Adaptive Time Delay for Improving Player Experience and Fairness in First-Person Shooter Games with Network Latency

Samin Shahriar Tokey
Worcester Polytechnic Institute
Worcester, MA, USA
sstokey@wpi.edu

Ben Boudaoud
NVIDIA
Durham, North Carolina, USA
bboudaoud@nvidia.com

Josef Spjut
NVIDIA
Durham, North Carolina, USA
josef.spjut@gmail.com

Mark Claypool
Worcester Polytechnic Institute
Worcester, MA, USA
claypool@wpi.edu

Abstract

In a multiplayer networked game, actions for players with higher latencies are received and (potentially) acted upon later than players with lower latencies, leading to unfairness, especially important in competitive games. Time delay is a latency compensation technique that can mitigate this unfairness by adding latency to players with lower latency so that all players experience the same latency. Although this provides equal latency to all players, it unnecessarily degrades the responsiveness for the lower-latency players when the players are not interacting. We propose an adaptive time delay technique that only adds latency to low-latency players when they are interacting with players with higher latency. We conducted three separate user studies assessing player performance and experience with network latency and different compensation techniques. Analysis of the results shows adaptive time delay improves average quality of experience compared to fixed time delay while preserving time delay's fairness.

CCS Concepts

• **Applied computing** → **Computer games**; • **Human-centered computing** → *User studies*; Laboratory experiments.

Keywords

First Person Shooter, Network Latency, Networked Game, Latency Compensation, Time Delay, Adaptive Time Delay

ACM Reference Format:
Samin Shahriar Tokey, Ben Boudaoud, Josef Spjut, and Mark Claypool. 2026. Adaptive Time Delay for Improving Player Experience and Fairness in First-Person Shooter Games with Network Latency. In *Foundations of Digital Games (FDG '26), August 10–13, 2026, Copenhagen, Denmark.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3815598.3815633

1 Introduction

The computer game industry has seen significant growth in recent years, with the number of gamers increasing from about 2 billion in 2015 to over 3 billion in 2024 [6]. The online gaming market is also thriving, with an estimated worldwide market size of $56 billion USD in 2021 and predictions that this market will grow at a 10.2% annual growth rate, reaching $132 billion USD by the year 2030 [3].

Among the most popular genres in the online gaming industry are first-person shooter (FPS) games which are enjoyed by both casual and competitive gamers. Multiplayer FPS games typically use a client-server architecture with an authoritative server that coordinates and manages all the actions and interactions between players. The authoritative server is responsible for processing the actions of all players, allowing them to interact and compete in real-time.

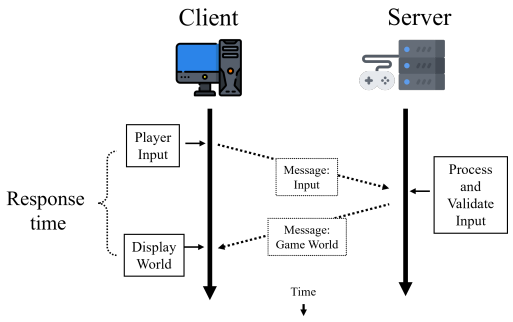


Figure 1: Latency between client and server.

Figure 1 illustrates an online game where the client sends player input to the server, which processes the input and returns an update. The time from when the player provides input to the client until the resulting change is displayed on the screen is the response time. In an online game, the response time is impacted by the network latency - the time it takes for a packet of data to travel from a player's client to the server and back. Physical distance and routing impacts latency, as the more network devices ("hops") between a client and server, the longer it takes for a packet to travel. As a result, players who are farther away from the server may experience higher latencies compared to players who are closer. High latency means greater delayed receipt of actions by the server and subsequent responses sent and displayed by the client, making it difficult for the player to react and make split-second decisions in fast-paced games. Latency can also cause de-synchronization between the

FDG '26, Copenhagen, Denmark

ACM ISBN 979-8-4007-2495-4/26/08
https://doi.org/10.1145/3815598.3815633

player's actions and the game's state, leading to inconsistencies between the authoritative state at the server and the state displayed at the client.

In a multiplayer FPS, differences in latency between players can lead to unfair advantages. Figure 2 illustrates two players connected to a server, with Client A positioned farther from the server than Client B. The server sends each client the other's position, but Client A receives the information later than Client B due to higher latency. Both players attempt to shoot each other upon seeing their opponent's avatar, but Client B's shot is a confirmed hit first because of its lower latency, causing Client A to get an eliminated message. This latency discrepancy impacts fairness by favoring players with lower latency.

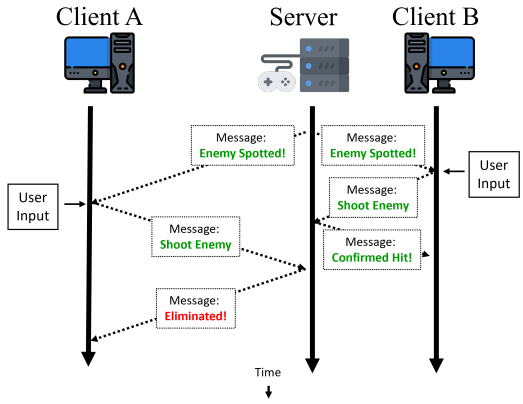


Figure 2: Latency and fairness.

Time delay is a latency compensation technique used in multiplayer games to improve fairness in gameplay by equalizing the latency between players [19]. The basic idea is to artificially add latency to the player with the lower latency, bringing their total latency equal to that of the player with higher latency. Doing so negates the unfair advantage that the player with lower latency may have had due to their faster response time by, effectively, providing both players with the same high latency. Figure 3 show the same scenario as in Figure 2 but with time delay. The server's response to actions from Client B are delayed to equal the higher latency for Client A, which improves fairness between the players. However, while improving fairness, time delay's decrease in responsiveness for the player at Client B has degraded their quality of experience (QoE) by making both players have high latency.

We propose adaptive time delay as a potential improvement over time delay to preserve fairness but mitigate some of time delay's increased latency for low-latency clients. With adaptive time delay, instead of adding latency to the low latency player unconditionally, adaptive time delay only adds delay during player interactions with other players. This means that when players are not interacting, time delay is turned off, allowing the low latency player to enjoy their original low latency. By only adding time delay during player interactions, adaptive time delay can improve the average response time for the low latency player and their gaming experience, while preserving the fairness in gameplay between players.

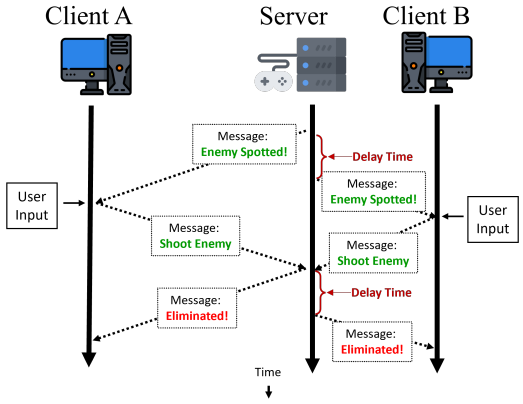


Figure 3: Time Delay.

The aim of this research is to evaluate adaptive time delay. While adaptive time delay should maintain the fairness offered by time delay, the addition and removal of artificial latency during gameplay may in itself degrade the gaming experience. Thus, we have two primary research questions to explore: R1 - does adaptive time delay preserve the fairness of time delay, and R2 - how does adaptive time delay affect player experience compared to time delay.

