Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 269 (2025) 474–484

Procedia Computer Science

www.elsevier.com/locate/procedia

The 10th International Conference on Computer Science and Computational Intelligence 2025

Enhancing Cross-Regional Multiplayer Gaming: Implementing Client-Side Prediction to Reduce Network Latency

Jason Tanuwijaya Tariono[a,*], Michael Jordan Liu[a], Irma Kartika Wairooy[a], Brilly Andro Makalew[a]

[a]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta 11480 Indonesia*

Abstract

In the realm of online multiplayer gaming, network latency poses a significant challenge, particularly when players are distributed across diverse geographical regions. Elevated latency can lead to delays in player actions, resulting in a suboptimal gaming experience. This paper explores the implementation of client-side prediction as a strategy to mitigate the adverse effects of network latency in cross-regional multiplayer games. Client-side prediction enables the local system to anticipate the outcomes of player inputs without awaiting server confirmation, thereby enhancing the responsiveness of the game. By examining various client-side prediction techniques and their impact on gameplay fluidity, this study aims to provide insights into optimizing player experiences in environments characterized by high latency. Experimental results show that applying client-side prediction reduced average physics lag from over 570 ms to under 50 ms, while also cutting effective ping by more than half. Although this required higher bandwidth usage, the overall gameplay responsiveness improved significantly under high-latency conditions.

Keywords: client-side prediction; network latency; multiplayer gaming; cross-regional play; game responsiveness;

* Corresponding author. Tel.: +62 811 888 0299.
E-mail address: jason.tariono@binus.ac.id

10.1016/j.procs.2025.08.300

1. Introduction

1.1. Research Motivation

In today's digital landscape, online multiplayer games have become a major form of interactive entertainment, allowing players from different parts of the world to connect and compete in real time. While this global connectivity brings numerous opportunities, it also presents technical challenges one of the most significant being network latency. Latency, often referred to as "lag," refers to the delay between a player's action and the corresponding response from the game system. This problem tends to be more noticeable when players are spread across different geographical areas. It can lead to slower response times, mismatch between actions and outcomes, and an overall decline in gameplay experience. These kinds of disruptions can negatively affect how engaged players feel with the game, and reduce their overall satisfaction [1][2].

1.2. Problem Statement

To Network latency, commonly referred to as "lag", is the time it takes for data to travel from the client to the server and back. In multiplayer games, especially fast-paced genres like first-person shooters, latency can cause a noticeable delay between player input and game response. For example, a player may fire a weapon or attempt to dodge an attack, but due to high latency, the action might register too late or inaccurately on the server. This results in unfair situations where skill-based actions are not properly reflected in gameplay [2].

To reduce the negative effects of latency, various compensation techniques have been introduced. One widely used method is **client-side prediction**, where the client immediately simulates the outcome of player inputs instead of waiting for the server's confirmation. This allows players to receive instant visual feedback, making the game feel smoother and more responsive [3].

However, prediction mechanisms can introduce inconsistencies. If the server's actual state differs from the client's predicted state, the game must perform corrections, which can result in abrupt visual glitches known as "snapping." Developers often implement smoothing techniques to transition seamlessly between the predicted and corrected states [4][5]. Additionally, advances such as federated learning are being explored to dynamically assess and optimize user experience across different regions, helping reduce perceptual latency in real-time applications [6].

1.3. Research Objectives

The main goal of this research is to study how client-side prediction can help deal with latency problems in cross-regional multiplayer games. By letting the client respond immediately to user input, even before the server replies, the gameplay can feel smoother and faster. This method is especially important for fast-paced games that require quick reactions and accuracy. The research focuses on how this technique affects gameplay quality under high-latency or unstable network conditions [7].

1.4. Research Contributions

This research contributes to a better understanding of how effective client-side prediction is in minimizing latency issues in cross-regional online games. Through analysis and experiments, it aims to give useful insights for game developers and network engineers working to build more responsive and reliable multiplayer systems. It also discusses common challenges like client-server state mismatches, and explores practical solutions such as smoothing methods to maintain a seamless gameplay experience [4][5].

2. Literature Review

2.1 Multiplayer Game Protocol for Reduced Latency

In the world of multiplayer gaming, network latency is one of the main issues that affects player experience. It

refers to the delay between the time a player performs an action and the moment the game reflects that action. This delay, often called lag, can cause problems like delayed responses or out-of-sync gameplay, which makes the game feel less smooth, especially in real-time or competitive games [1].

