Towards Delay-Efficient Game-Aware Data Centers For Cloud Gaming

MARYAM AMIRI, University of Ottawa
HUSSEIN AL OSMAN, University of Ottawa
SHERVIN SHIRMOMHAMMADI, University of Ottawa
MAHA ABDALLAH, Sorbonne Universités, UPMC Univ Paris 06

Gaming on demand is an emerging service that has recently started to garner prominence in the gaming industry. Cloud based video games provide affordable, flexible and high performance solution for end-users with constrained computing resources and enables them to play high-end graphic games on low-end thin clients. Despite its advantages, cloud gaming’s Quality of Experience (QoE) suffers from high and varying end-to-end delay. Since the significant part of computational processing, including game rendering and video compression, is performed in data centers, controlling the transfer of information within the cloud has an important impact on the quality of cloud gaming services. In this paper, a novel method for minimizing the end-to-end latency within a cloud gaming data center is proposed. We formulate an optimization problem for reducing delay, and we also propose a Lagrangian Relaxation (LR) time-efficient heuristic algorithm as a practical solution. Simulation results indicate that the heuristic method can provide close-to-optimal solutions. Also, the proposed model reduces end-to-end delay and delay variation by almost 11% and 13.5%, respectively, and outperforms the existing server-centric and network-centric models. As a byproduct, our proposed method also achieves better fairness among multiple competing players by almost 45% on average in comparison with existing methods.

Categories and Subject Descriptors: K.8.0 [Personal Computing]: General – games; C.2.4 [Computer-Communication Networks]: Distributed Systems - Client/server

Additional Key Words and Phrases: Cloud gaming; Software Defined Network (SDN);

ACM Reference Format:

1. INTRODUCTION

Cloud gaming [Shirmohammadi, et al. 2015], 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...

Cloud gaming is considered among the most profitable cloud computing services today. The revenue from cloud gaming services will reach 476 million US Dollars by the end of 2015 and is expected to rise to 650 million US Dollars by the end of 2020 [Tariq Shaik. 2015].

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 1. 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 [Huang, et al. 2013]. Also, cloud games introduce about 1.7 times higher latency than their console counterparts [James Wang. 2012], which also could adversely impact the players’ QoE. In fact, cloud games typically require a maximum network latency of 80 msec, a requirement that only 70% of end-users are able to meet [Choy, et al. 2014].

Cloud Gaming

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) [David Stonecipher. 2012].

Since public cloud infrastructures (e.g Amazon) have failed to guarantee the QoS requirements of cloud gaming systems [Choy, et al. 2014], some of the existing cloud gaming vendors, like Gaikai, have invested in their own proprietary data centers [Nelson. 2010]. 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

ACM Transactions on xxxxxxxx, Vol. xx, No. x, Article x, Publication date: Month YYYY
delay [Claypool and Claypool. 2010, Claypool, et al. 2012, Dick, et al. 2005]. All of these factors must be taken into account when planning and deploying a data center for cloud gaming [Jarschel, et al. 2013]. 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 [Shi, et al. 2011, Tizon, et al. 2011, Semsarzadeh, et al. 2014, Cai and Leung. 2012, Sun and Wu. 2015, Slivar, et al. 2015]. Only a handful of works consider network routing or/and resource management.

In our previous work, we proposed 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. 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 [Amiri, et al. 2015a]. However, the high complexity of the method proposed in [Amiri, et al. 2015a] places high computational constraints on the SDN controller. Therefore, in this paper, we propose a Lagrangian Relaxation (LR) heuristic method to solve our 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 in our heuristic method. Such model is essential since a reduction in the bit rate can have an adverse effect on the QoE.

The rest of this paper is organized as follows: section 1 discusses relevant background and related work; section 2 presents the proposed user utility and processing delay models; section 4 details the optimization method presented in [Amiri, et al. 2015a]; section 5 introduces the heuristic method that reduces the computational complexity of the optimization method of section 4; section 6 shows our performance evaluation results; and section 7 concludes the papers and discusses future research avenues.