To address these research questions, we conducted three separate user studies. For the first user study, we built a single player FPS game called Zombiefield to evaluate the impact of adapting delay on player QoE. For the second and third user studies, we built two separate multiplayer FPS games called Last Stand and Color Clash to evaluate both fairness and QoE. For all games, adaptive time delay detects possible player interaction (e.g., by line of sight and proximity between players) and then adds delay either abruptly or gradually. Experimental design also controlled the network latency and provided fixed (non-adaptive) time delay for comparison.

Analysis of the results for nearly 100 users shows adaptive time delay can improve player QoE compared to fixed time delay when latency is high, and improves fairness compared to no time delay. Our results also suggest that adding latency gradually with adaptive time delay is important to improving QoE compared adding latency abruptly.

The rest of this paper is organized as follows: Section 2 provides related work, Section 3 details our user study methodologies, Section 4 analyzes the results, Section 5 discusses the implications of the research, Section 6 mentions some limitations and possible future work, and Section 7 summarizes our conclusions.

2 Related Work

This section summarizes studies related to our work in two main areas: latency and first-person shooter (FPS) games and latency compensation to improve fairness.

2.1 Latency and First-Person Shooter Games

Numerous studies have detailed the effects of network latency on games, highlighting its impact on gameplay and player experience [1, 2, 5, 8, 10, 25, 30]. Many of these studies, including our

approach in this paper, utilize games with controlled levels of network latency in a laboratory environment.

In particular, several studies have shown that latency degrades gameplay and player experience in FPS games. For example, Amin et al. [1] demonstrate that player experience defines and determines sensitivity to latency in *Call of Duty*, with competitive gamers being more adept at compensating for impaired conditions. Armitage et al. [2] estimate the latency tolerance threshold for *Quake 3* to be about 150-180 ms. Quax et al. [26] show that for players of *Unreal Tournament 2003*, latency and latency jitter under 100 ms can degrade performance and quality of experience (QoE). Liu et al. [18] measure player performance and QoE in *Counter-Strike: Global Offensive*, finding accuracy decreases by 3% and score decreases by around 17% as latency increases to 150 ms. QoE follows suit, degrading 25% between 25 ms and 150 ms. Their subsequent work shows without latency compensation network latencies manifest similarly to local system latencies [17] but generally, local latency has a greater impact. Our earlier work [33] investigated the impact of FPS "peeker" player advantage across different network latencies, finding defending players with latencies of 120 ms had about a 30% disadvantage. While high base latency is known to degrade performance, a study by Schmid et al. [29] on FPS games suggests that small, continuous latency variations (jitter) have no significant impact on player accuracy or game experience.

2.2 Latency Compensation to Improve Fairness

Various techniques have been developed to mitigate the effects of network latency in games. Time manipulation, for example, adjusts the virtual time used for computing the game state and resolving player actions, improving fairness in gameplay [19]. Time warp (or rollback) and time delay are two well-known methods that leverage time manipulation to combat the effects of latency.

2.2.1 Time Warp. Time warp rolls back game state to when a player action occurred on the client, applies the action, then rolls the game state forward to the current time. Originally proposed by Jefferson [11] as an implementation of virtual time for distributed computation, time warp has since been formalized for continuous interactive media such as computer games [20, 22, 23] and identified as a means to overcome inconsistencies caused by client-side prediction [12]. In commercial FPS games, time warp is associated with the "shot around the corner" problem, where a server rolls back time to validate a hit on a player who has already moved to safety [14–16, 28]; advanced variants attempt to mitigate this by limiting rollback when a client identifies itself as currently safe [15]. Sun and Claypool [31] evaluate time warp for cloud-based game streaming, finding it can mitigate latency effects on player performance with projectile weapons at the cost of more visual inconsistencies. While our study does not focus on time warp, it is relevant because improving fairness is one of our primary objectives, and time warp is known to enhance fairness in gameplay.

2.2.2 Time Delay. Time delay is a technique of adding delay to incoming or outgoing packets to make latency equal for all players.

Several studies have explored latency compensation techniques and their impact on fairness in gaming environments. Paik et al. [24] proposed two methods for adjusting server-side delays based on player proximity and interaction probability, aiming to balance responsiveness and fairness in networked virtual environments (NVEs) with large player counts. Zander et al. [36] implemented a Self-Adjusting Game Lagging Utility to add outgoing delays, using bots to simulate player behavior and assess fairness in games. Kaiser et al. [13] increased fairness in Quake 3 Arena by combining game update messages into larger packets before transmission, implementing time delay to balance gameplay. Brun et al. [4] used a heuristic approach to select and synchronize distributed servers with incoming latencies, enhancing fairness while slightly increasing response time. Savery et al. [27] tested server-side and client-side delays, suggesting the local latency approach improves fairness in real-time strategy games where slight response delays are acceptable. Mauve et al. [21] demonstrated that introducing a 250 ms local latency in a 3D teleoperation application reduced inconsistencies and improved fairness by eliminating short-term artifacts. More recent work continues to address latency and fairness. Ferreira et al. [7] propose fairness-oriented multicast routing that balances per-user path latency at the network layer for distributed interactive applications. Halbhuber et al. [9] use an artificial neural network to compensate for high latency in cloud gaming, improving performance and experience but without targeting cross-player fairness.

Our adaptive time delay differs from prior server-side methods and from Paik et al. [24] in particular: Paik et al. tune a server waiting period for large networked virtual environments using map-based interaction probability and delay ranking, whereas we target pairwise FPS combat in real playable games, switching delay on only when a line-of-sight or proximity interaction is detected so low-latency players keep their native responsiveness when not engaging an opponent. We also evaluate with human players in a controlled user study, rather than through simulation.

Our preliminary work with the single-player game Zombiefield showed that adaptive time delay can improve overall responsiveness compared to regular time delay [32]. In this paper, we build on that result by implementing both time delay and adaptive time delay in three FPS games – one single-player and two multiplayer – and comparing their effects on fairness and responsiveness in tests using human players.

3 Methodology

To evaluate the effectiveness of adaptive time delay in improving fairness and responsiveness in FPS games, we conducted 3 user studies, each independently designed but with overlapping parameters to allow for comparisons across studies. The user studies are each within-subjects, using a full-featured, playable game in a laboratory environment with controlled amounts of latency and time delay. However, they differ in the game details and participant sample, and differ slightly in the methods of enabling adaptive time delay and added network latency. The similarities allow for comparisons across studies, while the differences strengthen the potential that the analyzed results will hold beyond just the games and environments each tests individually.

The overall approach used for each study had the following steps in common:

(1) Make custom FPS game. The player experience in an FPS varies based on the game design. Our studies used three different FPS games, primarily differing in objective, map size and terrain, weapon characteristics, and graphics fidelity. One game was single player to evaluate the responsiveness, and two were multiplayer to evaluate fairness and responsiveness.

(2) Mimic client-server networking. The studies reproduced (or emulate, for the single player game) a client-server network architecture with an authoritative server. In this architecture, all player actions were transmitted to the server, acted upon and sent back before the player sees the outcome.

