JOURNAL OF Emerging Paradigms in Computing Systems JEPCS

Journal of Emerging Paradigms in Computing Systems (JEPCS) | Volume 03, Issue No. 02, 2026

ARTICLE TYPE: Research Article

Article ID: JEPCS-2026-0302-0021

Latency Optimization in Cloud Gaming Using Edge-Assisted Resource Allocation Techniques

Muhammad Rafay Khalil Chaudhary[1,*], Haider Ejaz Bhatti[1], Muhammad Noman[1]

[1] Faculty of Information Technology & Computer Science, University of Central Punjab, Pakistan

*Corresponding author: L1F23BSCS0306@ucp.edu.pk

Contributing authors: L1F23BSCS0306@ucp.edu.pk, L1F23BSCS0206@ucp.edu.pk, mohammad.noman@ucp.edu.pk

Received: August 20, 2026 **Accepted:** September 25, 2026

Abstract- Cloud gaming moves game execution to remote servers so that players can run high-performance games on low-end devices. However, server overload and network delay keep latency high and degrade the quality of experience (QoE). This paper proposes an edge-assisted resource allocation strategy that places latency-critical processing on edge nodes close to users, while the central cloud handles computation-heavy and globally coordinated tasks. A multi-factor scoring rule based on proximity, current load, and predicted demand selects the serving edge node, and a feedback-driven offloading policy decides where each task runs. An illustrative evaluation, based on explicitly stated assumed parameters rather than measurements, indicates that the edge-assisted design yields lower latency and higher QoE than a cloud-only baseline across different user counts and bandwidth levels.

Keywords- Cloud gaming; edge computing; resource allocation; latency; quality of experience.

1. Introduction

Cloud computing provides on-demand access to scalable infrastructure over the Internet [14]. One of its most demanding applications is cloud gaming, in which game logic and rendering run on remote servers and the resulting video is streamed to the player in real time [4, 2]. This removes the need for expensive client hardware, but it makes the user experience depend on the network and the server. The main obstacle is latency, i.e., the delay between a player's input and the corresponding response on screen as shown in Figure 1. Even small increases in delay measurably reduce player performance, especially in fast-paced games [1], and measurement studies show that cloud-to-end-user latency is a limiting factor for deployment [5]. Server workload, network congestion, and limited bandwidth further degrade perceived quality [3, 6]. Edge computing places computational resources closer to end users and reduces transmission delay [7, 8]. Several works apply it to cloud gaming [9, 10, 11, 12], and offloading has been studied as a way to balance local and remote computation [13]. Most existing designs, however, do not jointly consider proximity, instantaneous load, and predicted demand in a simple, transparent decision rule.

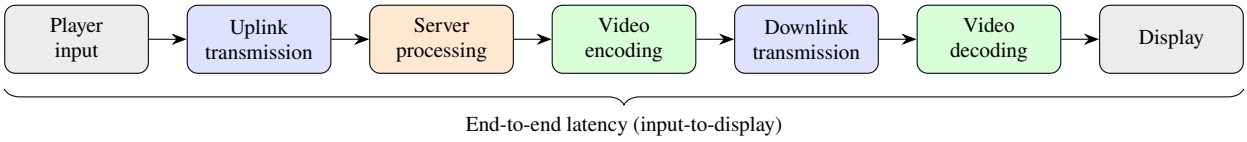


Figure 1: *Components of end-to-end latency in cloud gaming: network transmission (blue), server processing (orange), and codec delay (green).*

The main contributions of this paper are as follows:

- We partition the gaming process so that latency-critical functions (input handling, frame rendering, encoding, input prediction) run at the edge, while compute-intensive and globally coordinated functions run in the cloud.
- We formulate a scoring rule that combines user-edge proximity, current node load, and predicted demand, and give the decision procedure as an algorithm (Algorithm 1).
- We define a task-level offloading rule that uses live measurements of latency, jitter, packet loss, and throughput to decide between edge and cloud execution.
- Under an explicitly stated set of assumed parameters, we compare the proposed scheme with a cloud-only baseline in terms of latency and QoE under varying user counts and bandwidth, and discuss the limitations of this evaluation.

The rest of the paper is organized as follows. Section 2 reviews related work, Section 3 states the problem and research gap, Section 4 presents the architecture, Section 5 the allocation strategy, Section 6 the workflow, Section 7 the evaluation setup, Section 8 the results, and Sections 9 and 10 the limitations and conclusion.

