Performance Evaluation in Cloud-Edge Hybrid Gaming Systems

Athanasios Tsipis, Konstantinos Oikonomou
Dept. of Informatics
Ionian University
Corfu, Greece
Email: {atsipis, okon}@ionio.gr

Vasileios Komianos
Dept. of Audio & Visual Arts
Ionian University
Corfu, Greece
Email: vkomianos@ionio.gr

Ioannis Stavrakakis
Dept. of Informatics & Telecom.
Nat. & Kap. University of Athens
Athens, Greece
Email: ioannis@di.uoa.gr

***Abstract*—Cloud gaming architectures have emerged in efforts to provide efficient execution of computer video games in computer machines and mobile devices with low processing/rendering capabilities. Such approaches are prone to network delays since the rendering process is offloaded to the cloud, hence, increasing cloud's computational needs and decreasing user coverage. To tackle this challenge cloud-edge hybrid gaming systems were proposed in order to reduce network latency and improve user experience. By deploying rendering devices at the edge of the cloud and in close proximity to the end users, cloud-edge hybrid systems aim at augmenting the cloud's computational capacity and reducing the game's response time. In current work a preliminary simulation model to evaluate the performance of such systems and test the above hypothesis is presented, showing that cloud-edge hybrid gaming systems are able to outperform conventional cloud gaming architectures in latency reduction.**

***Index Terms*—Cloud Gaming, Cloud Edge Systems, Latency.**

I. Introduction

In this paper, performance issues in cloud gaming systems are studied. Video games are processing demanding, memory consuming computer applications and cloud gaming systems are proposed in order to provide gamers the ability to play video games in unresourceful computer machines. In addition, cloud gaming solutions are able to improve the quality of gaming experience by increasing the provided rendered *frames per second* (fps) and reduce the time needed for the game to respond to their actions (response time).

Response time is an important aspect of performance affecting the quality of user experience in cloud gaming environments and the term *latency* [1] is used in order to measure it. Previous works have shown that latency depends on the processing power of both client and server computer machines while it is affected by the network connection used for client-server communication [2], [3]. Cloud gaming providers are able to build powerful cloud gaming systems but the network capacity and its existing conditions are often imponderable and they can be the source of 80% of latency [4] as the rendered game frames and audio segments, which are of considerable size, have to be streamed over the network. Given that in cloud gaming systems the servers are in most cases placed in relatively great distances [5], many works are addressing this issue by placing rendering servers with potentially lower capabilities and cost, as near as possible to the game client in order to reduce the network latency, thus calling this approach cloud-edge hybrid gaming system [4], [6]. This difference can be observed in Fig. 1 which depicts a conventional cloud gaming system versus a cloud-edge hybrid gaming system after deploying rendering devices to assist the server at the edges of the cloud and in close proximity to the user clients.

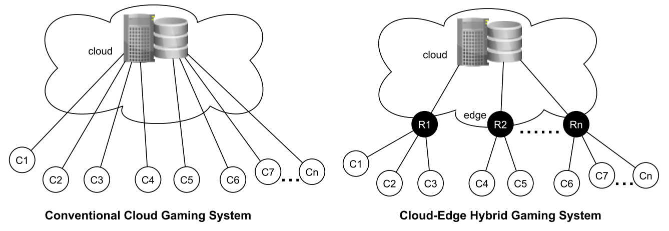
cloud
C1
C2
C3
C4
C5
C6
C7
Cn
Conventional Cloud Gaming System
cloud
edge
R1
R2
Rn
C1
C2
C3
C4
C5
C6
C7
Cn
Cloud-Edge Hybrid Gaming System

Figure 1. A conventional cloud gaming system versus an Edge device rendering-assisted cloud hybrid gaming system.

In this paper a novel evaluation model designed to facilitate study of cloud-edge performance issues is presented. The presented model's main purpose is to examine the resulted latency in different cloud-edge hybrid gaming setups with varying number of game renderers and in comparison with the solely cloud gaming approach. For this purpose a detailed end to end description of the data flow alongside with a latency analysis is conducted in order to decompose and identify critical points where bottlenecks and delays may occur due to the resource demanding nature of these systems. It is hypothesized that there exists a dependency between the number of deployed edge game renderers and latency reduction.