(3) Simulate network latency. Network latency was simulated in each game in order to evaluate the effects of latency on the players with and without different compensation techniques. Figure 4 shows the latency simulation method. Player inputs were intercepted and put into queues before being acted upon along with an execution time equal to the current system time plus added player latency. The queue was checked each tick of the game loop and when the execution time for any action was reached or surpassed, the input was taken from the queue and processed normally.

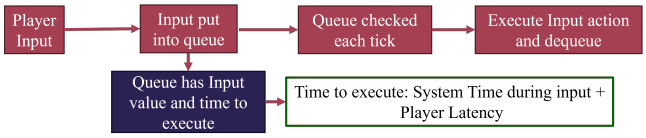


Figure 4: Simulating network latency.

(4) Implement latency compensation. For our studies, two kinds of latency compensation were implemented in the games.

- **Time Delay.** Time delay was implemented by delaying each player's inputs by the same amount, providing equal latency.
- **Adaptive Time Delay.** Adaptive time delay was implemented by only adding latency to a player's input when potentially interacting with other players. The method of detecting player interaction was slightly different in each study. In addition, the methods for adjusting the latency varied in each study – latency could be increased gradually over time or abruptly all at once, or after adjusting, latency could be kept at a fixed level before readjusting regardless of potential player interactions. Our study compared four different adaptive time delay implementations:
  - *Instant.* Changed latency to the target immediately when activated – i.e., in the next game frame.
  - *Abrupt.* Changed latency quickly within 50 to 100 ms, based on the target or configuration.
  - *Smooth.* Changed latency slowly over 1 to 2 seconds, depending on target or configuration.
  - *Keep on/off.* Once a target is reached (on or off), latency was fixed for 2 seconds.

  Adaptive time delay in all studies was activated when players (or AI opponents) were able to potentially interact, as determined by visibility or proximity checks; when such conditions are met, latency was enabled, and when they are not, latency was disabled. Figure 5 shows adaptive time delay activated while there was no obstacle between the players, and deactivated when there was an obstacle between the players and/or they were not facing each other.

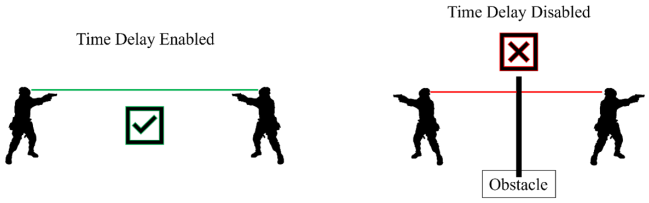


Figure 5: Adaptive time delay activation based on potential player interaction.

(5) Devise experimental harness. Each study had a test harness to control parameter settings for the user from round to round. The harness read in parameters from disk at the start of the session and then controlled settings such as latency, round length, and type of latency compensation with randomization of experimental condition order across users.

(6) Build log system. The games had logging systems that recorded player telemetry and performance metrics during gameplay.

(7) Hardware Used. All studies used high-performance gaming PCs equipped with modern multi-core Intel processors, NVIDIA graphics cards, and at least 32 GB of DDR4 RAM, with storage on SSDs. Input was provided by Logitech G502 gaming mice (1000 Hz polling rate), with output to high refresh rate gaming monitors (240–360 Hz) at 1080p resolution. Systems were connected via wired Ethernet where applicable. Local end-to-end latency was measured using either a high-speed camera (1000 Hz Casio EX-ZR100) or the NVIDIA Reflex Latency Analyzer, with average latencies in the range of 25–31 ms and standard deviations around 3 ms.

(8) Recruit users. All studies were approved by our university's Institute Review Board (IRB). Volunteers were recruited through university mailing lists with various incentives (class credit, gift card raffle) offered for participation.

(9) Conduct Study. Participants signed consent forms, provided demographic data, completed a reaction time test [35] and familiarized themselves with the game controls. They then played short game rounds under different conditions while the system recorded various performance metrics in the background. After each round, participants provided Quality of Experience (QoE) and fairness assessments before proceeding to the next round.

(10) Analyze Data. The studies each gathered a variety of data, with this paper primarily focusing on assessing fairness and QoE.

3.1 Study 1 - Zombiefield

The goal of Study 1 is to study potential improvement to QoE for adaptive time delay, along with adaptation strategies.

Study 1 used a custom game *Zombiefield* – a single player FPS game developed with Unreal Engine 5 – extended to study adaptive time delay. The goal of Zombiefield was to survive unending waves of zombies, eliminating as many as possible. The player's avatar had 100 health and took rapid damage by close zombies. When the avatar health reached zero, it respawned to its original location,

and the zombies also reset. Since Zombiefield was single player, interaction-like encounters triggered adaptive time delay were done by a special type of "network" zombie that were faster and stronger than standard zombies and had more health.

Zombies spawned in cycles, each cycle repeating every 5-10 seconds. In a cycle, there was a 50% chance of spawning either three standard (non-networked) zombies with 2 hit-points (HP) or one strong (networked) zombie with 5 HP, with an additional 10% chance to spawn a super strong (networked) zombie with 15 HP Each bullet inflicted 1 HP damage on any type of zombie. Player score increased by 1 point each second of survival and 10 points for each zombie kill. If the player avatar died, the score reset back to 0.

Zombiefield used a custom-built control rig to control the inverse kinematics of the player avatar. This made the weapon animations (sway, recoil, aiming) procedural. The avatar's gun was a full auto rifle with a laser pointer attached. The bullets were projectiles with tracers to make it easier for the player to see where the bullets were going. The player could also aim down the scope to better aim at distance. WASD controls were used for moving, and aiming and firing with the mouse. The player could make the avatar jog, sprint, and strafe to avoid Zombies.

Figure 6 shows a Zombiefield screenshot. The heads-up display (HUD) shows score, high-score, duration, health, rounds, ammo, kills and deaths.

Figure 6: Zombiefield in-game screenshot.

In the user study, participants played a round for 4 minutes to practice and familiarize themselves with the game mechanics. Data collected from the practice round was not analyzed. Then, the participant played a total of 16 rounds with various latency conditions. They are:

- **No latency.** 1 round with 0 ms of added latency.
- **Fixed time delay.** 3 rounds with fixed time delay of 50 ms, 100 ms or 150 ms of latency.
- **Adaptive time delay.** 12 rounds with adaptive time delay:
  - Instant - 3 Rounds with a base latency of 0 ms, adapting to 50 ms, 100 ms or 150 ms. Another 3 rounds going from half of target latency to 50 ms, 100 ms or 150 ms.
  - Smooth - 3 rounds going from 0 ms to 50 ms, 100 ms or 150 ms over 2 seconds.
  - Keep on/off - 3 rounds going from 0 ms to 50 ms, 100 ms or 150 ms but keeping adaptive time delay enabled or disabled for at least 2 seconds each change.

Each round was 75 seconds long. After the round ended, users were asked two questions regarding their experience in that round:

- *Rate your Quality of Experience for this round* - 1 (Low) to 5 (High) via a slider.
- *Was this experience acceptable?* (Yes/No) via on-screen buttons.

3.2 Study 2 - Last Stand

The goal of Study 2 is to also to study potential improvement to QoE from adaptive time delay, but also study fairness along with adaptation strategies.

