Improving Player Experience and Fairness Using adaptive Time-Delay on First-Person Shooters

Master's Thesis

By
Samin Shahriar Tokey

Advisor
Mark Claypool

Reader
Charles Roberts

Interactive Media and Game Development
Worcester Polytechnic Institute
100 Institute Road
Worcester MA 01609

October 2024

Abstract

Multi-player gamers often have different latencies to the server, even within the same game. For competitive games, this can result in unfairness. Time delay is a latency compensation technique that can mitigate the unfairness by adding latency to some players so that everyone has the same latency. Although this makes the game fair when players interact, it degrades the responsiveness for the player with the lower latency even when the players are not interacting. We propose adaptive time delay that only adds latency for interacting players. We implemented adaptive time delay in two different games and compare time delay to adaptive time delay on both games with different amounts of latency. Analysis of the results suggest adaptive time delay improves overall quality of experience compared to fixed time delay while preserving time delay's fairness.

Table of Contents

Abstract 2
1. Introduction 5
2. Background 6
3. Methodology 11
3.1 Zombiefield – Single Player Study 13
3.1.1 Overview 13
3.1.2 Gameplay 13
3.2.3 Latency Simulation 14
3.1.4 Simulating Fixed Time Delay Conditions 15
3.1.5 Adaptive Time Delay 15
3.1.6 Adaptive Time Delay Activation System 15
3.1.7 Configuration Files 16
3.1.8 Logging System 16
3.1.9 Experiment Design 17
3.1.10 User Study 18
3.1.11 Hardware Used 19
3.2: Last Stand – Two Player Study 20
3.2.1 Overview 20
3.2.2 Gameplay 20
3.2.3: Networking Implementation 21
3.2.4: Implementing Time Delay 22
3.2.5 Log System 24
3.2.6 User Study 24
3.2.15 Hardware Used 25
4. Analysis 26
4.1 Demographics 26
4.1.1 Zombiefield Single-Player User Study 26
4.1.2 Last Stand Two-Player User Study 26
4.2 Responsiveness 27
4.2.1 Quality of Experience – Zombiefield 27
4.2.2 Quality of Experience – Last Stand 28
4.3 Fairness 28
4.3.1 Accuracy 28

4.3.2 Score 29
4.4 Comparison of Adaptation Strategies 29
4.4.1 Zombiefield 30
4.4.2 Last Stand 30
4.5 Comparison with Previous Work 31
4.6 Summary 32
4.7 Limitations 32
5. Conclusion 33
6. Future Work 33
6.1 Short Term 33
6.2 Medium Term 34
6.3 Long Term 34
7. References 35
8. Appendices 38
8.1 Demographic Survey 38
8.2 IRB Approval Letter 39

Improving Player Experience and Fairness Using Adaptive Time-Delay on First-Person Shooters

1. Introduction

The video gaming industry has seen significant growth in recent years, with the number of video gamers increasing from 2.03 billion in 2015 to 3.32 billion in 2024 [1]. The online gaming market is also thriving, with an estimated worldwide market size of $56 billion USD in 2021. Predictions suggest that this market will continue to grow, experiencing a 10.2% annual growth rate and reaching a value of $132 billion USD by the year 2030 [2]. Among the most popular genres in the online gaming industry is the first-person shooter which is enjoyed by both casual and competitive gamers. [25]

First-person shooter games that 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 packets and actions of all players, allowing them to interact and compete in real-time.

Latency, is the time it takes for a packet of data to travel from a player's client to the server and back. Physical distance impacts latency, as the further the distance, the longer it takes for a packet to travel. As a result, players who are further away from the server may experience higher latencies compared to players who are closer. However, latency is also effected by factors such as network capacity, congestion, and routing efficiency, which can further contribute to delays. High latency, means delayed receipt of action by the server and responses by the clients, making it difficult for the player to react and make split-second decisions in fast-paced games. Latency can also cause desynchronization between the player's actions and the game's state, leading to inconsistencies and inaccuracies in the game world. In competitive first-person shooter games, high latency can thus put the player at a significant disadvantage, as they may not be able to respond as quickly or accurately as players with lower latencies, negatively impacting their overall performance and leading to a less enjoyable gaming experience.