The evaluation model is then put to the test under different configurations and gaming conditions for various simulation scenarios. Preliminary simulation results show that cloud edge hybrid gaming systems are indeed able to outperform conventional cloud gaming architectures with considerable rendering power in latency reduction, depending on the number of deployed edge game renderers. However, after a certain point the reduction rate is significantly reduced.

In Section II, a review of relevant works is provided focusing in study of cloud gaming latency issues and cloud-edge architectures. In Section III, the flow of users input commands and the resulted streaming media between clients and servers in cloud-edge gaming architectures is presented. In Section IV, a preliminary analysis of the latency factor based on the considered data flow is presented. In Section V, the model setup that is used for the simulation is resented and its results are explained in Section VI. Finally, Section VII concludes the findings of this work and draws the guidelines for future work.

II. Related Work

Studies like [7], [8] indicate that low-latency is a force driver in order to achieve acceptable user-experience, in cloud gaming systems. This is also the case in [9], where the authors provide a suite of measurement techniques to decompose and analyze the Quality of Experience (QoE) of existing cloud gaming systems. In fact, in works such as [10]–[12] it is shown that there exists a threshold of maximum latency tolerance, above which players begin to notice significant delay in games. This delay refers to a range of 100 to 150 ms of response time for a seamless gameplay in fast-paced games (e.g., first-person shooter games) and to a range of 200 to 300 ms for slower-paced games (e.g., role playing games) [13], [14].

Moreover, previous research [1], [11], [12] points out that uploading generated command input data from the players' client to the cloud, i.e., the upstream latency, does not seriously affect the QoE since the data are small in size. On the other hand in the case of the downstream latency, the size of the data packets that encapsulate the newly rendered game frames have a significantly larger size. To address this problem Lin and Shen [15] in their model, named *CloudFog*, propose a solution to reduce the downstream latency by installing powerful devices, called supernodes, which are situated near the end-users and are responsible for rendering new game frames. In a similar approach Choy et al. [4], [6] present a hybrid cloud gaming system, called *EdgeCloud*, which incorporates resources from content delivery network (CDN) servers which are in close proximity to the end-users with the aim of supporting latency-sensitive applications. In their experiments they discovered that this approach can lead up to a 90% user coverage in contrast with conventional cloud gaming systems where only 70% of the users were able to meet the 80 ms latency target.

III. System Data Flow

Based on the studies mentioned in previous section, a cloud-edge hybrid gaming system data flow model, as illustrated in Fig. 2, is described here.

Upon data generation at the *game client* by the player's gaming command input, the player's actions are sent using a Command Data Packet (`CDP`). The `CDP` contains a few Bytes of information regarding the game actions issued by the player and is sent to a device (hereafter named as the *game renderer*) at the edges of the cloud, in close proximity to the player's game client, and then forwarded into the cloud towards the *game server* hosting the game world.

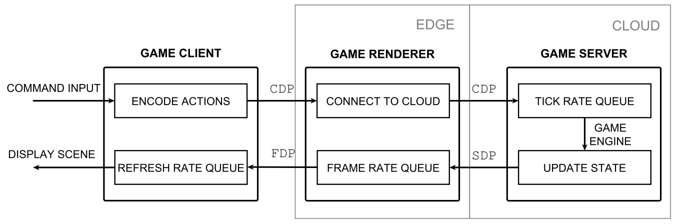

Figure 2. Game Data flow in cloud-edge hybrid gaming systems where the game client sends commands to the cloud game server and the server responds with update actions rendered by the edge game renderer and streamed back to the client for display.

When the `CDP` reaches the game server it is queued inside a buffer containing `CDPs` from other game clients along with actions generated by the game's AI engine. At regular intervals, which correspond to the game server's *tick rate*, the game server computes all of these actions and translates them into a new game world status update. The update is then encoded into a new State Data Packet (`SDP`), and sent back, as a response, to the game renderer. The size of the `SDP` varies depending on spikes in gameplay and fluctuations in the amount of updated elements inside the game world. However, its size is relatively small, usually some hundreds of Bytes. Upon reception at the game renderer, the `SDP` enters the render queue. At a fixed rate, i.e., the *frame rate*, the renderer uses the information included in the `SDP`, in order to create a new video image frame that captures the game status update. It then encapsulates the newly rendered game frame inside a Frame Data Packet (`FDP`), which is streamed back to the game client. In contrast with the previous packets (i.e. `CDP` and `SDP`) a `FDP` is large in size, in a factor of KBytes depending on video image quality and resolution, containing all information regarding the newly rendered game scene. When the `FDP` reaches the game client it enters its queue, which contains buffered frames arriving from the renderer. The display device of the game client displays these frames in a fixed *refresh rate* which depends on the hardware used.

