cvxr.com/. [Accessed: 13-Jan-2017].

[112] “Man page of IPTABLES.” [Online]. Available: http://ipset.netfilter.org/iptables.man.html. [Accessed: 13-Jan-2017].

[113] “MSU Video Group / Video data filtering and compression.” [Online]. Available: http://www.compression.ru/video/. [Accessed: 13-Jan-2017].