2. BACKGROUND AND RELATED WORK

2.1 Effect of Network Latency and Jitter

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 [Claypool and Claypool. 2006]. Also, a delay of less than 100 msec is highly desirable for high action paced games, whereas 150 msec is the delay threshold for slow paced games [Jarschel, et al. 2013]. In general, previous works have found that a delay of more than 100 msec becomes noticeable by gamers [Beskow, et al. 2007a].
and has a negative impact on the QoE. Therefore, ideally, the network delay threshold should be set at 80 msec to account for the additional computational processing delay on game servers [Choy, et al. 2014]. In addition to latency, the amount of latency variation, known as jitter, also has detrimental effects on gamers’ QoE [Cai and Leung. 2012, Dick, et al. 2005]. In [Lampe, et al. 2014], 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 msec in cloud games. However, a jitter of more than 70 msec has a significant adverse effect on players’ QoE [Beskow, et al. 2007a].

2.2 Cloud Gaming Delay Reduction Methods

The overall delay in cloud gaming service consists of processing delay and network delay. An empirical study of a data center network consisting of 150 inter-switches and 1500 servers has shown that 15% congestion situations can last for more than 100 seconds while many network links are often underutilized [Kandula, et al. 2009]. We divide the delay reduction solutions for cloud gaming services into two broad groups:

- Game processing techniques for delay reduction through game engine, video rendering and coding optimizations.
- Networking techniques consisting of server selection and resource provisioning methods.

In terms of video coding delay reduction, various works can be found that employ video compression techniques such as wrapping motion estimation, view compensation and object information extraction to facilitate video compression [Hemmati, et al., Shi, et al. 2011]. Beside the video encoding techniques, there have been a number of studies that focus on game hosting server selection schemes within cloud gaming data centers. The heuristic method presented in [Choy, et al. 2014] is an example of such scheme. It introduces a voting-based game-placement strategy in which end users vote for a game to be hosted on a server in order to increase gamers’ coverage. Similarly, Beskow et al. propose another server selection method to find the optimal servers to cover a group of gamers belonging to the same geographic region with the goal of minimizing latency in data transfer [Beskow, et al. 2007b]. Chen et al. present a method for allocating virtual machines (VM) to physical machines (PM) in the cloud gaming infrastructure that tackles the challenge of trade-off between users’ QoE and providers’ net profit. They propose a heuristic method that maximizes the net profit of cloud gaming providers while maintaining a just good enough QoE level for gamers [Hong, et al. 2014].

Other works focus on delay reduction for cloud gaming services using the Software Defined Networking (SDN) paradigm. In our previous work, we propose a load sharing method implemented on an SDN network controller to tackle the problem of latency reduction between game servers and core switches in a cloud gaming data center [Amiri, et al. 2015b]. In [Beskow, et al. 2007a], the authors present a method to routing the game users to the nearest game server among geographically distributed servers. Also, in [Amiri, et al. 2015a], a Linear Programing (LP) optimization-based method is proposed for optimally assigning game servers to gaming sessions and selecting the best communication path within a cloud gaming datacenter.
2.3 Data Center Architecture

With a global increasing trend in data center workload and IP traffic, issues pertaining to the underlying network architecture of data centers as well as data transfer within their boundaries have garnered considerable attention from the industry and academia alike [Wang, et al. 2015, Alizadeh, et al. 2011, Wilson, et al. 2011, Vamanan, et al. 2012].

It is estimated that the global data center IP traffic reached 3.4 ZB/year by the end of 2014 and is expected to rise to 10.4 ZB by the end of 2019. Also, the overall workload of cloud data centers is expected to grow more than three times over the next five years [Cisco. 2015].