The above outlined game data flow is important because it frees cloud game servers from having to compute and render complex graphics by their own and then having to stream huge generated frames to long distances over the Internet to the game clients. Instead, with the intervention of the edge game renderers, the rendering takes place near the game clients reducing game latency and increasing user coverage.

IV. Latency Analysis

Cloud gaming offers hardware independence, allowing end-users to play complex games by off-loading computational resource demanding processes to the cloud [4]. In this way cloud gaming offers the potential of playing games on the go without the need for specialized equipment or/and hardware (e.g., powerful GPUs). Therefore, players may engage with the gaming world through various display devices, even while moving in and out of network coverage. As already mentioned, low latency is essential for unimpaired game play and a fundamental design challenge for cloud gaming developers, and therefore it is considered as a crucial measure for high QoE [7].

For the data flow model described in previous section, latency L is considered to be the total response delay, i.e., the time period that elapses between the moment the player hits his/hers keyboard and issues a command action input to the game system and the moment he/she can perceive the result and notice an update in the status of the game world. According to [1] this latency comprises three basic components: Processing delay, playout delay and network delay. However, for the hybrid system in question another component is introduced, which is the rendering delay, and therefore, L is equal to the sum of:

- *Processing delay*: denoted here as d_{pr}, the time it takes for the cloud game server to decode a `CDP` (forwarded by an edge device, which holds information regarding the player's recent input actions), update the game state and respond with a `SDP`, which encapsulates the update status of the game world.
- *Rendering delay*: symbolized as d_r, the time it takes for an edge device to decode and translate a received `SDP` and render it into a new video image frame before streaming it to the player's game client.
- *Playout delay*: given by d_{pl}, the time it takes to issue a command and create the `CDP` plus the difference between time the player's game client receives a new `FDP`, corresponding to his/her initial command input, and the time it takes for his display to present the new game image, i.e., the display lag.
- *Network delay*: declared here as d_n, the time elapsed between transmission of a `CDP` and the reception of a `FDP`. It is usually referred to as the network round trip time (RTT). In current work it is associated with the sum of transmission and propagation delays between the player's game client and the edge game renderer and vice-versa and between the latter and the cloud game server and vice-versa.

Hence, L is described by:

$$L = d_{pr} + d_r + d_{pl} + d_n, \tag{1}$$

for a particular set of game client, renderer and server. However, d_n, as already mentioned is equal to the sum of total transmission delay d_t and total propagation delay d_p in the aforementioned set, and so Eq. (1) can be written as:

$$L = d_{pr} + d_r + d_{pl} + d_t + d_p, \tag{2}$$

where $d_t = \delta_t(\text{client} \rightarrow \text{renderer}) + \delta_t(\text{renderer} \rightarrow \text{server}) + \delta_t(\text{server} \rightarrow \text{renderer}) + \delta_t(\text{renderer} \rightarrow \text{client})$ and $d_p = \delta_p(\text{client} \rightarrow \text{renderer}) + \delta_p(\text{renderer} \rightarrow \text{server}) + \delta_p(\text{server} \rightarrow \text{renderer}) + \delta_p(\text{renderer} \rightarrow \text{client})$ respectively.