Time delay is a technique used in multiplayer games to improve fairness in gameplay by equalizing the latency between players. The basic idea behind this technique is to add latency to the player who has lower latency, bringing their latency closer to the player with higher latency. By doing so, the two players' latencies are the same, mitigating the unfair advantage that the player with lower latency may have had due to their faster responsiveness. However, time delay has the disadvantage that, increasing the latency of a player with low latency degrade their gaming experience, making the game less enjoyable.

We propose adaptive time delay as is a potential improvement over traditional time delay techniques in multiplayer games. Instead of adding latency to the low latency player continuously, adaptive time delay activates time delay only

during player interactions. This means that when players are not interacting with each other, time delay is turned off, allowing the low latency player to enjoy the original low latency that they had before the time delay was introduced. By only adding time delay during player interactions, adaptive time delay can effectively improve the responsiveness of the low latency player, giving them a more responsive gaming experience while still maintaining fairness in gameplay.

The aim of this research is to evaluate adaptive time delay. Adaptive time delay has the potential of maintaining the fairness offered by time delay and it can improve responsiveness by turning off time delay while players are not interacting with each other. However, this adaptation may feel unnatural to the players, affecting gaming experience. Thus, we have two primary research areas to explore. First, to evaluate how adaptive time delay affects player experience and second how adaptive time delay affects fairness. This was achieved by doing two separate case studies. We built a single player zombie shooter game called Zombiefield, where we evaluated improvement in player quality of experience (QoE). We then built a multiplayer first-person shooter game called Last Stand, where we evaluated both fairness and QoE. Adaptive time delay was introduced by shooting rays from the players' current position, and activating time delay if the ray is not interrupted by any obstacle. Delay was introduced either abruptly or gradually for both games. There were options in the game designed to support the experimental design, including user testing with compensation turned on and off, adaptive time delay on and off, and for different latencies.

The Zombiefield user study had 38 users and Last Stand user study had 23 users. Most of the players were male gamers with many of them being experienced FPS players. Players experienced no latency, fixed latency, and our adaptive time delay, each with varying amounts of simulated network latencies. Analysis of the results show promising results where adaptive time delay improved player QoE over fixed time delay in both studies when latency was high and improved fairness in the Last Stand study over no time delay. Our results also suggest that adding delay smoothly has the potential to improve QoE even further compared to adding delay abruptly.

This report is structured as follows: Chapter 2 offers an overview of latency, its impact on games and works related to time delay. Chapter 3 details how we conducted both of our user studies. Next, chapter 4 presents the results and our interpretation of the data. Chapter 5 encapsulates our key discoveries and insights. Finally, chapter 6 talks about possible future work regarding adaptive time delay.

2. Background

Latency in computer games is the delay between the players action and the result of that action in game. This delay can be local or network latency.

Local latency is the delay between when a player takes an action and when the game responds on the player's local device, such as a PC, console, or mobile device. This delay is caused by the processing time of the player's local device, including the time it takes for the CPU and GPU to render the game graphics, as well as the time it takes for the local

input device, such as a keyboard or controller, to transmit the player's input to the device. It can also be affected by display response time. For high-performance gaming PCs the system can have local latency as low as 25ms [3] and for TV and console combo it can have a local latency above 200ms [4] [5].

Network latency on the other hand is the round-trip time between the client and the game server. Round-trip time (RTT) is the time it takes for a packet of data to travel from a client to a server and back again. RTT can be effected by various factors such as distance between the source and the destination, network congestion, bandwidth. Figure 1 shows 4 players on a client-server network model with different latencies due to their geographic location. Players further away have much higher RTT (depicted in ms) rather than the players that are geographically closer to the server. This is because the physical distance that data must travel between the player and the server is longer, resulting in a longer RTT. As a result, players who are further away may experience more delay or lag in their gameplay, while players who are closer may have a more seamless and responsive experience.

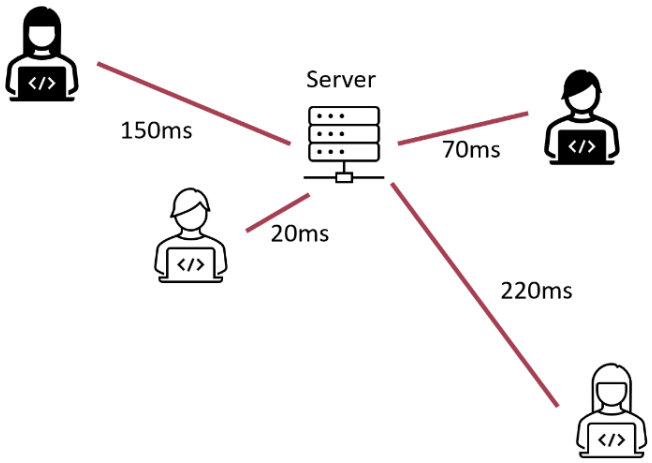


Figure 1: Latency due to geographic location of players.