While various network architectures have been proposed, the simple 3-tier model is commonly used in modern data centers. This model consists of three layers: core, aggregation and access layers as shown in figure 2. Using this architecture, the scalability of a data center is no longer dependent on its topology. A data center can be scaled up by scaling the switches. Additionally, VL2 [Greenberg, et al. 2009], PortLand [Niranjan Mysore, et al. 2009], BCube [Guo, et al. 2009] and fat-tree [Al-Fares, et al. 2008] are alternative data center network architectures that have been proposed recently [Cisco. 2008].

2.4 Software Defined Networking and Cloud Gaming

Although several methods have been proposed to minimize the computational and network cost within a data center, these methods usually take into consideration either network status or computational resource status of servers [Wang, et al. 2012, Beloglazov and Buyya. 2010, Liu, et al. 2009, Greenberg, et al. 2008]. Moreover, they determine the optimal solution based on static network metrics, and they often fail to take into consideration 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.

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. This adaptability 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 in order to not only accommodate current network conditions, but also optimize QoS parameters.

![Fig. 2 Typical Datacenter 3-tier Model](image-url)
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.

3.1 Utility Model

Game delay sensitivity is dependent on various factors, one being the game genre [Claypool and Claypool. 2006]. 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 [Claypool. 2009], the same model derivation method can be applied to other games in these measured games’ genres.

We perform the study using GamingAnywhere (GA) [Huang, et al. 2013], 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 [March 2014]. We calculate PSNR and SSIM index as objective measures of the normalized gamers’ quality perception. The results of the measurements are plotted in figure 3. 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 [Fox. 2015, Funge, et al. 2011]. In fact, the sigmoidal (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) [Funge, et al. 2011, Lee, et al. 2005, Liu, et al. 2007]. We report the mean square error between the fitted curve and collected data for each game in table 1.

![Figure 3: PSNR and SSIM graphs](image)
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 3, where $l$ and $k$ are model parameters derived through regression and $r$ is the bit rate (see equation 1). Also, we denote $(\alpha_i)$ as a priority factor obtained from the derivative of the utility function $U(r)$ (see equation 2). Hence, $\alpha_i$ 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 paper are described in table 2.

$$U(r) = \frac{l}{1 + e^{-kr}}$$  \hspace{1cm} (1)

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

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

Since measurement studies in [Hong, et al. 2014] 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 $r_p$, hence $d_{p_{ij}}(r_p)$ can be modeled as follows, where the processing resource requirement of game $i$ and $u_i$, $\sigma_i$ and $\tau_i$ are model parameters that are derived from regression.

$$d_{p_{ij}}(r_p) = u_i/(1 + \sigma_i e^{-\tau_i r_p})$$  \hspace{1cm} (3)
### Table 2. Table of Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U(r)$</td>
<td>Utility function</td>
<td>$N_p$</td>
<td>number of available paths</td>
</tr>
<tr>
<td>$i$</td>
<td>Gaming session index</td>
<td>$j$</td>
<td>Gaming server index</td>
</tr>
<tr>
<td>$k$</td>
<td>Available paths index</td>
<td>$D_{max}$</td>
<td>Maximum tolerable delay</td>
</tr>
<tr>
<td>$d_{p ij}$</td>
<td>Processing delay of game $i$ servers by game server $j$</td>
<td>$c_j$</td>
<td>Capacity of server $j$</td>
</tr>
<tr>
<td>$d_{njk}$</td>
<td>Network delay of path $k$</td>
<td>$r_{bh}$</td>
<td>Required bandwidth of game $i$</td>
</tr>
<tr>
<td>$\alpha_i$</td>
<td>Weight factor of network delay for game $i$</td>
<td>$\beta_i$</td>
<td>Weight factor of processing delay for game $i$ (1 - $\alpha$)</td>
</tr>
<tr>
<td>$N_s$</td>
<td>number of active game servers</td>
<td>$z_{ijk}$</td>
<td>binary variable for selection of a path</td>
</tr>
<tr>
<td>$N_g$</td>
<td>number of gamers</td>
<td>$Z_{LR}$</td>
<td>Lagrange relaxation of $O_{total}$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Lagrangian multipliers parameter</td>
<td>$\delta_{ijk}$</td>
<td>$z_{ijk} - x_{ij}$</td>
</tr>
<tr>
<td>$\phi^t$</td>
<td>subgradient step size</td>
<td>$\gamma_t$</td>
<td>Subgradient scalar (between 0 to 2)</td>
</tr>
<tr>
<td>$t$</td>
<td>Number of iterations</td>
<td>$\rho$</td>
<td>Number of available ports on each switch</td>
</tr>
<tr>
<td>$\nu_i, \sigma_i, \tau_i$</td>
<td>Processing delay model parameters</td>
<td>$k, 1$</td>
<td>Utility function model parameters</td>
</tr>
</tbody>
</table>