Study 2 used a custom two player game *Last Stand* [34]. Last Stand was built using Unity HDRP, which included hitscan-based gunplay and positional audio. Players were pitted against each other in a 1v1 setting with rounds of deathmatch. Dying made the player's avatar respawn at the most distant of six predetermined spawn points, chosen to be as far away from the other player's avatar as possible. On respawn, player's health and ammo was replenished. There was also 1.5 seconds of invincibility period on respawn, which was indicated by the green flashing health icon on the bottom left side of the screen.

Although the main game had multiple weapons of varying types and firing rates, our study used only a semi-automatic rifle with 11 bullets and unlimited magazines. A laser was projected from the muzzle of the gun for aiming assistance. Players could also aim down sight for better aiming at distance.

The FPS Animation Framework[1] was used to animate the gun and the player's avatar. Recoil, weapon sway, look and leaning, among others, were procedurally animated. All of these animations were replicated over the network.

WASD were used for moving, and aiming and firing with the mouse. Players could walk, sprint and lean left or right. Although the game auto reloaded if the player ran out of ammo, the player could also do a tactical reload before the ammo runs out.

Figure 7 shows a screenshot of the game with some of the UI elements. The gameplay took place in an abandoned hanger, surrounded by large trees, ground foliage and terrain. Trees were strategically planted to provide cover. The game displayed score, health, kills, deaths, ammo, rounds and duration. Yellow indicators lit up when the player got close to an opponent, marking the direction and proximity of the opponent. For example, a yellow bar appeared at the top of the screen if the opponent avatar was in front of the player. The yellow bar got bigger and brighter as the player got closer to the opponent. While being shot, the yellow markers turned red. For shooting, a white hit marker was displayed on a confirmed hit, yellow for headshots and red for eliminations.

For each confirmed hit, players got 1 point, headshots were 5 points and kills were 10 points. Score, kills and deaths were reset after each round.

For the user study, participants had 1 practice round followed by 20 rounds with different latency configurations. They were:

- **No Latency.** A condition with no latency for both players. 1 round.
- **Unfair Conditions.** One player had no latency while the other had a fixed 100 ms and 200 ms latency. 2 rounds.

[1]FPS Animation Framework - https://assetstore.unity.com/packages/tools/animation/fps-animation-framework-238641

Figure 7: Last Stand in-game screenshot.

- **Latency Conditions.** Both players had a fixed latency. 3 rounds with 100 ms vs. 100 ms, 200 ms vs. 200 ms and 100 ms vs. 200 ms.
- **Adaptive Time Delay.** The unequal latency conditions were replayed but with adaptive time delay enabled.
  - Abrupt - Abruptly made one player go from 0 ms to 100 ms or 200 ms over 3-6 frames. 2 rounds.
  - Smooth - Smoothly made one player go from 0 ms to 100 ms or 200 ms over a few seconds. 2 rounds.
- *10 more rounds were with Player 1 and Player 2 configurations swapped.*

Each round lasted 80 seconds. After each round ended, users were asked two questions regarding their experience in that round:

- *Rate how laggy did you feel the round was* - (1) Very Laggy to (5) Very Smooth via a slider.
- *Was the round fair?* (Yes/No) via on-screen buttons.

3.3 Study 3 - Color Clash

The goal of Study 3 is to continue study of potential improvement to QoE and fairness from adaptive time delay, but incorporate an additional game mechanic that encourages gameplay that does not necessarily interact with other players.

Study 3 used our custom two-player game *Color Clash*. Although its networking architecture was similar to Last Stand, Color Clash features simpler graphics, a map with verticality, a unique 'painting' mechanic to encourage gameplay without interacting with the other player, and a different method for detecting player interaction. In Last Stand, both players had symmetric configurations, however, Color Clash both players had a control player and an experimental player.

Color Clash was developed with Unity using the NetCode for Game Objects (NGO) along with a 60x60 map, shown in Figure 8, that consisted of obstacles, tiles, ramps, and platforms. WASD controls were used for moving, the spacebar for jumping, and aiming and firing with the mouse. If a player's avatar was hit, it was eliminated and the opposing player gained a point. Once eliminated, the opponent's screen displayed 'Respawning', and the opponent avatar respawned at a random location away from the other player after three seconds.

In addition to shooting, the game included a painting mechanic that provided an opportunity for players to aim and shoot at fixed floor targets, in addition to when they interact with the opponent player. All tiles, except the ramps, contained dark gray circular

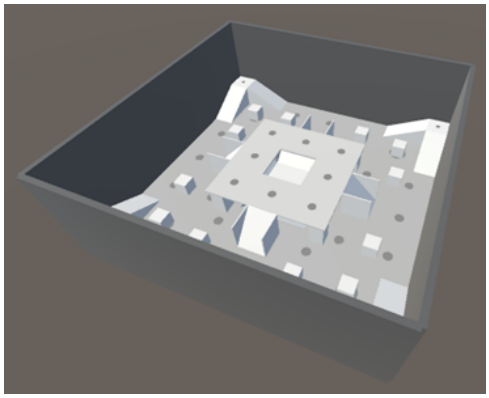


Figure 8: An overview of Color Clash's map.

targets. Players painted their assigned color – red or blue – by right clicking on the mouse. If a player hit a target, the whole tile was painted their color (depicted in Figure 9). Six tiles were painted with each color (twelve total) randomly before each round began. When players moved across a tile of their color, their speed increased by 30%, and when they moved across a tile of their opponent's color, their speed decreased by 20%.

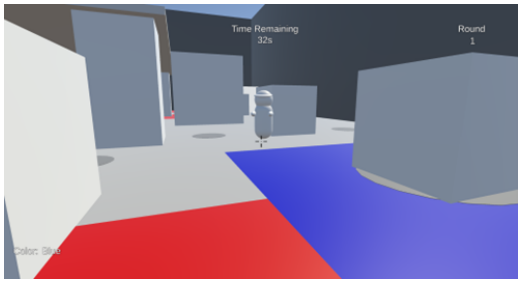


Figure 9: Painted tiles in the Color Clash map.

Players had one crosshair for aiming when painting and shooting. The paint gun had a delay of 250 ms between shots and unlimited ammo.

Before the round starts, each player pressed a "ready" button on their screens and, when both players were ready, a timer counted down the start from 3 seconds. Each game round was 50 seconds and a countdown timer showed the time remaining at the top of the screen. The player with the most eliminations at the end of the round won.

For this study, one client PC was always the control player and the other the experimental player. The control player had different fixed latency values, while the experimental player had different time delay settings. When the setting for a particular round was adaptive time delay, the experimental player also had a different adaptation time applied to it.

- **No Latency.** No latency for both players. 1 round.
- **Unfair Conditions.** Experimental player had no latency while control player had a fixed 50 ms, 100 ms and 150 ms latency. 3 rounds.

- **Latency Conditions.** Both players had fixed latency, with both having 50 ms, 100 ms and 150 ms latency. 3 rounds.
- **Adaptive Time Delay (Smooth).** The condition with different latencies was replayed but with adaptive time delay for the experimental player. When adapting, latency went from low to high and high to lower over 1 s for the experimental player.
- **Extra Conditions.** 1 round of adaptive instant time delay and 1 round of adaptive smooth time delay over 2 s with the control player having 150 ms latency.

Each latency configuration was played twice during a session, meaning that users played a total of 24 experimental rounds along with the 1 practice round in a session.

After the round ended, players were asked two questions regarding their experience in that round. They were:

- *How much lag did you feel this round?* - 1 (Low) to 5 (High) via a slider.
- *Rate the fairness of the round.* - Between - 5 (Unfair) and 5 (Fair) via a slider.