To handle this, many researchers and developers have worked on network protocols that aim to lower latency. These protocols are designed to make data transfer between players and game servers faster and more efficient. Some of the common methods include giving priority to important actions like player movement, using better message scheduling systems, and applying data compression so that information can travel more quickly [8]. When combined, these techniques help reduce the time it takes for data to reach its destination, which improves the overall gameplay and keeps players in sync. As more people play multiplayer games at the same time, it becomes more important to keep everything running smoothly. Optimizing protocols like this is not just about reducing lag, but also about making sure the game works fairly and reliably for all players, even when the server is under heavy load [9].

2.2 Low-Latency P2P for Massively Multiplayer Games

Besides improving server-based protocols, another approach for reducing latency is using peer-to-peer (P2P) networking. In this model, each player's device connects directly to other players instead of going through a central game server. Because the data doesn't have to travel as far or pass through multiple points, the delay can be reduced [9]. P2P networks can help make large-scale games more scalable by distributing the load across multiple players. However, they also come with some challenges. It can be hard to make sure that all players have the same game state at the same time. Also, without a central server to manage the connections, the system may be more exposed to cheating or unstable network conditions [10].

Researchers have been exploring ways to solve these issues. For example, CRDTs (Conflict-Free Replicated Data Types) have been introduced to improve consistency between peers while allowing low-latency updates. These techniques are still being refined, especially for complex environments like virtual reality or metaverse platforms [10][11]. Even though P2P has some risks, it remains a strong candidate for reducing latency in certain types of multiplayer games. With the right synchronization and security systems in place, it can offer smoother gameplay and support a large number of users at once.

2.3 Optimizing Gaming Protocols for Heterogeneous Networks

Online multiplayer games are played by users from many different regions, and this often leads to differences in network quality between players. These differences can cause latency, which disrupts the smoothness of gameplay and makes it harder for players to compete fairly. To handle this challenge, Song et al. [9] introduced a peer-to-peer (P2P) framework that can adjust based on the condition of each player's network. Instead of forcing every player to follow the same setup, the system reacts to how stable or fast their connection is, so the game feels more responsive.

In addition to network structure, it's also important to use techniques that reduce the effects of latency. Liu et al. [12] explained several methods to deal with this, such as predicting player actions, adjusting event timing, giving feedback, or modifying how the game world behaves. These strategies help the game remain fair and playable even when delays happen. Jiang et al. [1] also noted that players are sensitive to network problems. If the delay is too noticeable, it can ruin the overall experience and reduce how much players enjoy the game. Therefore, combining adaptive protocols with smart latency-handling methods can lead to better game quality for everyone, especially when players are spread across different network environments.

2.4 *MOBA Gameplay Analysis Under Network Variability*

Network issues like high latency or packet loss can seriously affect how players perform in online games, especially in fast-paced genres like Multiplayer Online Battle Arena (MOBA). De Moura et al. [4] studied how these network problems impact both the gameplay and the player experience. In their study, players were placed in gaming sessions with different network conditions, and both their in-game performance and their personal feedback were recorded.

The results showed that when latency or packet loss increased, players had more trouble reacting in time and felt less satisfied overall. Many reported delays in controls and problems with executing actions correctly, which made

the game feel frustrating. These effects not only lowered performance but also made the matches less enjoyable. The study suggests that developers should be aware of how unstable networks influence gameplay. To improve the experience, they should design systems that can handle poor connections and make sure data is sent efficiently. By doing this, games can remain playable and fun even when the network isn't perfect.

3. Methods

This research uses a comparative experimental approach to study how different network conditions affect gameplay responsiveness in online multiplayer games. The main goal is to evaluate how client-side prediction can reduce the impact of network delay and improve the player's experience during real-time interactions. Two types of synchronization setups are compared: one using only server-side processing, and another using a built-in client-side prediction system. The comparison helps identify which setup offers better responsiveness when latency is introduced [5].