Taking it a step further, based on the works found in [16] and [17], it is assumed that the game server computes new game state updates, based on the combination of queued player command inputs and game AI actions, at a fixed tick rate t, such that the game server is able to send a new `SDP` every $T = 1/t$ seconds. Correspondingly, the game renderer, renders new `SDP`s into `FDP`s with a fixed frame rate f, hence, every $F = 1/f$ seconds a new game scene is generated. Moreover, the game client displays new buffered game frames at a steady refresh rate r which depends on the display capabilities of the client's hardware, such that the game client can display a new `FDP` every $R = 1/r$ seconds. Taking into account that a new `CDP` can arrive at the game server in the interval $[nT, (n+1)T]$, after being processed it will have to wait for a time lag period $\tau_s \leq T$ before being encapsulated and sent in a new `SDP` at time $(n+1)T$. Similarly, a new `SDP` arriving at the game renderer in the interval $[nF, (n+1)F]$, it will have to wait in queue for a time lag period $\tau_r \leq F$ before being rendered at the $(n+1)F$, and in sequence a `FDP` arriving at the client in the interval $[nR, (n+1)R]$, after being decoded, it will have to wait in queue for a time lag period $\tau_c \leq T$ before being displayed at the $(n+1)T$.

Consequently, after adding the three lag periods, latency L perceived by a player in his game client is given by:

$$L = d_{pr} + d_r + d_{pl} + d_t + d_p + \tau_s + \tau_r + \tau_c, \tag{3}$$

Assuming no further delays induced by network protocols and network traffic are present, and the fact that d_{pr}, d_r, d_{pl}, τ_s, and τ_c are unknown, but have an upper boundary, since they are subject to hardware constraints, it follows that in order to reduce latency in the hybrid system we must decrease the values of d_t and d_p, i.e. the d_n, and the value of the rendering lag τ_r. It is hypothesized that a way to accomplish this is by deploying more edge devices near the end-users that will support the rendering procedure and reduce the distance the `FDP`s must travel to reach the game clients.

V. Evaluation Model Setup

To evaluate the reduction of game clients' latency triggered by the deployment of more game renderers in the hybrid cloud-edge gaming system, a simulation program in C++ has been developed using OMNeT++. During network initialization a random topology with 1000 uniformly and independently distributed end-user nodes, i.e., the game clients, are placed in a plane $[0, 1] \times [0, 1]$. The cloud game server and the edge game renderers are also placed in random positions inside the plane and a connection link exists between every game renderer and the cloud game server. However, a connection between any game client and an edge game renderer exists only if their euclidean distance is equal to or smaller than a given connectivity radius r_c. The criteria for selecting the appropriate renderer in the given range is based on a distance comparison of all possible connections, where the link with the minimum distance weight is selected.

For simplicity reasons it is assumed that all transmissions take place over an error-free channel with Internet speeds capable of supporting the continuous stream of packets transmitted and received by the network nodes. Each simulation scenario lasts for 1000 sec and was conducted under three

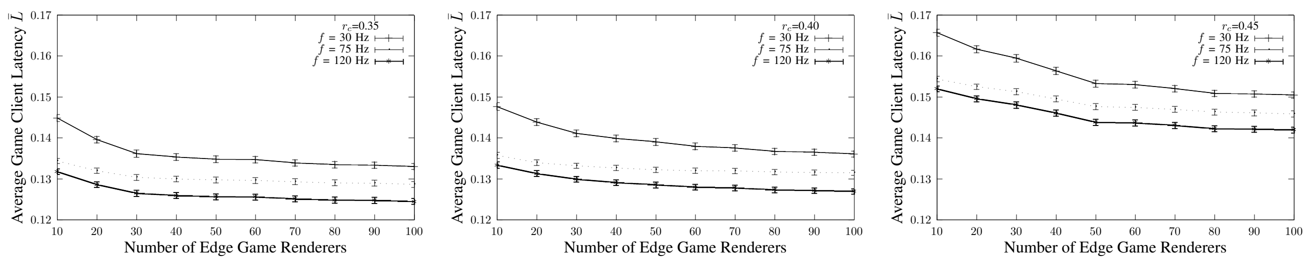

Figure 3. Simulation results of average game client latency $\bar{L}$ as a function of the number of edge game renderers for different network topologies based on the value of r_c and under different frame rates based on the value of f.

Table I
CLOUD-EDGE NETWORK SIMULATION PARAMETERS

Parameter	*Value*
Simulation Time	1000 sec
Game Clients	1000
`CDP` Length	60 Bytes
`SDP` Length	1000 Bytes
`FDP` Length	4000 Bytes
Data Rate Transmission	5 Mbps
d_{pl}	Stochastic Distribution
d_r	Normal Distribution
d_{pr}	Normal Distribution
t	128 Hz
f	30, 75, and 120 Hz
r	60 Hz
r_c	0.35, 0.40, and 0.45

different topologies based on different seed sets and different values for the r_c. The parameters of the simulation model are summarized in Table I.