Table 1: Summary of the user studies.

	Zombiefield	Last Stand	Color Clash
Players	One	Two	Two
Latency (ms)	0, 50, 100, 150	0, 100, 200	0, 50, 100, 150
Interaction criteria	Ray from chest	Ray from head	Proximity and sight
Adapt. strategies	Instant, smooth, on/off	Abrupt, smooth	Instant, smooth
Weapon fire type	Full auto	Semi auto	Semi auto
Fire rate (rounds/s)	11	3	4
Weapon actions	Shoot, aim, reload	Shoot, aim, reload, lean	Shoot, paint
Reload duration (s)	2.1	3.3	Not needed
Player movement	Run, sprint	Run, sprint	Run, jump
Speed (m/s)	6, 10	4, 8	3
Map size	120m × 120m	100m × 70m	60m × 60m
Map verticality	Uneven terrain	Uneven, mounds	Steps, raised

3.4 Summary

Table 1 summarizes the 3 user studies highlighting the key differences. The round times were chosen based on pilot studies considering the pace of the game, influenced by factors such as map size, fire rate, reload duration, and the type of gun used. The variability in interaction detection methods allows for a more complete evaluation of adaptive time delay across multiple gaming scenarios.

4 Analysis

This section first presents demographics for the Zombiefield, Last Stand and Color Clash user studies then analysis of the impact of adaptive time delay on responsiveness and fairness followed by a comparison between different adaptation strategies.

4.1 Demographics

Table 2 summarizes the demographics of all three studies in one table. Gaming skill is a self-rating from 1 (low) to 5 (high). Skill and Reaction Time show mean values, with standard deviation in parentheses. Generally, the population consisted of graduate and undergraduate students, mostly skilled first-person shooter players. For all studies, the participants' average reaction times were generally fast, typical of experienced gamers.

Table 2: Participant demographics summary.

Study	Users	FPS Gaming Skill (1-5)	Reaction Time (ms)
Zombiefield	35	2.9 (1.1)	195.6 (24.1)
Last Stand	23	2.7 (1.4)	204.3 (39.7)
Color Clash	38	3.5 (0.9)	187.0 (18.5)

Thirty-five (35) participants completed the Zombiefield study, with full participation and data for each round. The average age was 20 (M=20.2, SD=3.8) years, ranging from 17 to 37 years and 33 of them were male and 2 female. General and FPS gaming skills were self-reported (1 low to 5 high); most were experienced gamers and many were experienced in FPS games.

Twenty-three (23) participants completed the Last Stand study. They took part in 14 sessions in total: 9 sessions with two users playing against each other, and 5 sessions where a single user played against the study proctor. The average age of the participants was approximately 24 years (M = 24.6, SD = 4.9), with a range of 19 to 38 years and 16 of them were male, 6 were female with one non-binary. General and FPS gaming skills were self-reported (1 low to 5 high) - players were experienced gamers with many being experienced in FPS.

Thirty-eight (38) participants completed the Color Clash study. Experience with FPS games and the impact of lag on gameplay were self-reported answers (1 low to 5 high). Users reported how much lag effected their gameplay (1 low to 5 high), with an average rating of 3.5 (SD=0.9).

4.2 Responsiveness

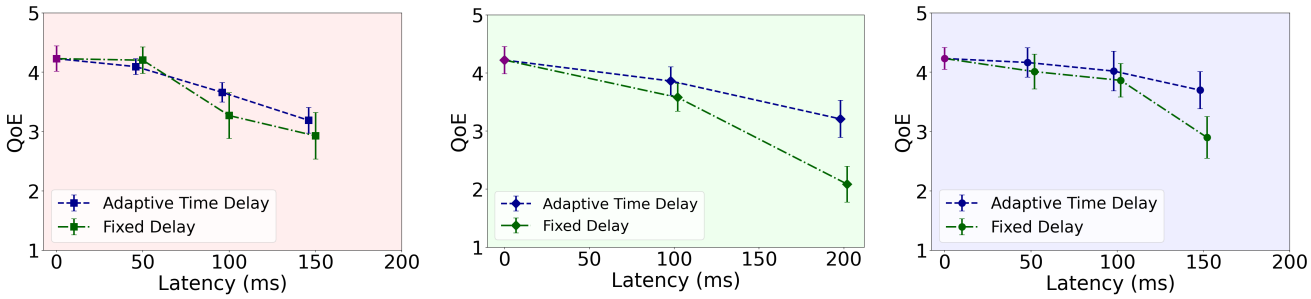
To evaluate responsiveness, users rated their gaming experience, which we call Quality of Experience (QoE) at the end of each round.

Figure 10 shows shows graphs analyzing QoE versus latency for each user study. The X axis shows the latency, and the Y axis shows the reported QoE where a higher score means the player had a better experience. The graphs, one per study, shows mean QoE value with a 95% confidence interval. The green line represents fixed time delay and the blue represents adaptive time delay.

Zombiefield. Figure 10a shows Zombiefield QoE across latencies of 0 ms, 50 ms, 100 ms, and 150 ms for both fixed and adaptive time delays. From the graph, adaptive time delay provides better average QoE for latencies above 50 ms, but the differences in means are not statistically significant.

Last Stand. Figure 10b shows QoE for no latency, fixed latency and adaptive time delay. Adaptive time delay provides better average QoE than fixed time delay, particularly for 200 ms of latency where the mean QoE with adaptive time delay is 50% greater than the mean QoE with fixed time delay.

Color Clash. In Color Clash, QoE is provided by a "perceived latency rating", which is then inverted to calculate the QoE. Figure 10c shows QoE for the Color Clash study. Similar to Last Stand, adaptive time delay provides better average QoE than fixed time delay, particularly for the highest latency – here, 150 ms – where the



(a) Zombiefield user study. (b) Last Stand user study. (c) Color Clash user study.

Figure 10: QoE comparison between adaptive time delay and fixed delay.

mean QoE with adaptive time delay is 25% greater than the mean QoE with fixed time delay.

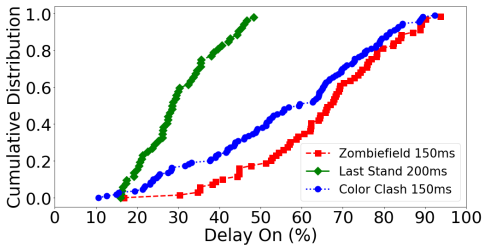


Figure 11: Cumulative distributions of delay on (%) for the highest latencies.

Summary. In the Zombiefield study, at a latency of 150 ms, the QoE results show a smaller difference between adaptive time delay and fixed delay compared to the other two studies. This may be due to the adaptive time delay being active for a longer, leading to a player experience more similar to that with fixed time delay. To explore this, Figure 11 shows the cumulative distributions of delays for each study for adaptive time delay rounds at the highest latencies, computed as a percentage of the players total round time. Zombiefield's distribution is shifted to the right of the other two studies, which means player had more time with higher delays than the other two studies, closer to that in fixed time delay.

4.3 Fairness

To evaluate fairness among players, score was used in Last Stand and in Color Clash. The following sub-sections present these results.