For networked games, this network latency is added on top of local delay. This can affect the game's responsiveness, and it can also make the game world inconsistent [6].

Increasing latency leads to reduced responsiveness, as shown in Figure 2. When a player takes a shot in an online game, the input (shot information) is sent to the server for processing. The server verifies the shot and sends a confirmation message back to the player, registering the shot in the game world. If the player has high round-trip time (RTT), the delay in sending and receiving these messages can cause a delay between the player's input and the game's

response. This results in a delayed shot registration, making the game feel unresponsive and negatively impacting the player's experience.

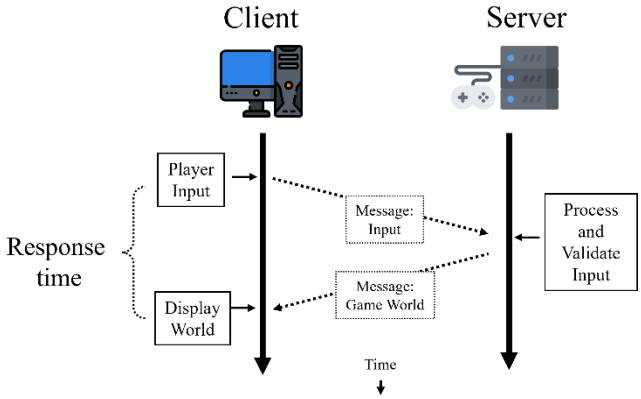


Figure 2: Response time with client-server latency.

Increased latency can cause inconsistencies in the game world [6], as illustrated in Figure 3. In this scenario, the server controls the game world, including the movements of an NPC, and sends updates to clients to keep them synchronized. However, if client 1 has a latency of 200ms and client 2 has a latency of only 20ms, client 1 will receive the updates much later than client 2. As a result, client 1 will see the NPC in an older position compared to client 2, who will see the NPC in a more recent position. This inconsistency in the game world can cause confusion and detract from the user's experience.

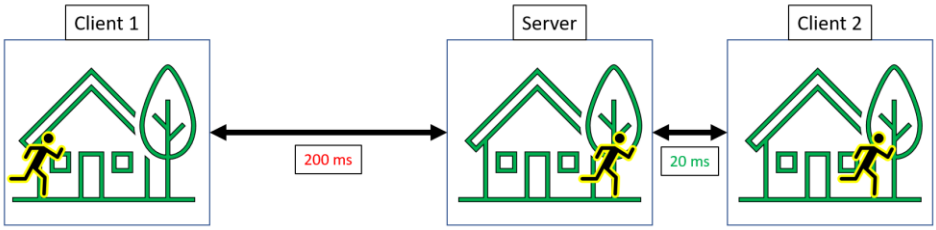


Figure 3: Inconsistency in game world due to latency.

To mitigate these issues, various latency compensation techniques are introduced in games. These can mainly be divided into 4 classes: feedback, prediction, world adjustment and time manipulation [6].

Feedback offers the player auditory or visual cues about latency, without altering the actual state of the game world. An instance of this approach is displaying visual indicators immediately when a player inputs a command, instead of waiting for server validation. Another example could be visually displaying the round-trip time at the corner of the screen.

Prediction involves approximating the game's state at a client without relying on the official game state provided by the server. Interpolation is a form of prediction method that can enhance player accuracy by 2% [7], as it estimates the current state of the game by interpolating between past states, such as position, animation, and rotation. Another form of prediction known as Extrapolation predicts future states for objects that are under the control of other players, under the

assumption that their current behavior will persist. Several studies have shown that using extrapolation can mitigate the inconsistencies that arise between the game states of the client and the server [8] [9] [10] [11] [12].

World adjustment alters the game state to reduce the level of difficulty, similar to configurations with lower latency. Control assist is an example of world adjustment technique where outcome of the player inputs are adjusted to combat inaccuracies due to latency. Attribute scaling is also a world adjustment technique where the numeric attributes of game objects and world parameters are adjusted to modify game difficulty, thereby making it easier for players to complete actions even with higher latency. Studies have shown that this technique can enhance both the quality of experience by 20% [15] and player performance by 50% [14].