2. Background and Related Work

2.1 Latency in Cloud Gaming

In cloud gaming, latency is the time between a user's input and the matching frame appearing on screen. It accumulates from server processing, video encoding and decoding, and network transmission [2, 5]. High latency causes input lag and reduced responsiveness, which is most harmful in racing and first-person-shooter games [1]. Subjective studies link these delays directly to lower QoE [6, 3]. Table 1 shows the existing latency-reduction techinques and their limitations.

2.2 Existing Optimization Techniques

Prior work reduces latency through adaptive bitrate streaming [17, 18], predictive input modeling [19], content delivery networks (CDNs), network prioritization, and stream compression [4]. Resource allocation and load balancing are also used to avoid server overload. Table 1 summarizes the main limitation of each technique.

Table 1: *Existing latency-reduction techniques and their limitations.*

Technique	Limitation	Ref.
Adaptive bitrate streaming	May reduce visual quality under low bandwidth	[17, 18]
Predictive input modeling	Needs accurate prediction models	[19]
CDN-style placement	High deployment cost for global coverage	[5]
Network prioritization (QoS)	Depends on operator support	[8]
Stream compression	Compression artifacts may affect quality	[4]

2.3 Edge Computing for Latency Reduction

Edge servers handle time-sensitive tasks near the user and forward heavier computation to the central cloud [7, 8]; the same idea is used for other latency-sensitive applications [16], and edge intelligence is an active direction [15]. For cloud gaming, EdgeGame moves rendering to the network edge and adapts video bitrate with deep reinforcement learning [9]. E-Render separates the graphics engine and deploys it at the edge to support UHD-quality streaming [11]. An edge-cloud selection scheme has been proposed for virtual-reality multiplayer games [12] and a reinforcement-learning approach balances latency, fairness, and load [10]. Computation offloading itself was first studied for mobile devices [13]. In contrast to learning-based schemes, our approach uses an explicit, transparent scoring rule that needs no training phase.

2.4 Limitations of Existing Methods

Adaptive streaming lowers latency by sacrificing quality. Static allocation and load-balancing schemes adapt poorly to changing demand and network conditions. Edge deployments add infrastructure cost and coordination complexity [7, 8]. These limitations motivate a more flexible allocation method.

3. Problem Statement and Research Gap

Problem. Latency in cloud gaming comes from encoding/decoding, server processing, and network transmission. High latency causes input lag, inconsistent responsiveness, and unfair competition, so it must be minimized to guarantee real-time interaction [1, 2].

Research gap. (i) Methods such as predictive input modeling and adaptive bitrate streaming work well under typical conditions, but degrade for users far from data centers or with varying bandwidth [5]. (ii) Existing edge and CDN deployments are often provisioned statically and rarely react to instantaneous network conditions, server load, or user mobility [5, 8]. (iii) Few studies combine several latency-reduction strategies in one framework; most edge-gaming

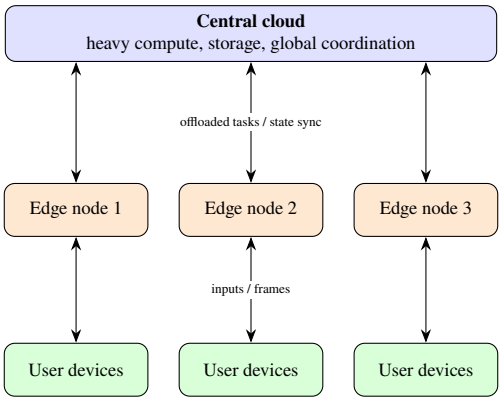


Figure 2: *Proposed edge-assisted cloud gaming architecture.*

works target a single aspect, such as rendering placement [11], bitrate adaptation [9], or edge selection [12]. An adaptive, edge-assisted allocation scheme that balances latency, computational load, bandwidth efficiency, and QoE is therefore still needed.

4. System Architecture

The system has three layers as shown in Figure 2. The first layer is user devices, second is edge nodes, and last is a central cloud. This layered architecture follows the edge-cloud hierarchy commonly used in edge computing [7, 8].