### 4. PROPOSED 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 datacenter for a game session’s data streams. Hence, in order 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 OpenFlow protocol as explained in section (3.3). 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.

![Fig. 4. Game-Aware Optimization Method [Amiri, et al. 2015a]](image-url)
4.1 OPTIMIZATION PROBLEM FORMULATION

In our previous work [Amiri, et al. 2015a], we 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 (4).

$$O_{total} = \sum_{i}^{N} \sum_{j}^{Nv} ((\beta_{i} d_{ij} x_{ij}) + (\alpha_{i}) \sum_{k}^{Np} d_{nijk} z_{ijk})$$

(4)

Subject to:

ai. $$\sum_{i}^{N} r_{pi} x_{ij} \leq C_{sj} \quad \forall j \in \{1, ..., N_{g}\}$$

aii. $$\sum_{i}^{N} z_{ijk} = 1 \quad \forall j \in \{1, ..., N_{s}\}, \forall k \in \{1, ..., N_{p}\}$$

aiii. $$\sum_{i}^{N} x_{ij} \leq 1 \quad \forall i \in \{1, ..., N_{g}\}$$

aiv. $$\sum_{i}^{N} \sum_{k}^{Np} r_{pi} z_{ijk} \leq BW_{max} \quad \forall j \in \{1, ..., N_{g}\}$$

av. $$\sum_{i}^{N} d_{ij} x_{ij} + \sum_{k}^{Np} d_{nijk} z_{ijk} \leq D_{max} \quad \forall i \in \{1, ..., N_{g}\}, \forall j \in \{1, ..., N_{s}\}, \forall k \in \{1, ..., N_{p}\}$$

avii. $$z_{ijk} \in \{0, 1\} \quad \forall i \in \{1, ..., N_{g}\}, \forall j \in \{1, ..., N_{s}\}, \forall k \in \{1, ..., N_{p}\}$$

aviii. $$x_{ij} \in \{0, 1\} \quad \forall i \in \{1, ..., N_{g}\}, \forall j \in \{1, ..., N_{s}\}$$

In (4), we define two binary decision variable $$x_{ij}$$ and $$z_{ijk}$$. Binary variable $$z_{ijk}$$ is equal to 1 if the kth path is chosen among the $$N_{p}$$ available paths and 0 otherwise. Also, binary variable $$x_{ij}$$ will take value 1 if the jth server is selected to host the ith 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 (avii) restricts the total amount of delay (processing and network delay) to the maximum tolerable delay (Dmax) within the datacenter.

In section (3) we provided a model for the calculation of delay processing ($$d_{pij}(rp_{j})$$). Also, we provided the utility function ($$U(r)$$) to measure the priority factor ($$\alpha_{i}$$) for individual games.

For the network delay, we use the method proposed in [Amiri, et al. 2015b] to compute the network delay ($$d_{nijk}$$) using an SDN controller.

The objective function of (4) presents two splittable flow problems. Hence, the resolution of (4) is an NP hard problem [Baier, et al. 2002]. To address the complexity
issue of (4), we will propose a time-efficient heuristic method for the calculation of $O_{\text{total}}$ in section (5).