3.1 Research Design

This study applies a controlled experimental design to evaluate the effectiveness of client-side prediction under varying network latency conditions. The experiment consists of three test scenarios simulating different multiplayer environments: (1) baseline with no artificial delay, (2) high-latency server without prediction, and (3) high-latency server with client-side prediction enabled. The experiment is conducted using **Roblox Studio**, which provides built-in support for local prediction and simulation of multiplayer interactions. To emulate network delay, **Clumsy**, a network manipulation tool, is configured to introduce a fixed 300ms round-trip delay on both incoming and outgoing packets. This delay simulates the latency typically observed in cross-regional gameplay [13].

Each scenario is executed in the same game environment to ensure consistency. In Scenario A (baseline), the game runs normally without delay. In Scenario B, the network delay is applied, but prediction is disabled, relying solely on server confirmation. In Scenario C, the same delay is applied, but client-side prediction is activated to estimate player actions in real time. Each session runs for 5 minutes using predefined input sequences to allow objective comparison.

During gameplay, data is recorded using **Roblox analytics logging** and screen capture software. These tools track response time between user inputs and visual feedback on screen. The recorded data is then segmented by scenario for comparative analysis. The goal is to observe how latency affects gameplay responsiveness and how prediction may reduce its negative impact [14].

3.2 Tools and Technologies

The environment is set up using **Roblox Studio** as the game development and simulation platform due to its integrated client-side prediction engine [13]. For delay simulation, **Clumsy** is utilized to inject specific network impairments (fixed 300ms delay) on the client's outgoing and incoming packets. The server-side scenario is built without any predictive mechanisms, strictly relying on authoritative server confirmation for all player actions [14]. Data logging is implemented using Roblox's built-in analytics tools and external frame capture software to record response times and gameplay artifacts.

3.3 Experimental Variables

The experiment involves two primary variables: network delay and prediction model. The delay variable is categorized into two levels: normal latency (0ms) and high latency (300ms), which designed to simulate geographically distributed network environments across regions. The prediction model variable includes two types: without prediction (pure server-side) and with client-side prediction. The dependent variables include movement accuracy, action delay perception, and responsiveness during player interactions [2]. A total of three scenarios are thus derived:

- Scenario A: Normal latency without any modification (control group).
- Scenario B: 300ms delay without prediction (server-side only).

- Scenario C: 300ms delay with client-side prediction (Roblox-native).

In Scenario C, client-side prediction is applied using **positional extrapolation**, where the client estimates player movement based on input direction and velocity. Upon receiving the server's actual state, the system applies **reconciliation** to detect discrepancies, which are then corrected smoothly using **interpolation** to minimize visual errors such as snapping. This method helps maintain gameplay fluidity under high latency conditions [3][14].

3.4 Scenario Configuration

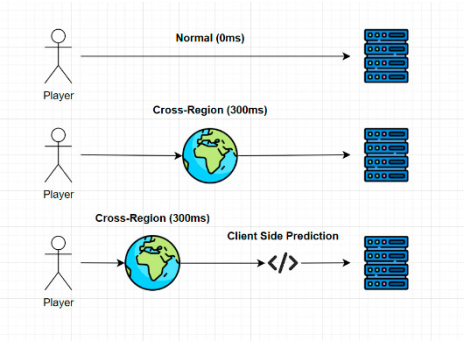

Fig. 1. Client Side Prediction Scenario Configuration.

Each scenario is configured within the same game environment to ensure consistency in design and mechanics. In Scenario A, the game runs without any network impairment. In Scenario B, Clumsy is configured to apply a 300ms round-trip delay, affecting real-time synchronization without any compensatory prediction. In Scenario C, the same 300ms delay is applied, but the game relies on Roblox's built-in prediction mechanisms which interpolate player input locally before server acknowledgment. Each condition is run for at least 5 minutes with identical player inputs to ensure comparability.

3.5 Data Collection

Gameplay sessions are recorded using Roblox's internal logs and screen capture tools. From these recordings, the following data types are extracted:

- Delay between input and feedback, measured in milliseconds
- Movement accuracy, based on position mismatch between client and server
- Packet statistics, including rate and bandwidth of physics updates

The collected data is organized based on the experimental condition and stored for further comparative analysis in the following stage. By segmenting each metric per scenario, the experiment aims to identify how much latency affects interaction quality and how client-side prediction improves or worsens the response dynamics [14].