- Cloud server: The cloud provides high-performance computation (large-scale rendering, AI, complex physics), centralized storage (user profiles, game progress, textures, models, maps), and global coordination (matchmaking, multiplayer synchronization, leaderboards) [4, 2]. It also acts as a fallback when edge nodes are overloaded or unavailable. Because it is usually far from users, relying on it alone causes high network latency [5], which is why latency-sensitive work is moved to the edge.
- Edge nodes: Edge nodes are located in local data centers, ISP facilities, or telecom infrastructure close to users [7, 8]. These nodes execute latency-critical tasks (input handling, frame rendering, video encoding, predictive input modeling). As well as monitor CPU, GPU, memory, and bandwidth usage to support dynamic session assignment. Also they supervise cache frequently used assets such as textures, models, and recent game state and reroute requests to neighboring nodes or the cloud if they become overloaded.
- User device: The device captures inputs (touch, controller, keyboard, mouse), sends them to the nearest edge node, decodes and displays the received video, and reports latency, jitter, packet loss, and available bandwidth to the allocation module. Since it only performs lightweight work, low-end devices can be used [4], but the experience becomes sensitive to network conditions.

5. Dynamic Resource Allocation Strategy

5.1 Edge Server Selection

Let $\mathcal{E}$ be the set of edge nodes. When user u starts a session at time t, the allocation module scores every node $e \in \mathcal{E}$ using three factors.

Proximity. With $\mathrm{RTT}_{u,e}$ the measured round-trip time between u and e and $R_{\max}$ the maximum tolerated value,

$$\tilde{d}_{u,e} = \min\left(1,\ \frac{\mathrm{RTT}_{u,e}}{R_{\max}}\right). \tag{1}$$

Current load. With $\upsilon_{e,r}(t) \in [0,1]$ the utilization of resource $r \in \{\mathrm{CPU}, \mathrm{GPU}, \mathrm{memory}, \mathrm{network}\}$ and weights $\omega_r \geq 0$, $\sum_r \omega_r = 1$,

$$\ell_e(t) = \sum_r \omega_r\, \upsilon_{e,r}(t). \tag{2}$$

Predicted demand. Let $\lambda_e(t)$ be the session arrival rate observed at e. Any short-term predictor can be used; a simple choice is exponential smoothing,

$$\hat{\lambda}_e(t{+}1) = \alpha\, \lambda_e(t) + (1-\alpha)\, \hat{\lambda}_e(t), \quad \alpha \in (0,1], \tag{3}$$

with normalized predicted load $\hat{\rho}_e(t) = \min\big(1, \hat{\lambda}_e(t{+}1)/\Lambda_e\big)$, where Λ_e is the session capacity of e per interval.

Algorithm 1 Edge server selection for a new session

Require: user u, edge set $\mathcal{E}$, weights w_1, w_2, w_3, thresholds $R_{\max}, \tau$
Ensure: serving node for u
$\mathcal{E}_u \leftarrow \{e \in \mathcal{E} : \mathrm{RTT}_{u,e} \le R_{\max} \ \wedge \ \ell_e(t) \le \tau\}$
if $\mathcal{E}_u = \emptyset$ **then**
    return cloud
end if
for all $e \in \mathcal{E}_u$ **do**
    compute $S_{u,e}$ using Eq. (5)
end for
$e^\star \leftarrow \arg\min_{e \in \mathcal{E}_u} S_{u,e}$
assign session of u to $e^\star$
return $e^\star$

Decision rule. Only nodes that satisfy the proximity and load constraints are candidates:

$$\mathcal{E}_u = \{e \in \mathcal{E} : \ \mathrm{RTT}_{u,e} \le R_{\max}, \ \ell_e(t) \le \tau\}, \tag{4}$$

where τ is a load threshold. Each candidate receives the score

$$S_{u,e} = w_1 \, \tilde{d}_{u,e} + w_2 \, \ell_e(t) + w_3 \, \hat{\rho}_e(t), \quad \sum_i w_i = 1, \tag{5}$$

and the session is assigned to

$$e^\star = \arg\min_{e \in \mathcal{E}_u} S_{u,e}. \tag{6}$$

If $\mathcal{E}_u = \emptyset$, the session is served by the cloud. The procedure is summarized in Algorithm 1.