5. PROPOSED HURISTIC 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 Lagrangian 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 [Floudas and Pardalos. 2008]. In practice, the new problem resulting from LR is simpler than the original objective function through the construction of the lower bounds [Geoffrion. 1974].

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 obtain.

In section (5.1), we propose the Game Routing Heuristic (GRH) method based on the Lagrangian Heuristic approach. We dualize the constraint set (av) to generate the lower bounds. In addition, the subgradient optimization method can be employed to get the upper bounds.

5.1 Lagrange relaxation of $O_{\text{total}}$

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

$$Z_{LR}(\lambda) = \min \left( \Sigma_i^{N_s} \Sigma_j^{N_p} (\alpha_i d_{nijk} + \lambda_{ijk}) z_{ijk} + \Sigma_i^{N_s} \Sigma_j^{N_p} (\beta_i d_{pij} - \Sigma_k^{N_p} \lambda_{ijk}) x_{ij} \right)$$

Subject to:

i. $\Sigma_i^{N_s} r_i x_{ij} \leq C_{ij} \quad \forall \; j \in \{1, \ldots, N_g\}$

ii. $\Sigma_i^{N_s} z_{ijk} = 1 \quad \forall \; j \in \{1, \ldots, N_s\}, \forall \; k \in \{1, \ldots, N_p\}$

iii. $\Sigma_i^{N_s} x_{ij} \leq 1 \quad \forall \; i \in \{1, \ldots, N_g\}$

iv. $\Sigma_i^{N_s} \Sigma_k^{N_p} r_{bij} z_{ijk} \leq B W_{\text{Max}} \quad \forall \; j \in \{1, \ldots, N_s\}$

v. $\Sigma_i^{N_s} d_{pij} x_{ij} + \Sigma_k^{N_p} d_{nijk} z_{ijk} < D_{\text{Max}} \quad \forall \; i \in \{1, \ldots, N_g\}$

vi. $z_{ijk} \in \{0,1\} \quad \forall \; i \in \{1, \ldots, N_g\}, \forall \; j \in \{1, \ldots, N_s\}, \forall \; k \in \{1, \ldots, N_p\}$

vii. $x_{ij} \in \{0,1\} \quad \forall \; i \in \{1, \ldots, N_g\}, \forall \; j \in \{1, \ldots, N_s\}$

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:

$$Z_{LR1}(\lambda) = \min \Sigma_i^{N_s} \Sigma_j^{N_p} \Sigma_k^{N_p} (\alpha_i d_{nijk} + \lambda_{ijk}) z_{ijk} = \Sigma_i^{N_s} \Sigma_j^{N_p} (\min \Sigma_k^{N_p} (\alpha_i d_{nijk} + \lambda_{ijk}) z_{ijk})$$

Subject to:

i. $\Sigma_j^{N_s} \Sigma_k^{N_p} z_{ijk} = 1 \quad \forall \; i \in \{1, \ldots, N_g\}$
Subject to:

\[ \sum_{i}^{N_g} \sum_{j}^{N_p} r_{ij} z_{ijk} \leq BW_{Max} \quad \forall \, j \in \{1, ..., N_s\} \]

\[ z_{ijk} \in \{0,1\} \quad \forall \, i \in \{1, ..., N_g\}, \forall \, j \in \{1, ..., N_s\}, \forall \, k \in \{1, ..., N_p\} \]

Constraint (cii) can be considered as a Generalized Upper Bound (GUB) constraint such that the 1\textsuperscript{st} Lagrangian problem consists of \(N_g N_p\) multiple choice problems and can be solved in a time proportional to \(N_g N_p\) where \(i = \text{argmin}\{\alpha_i d_{ij} + \lambda_{ijk}\} \) for \(j = 1, ..., N_s\), \(k = 1, ..., N_p\) and \(z_{ijk} = 1\) [Guignard. 2003]. The second sub-problem is obtained by dualizing constraint (v), which consists of the term \(x_{ij}\), and is given as follows:

\[ Z_{LR2}(\lambda) = \min_{\sum_{i}^{N_g} \sum_{j}^{N_p} (\beta d_{ij} - \sum_{k}^{N_p} \lambda_{ijk}) x_{ij} } \]  