Time manipulation indicates the techniques that alter the game time to calculate game state and/or process player actions. Time warp or rollback is a time manipulation technique that involves rolling back the game state to incorporate a player's action, and then advancing the game forward after applying the action. In an FPS game, this technique can enhance a player's aiming accuracy by 5% for up to 150 ms of latency and overcome network latency of up to 100 ms [7] [13].

Time delay is also a time manipulation technique. Network delay is added to the player having lower latency to match the player with higher latency. There are two methods for implementing time delay: incoming delay and outgoing delay. Incoming delay involves adding a delay to incoming packets before they are applied to the game, while outgoing delay adds delay to outgoing packets before they are sent out. This happens in both server and client side.

Paik et al. implemented incoming delay or waiting period to their server based and adjusted it based on the player count and their proximity. They derived an area of interest where participants might interact with each other. In their research they proposed two methods of adjusting waiting periods. One method is based on probability of interaction, where one participant waiting time is proportional to the weighted average delay for all the participants that are in the area of interest. The second method is a rank-based waiting period decision method where the waiting period of a participant is set by taking the highest latency of the participants currently present in a certain portion of the area of interest. To evaluate they build an NVE and performed simulations with a map size of 100x100 and 2500 participants. They found that instead of taking the maximum latency of the participants, their methods seemed to strike a balance between good responsiveness and fairness. They also suggested that combining both of their methods might yield a better result [19].

Zander et al. used a Self-Adjusting Game Lagging Utility that added outgoing delay to a server. They conducted their experiments using bots instead of human players, which allowed for a more controlled and precise testing environment where the bots faced similar unfairness as human players. The authors evaluated their experiment using pre-made games, and instead of using time delay as a latency compensation technique, they used it to assess the behavior of bots in their experiments [16].

Kaiser et al. utilized a technique for implementing time-delay in an online game, which involved combining game update messages into a larger packet before transmission. This approach was tested in a first-person shooter game called Quake 3 arena. The study found that their implementation of time-delay led to an increase in fairness among players [17].

Brun et al. implemented server-side incoming delay over distributed servers and synchronizes between the server by adding the delay. They proposed a heuristic to select the servers which starts from an initial two-server solution optimized for the two most distant players (from a network point of view). It is then expanded at each iteration to minimize the response time of the worst-off player by adding a new server. The process ends when no more improvement is possible. The heuristic method described can enhance fairness while only slightly increasing the response time for players, in contrast to the central server approach which can further improve fairness but at the expense of degraded response time, or alternatively, it may improve response time but at the cost of fairness [20].

Savery et al. tested incoming delay in both server-side and client side. They found that, when low state divergence and minimal corrections are crucial, and a slight delay in response time can be tolerated, using the local lag approach is a suitable option. This method is particularly well-suited for real-time strategy games where game entities' target positions are designated using "click to move" input, making it an ideal match [21].

Time delay has been known to help reduce inconsistencies in teleoperation applications as well. Mauve et al. conducted a study using a 3D teleoperation application called TeCo3D, in which they introduced a local lag of 250ms to address inconsistencies. Prior to the implementation of local lag or incoming delay, the TeCo3D application experienced artifacts and animation freezes lasting between 100-200ms. The study found that the introduction of local lag helped eliminate short-term inconsistencies, as long as all network packets successfully reached their destination [18].

The utilization of time-delay as a latency compensation technique is widely recognized as an effective means of reducing inconsistency of states and enhancing fairness in games. However, a significant drawback of this approach is its negative impact on response time, which can degrade quality of experience. Our proposed approach seeks to address both fairness and response time concerns by introducing an enhanced version of time-delay that activates during player interactions. While Paik et al have previously investigated a similar approach based on player proximity [19], our approach considers not only proximity, but also the possibility of interaction, which we determine by detecting the presence of obstacles between players. Additionally, we propose a gradual introduction of time-delay as well as an immediate activation, which may further enhance the overall quality of experience. Unlike many other studies that have evaluated their methodology using pre-existing games or third-party software and computer simulations, we developed our own game, incorporate time-delay directly into the game mechanics and conduct tests with human subjects. While developing our own game gave us greater control over the testing environment, user study with human subjects can provide insights into how actual players interact with the game and how adaptive time-delay approach affects their gaming experience.