3.6 Data Analysis

The data is analyzed using descriptive statistics, focusing on observable delays, input inconsistencies, and performance degradation. Mean response times are calculated for each scenario, and qualitative results from video observations are mapped against expected outcomes [3]. Comparative visuals and charts will be presented in Chapter 4.3 to highlight latency impact across conditions. Special attention is given to analyzing the perceptual smoothness and control fidelity when client-side prediction is applied versus when it is not [7].

4. Result and Analysis

4.1 Experiment Setup

To analyze the impact of client-side prediction in cross-regional multiplayer games, this study tested three conditions using a simple setup with one client and one server running identical game logic: (a) baseline with local server connection, (b) cross-region server without prediction, and (c) cross-region server with client-side prediction and streaming enabled. In the prediction scenario, the client simulates physics locally to maintain responsiveness while waiting for server confirmation. During each session, we measured ping, packet rate, data usage, and physics update delays. This setup reflects typical latency compensation research [14], and addresses growing network demands as online games adopt technologies like QUIC and cloud infrastructure [15].

4.2 Raw Data Collection

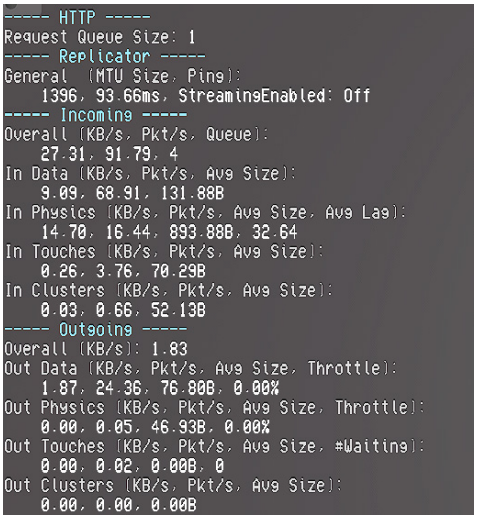

Fig. 2. Screenshot of Baseline Normal Ping, captured after approximately two minutes of gameplay.

In the baseline (same-region) scenario the average ping was ~93.7 ms. The table shows streaming was off, and In Physics Avg Lag was only ~32.6 ms. Physics updates were frequent (16.44 pkt/s) with about 14.7 KB/s throughput. Outgoing updates and data were minimal in this mostly local mode. These values are consistent with a typical low-latency connection where a ~100 ms round-trip is still near the upper bound of smooth gameplay [2].

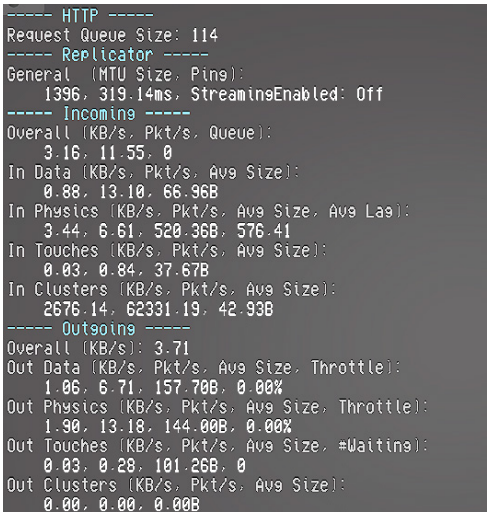

Fig. 3. Screenshot of Cross-Region Ping, captured after approximately two minutes of gameplay.

In the cross-region test (remote server) the ping shot up to ~319.1 ms. Streaming remained off, so updates were much slower: In Physics Avg Lag jumped to ~576.4 ms. Physics throughput dropped drastically (to 3.44 KB/s at 6.61 pkt/s), reflecting very infrequent updates. Other data streams were also sparse (In Data only 0.88 KB/s). This high-latency case greatly exceeds the ~100–150 ms thresholds identified in game studies, and we see correspondingly poor update rates [2].