5.2 Task Offloading Rule

Tasks are split into latency-critical tasks $\mathcal{K}_{\mathrm{lc}}$ (input handling, frame rendering, encoding, input prediction) and the rest (AI, physics, leaderboards, synchronization). Tasks in $\mathcal{K}_{\mathrm{lc}}$ always run at the selected edge node. Any other task k with workload p_k is offloaded to the cloud if offloading is enabled, edge resources are insufficient, and the estimated cloud completion time is lower:

$$T_k^{\mathrm{cloud}} = \mathrm{RTT}_{e,c} + \frac{p_k}{f_c} \ < \ T_k^{\mathrm{edge}} = q_e + \frac{p_k}{f_e}, \tag{7}$$

where f_e, f_c are the processing rates of the edge and cloud, q_e is the queueing delay at e, and $\mathrm{RTT}_{e,c}$ is the edge–cloud round-trip time. Figure 3 shows the decision flow.

5.3 Feedback Loop

Each user device continuously reports latency, jitter, packet loss, and throughput. If the measured latency of a session exceeds its budget, the allocation module re-runs Algorithm 1 and migrates the session to a less loaded or closer node, and the encoder adapts the compression level to the available bandwidth. This keeps the selection adaptive instead of static.

6. Request Handling Workflow

A request follows six steps:

1. **Input capture.** The device sends the user input to the nearest suitable edge node as shown in Algorithm 1.
2. **Edge processing.** The edge performs input interpretation, predictive input modeling, frame generation, and video compression.
3. **Cloud offloading.** Computation-heavy or globally coordinated tasks are sent to the cloud according to Eq. (7).
4. **Response aggregation.** The edge merges locally produced frames with cloud results (e.g., updated game state).
5. **User display.** The device decodes and displays the compressed frames.
6. **Feedback.** Measured metrics return to the allocation module, which adjusts allocation, routing, and compression.

Together, these steps keep time-sensitive work close to the user, use the cloud only for non-critical work, scale with the number of users, and adapt to network fluctuations.

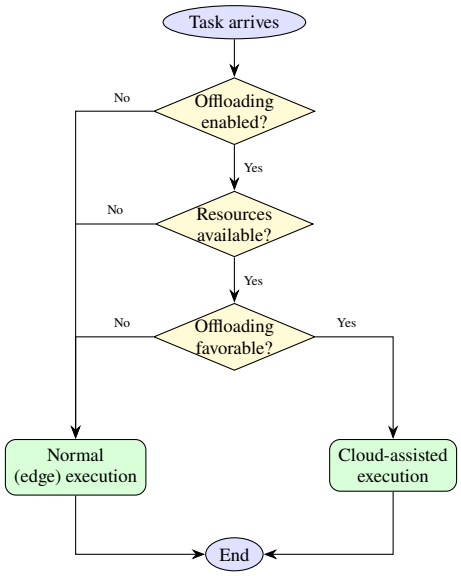


Figure 3: *Offloading decision flow at an edge node.*

7. Evaluation Setup

The results show the expected qualitative behavior of the proposed scheme under the assumed parameters in Table 2, and they are not measurements from a deployed system or a validated simulator. We compare the proposed edge-assisted system against a cloud-only baseline, in which all processing, rendering, and streaming is done by central cloud servers. The number of concurrent users is varied from 10 to 500 and the available bandwidth from 5 to 100 Mbps. Interactive cloud gaming commonly targets response delays below about 100 ms, which measurements show to be achievable when the backend is placed near the user [20]. Table 2 lists the setup.

Table 2: *Evaluation setup. Values marked "assumed" are representative assumptions, not measurements.*

Parameter	Value
Platform	None: illustrative curves under assumed parameters (no simulator or testbed)
Compared systems	Cloud-only (baseline); edge-assisted (proposed)
Concurrent users	10-500
Bandwidth	5-100 Mbps per user
Edge nodes (assumed)	5 nodes, each with 16 CPU cores, 1 GPU, 64 GB RAM
Cloud (assumed)	1 central data center with 128 CPU cores, 8 GPUs, 512 GB RAM
RTT (assumed)	User-edge 10-20 ms; edge-cloud 30 ms; user-cloud 40-80 ms
Workload (assumed)	One fast-paced game, 1080p at 60 fps, Poisson session arrivals
Model parameters (assumed)	$(w_1, w_2, w_3) = (0.4, 0.35, 0.25)$, $\omega_r = 0.25$, $\alpha = 0.5$, $\tau = 0.8$, $R_{\max} = 50$ ms
Metrics	Latency (ms), QoE (1-10), performance score (%)

We evaluate latency, QoE (a user-centric score (1-10) covering responsiveness, fluidity, video quality, and absence of lag or buffering) and an overall performance score.