3. Methodology

This chapter presents the methodology used to conduct our study, focusing on 3 primary sections: hypothesis, single player user study and multi-player user study.

Hypothesis 1:

There will be improvement in fairness with time delay on compared to time delay off than off for players with a large difference in latency.

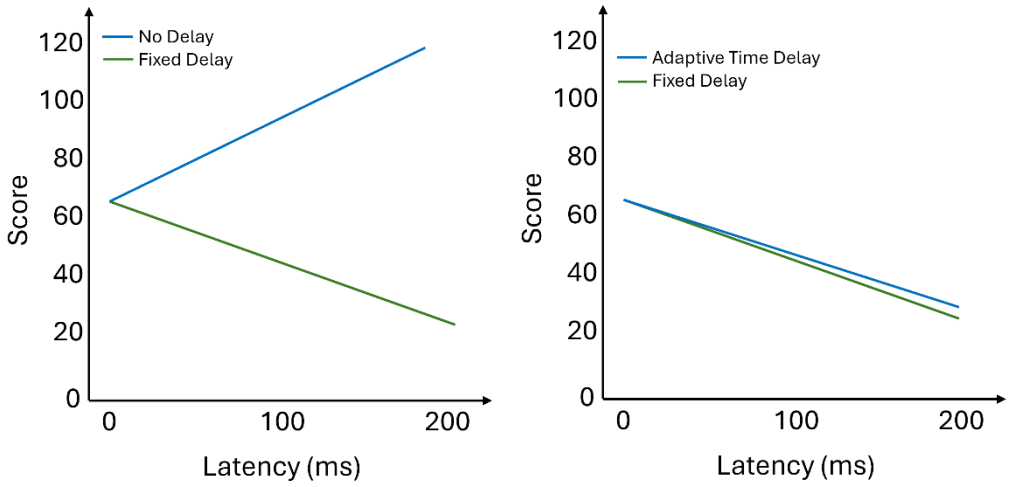


Figure 4: Performance difference between players with no delay versus fixed delay and adaptive time delay versus fixed delay.

Figure 4 depicts the hypothesis that if adaptive time delay is not used (left), players with high latency differences may experience a significant decrease in fairness shown as difference in score. However, the figure on the right suggests that scores will remain fair as difference goes down for all players if adaptive time delay is used, regardless of their latency difference.

Hypothesis 2:

For players having a large latency difference, adaptive time delay will result in better player experience for the player with lower latency compared to regular time delay.

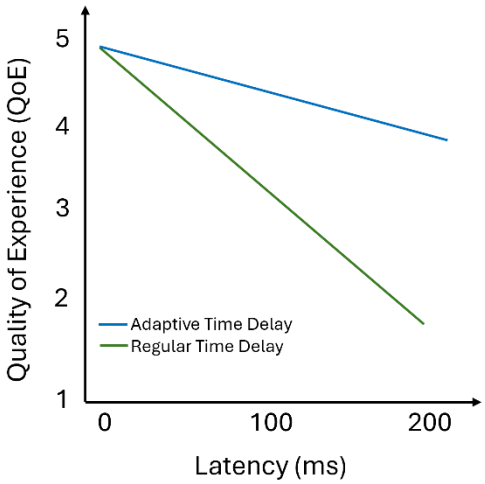


Figure 5: Improvement of player experience due to adaptive time delay.

Figure 5 depicts the hypothesis that with regular time delay, the player who had lower latency, will have higher latency all the time. This decrease in latency can be improved by using adaptive time delay, which will increase the average responsiveness. Increased responsiveness will also result in a better quality of experience.

Hypothesis 3:

Avoiding frequent and abrupt transitions in latency for adaptive time delay will result in a better player experience.

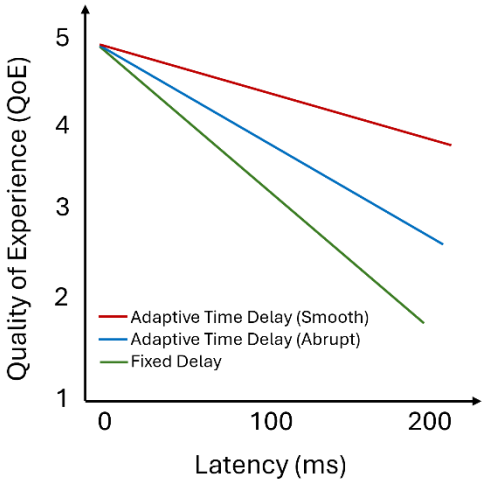


Figure 6: QoE for adaptive smooth, adaptive abrupt and fixed delay.