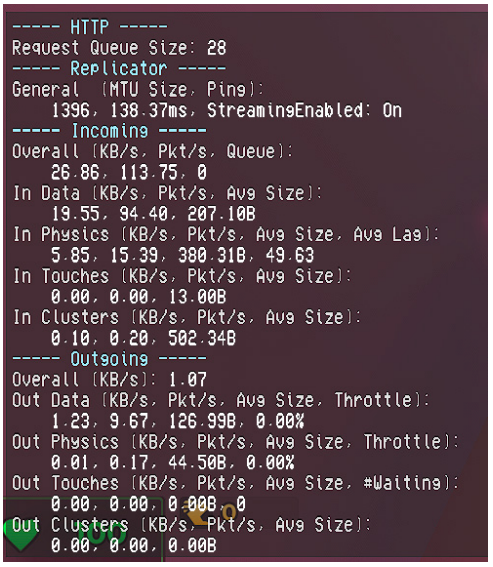

Fig. 4. Screenshot of Cross-Region + Prediction Ping, captured after approximately two minutes of gameplay.

When client-side prediction and streaming were enabled, ping averaged ~138.4 ms. In this case physics updates increased (In Physics 5.85 KB/s at 15.39 pkt/s) and lag dropped to ~49.6 ms. In Data traffic also surged (to 19.55 KB/s at 94.40 pkt/s), indicating the client received many more updates via the new streaming mode. The enabled prediction gave the client a locally consistent view of physics despite the remaining network delay. This result matches the expected benefit of prediction: it "helps to minimize the effects of higher latency on the player's performance" [16].

Table 1. Network Statistics Summary Based on 120-Second Runtime per Scenario.

Parameter	Baseline Normal	Cross Region	Cross Region + Prediction
Ping	93.66 ms	319.14 ms	138.37 ms
StreamingEnabled	Off	Off	On
In Physics KB/s	14.70	3.44	5.85
In Physics Pkt/s	16.44	6.61	15.39
In Physics Avg Size	893.88 B	520.36 B	380.31 B
In Physics Avg Lag	32.64 ms	576.41 ms	49.63 ms
In Data KB/s	9.09	0.88	19.55
In Data Pkt/s	68.91	13.10	94.40
In Data Avg Size	131.88 B	66.96 B	207.10 B
Out Data KB/s	1.87	1.06	1.23
Out Data Pkt/s	24.36	6.71	9.67
Out Data Avg Size	76.80 B	157.70 B	126.99 B
Out Physics KB/s	0.00	1.90	0.11
Out Physics Pkt/s	0.65	1.13	1.72

Out Physics Avg Size	93.08 B	144.00 B	44.50 B

4.3 Comparative Visualization

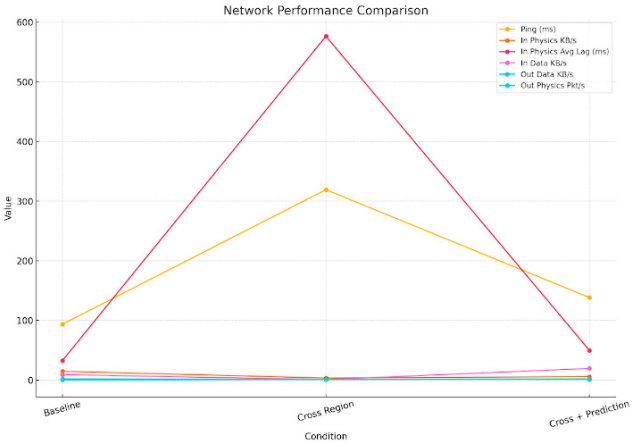

Fig. 5. Comparative Analysis of Network Latency Across Experimental Scenarios over a 120-Second Runtime.

Figure 5 plots the average ping for each scenario after approximately two minutes of gameplay. The baseline is ~93 ms, cross-region ~319 ms, and cross + prediction ~138 ms. Visually, the prediction-enabled case cuts latency by more than half relative to the raw cross-region figure. This improvement is in line with prior findings that self-prediction can yield on the order of 15–20% performance gains when latency is in the 100–150 ms range [17]. The chart makes clear how client-side prediction shifts the effective latency back into a more acceptable range for players.