8. Results and Analysis

8.1 Latency versus Number of Users

Figure 4 compares both systems as the number of concurrent users grows. Curve of cloud-only latency rises steeply, reaching 300 ms at 500 users, while the edge-assisted latency curve rises more slowly, reaching 160 ms, as expected when sessions are served by nearby nodes. In these curves, the edge-assisted latency stays within the 100 ms target discussed in Section 7 up to about 200 users; beyond that, additional edge capacity would be needed.

8.2 Bandwidth versus Performance

Figure 5 shows that the overall performance score improves quickly up to roughly 50 Mbps and then saturates, indicating diminishing returns from additional bandwidth.

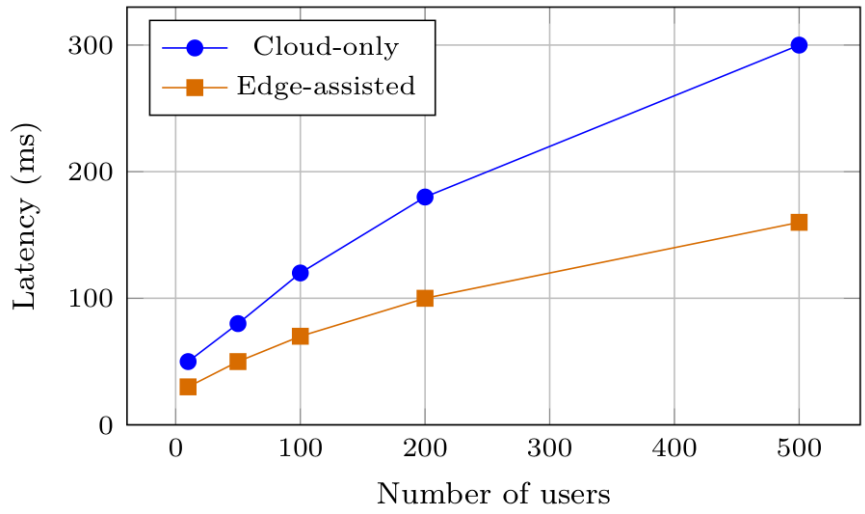


Figure 4: *Latency versus number of users for cloud-only and edge-assisted systems*

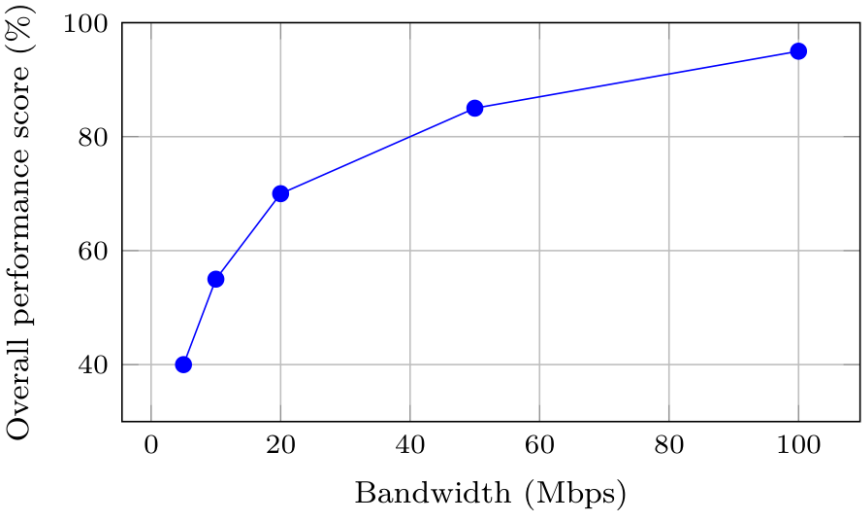


Figure 5: *Overall system performance score versus available bandwidth (illustrative).*

8.3 Average Latency

Averaged over the evaluated user counts in Figure 4, latency drops from 146 ms (cloud-only) to 82 ms (edge-assisted), a reduction of about 44% as shown in Figure 6.

8.4 Bandwidth versus Quality of Experience

Figure 7 shows that the edge-assisted system gives a higher QoE score at every bandwidth level, with the largest gap at low bandwidth, where cloud-only delivery suffers most from delay and congestion.