Subject to:

\[ \sum_{i}^{N_g} r_{ij} x_{ij} \leq C_{s_j} \quad \forall \, j \in \{1, ..., N_s\} \]

\[ \sum_{i}^{N_g} x_{ij} = 1 \quad \forall \, i \in \{1, ..., N_g\} \]

\[ x_{ij} \in \{0,1\} \quad \forall \, i \in \{1, ..., N_g\}, \forall \, j \in \{1, ..., N_s\} \]

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 \(Z_{LR2}(\lambda)\).

\[ z_{ijk} \leq \sum_{i}^{N_g} x_{ij} \quad \forall \, j \in \{1, ..., N_s\} \]

\[ \sum_{k}^{N_p} x_{ij} \geq 1 \quad \forall \, j \in \{1, ..., N_s\} \]

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

\[ Z_{LR2}(\lambda) = \min_{\sum_{i}^{N_g} \sum_{j}^{N_p} (\beta d_{ij} - \sum_{k}^{N_p} \lambda_{ijk}) x_{ij} } \]  

Subject to:

\[ \sum_{i}^{N_g} r_{ij} x_{ij} \leq C_{s_j} \quad \forall \, j \in \{1, ..., N_s\} \]

\[ \sum_{i}^{N_g} x_{ij} = 1 \quad \forall \, i \in \{1, ..., N_g\} \]

\[ x_{ij} \in \{0,1\} \quad \forall \, i \in \{1, ..., N_g\}, \forall \, j \in \{1, ..., N_s\} \]

\[ \sum_{i}^{N_g} x_{ij} \leq 1 \quad \forall \, i \in \{1, ..., N_g\} \]

\[ \sum_{i}^{N_g} x_{ij} \geq 1 \quad \forall \, j \in \{1, ..., N_s\} \]

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 [Chalmet and Gelders. 1976], it can be solved in time proportional to \(N_s \sum_{i}^{N_g} C_{s_j}\).

### 5.2 Determining \(\lambda\)

A subgradient optimization algorithm is used to iteratively adjust the value of the Lagrange multiplier \(\lambda\) [Held, et al. 1974] 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.
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 1.

**ALGORITHM 1:** Subgradient Optimization Algorithm

1: **Inputs:**
   \( \lambda_0 \): initial value for Lagrangian multipliers
   \( t \): number of iterations
   \( z_{LB}^* \): lower bound value
   \( V_{UB} \): upper bound value
   \( \gamma_0 \): scalar parameter
   \( \hat{\epsilon} \): error index

2: **Initialize** \( \lambda = 0, t = 0, t_{\text{max}} = 100, z_{LB}^* = -\infty, V_{UB} = +\infty, \gamma_0 = 2, \hat{\epsilon} = +\infty \),

3: **while** \( t \leq t_{\text{max}} \) **&** \( \hat{\epsilon} \geq 0.0001 \) **do**
   // computing lower bound
4:   **Solve** \( Z_{LR1}(\lambda_t), Z_{LR2}(\lambda_t) \), set \( \bar{X}_t \leftarrow \text{output } Z_{LR2}(\lambda_t) \) // feasible solution \( \bar{X}_t \in \{x_{ij} \mid i \in \{1, \ldots, N_g \}, \forall j \in \{1, \ldots, N_p \} \} \)
   Compute \( Z_{LB}(\lambda_t) = Z_{LR1}(\lambda_t) + Z_{LR2}(\lambda_t) \)
   Set \( z_{LB} = \max \{ z_{LB}^*, Z_{LB}(\lambda_t) \} \)
   // computing upper bound
5:   **Solve** \( O_{total} \forall x_{ij} \in \bar{X}_t, \text{set } \bar{Z}_t \leftarrow \text{output } O_{total} \) // the optimal value of \( \bar{Z}_t \) can be found by setting \( Z_{ijk} = 1 \) where \( i = \arg \min (\alpha d_{ijk}) \) for \( j = 1, \ldots, N_s, k = 1, \ldots, N_p \).
   Compute \( \bar{O}_{total} = O_{total}(\bar{X}_t, \bar{Z}_t) \)
   Set \( V_{UB} = \min \{ V_{UB}, \bar{O}_{total} \} \)