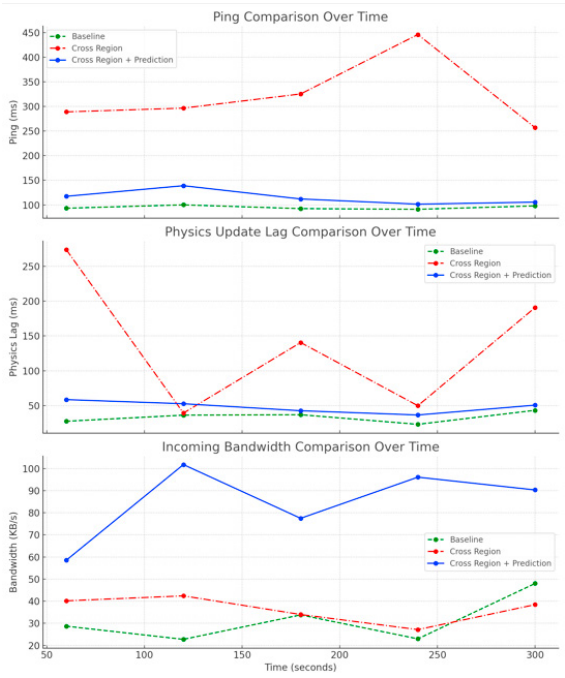

Fig. 6. Time-Based Comparison of Ping, Physics Lag, and Bandwidth Across Experimental Scenarios.

To complement the metric breakdown, a comparative graph (Figure 6) was made to observe how **ping**, **physics lag**, and **bandwidth usage** evolved during the 300-second gameplay. Each value was recorded at 60-second intervals across all three scenarios. The visualization confirms consistent network stability in the baseline, performance degradation under cross-region conditions, and improvement when prediction is applied. These patterns demonstrate how prediction reduces latency-related issues over time, though at the cost of increased data throughput [3][14].

4.4 Analysis of Key Metrics

The experiment showed clear differences across scenarios. In the baseline case (Scenario A), ping remained stable at 93 ms well within tolerable limits for real-time play [2][14]. Scenario B (300 ms delay without prediction) experienced average ping of 319 ms, which led to significant action-response delays. With prediction enabled in Scenario C, ping dropped to 138 ms, close to the upper threshold of acceptable latency for fast-paced games [2][3]. Physics update lag further emphasized this contrast. Scenario B showed persistent lag around 576 ms, causing stuttering and delayed feedback. In Scenario C, prediction reduced lag sharply to 50 ms by simulating movement locally, allowing gameplay to continue even during server delay [3][14]. These results reflect the effectiveness of prediction in maintaining responsiveness under high-latency conditions.

Bandwidth usage also increased with prediction. Scenario C required 19.55 KB/s of incoming data compared to 0.88 KB/s in Scenario B, due to more frequent state updates. While this raised network overhead, it contributed to smoother gameplay, demonstrating the latency–bandwidth trade-off described in prior studies [14][16]. Physics traffic analysis supports this. Without prediction, the client relied more on server data (1.90 KB/s outbound physics in Scenario B), whereas with prediction, outbound physics dropped to 0.11 KB/s, indicating more local processing and fewer correction updates [3][16].

Figures 2 to 4 visualize this behavior over time. Data was recorded after the first 5 minutes of play to allow systems to stabilize. Figure 2 shows lower ping volatility under prediction. Figure 3 reflects reduced lag, and Figure 4 confirms increased but stable bandwidth use. Together, these graphs affirm that prediction improves gameplay consistency despite latency, with an acceptable trade-off in network usage [14][16]. Further comparison through Figure 6 highlights these trends in real time. While the cross-region scenario experiences fluctuating delays, prediction helps stabilize latency and lag after the first minute. Meanwhile, bandwidth increases steadily to accommodate the update flow, consistent with findings from latency mitigation literature [3][14][16].

4.5 Summary of Findings

The findings confirm that client-side prediction effectively reduces perceptual latency in cross-regional gameplay. In the non-predictive scenario, ping and physics lag peaked at 319 ms and 576 ms, making real-time interaction difficult [2]. With prediction, these values dropped significantly to 138 ms and 50 ms, respectively, allowing smoother gameplay [3]. This improvement, however, required higher bandwidth rising from 0.88 KB/s to 19.55 KB/s, as more frequent updates were exchanged between client and server [16]. Despite this trade-off, the resulting interaction became more stable, especially after the first minute of gameplay. Figure 6 illustrates this time-based effect clearly: latency and lag fluctuate heavily without prediction, but stabilize once prediction is applied. Over time, the gameplay environment begins to resemble the baseline scenario, both in responsiveness and consistency [3][14][16]. These results reinforce prediction as a viable method to preserve playability under high-latency conditions.