Going from low latency to high can be done smoothly or abruptly. Figure 6 depicts the hypothesis that smooth adaptive time delay will provide improved QoE over abrupt adaptive time delay.

Adaptive time delay activates while the players are interacting with each other. When activated, it will increase the latency of the low latency player to match the high latency player. Avoiding activating this feature too frequently and increasing the latency gradually will improve the overall player experience.

To test our hypothesis, we conducted two separate user studies. In the first study we can evaluate the impact on user experience. As opposed to regular time delay, adaptive time delay change player latency frequently. Frequent change in latency may be detrimental to player experience. Therefore, we first evaluate the impact of adaptive time delay on player experience. This study was a single player user study, where players were exposed to no delay, fixed delay and adaptive time delay in a shooter game. Their reported QoE was recorded and logged by the game.

Another second user study was conducted to study the fairness aspect of adaptive time delay alongside QoE. This was a 1v1 multiplayer user study where both users faced combinations of no delay, fixed delay and adaptive time delay in a shooter game. Their performance and reported QoE was logged.

3.1 Zombiefield – Single Player Study

3.1.1 Overview

Zombiefield is a single player first person shooter developed with Unreal Engine 5 to study adaptive time delay. Our goal with Zombiefield was to test hypothesis 2 and 3. Zombiefield is a single player game, where player had to survive waves of zombies. There were two types of zombies used in Zombiefield, networked – which simulated adaptive time delay on interaction for the player, and local zombie.

3.1.2 Gameplay

Zombiefield has weak, strong and ultimate zombies. The weak is a standard (local) zombie and the other two are networked. All 3 have different speed and health. The Player has to avoid, survive and eliminate as many zombies as possible. The player has 100 health and zombies quickly drain their health if it gets close. When the health reaches zero, player respawns to its original location and the zombies reset as well.

Zombie spawning occurs in cycles, each lasting between 5 to 10 seconds, during which zombies are randomly spawned based on specific probabilities: there's a 50% chance of spawning three weak (non-networked) zombies, which are the slowest and have 2 HP each, or one strong (networked) zombie with 5 HP, along with a 10% chance for the spawn of an ultimate (networked) zombie, the fastest, boasting 15 HP. Each bullet inflicts 1 damage on any type of zombie.

A video of the game can be found here: https://youtu.be/aBNj0L3aLV8?si=LSywpfEts0R-Hhy3

Zombiefield uses a custom-built control rig to control the inverse kinematics (IK) of the player. This makes the weapon animations (sway, recoil, aiming) procedural. The gun used in Zombiefield is a full auto rifle with a laser pointer attached. The bullets are projectile tracers, making it easier to see where the bullets are going.

Player’s score increases by 1 point as the player avoids and survives the zombies. When player kills any zombie, 10 points are awarded. However, if the player dies, the score resets back to 0.

Player was able to jog, sprint and strafe to avoid the Zombies. Besides shooting player was able to aim down sight and tactically reload.

The game has text based UI for score, high-score, duration, health, rounds, ammo, kills and deaths. These are shown in Figure 7.

Figure 7: Zombiefield in game screenshot.

3.2.3 Latency Simulation

We added functionality to simulate latency in game by introducing input delay to all player actions. Figure 8 shows the latency simulation process. All of the player inputs are intercepted and put into input queues. The queues had two values – the actual value for that input and the time to execute with added latency. Time to execute is calculated by getting the system time of when the input was pressed and adding desired delay to it. These input queues are checked every tick. Whenever the execution time is reached or surpassed, the input value is taken from the queue and applied to the game. Afterwards that input value and execution time is dequeued from the queue.