Figures 12 and 13 show mean score for Last Stand over all rounds with a 95% confidence interval. The X axis is latency and the Y axis is score. The left graph represents the "unfair" conditions, where the blue line represents users with no latency and the green line represents users with fixed delay (i.e., games with a no-delay player that had an unfair advantage over the other player). The latency for the fixed delay users is shown on the X axis. The right graph represents the "fair" conditions, with adaptive time delay in blue and fixed delay in green, for the same latencies (i.e., games where both users had the same latency either fixed, or when interacting with adaptive). Figure 13 shows the same data but for the Color Clash study.

In the graphs on the left ("unfair"), in the case where one user had no latency and the other had a fixed 200 ms of latency, there is a statistically significant difference in scores. The user with higher latency scored significantly lower, indicating that the game was less fair as the higher-latency users were disadvantaged.

In the graphs on the right ("fair"), scores decrease with latency, but the score differences between adaptive time delay and fixed time delay are similar for all latencies. This indicates that adaptive time delay preserves fairness comparable to fixed time delay.

4.4 Adaptation Strategy

Each study had slightly different algorithms for how rapidly latency was added/removed from low-latency players when they interacted with high-latency players. The following results show comparisons between these adaptation strategies.

Figures 14 shows the average QoE for the highest latency values with different adaptive time delay algorithms on the X axis. All points are means with 95% confidence intervals. From the graphs, adaptive time delay using a smooth algorithm provides the best QoE over all other algorithm choices tested. Moreover, instant adaptive delay – where the latency immediately jumps to the highest or lowest value – provides for a worse QoE experience on average than does fixed delay. This suggests that potential gains to average QoE from using adaptive time delay can be undone if abrupt latency transitions are used.

4.5 Summary

As a summary, this section provides analysis for responsiveness and fairness with results from all three studies on one graph.

Responsiveness. Figure 15 show result of QoE difference between adaptive time delay and fixed delay for all 3 studies, with mean and 95% confidence interval. Values above 0 indicate adaptive time delay provides better QoE. For 100 ms or higher latency, all studies show adaptive time delay provides better QoE than fixed delay.

Fairness. Figure 16 show score difference between no delay and fixed delay on left, and adaptive time delay and fixed delay on the right. Values near 0 indicate a more fair experience. Here, according to the graphs, adaptive time delay provides a fairer experience

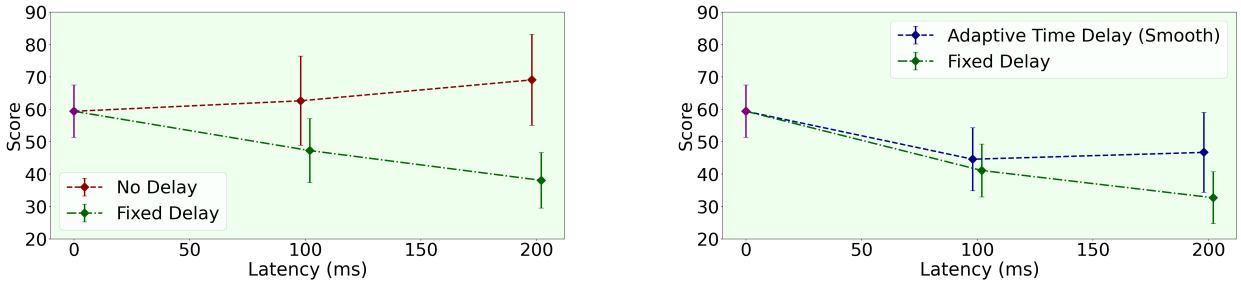


Figure 12: Score for no delay versus fixed delay (left) and adaptive time delay versus fixed delay (right) for Last Stand.

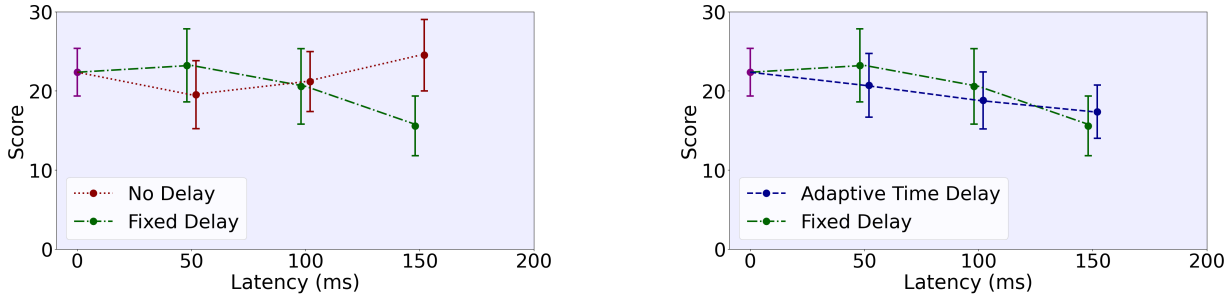
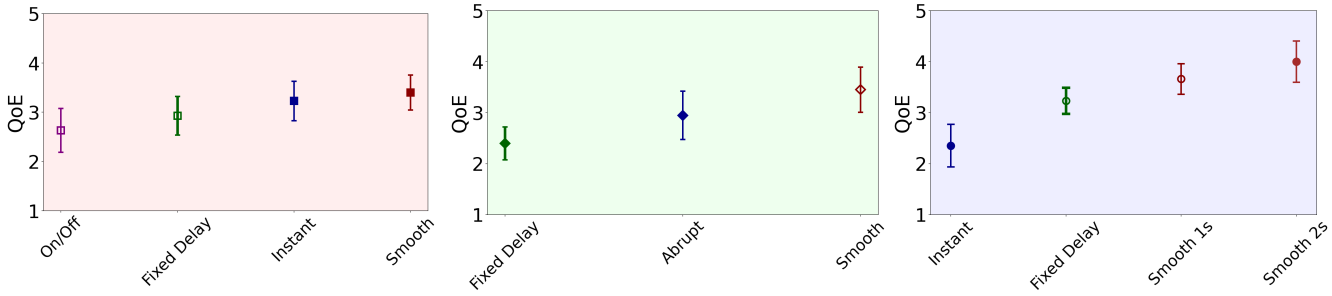


Figure 13: Score for no delay versus fixed delay (left) and adaptive time delay versus fixed delay (right) for Color Clash.



(a) Zombiefield at 150 ms. **(b) Last Stand at 200 ms.** **(c) Color Clash at 150 ms.**

Figure 14: Adaptation algorithm for adding/removing delay, highest latency.

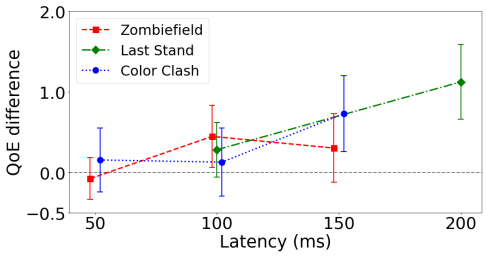


Figure 15: QoE difference between adaptive time delay and fixed delay for all studies.