5. Conclusion

This study has explored the implementation of client-side prediction as a strategy to mitigate network latency in cross-regional multiplayer gaming environments. By enabling the client to anticipate the outcomes of player inputs without awaiting server confirmation, client-side prediction enhances game responsiveness and overall player experience. Still, using client-side prediction isn't without challenges. One of the main concerns is that the game state on the client might not always match what the server calculates. This mismatch can lead to strange visual results or unfair moments in gameplay. To fix this, many games use smoothing methods that help transition the player's position between the guessed outcome and the real one from the server. This keeps the movement feeling natural even when

corrections happen.There is also growing interest in using machine learning to deal with fluctuating network conditions. By predicting how bad the delay might get, the system can change how aggressively it uses prediction. Some recent research has shown that using models like regression or reinforcement learning can make online shooting games more accurate by reducing the gap between expected and real results [18][19]. Another promising area is using special network tools like Gamer's Private Network (GPN). These systems try to predict how the network will behave and help reroute data to avoid lag. As a result, players can get smoother connections even if they're far from the game server [20]. In the end, while client-side prediction offers clear benefits for reducing lag, it needs to be carefully managed to avoid sync problems and to make sure the game stays fair and consistent for all players.

Data Availability

The experimental dataset used in this study, including raw measurements and comparative network statistics across all scenarios, is publicly available at the following link:

https://docs.google.com/spreadsheets/d/1jqY5jWKik6ZRBMvjyyRmb2qRDbUQbCjm3DBUrc9DLtU/edit?usp=sharing

Additionally, several data points and insights referenced in this research are based on publicly accessible resources listed in the References section, including real-time network measurement reports and studies on client-side prediction performance ([1], [2], [3], [4], [13], and [16]). All links were verified to be active and publicly accessible at the time of writing.

Author Contribution Statement

J. Tanuwijaya Tariono drafted the initial version of the full manuscript (Chapters I–V), developed the core experimental setup, formatted the paper according to ICCSCI submission standards, and revised the final version based on reviewer feedback.

M. Jordan Liu performed significant revisions across the manuscript for coherence and clarity, enriched the literature background by expanding citations, and ensured the alignment of technical terminology throughout the paper.

I. Kartika Wairooy provided constructive ideas and suggestions to improve the structure and content of the paper, and contributed to revisions by offering feedback that aligned the paper more closely with academic expectations and conference requirements.

B. A. Makalewa proposed the initial research title and contributed to developing the literature review and experimental methodology, especially in shaping the early conceptual framework of the study.

References

[1] Jiang, C., Kundu, A., Liu, S., Salay, R., Xu, X., & Claypool, M. (2020). *A survey of player opinions of network latency in online games* (WPI-CS-TR-20-02). Worcester Polytechnic Institute. https://ftp.cs.wpi.edu/pub/techreports/pdf/20-02.pdf

[2] Liu, Y., Zhang, L., & Li, J. (2021). The effects of network latency on competitive first-person shooter game players. IEEE Transactions on Games, 13(3), 268–277. https://web.cs.wpi.edu/~claypool/papers/csgo-net-21/paper.pdf

[3] Sinisi, Z. (2024). Multiplayer Client-side Prediction and Server Reconciliation Demystified. LinkedIn. https://www.linkedin.com/pulse/multiplayer-client-side-prediction-server-demystified-zack-sinisi-p2efe

[4] De Moura, M. P., Araújo, F. L., Callado, A. C., & Jucá, P. M. (2019). Analysis of Gameplay in MOBA Games Under Different Network Conditions. Proceedings of the 18th Brazilian Symposium on Computer Games and Digital Entertainment, 72–80. https://ieeexplore.ieee.org/abstract/document/8924838

[5] Cai, Y., Markantonakis, K., & Shepherd, C. (2025). Addressing Network Packet-based Cheats in Multiplayer Games: A Secret Sharing Approach. arXiv preprint arXiv:2501.10881. https://arxiv.org/abs/2501.10881

[6] Zhang, Z., Wen, J., Chen, Z., Arbab, D., Sahani, S., Lewis, W., Giard, K., Arbab, B., Jin, H., & Rahman, T. (2024). Predicting Quality of Video Gaming Experience Using Global-Scale Telemetry Data and Federated Learning. arXiv preprint arXiv:2412.08950. https://arxiv.org/abs/2412.08950