6:   \( \left\| \delta_{ijk} \right\| = z_{ijk} - x_{ij} \) // positive scalar called step size in the \( t \)th iteration based on Fisher’s formula [Fisher, 1985]
   \( \gamma \left( V_{UB} - z_{LB} \right) \)
   Calculate \( \phi^t = \gamma \left( V_{UB} - z_{LB} \right) \)
   \( \lambda_{ijk}^{t+1} = \max \{ \lambda_{ijk} + \phi^t \delta_{ijk}, 0 \} \) // \( \forall i \in \{1, \ldots, N_g \}, \forall j \in \{1, \ldots, N_p \}, \forall k \in \{1, \ldots, N_p \} \)
   \( \hat{\epsilon} = | V_{UB} - z_{LB} | \)
   \( t = t + 1 \)
7: **End while;**

### 6. EXPERIMENTAL SETUP AND EVALUATION

#### 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 (4) [Amiri, et al. 2015a] (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) [Grant, et al. 2008]. In order 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 \( p \) ports, the Data Center Network (DCN)
consists of $\rho$ 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 [POX. 2015]. 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:

1) 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.

2) 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 $\lambda_0$, $t$, $\gamma_0$ and $t$ are mentioned in Algorithm 1.

<table>
<thead>
<tr>
<th>Class I Problem</th>
<th>Scenario</th>
<th>Number of Gaming Sessions</th>
<th>Gaming Session Ratio to Gaming Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Small Size</td>
<td>40</td>
<td>2.5</td>
</tr>
<tr>
<td>2</td>
<td>Medium Size</td>
<td>80</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Large Size</td>
<td>160</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class II Problem</th>
<th>Scenario</th>
<th>Number of Gaming Sessions</th>
<th>Gaming Session Ratio to Gaming Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Small Size</td>
<td>625</td>
<td>2.5</td>
</tr>
<tr>
<td>2</td>
<td>Medium Size</td>
<td>1250</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Large Size</td>
<td>2500</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Games</th>
<th>Genre</th>
<th>Upstream traffic</th>
<th>Downstream traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean Packet Size (bytes)</td>
<td>Mean Inter-Departure time (ms)</td>
<td>Mean Packet Size (bytes)</td>
</tr>
<tr>
<td>assault cube</td>
<td>FPS</td>
<td>35.03</td>
<td>2.4</td>
</tr>
<tr>
<td>Warcraft</td>
<td>RTS</td>
<td>59.83</td>
<td>1.5</td>
</tr>
</tbody>
</table>

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 (4) (i.e. GRO) [Amiri, et al. 2015a] with the one proposed in section (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 iterations number that yielded the best solution, the average computational time per iteration, and the time required to reach the best solution. These results showcase the advantage of relieving some of the constraints of $O_{\text{total}}$ using the proposed
heuristic method. The best solution is reached using the GRH method far faster than the optimal one using the GRO approach.

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

Furthermore, for the GRH method, figures 5a and 5b show the semilogarithmic 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.