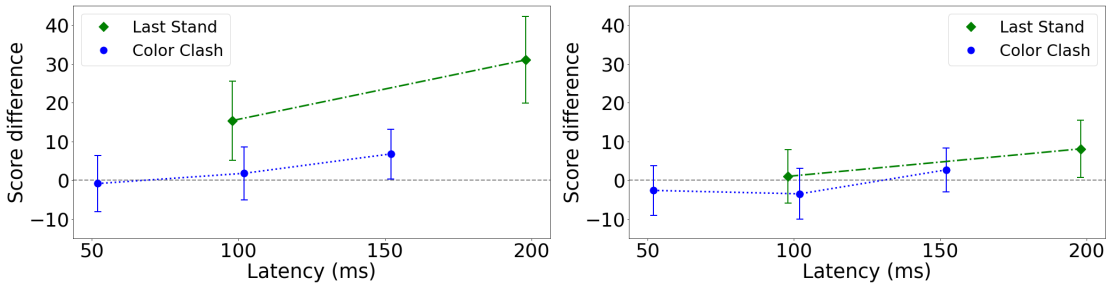
compared to having no delay, with performance fairness akin to that of fixed delay.

5 Discussion

Our studies show that for FPS games with players with different latencies, adaptive time delay can improve fairness compared to no added delay and improves responsiveness for low-latency players compared to fixed delay. This is because adaptive time delay only adds latency when players are interacting which provides for a more responsive experience for low-latency players when they are not in combat, yet by adding delay when players are interacting provides performance fairness.

However, the benefits to fairness and responsiveness are only pronounced at latencies above 100 ms. That is not to say that benefits do not exist for latencies below 100 ms, but the effort to implement adaptive time delay may not be worth it if all players have sub-100 ms of latency to the server.

When implementing adaptive time delay, care must be taken in the adaptation algorithm. The simple approach to immediately add



(a) No delay - fixed delay

(b) Adaptive time delay - fixed delay

Figure 16: Score difference.

latency to low-latency players when interacting with high latency players can actually result in a *worse* experience than simply having fixed time delay in the first place. Algorithms that apply latency adjustments smoothly over one or more seconds seem to provide the best experience, although care here too must be taken so as not to add latency too slowly, exposing unfairness. Future work could determine the "sweet spot" for latency adjustment algorithms.

6 Limitations

One limitation of our study is that we only tested our adaptive time delay technique with two player games. Many multiplayer FPS game modes have more than two players, including Battle Royale modes that have up to 100 players. Managing adaptive time delay in these scenarios is more complex, not only because of an increased number of interactions, but also because there may be "chain" effects from latency needing to be added to players because of a transitive interaction.

Another limitation is that all our fixed latency conditions were constant for that round. In practice, network latencies between clients and server can often vary over time with network congestion, last-mile link variance and server/client loads. Ideally, both fixed and adaptive time delay would adjust to variations in latency, probably by smoothing out any added latency over a short period.

In our Last Stand and Color Clash studies, some pairs of participants had different levels of skill. These skill gaps are confounding factors since they add apparent unfairness to the results when comparing objective performance, such as score. Future studies could try to account for skill differences but selecting matches based on skill level is a challenging process in practice and in user studies. Additionally, our participant pool skewed heavily young and male across all three studies, so our findings may not generalize across broader player populations.

In most FPS games, players have access to a variety of weapons, such as sniper rifles, rocket launchers with an area of effect, and melee weapons, allowing players to adjust their tactics based on latency, should they so choose. In contrast, our studies had only a single weapon – a rifle – with no option for the players to pick a weapon better suited to the network conditions.

Our map design and gunplay intentionally encouraged close-quarters combat to concentrate action as quickly as possible for the user study. This, in effect, creates a "worst case" scenario for adaptive time delay in that latencies need to adapt frequently. A larger map would reduce the number of player interactions, which would have allowed low-latency players to enjoy even better responsiveness.

7 Conclusion

Network latency significantly impacts player experience, degrading fairness and responsiveness in first-person shooter games. To address unfairness, time delay as a latency compensation technique adds a fixed amount of delay to players with lower latency, equalizing latency for all to that of the highest-latency player. This improves fairness but degrades the responsiveness of the low-latency player even when they are not interacting with others.

We propose adaptive time delay, which adds delay to the low-latency player only when they have a chance of interacting with another player. This has the potential to make the game more responsive for the low-latency player when they are not interacting with others while preserving the fairness of fixed time delay when they are.

We conducted a 35-person user study with a single-player game (Zombiefield) to evaluate adaptive time delay across different latencies and compare it with fixed time delay. Two further user studies, with 23 and 38 participants, used two-player games (Last Stand and Color Clash) to evaluate the impact of adaptive time delay on player experience and fairness. After each round, participants rated their experience and the game recorded player performance.

Analysis across all studies suggests adaptive time delay improves average player experience compared to fixed time delay. The two-player studies further show that adaptive time delay provides fairness comparable to fixed time delay, with both offering improved fairness over no added delay. Among the adaptation algorithms evaluated – smooth, abrupt, instant, and on/off – smooth performs slightly better than the others, while instant can perform worse than fixed time delay.

Future work can refine the smooth adaptation strategy by testing different time ranges and on/off heuristics. Interaction detection may be improved by using multiple rays and adapting the approach for larger game modes (e.g., Battle Royale). Other directions include exploring adaptive time delay in different game genres and combining it with other latency compensation techniques to further improve fairness and experience.

References

[1] Rahul Amin, France Jackson, Juan E. Gilbert, Jim Martin, and Terry Shaw. 2013. Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2. In *Proceedings of HCI – Users and Contexts of Use*. Las Vegas, NV, USA, 97–106.

[2] Grenville Armitage. 2003. An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3. In *Proceedings of the 11th IEEE International Conference on Networks (ICON)*. Sydney, Australia.

[3] Karthik Balasubramanian. 2022. The Rise of Online Multiplayer. Online: https://www.gameopedia.com/online-multiplayer-games. (Accessed 15th March 2023).

[4] Jeremy Brun, Farzad Safaei, and Paul Boustead. 2006. Managing Latency and Fairness in Networked Games. *Commun. ACM* 49, 11 (nov 2006), 46–51. https://doi.org/10.1145/1167838.1167861

[5] Matthias Dick, Oliver Wellnitz, and Lars Wolf. 2005. Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. In *Proceedings of the 4th ACM Network and System Support for Games (NetGames)*. Hawthorne, NY, USA.

[6] Marko Dimitrievski. 2023. Gaming statistics – 2023. Online: https://truelist.co/blog/gaming-statistics/. (Accessed 26h April 2024).

[7] Ibirisol Fontes Ferreira, Maycon Leone Maciel Peixoto, and Gustavo Bittencourt Figueiredo. 2024. Fairness-oriented Multicast Routing for Distributed Interactive Applications. *Computer Communications* 219 (2024), 229–242. https://doi.org/10.1016/j.comcom.2024.03.015

[8] Tobias Fritsch, Hartmut Ritter, and Jochen H. Schiller. 2005. The Effect of Latency and Network Limitations on MMORPGs: a Field Study of Everquest 2. In *Proceedings of the 4th ACM Network and System Support for Games (NetGames)*. Hawthorne, NY, USA.

[9] David Halbhuber, Niels Henze, and Valentin Schwind. 2021. Increasing Player Performance and Game Experience in High Latency Systems. *Proceedings of the ACM on Human-Computer Interaction* 5, CHI PLAY (2021), 1–20. https://doi.org/10.1145/3474710

[10] O. Hohlfeld, H. Fiedler, E. Pujol, and D. Guse. 2016. Insensitivity to Network Delay: Minecraft Gaming Experience of Casual Gamers. In *Proceedings of the International Teletraffic Congress (ITC)*. Wurzburg, Germany.