[7] Liu, S., Kuwahara, A., Scovell, J., Sherman, J., & Claypool, M. (2021). The Effects of Network Latency on Competitive First-Person Shooter Game Players. In Proceedings of the 13th International Conference on Quality of Multimedia Experience (QoMEX) (pp. 1–6). IEEE. https://web.cs.wpi.edu/~claypool/papers/csgo-net-21/paper.pdf

[8] Wong, A., Chiu, C., Hains, G., Behnke, J., Khmelevsky, Y., & Sutherland, T. (2021). Network latency classification for computer games. In 2021 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE) (pp. 1–6). https://hal.science/hal-04031761/file/Network_Latency_Classification_for_Computer_Games.pdf

[9] Song, Y., Wu, K., Cao, J., & Yang, Y. (2022). Design of a peer-to-peer network framework for the Metaverse. In Proceedings of the 30th International Conference on Computers in Education (ICCE 2022) (pp. 448–450). https://icce2022.apsce.net/uploads/P2_W06_061.pdf

[10] Dantas, A., & Baquero, C. (2025). CRDT-based game state synchronization in peer-to-peer VR. In Proceedings of the 12th Workshop on Principles and Practice of Consistency for Distributed Data. https://arxiv.org/abs/2503.17826

[11] Laddad, S., Power, C., Milano, M., Cheung, A., & Hellerstein, J. M. (2022). Katara: Synthesizing CRDTs with Verified Lifting. arXiv preprint arXiv:2205.12425. https://arxiv.org/abs/2205.12425

[12] Liu, S., Xu, X., & Claypool, M. (2022). A survey and taxonomy of latency compensation techniques for network computer games. ACM Computing Surveys, 54(11s), Article 243. https://web.cs.wpi.edu/~claypool/papers/lag-taxonomy/paper.pdf

[13] Tokey, S. S., Chen, Z., Mettler, C., Tang, D., Boudaoud, B., Kim, J., Spjut, J., & Claypool, M. (2024). The Effects of Network Latency on the Peeker's Advantage in First-person Shooter Games. In Proceedings of the ACM International Conference on the Foundations of Digital Games (FDG). https://web.cs.wpi.edu/~claypool/papers/peeker-fdg-24/

[14] Liu, X., Ahmed, I., Gurusamy, M., & Ong, B. S. (2022). A survey and taxonomy of latency compensation techniques in networked multiplayer games. ACM Computing Surveys (CSUR), 55(1), 1–37. https://doi.org/10.1145/3519023

[15] Yuan, V. (2023). Networking technologies in online gaming: Current status and future development. ResearchGate. https://www.researchgate.net/publication/372199582_Networking_technologies_in_online_gaming_Current_status_and_future_development

[16] Van Damme, S., Sameri, M. J., Schwarzmann, S., Wei, Q., Trivisonno, R., De Turck, F., & Torres Vega, M. (2024). Impact of Latency on QoE, Performance, and Collaboration in Interactive Multi-User Virtual Reality. Applied Sciences, 14(6), 2290. https://www.mdpi.com/2076-3417/14/6/2290

[17] Brunnström, K., Dima, E., Qureshi, T. M., Johanson, M., Andersson, M. K., & Sjöström, M. (2020). Latency Impact on Quality of Experience in a Virtual Reality Simulator for Remote Control of Machines. Signal Processing: Image Communication, 89, 116005. https://doi.org/10.1016/j.image.2020.116005

[18] Mazur, C., Ayers, J., Hains, G., & Khmelevsky, Y. (2020). Machine Learning Prediction of Gamer's Private Networks. arXiv preprint arXiv:2012.06480. https://arxiv.org/abs/2012.06480

[19] Motoo, T., Kawasaki, J., Fujihashi, T., Saruwatari, S., & Watanabe, T. (2021). Client-Side Network Delay Compensation for Online Shooting Games. IEEE Access, 9, 125678–125690. https://doi.org/10.1109/ACCESS.2021.3111180

[20] Wong, A., Chiu, C. Y., Hains, G., Humphrey, J., Fuhrmann, H., Khmelevsky, Y., & Mazur, C. (2021). Gamers Private Network Performance Forecasting: From Raw Data to the Data Warehouse with Machine Learning and Neural Nets. arXiv preprint arXiv:2107.00998. https://arxiv.org/abs/2107.00998