![Graph 5a](image1.png)

**Fig. 5.** 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)

![Graph 5b](image2.png)

![Graph 6a](image3.png)

**Fig. 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)

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

<table>
<thead>
<tr>
<th></th>
<th>GRH</th>
<th>GRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best solution</td>
<td>iteration</td>
<td>Total CPU</td>
</tr>
<tr>
<td>number</td>
<td></td>
<td>time</td>
</tr>
<tr>
<td>Class I</td>
<td>17</td>
<td>0.23</td>
</tr>
<tr>
<td>Problems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class II</td>
<td>31</td>
<td>3.19</td>
</tr>
<tr>
<td>Problems</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>GRH</th>
<th>GRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best solution</td>
<td>iteration</td>
<td>Total CPU</td>
</tr>
<tr>
<td>number</td>
<td></td>
<td>time</td>
</tr>
<tr>
<td>Class I</td>
<td>19</td>
<td>0.34</td>
</tr>
<tr>
<td>Problems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class II</td>
<td>38</td>
<td>7.25</td>
</tr>
</tbody>
</table>
Table 7. Computational results for the large size Class I and II problems

<table>
<thead>
<tr>
<th></th>
<th>GRH</th>
<th>GRO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best solution iteration number</td>
<td>Total CPU time</td>
</tr>
<tr>
<td>Class I Problems</td>
<td>24</td>
<td>1.50</td>
</tr>
<tr>
<td>Class II Problems</td>
<td>47</td>
<td>17.64</td>
</tr>
</tbody>
</table>

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 5a and 5b, we can conclude that $V_{UB}$ will converge faster to $z^*$ with adaptive step size. Moreover, the adaptive step size method outperforms the fixed step size method in terms of accuracy.

Figures 6a and 6b 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 problem instances. 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 $\alpha$ 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, $\alpha$ is set to 1.

![Fig. 7. Overall delay and Jitter for Class I problem instances a)Assault Cube b)Warcraft](image-url)
3) Proposed Method, where we calculate the weighted factor $\alpha$ for the two games in our experiment based on equation (2). The value of $\alpha$ was calculated as 0.67 and 0.253 for Assault Cube and Warcraft, respectively.

For the Class I problem, figure 7a 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 7b 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 7b 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 problem are shown in figures 8a and 8b. 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 players are playing Assault Cube are comparable with the proposed method, the average delay variation experienced by players using our 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 in order 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 [Zander and Armitage. 2004, Armitage. 2003]. We use the fairness index proposed in [Zander, et al. 2005] 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.

$$\mu(d_{\text{overall}}) = \frac{\sum_i^g p_i d_i}{N_g}$$

We measured the unfairness index $\mu(d_{\text{overall}})$ 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 9 show that gamers in different groups have a smaller difference in kill rate when our proposed method (GRH method) is used.

![Fig. 9. Unfairness for three different of gamers](image)

7. CONCLUSIONS

Since the major computational part of game processing is performed in data centers, a properly designed data center can provide high quality games to end-users and reduce costs. In this work, we focus on the data center's network resource management in a centralized fashion using Software Defined Networking (SDN). We present 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. Previously, we proposed an optimization model that takes into consideration the type of requested games, current server loads, and current path delays to make decision on which games server and communication path will minimize the delay with in data center. Due to the high complexity of the optimization method, in this paper, we designed a Lagrangian Relaxation (LR) heuristic method to drastically reduce the complexity of the problem such that it can be practically implemented in the real data center using the OpenFlow (OF) controller. The proposed method was evaluated with three different scenarios and the results indicate that it can provide close to optimal solution. Moreover, experimental results showed that the proposed method, on average, outperforms existing 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.
REFERENCES


Greenberg, A., Hamilton, J.R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D.A.,


POX, Available at: https://openflow.stanford.edu/display/ONI/POX+Wiki, 2015.


ALIZADEH M., GREENBERG A., MALTZ D.A., PADHYE J., PATEL P., PRABHAKAR B., SENGUPTA S.
AND SRIDHARAN M., "Data center tcp (dctcp)," ACM SIGCOMM computer communication review, vol. 41, no. 4, pp. 63-74.