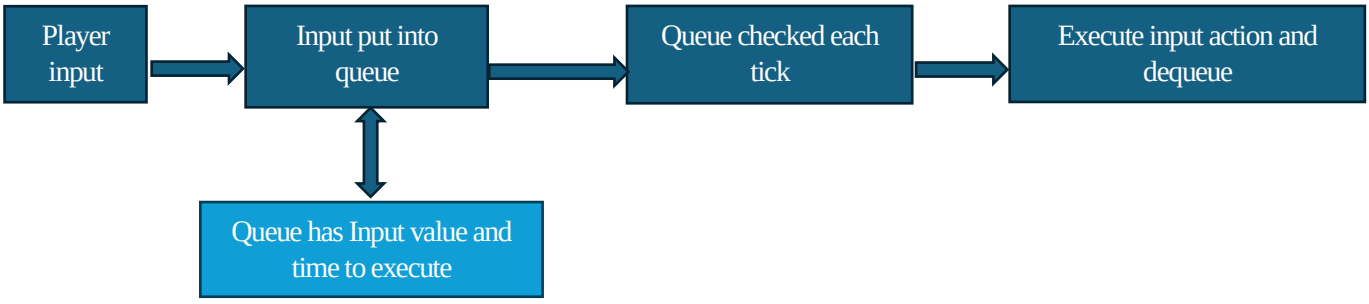


Figure 8: Latency Simulation for Player Inputs.

3.1.4 Simulating Fixed Time Delay Conditions

To simulate fixed time delay condition in Zombiefield, the latency simulation queue was used. As the queue can dynamically take in any latency and delay the inputs by that latency, we were able to set a fixed latency for the player to simulate fixed time delay conditions. During the study, this feature enabled us to try evaluate the effects of fixed time delay on player experience.

3.1.5 Adaptive Time Delay

Adaptive time-delay is an improvement over fixed time delay aimed at providing a fair and responsive gaming experience. This system is designed to only activate during player interactions and is turned off when the players are not interacting with each other. In Zombiefield, this activation is done by checking if the networked zombies can be interacted with or not. Rays from the players chest are fired to all the networked zombies, and if any of those rays are unobstructed, the game activates time delay. Otherwise, time delay is kept off.

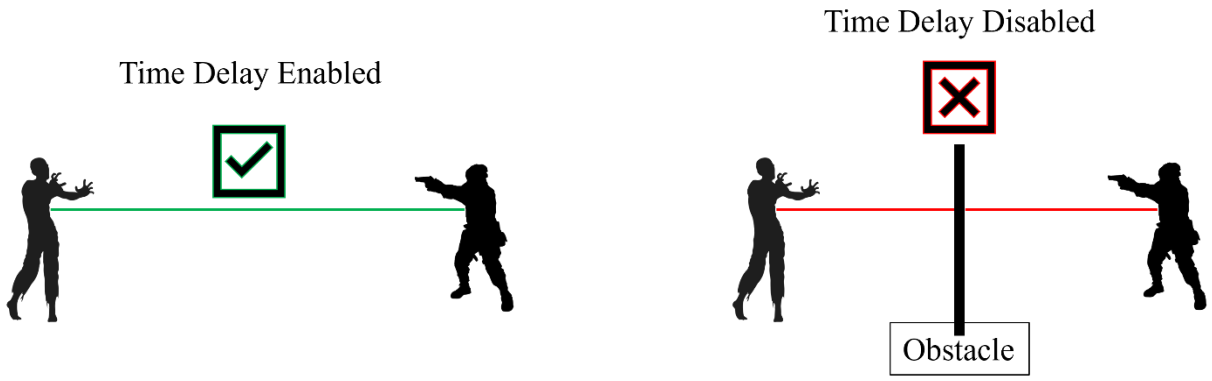


Figure 9: Time Delay Activated (Left) versus Time Delay Deactivated (Right).

3.1.6 Adaptive Time Delay Activation System

In Zombiefield, the activation of the time delay system can be gradual, instant or kept on/off for a certain amount of time, depending on the desired effect. Gradual or smooth adaptive time delay increases player latency slowly, mitigating the effect of sudden change in latency. Instant activation, on the other hand, can be used to quickly introduce latency into the game environment and maintain maximum fairness for players. Keeping adaptive time delay on or off for a certain amount of time ensures that the latency does not change too frequently. All these activation effects were achieved by dynamically changing the player latency variable based on desired conditions, which effected the latency simulation for player input shown in figure 6 in real time. As Zombiefield is a single player game, we had no way of assessing fairness. However, implementing different ways of activating adaptive time delay helped us to assess its impact on player experience.