[11] David R. Jefferson. 1985. Virtual Time. *ACM Transactions on Programming Language and Systems* 7, 3 (July 1985), 404–425. https://doi.org/10.1145/3916.3988

[12] Xinbo Jiang, Farzad Safaei, and Paul Boustead. 2005. Latency and Scalability: a Survey of Issues and Techniques for Supporting Networked Games. In *Proceedings of the 13th IEEE International Conference on Networks (ICN)*. Kuala Lumpur, Malaysia.

[13] Arnaud Kaiser, Nadjib Achir, and Khaled Boussetta. 2010. Improving Energy Efficiency and Gameplay Fairness for Time-Sensitive Multiplayers Games in MANET. In *2010 IEEE International Conference on Communications Workshops*. Cape Town, South Africa, 1–5. https://doi.org/10.1109/ICCW.2010.5503904

[14] Steven Lee and Rocky Chang. 2017. On 'Shot Around a Corner' in First-person Shooter Games. In *Proceedings of the 15th Annual Workshop on Network and Systems Support for Games (NetGames)*. IEEE, Taipei, Taiwan, 1–6.

[15] Steven W. K. Lee and Rocky K. C. Chang. 2018. Enhancing the Experience of Multiplayer Shooter Games via Advanced Lag Compensation. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*. Amsterdam, Netherlands, 284–293.

[16] Wai-Kiu Lee and Rocky Chang. 2015. Evaluation of Lag-Related Configurations in First-Person Shooter Games. In *Proceedings of the Workshop on Network and Systems Support for Games (NetGames)*. IEEE Press, Zagreb, Croatia.

[17] Shengmei Liu, Atsuo Kuwahara, James Scovell, Jamie Sherman, and Mark Claypool. 2021. Comparing the Effects of Network Latency versus Local Latency on Competitive First Person Shooter Game Players. In *Proceedings of the ACM Esports and High Performance HCI Workshop (EHPHCI)*. Virtual Conference.

[18] Shengmei Liu, Atsuo Kuwahara, Jamie Sherman, James Scovell, and Mark Claypool. 2021. Lower is Better? The Effects of Local Latencies on Competitive First Person Shooter Game Players. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*. Virtual Conference.

[19] Shengmei Liu, Xiaokun Xu, and Mark Claypool. 2022. A Survey and Taxonomy of Latency Compensation Techniques for Network Computer Games. *ACM Comput. Surv.* 54, 11s, Article 243 (sep 2022), 34 pages. https://doi.org/10.1145/3519023

[20] Martin Mauve. 2000. Consistency in Replicated Continuous Interactive Media. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*. Philadelphia, PA, USA.

[21] Martin Mauve. 2000. Consistency in Replicated Continuous Interactive Media. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (Philadelphia, Pennsylvania, USA) *(CSCW '00)*. Association for Computing Machinery, New York, NY, USA, 181–190. https://doi.org/10.1145/358916.358989

[22] Martin Mauve. 2000. How to Keep a Dead Man from Shooting. In *Proceedings of Interactive Distributed Multimedia Systems and Telecommunication Services Workshop (IDMS)*. Enschede, Netherlands.

[23] Martin Mauve, Jürgen Vogel, Volker Hilt, and Wolfgang Effelsberg. 2004. Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications. *IEEE Transactions on Multimedia* 6, 1 (2004), 47–57.

[24] Doowon Paik, Chung-Ha Yun, and Jooyeon Hwang. 2008. Effective Message Synchronisation Methods for Multiplayer Online Games with Maps. *Computers in Human Behavior* 24, 6 (2008), 2477–2485. https://doi.org/10.1016/j.chb.2008.03.004 Including the Special Issue: Electronic Games and Personalized eLearning Processes.

[25] Lothar Pantel and Lars C. Wolf. 2002. On the Impact of Delay on Real-Time Multiplayer Games. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. Miami, FL, USA.

[26] Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer, and Natalie Degrande. 2004. Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game. In *Proceedings of ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*. New York, NY, USA.

[27] Cheryl Savery, T. C. Nicholas Graham, and Carl Gutwin. 2010. The Human Factors of Consistency Maintenance in Multiplayer Computer Games. In *Proceedings of the 16th ACM international conference on Supporting group work - GROUP '10* (Sanibel Island, Florida, USA) *(GROUP '10)*. Association for Computing Machinery, New York, NY, USA, 187–196. https://doi.org/10.1145/1880071.1880103

[28] Cheryl Savery, T. C. Nicholas Graham, and Carl Gutwin. 2010. The Human Factors of Consistency Maintenance in Multiplayer Computer Games. In *Proceedings of the 16th ACM International Conference on Supporting Group Work*. Association for Computing Machinery, Sanibel Island, FL, USA, 187–196.

[29] Andreas Schmid, David Halbhuber, Thomas Fischer, Raphael Wimmer, and Niels Henze. 2023. Small Latency Variations Do Not Affect Player Performance in First-Person Shooters. *Proceedings of the ACM on Human-Computer Interaction* 7 (2023), 197–216. https://doi.org/10.1145/3611027

[30] Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. 2003. The Effect of Latency on User Performance in Warcraft III. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames)*. Redwood City, CA, USA.

[31] Jiawei Sun and Mark Claypool. 2019. Evaluating Streaming and Latency Compensation in a Cloud-based Game. In *Proceedings of the 15th IARIA Advanced International Conference on Telecommunications (AICT)*. Nice, France.

[32] Samin Shahriar Tokey, James Cannon, Saketh Dinasarapu, Ao Jiang, Hanzalah Qamar, and Mark Claypool. 2024. Effects of Adaptive Time Delay on Quality of Experience in First Person Shooter Games. In *Proceedings of the 19th International Conference on the Foundations of Digital Games* (Worcester, MA, USA) *(FDG '24)*. Association for Computing Machinery, New York, NY, USA, Article 61, 4 pages. https://doi.org/10.1145/3649921.3656993

[33] Samin Shahriar Tokey, Zesheng Chen, Colin Mettler, Dexuan Tang, Ben Boudaoud, Joohwan Kim, Josef Spjut, and Mark Claypool. 2024. The Effects of Network Latency on the Peeker's Advantage in First-person Shooter Games. In *Proceedings of the 19th International Conference on the Foundations of Digital Games (FDG '24)*. Association for Computing Machinery, Worcester, MA, USA, Article 1, 10 pages. https://doi.org/10.1145/3649921.3650002

[34] Samin Shahriar Tokey and Mark Claypool. 2024. Last Stand - A First Person Shooter Game for User Studies on the Effects of Network Delay on Players. In *Proceedings of the ACM International Conference on the Foundations of Digital Games (FDG)* (Worcester, Massachusetts, USA). ACM, New York, NY, USA.

[35] Xiaokun Xu and Mark Claypool. 2025. Reflex – An Open-source Tool for Measuring Human Reaction Times. In *Proceedings of the 17th International Conference on Quality of Multimedia Experience (QoMEX)*. Madrid, Spain.

[36] Sebastian Zander, Ian Leeder, and Grenville Armitage. 2005. Achieving Fairness in Multiplayer Network Games Through Automated Latency Balancing. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology* (Valencia, Spain) *(ACE '05)*. Association for Computing Machinery, New York, NY, USA, 117–124. https://doi.org/10.1145/1178477.1178493