Each game client follows a stochastic process for generating new `CDPs` which are then sent to its associated game renderer and in sequence to the game server with a steady data rate speed of 5 Mbps. This value was chosen based on the fact that OnLive [18] recommends at least this speed for a seamless gameplay experience [9]. In addition, the command generation at the game client has an upper limit which matches the tick rate of the game server, since the server would not be able to process more packets.

Considering d_{pr} at the server, based on [16], it is assumed that it follows a normal distribution with mean value ($\mu_{d_{pr}}$) of 3 ms and standard deviation value ($\sigma_{d_{pr}}$) of 0.1 ms. Similarly, based on [9], d_r also follows a normal distribution with mean value (μ_{d_r}) of 40 ms and standard deviation value (σ_{d_r}) of 1 ms. The great advantage of cloud gaming still remains the fact that it frees gameplay from being dependent on specific hardware, but instead enables players to access games through all kinds of devices and platforms. Taking into account this diversity in the game clients' screens, a random display delay value was generated based on a stochastic distribution, in addition to the delay caused by command input which is around 10 ms for standard input devices (e.g. keyboard). Therefore, the d_{pl} is distributed within the range of $[20, \ldots, 40]$ ms [16], including modern AR/VR headsets [19].

Regarding the tick rate t of the produced `SDPs` of the game server, a popular rate for moderate-demanding games was selected, i.e., 128 Hz, whereas in the case of the renderers, for the render rate f, three values were taken into consideration, i.e., 30 Hz, 75 Hz and 120 Hz, in order to show latency for various popular frame rates and based on the fact that higher frame rates that exceed the tick rate of the server won't be perceivable by the player, since no updates will be available for rendering in time. As for the refresh rate r for displaying `FDPs` in the game clients, the fixed value of 60 Hz was chosen corresponding to moderate screens.

Finally, the various sizes of the different data packets were chosen by monitoring the data exchanged between client and server on the Uniquitous open-source gaming system [20]. For this purpose the platform was installed alongside Unity's game engine in a closed network control environment, where data traffic was recorded and analyzed in order to estimate the size of the packets.

For simplicity reasons it is assumed that all players are engaged in the same game world and they all witness the same game scene changes. This simplification was adopted in order to avoid the necessity and complexity of producing frames with different perspectives for each game client.

VI. SIMULATION RESULTS

A set of experiments based on the aforementioned simulation model is conducted in order to examine the latency issues. Fig. 3 validates the behavior for various cases by plotting the average game clients' latency $\bar{L}$ in the network with 95% confidence interval as a function of the number of deployed edge game renderers for three different topologies (based on the value of connectivity radius r_c) and for the three different frame rates f. Both r_c and f play an important role in the initial value of the $\bar{L}$ since for lower r_c the clients can detect renderers at closer proximity and for higher f the rate of rendered `SDPs` is increased and so τ_r is reduced. Additionally, as expected the addition of gradually more edge game renderers, between the game server and the game clients, boosts the response time of the gaming system and decreases $\bar{L}$, but it is important to notice that from a certain point forward the reduction rate is significantly reduced.

Given that cloud gaming services and edge game renderers can be of variable processing power it is questionable whether

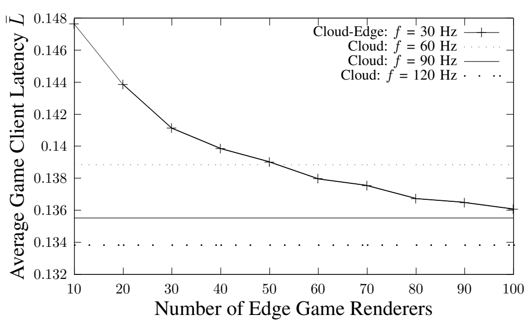

Figure 4. Performance comparison between hybrid cloud-edge gaming system for increasing number of renderers and conventional cloud gaming system for different rendering power. The straight lines refer to the conventional model.

cloud-edge architecture provides better performance than conventional cloud gaming architecture. In order to examine the previous research question a simulation scenario is tested in order to compare the resulted latency for conventional cloud gaming systems of different rendering capabilities versus a cloud-edge hybrid gaming system with varying number of game renderers, but with less rendering power, i.e., Fig. 4. Notice that for the comparison to be valid the coordinates of the clients and the cloud server remained the same for the two systems. The only difference was due to the addition of the intermediate renderers for connectivity radius r_c equal to 0.4.