9. Discussion and Limitations

Placing edge nodes near users reduces communication delay, and load-aware allocation avoids bottlenecks during high demand, which is expected to lower latency and improve responsiveness compared with cloud-only processing. Edge deployment nevertheless requires coordination between cloud and edge nodes, higher infrastructure cost, and management of distributed resources.

The current evaluation has the following limitations.

- First, the results are considered under assumed parameters, not measurements from a simulator or a deployed system, so validation by simulation or a real testbed is still required.
- Second, it covers up to 500 concurrent users, and large multi-region networks were not evaluated.

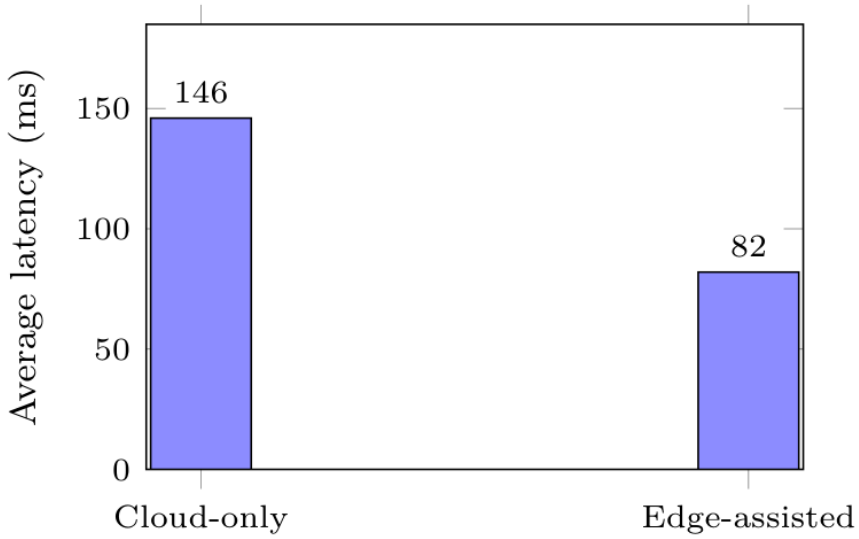


Figure 6: *Average latency of the cloud-only and edge-assisted systems*

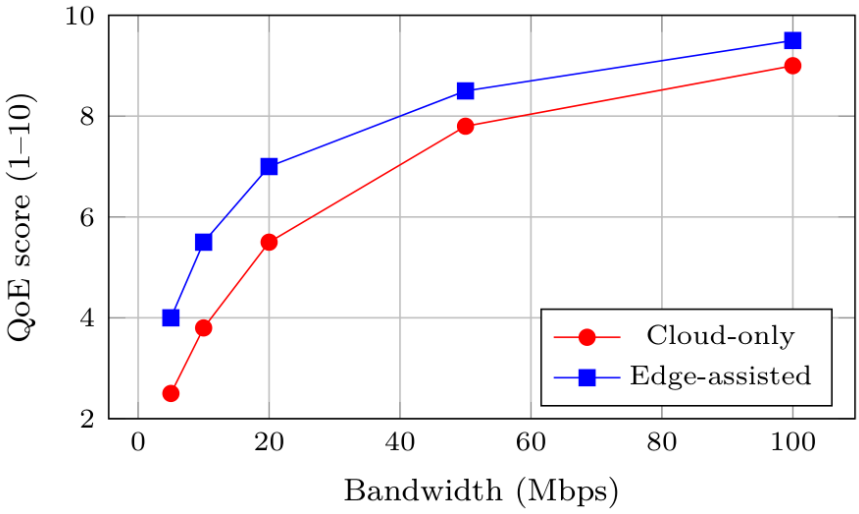


Figure 7: *QoE score versus bandwidth (illustrative).*

- Third, edge nodes are assumed to have sufficient compute capacity, which may not hold in practice.
- Fourth, the deployment and operating cost of edge nodes is not modeled.
- Fifth, the results depend on the workload and may differ for other games and player behavior.
- Sixth, only a cloud-only baseline was compared, the accuracy of demand prediction was not evaluated separately, and no real-world deployment was used.

10. Conclusion

This paper proposed an edge-assisted resource allocation strategy for cloud gaming. Latency-critical tasks run at nearby edge nodes, a proximity/load/prediction score selects the serving node, and a feedback-driven rule offloads heavy tasks to the cloud. The illustrative evaluation, based on assumed parameters, indicates lower latency and higher QoE than a cloud-only system. Future work includes validation through simulation and real testbeds, comparison with additional baselines, cost-aware allocation, and learning-based prediction and scheduling.

Author Declarations

Author Contributions (CRediT): Muhammad Rafay Khalil Chaudhary: Conceptualization; Methodology; Investigation; Literature Review; Writing - Original Draft. Haider Ejaz Bhatti: Methodology; Supervision; Validation; Writing - Review

& Editing. Muhammad Noman: Investigation; Literature Review; Validation; Visualization; Writing - Review & Editing. All authors have read and approved the submitted version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Ethics Statement: Not applicable.

Declaration of Generative AI: During the preparation of this manuscript, the authors used LLM tools for language refinement, grammatical correction, and improvement of readability. The authors subsequently reviewed and verified the resulting content and take full responsibility for the accuracy, integrity, and content of the manuscript.

References

[1] M. Claypool and D. Finkel, "The effects of latency on player performance in cloud-based games," in *Proc. 13th Annu. Workshop Network and Systems Support for Games (NetGames)*, 2014, pp. 1-6, doi: 10.1109/NetGames.2014.7008964.

[2] 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, Feb. 2014, doi: 10.1109/TMM.2013.2291532.

[3] S. Wang and S. Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, 2009, pp. 1-7, doi: 10.1109/GLOCOM.2009.5425784.

[4] 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, Jul. 2013, doi: 10.1109/MNET.2013.6574660.

[5] 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 *Proc. 11th Annu. Workshop Network and Systems Support for Games (NetGames)*, 2012, pp. 1-6, doi: 10.1109/NetGames.2012.6404024.

[6] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "An evaluation of QoE in cloud gaming based on subjective tests," in *Proc. 5th Int. Conf. Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2011, pp. 330-335, doi: 10.1109/IMIS.2011.92.

[7] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30-39, Jan. 2017, doi: 10.1109/MC.2017.9.

[8] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657-1681, 2017, doi: 10.1109/COMST.2017.2705720.

[9] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 178-183, Aug. 2019, doi: 10.1109/MWC.2019.1800440.

[10] X. Deng, J. Zhang, H. Zhang, and P. Jiang, "Deep-reinforcement-learning-based resource allocation for cloud gaming via edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 5364-5377, Mar. 2023, doi: 10.1109/JIOT.2022.3222210.

[11] Y. Kim, Y. Choi, Y. C. Lee, H. Han, and S. Kang, "E-Render: Enabling UHD-quality cloud gaming through edge rendering," *IEEE Access*, vol. 10, pp. 72107-72119, 2022, doi: 10.1109/ACCESS.2022.3187696.

[12] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, "Towards efficient edge cloud augmentation for virtual-reality MMOGs," in *Proc. 2nd ACM/IEEE Symposium on Edge Computing (SEC)*, 2017, Art. no. a8, doi: 10.1145/3132211.3134463.

[13] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Systems, Applications, and Services (MobiSys)*, 2010, pp. 49-62, doi: 10.1145/1814433.1814441.

[14] G. Pallis, "Cloud computing: The new frontier of Internet computing," *IEEE Internet Computing*, vol. 14, no. 5, pp. 70-73, 2010, doi: 10.1109/MIC.2010.113.

[15] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655-1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.

[16] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. 12th Annu. Int. Conf. Mobile Systems, Applications, and Services (MobiSys)*, 2014, pp. 68-81, doi: 10.1145/2594368.2594383.

[17] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. ACM SIGCOMM*, 2014, pp. 187-198, doi: 10.1145/2740070.2626296.

[18] T. Stockhammer, "Dynamic adaptive streaming over HTTP: Standards and design principles," in *Proc. 2nd Annu. ACM Conf. Multimedia Systems (MMSys)*, 2011, pp. 133-144, doi: 10.1145/1943552.1943572.

[19] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proc. 13th Annu. Int. Conf. Mobile Systems, Applications, and Services (MobiSys)*, 2015, pp. 151-165, doi: 10.1145/2742647.2742656.

[20] M. Xu, Z. Fu, X. Ma, L. Zhang, Y. Li, F. Qian, S. Wang, K. Li, J. Yang, and X. Liu, "From cloud to edge: A first look at public edge platforms," in *Proc. ACM Internet Measurement Conference (IMC)*, 2021, pp. 37-53, doi: 10.1145/3487552.3487815.