Simulation results indicate that as the number of renderers is increased the latency is reduced. Compared to the resulted latency of conventional architecture with higher frame rates, depicted by horizontal parallel lines, it is observed that conventional systems of sufficiently more rendering power outperform the hybrid system but as the number of renderers is increased the latter is able to perform better.

VII. Conclusion

In conclusion, current work showed that by deploying hybrid cloud-edge gaming systems it is possible to reduce the game clients' latency and to outperform the conventional cloud gaming architectures. According to preliminary simulation results it was observed that when increasing renderers the latency is reduced, however from a certain point forward the reduction rate is significantly reduced. This finding has to be further explored in future work where it will be studied as a facility location problem in order to research if further latency reduction can be achieved. An additional approach towards this direction will be the adaptation of the simulation model to more realistic rendering load parameters. Moreover, a mathematical formulation of its behavior will be explored and a demo testbed will be created to validate the findings.

Acknowledgment

This work was supported in part by project "A Pilot Wireless Sensor Networks System for Synchronized Monitoring of Climate and Soil Parameters in Olive Groves," (MIS 5007309) which is partially funded by European and National Greek Funds (ESPA) under the Regional Operational Programme "Ionian Islands 2014-2020." In addition, this work was supported in part by the European Commission as part of the ReCRED project (Horizon H2020 Framework Programme of the European Union under GA number 653417), by the Chair of Excellence UC3M - Santander Program and by the National and Kapodistrian University of Athens (S.A.R.G.).

References

[1] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 1269–1272.

[2] M. Claypool and K. Claypool, "On latency and player actions in online games," 2006.

[3] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. C. Leung, and C.-H. Hsu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.

[4] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "A hybrid edge-cloud architecture for reducing on-demand gaming latency," *Multimedia systems*, vol. 20, no. 5, pp. 503–519, 2014.

[5] P. J. Braun, S. Pandi, R.-S. Schmoll, and F. H. Fitzek, "On the study and deployment of mobile edge cloud for tactile internet using a 5g gaming application," in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2017, pp. 154–159.

[6] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "Edgecloud: A new hybrid platform for on-demand gaming," *Technical Report CS-2012–19, University of Waterloo*, 2012.

[7] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui, "Cloud gaming: architecture and performance," *IEEE network*, vol. 27, no. 4, pp. 16–21, 2013.

[8] M. Amiri, H. A. Osman, S. Shirmohammadi, and M. Abdallah, "Toward delay-efficient game-aware data centers for cloud gaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 5s, p. 71, 2016.

[9] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 480–495, 2014.

[10] 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*. ACM, 2010, pp. 215–222.

[11] 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*. IEEE, 2011, pp. 330–335.

[12] 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 *Proceedings of the 11th annual workshop on network and systems support for games*. IEEE Press, 2012, p. 2.

[13] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "Gaming in the clouds: Qoe and the users' perspective," *Mathematical and Computer Modelling*, vol. 57, no. 11-12, pp. 2883–2894, 2013.

[14] Y. Deng, Y. Li, R. Seet, X. Tang, and W. Cai, "The server allocation problem for session-based multiplayer cloud gaming," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1233–1245, 2018.

[15] Y. Lin and H. Shen, "Cloudfog: leveraging fog to extend cloud gaming for thin-client mmog with high quality of service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 431–445, 2017.

[16] F. Metzger, A. Rafetseder, and C. Schwartz, "A comprehensive end-to-end lag model for online and cloud video gaming," *5th ISCA/DEGA Work. Percept. Qual. Syst.(PQS 2016)*, pp. 15–19, 2016.

[17] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[18] 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)*. IEEE, 2012, pp. 1–6.

[19] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 2018.

[20] M. Luo and M. Claypool, "Uniquitous: Implementation and evaluation of a cloud-based game system in unity," in *2015 IEEE Games Entertainment Media Conference (GEM)*. IEEE, 2015, pp. 1–6.