3.1.7 Configuration Files

There were two types of configuration files that controlled various aspects of Zombifield gameplay. These configuration parameters also helped fine tune our study by allowing to set different latency settings for different round as well as in general settings. These configs are:

Global Configs – A file named *DefaultZFConfig.ini* was used to control various global settings of the game. The global configuration consisted of the following settings:

- RoundTimer – How long each round lasts.
- Camera Expsure – Controls the brightness of the camera. Default is 0 (Nighttime). Higher is brighter, 4 is daylight.
- WeakAISpawnCountPerCycle – How many weak zombies (Standard – non networked) spawn per cycle.
- StrongAISpawnCountPerCycle – How many strong zombies (Networked) spawn per cycle.
- UltimateAISpawnCountPerCycle – How many ultimate zombies (Networked) spawn per cycle.
- MinDelayPerCycle – Minimum delay between each spawn cycle.
- MaxDelayPerCycle – Maximum delay between each spawn cycle.

During each spawn cycle, there is 50% chance of the strong or the weak zombie spawning (either or) and 10% chance of an ultimate zombie spawning (exclusive).

Round Confisg – A file name RoundConfig.csv was used to control the latency settings of each round. Each row in that file represented a possible round setting. Latency values remained the same for the entire round. The columns of that file are:

- Before TD Latencies – The base player latency without time delay.
- After TD Latencies – Player target latency after activating time delay.
- ADT kick in delays – How long it takes for the player latency to go from base to the target latency after ATD is enabled.
- ADT let off delays – How long it takes for the player latency to go from target to the base latency after ATD is disabled.
- TD stay of inertia delays – How long time delay stays off once ATD is disabled.
- TD stay of inertia delays – How long time delay stays on once ATD is enabled.

3.1.8 Logging System

A logging system was developed to collect performance data in the back end as well as in game QoE survey answers. The logging system generated 4 files in each session. They are:

- *PerTickLog* logs player positional, performance, latency, pose, animation and other metrics per tick as a row of data. This file has one row of data for every tick.
- *RoundLog* logs the summary of each round including various performance metrics and the QoE answers. Each row of data is recorded after the end of each round. Most of the data analysis was done using this log file.
- *ProjectileLog* logs metrics for each bullet that's shot. Each row of this file is created whenever a projectile gets destroyed. It logs metrics such as round number, how much the bullet travelled, what it hit, where it was shot from, what was its destination, how long it survived and was it an aimed bullet or not.
- *EnemyLog* logs metrics for each enemy that spawns in the game. Each row is recorded whenever an enemy gets destroyed. This logs metrics such as spawn position, how long the enemy lived, how much damage it did, how long it travelled, what type of enemy it was, how much damage it took from the player.

3.1.9 Experiment Design

Table 1 outlines the specific latency values assigned to different rounds, as well as values affecting how adaptive time delay is implemented.

3 The Base group provided a baseline assessment with no delay, illustrating performance without external latency factors. Group 1, labeled Adaptive Time Delay (ADT), experienced latency spikes when encountering a networked enemy, with the delay dissipating after the enemy left. Group 2 faced a constant delay throughout the entire round, allowing for analysis of performance under consistent latency conditions. Group 3 combined Regular Time Delay and Adaptive Time Delay, with the base delay set at half when facing a networked enemy. Groups 4 and 5 explored various aspects of activating Adaptive Time Delay, examining the effects of latency adjustments on user experience.

Players played fixed length rounds. After the round ended, players were asked two questions regarding their experience in that round. They are:

- Rate your Quality of Experience for this round - (1) Low to (5) High – A slider was used.
- Was this experience acceptable? (Yes/No)

Table 1: Zombiefield experimental configurations.

	Rounds / 75s	Before TD Latencies	After TD Latencies	ADT kick in delays	ADT let off delays	TD stay off inertia delays	TD stay on inertia delays
Base Group	1	0	0	0	0	0	0
	2	0	50	0	0	0	0
	3	0	100	0	0	0	0
Group 1	4	0	150	0	0	0	0
	5	50	50	0	0	0	0
	6	100	100	0	0	0	0
Group 2	7	150	150	0	0	0	0
	8	25	50	0	0	0	0
	9	50	100	0	0	0	0
Group 3	10	75	150	0	0	0	0
	11	0	50	2	2	0	0
	12	0	100	2	2	0	0
Group 4	13	0	150	2	2	0	0
	14	0	50	0	0	2	2
	15	0	100	0	0	2	2
Group 5	16	0	150	0	0	2	2

3.1.10 User Study

A User study using Zombiefield experimental config was conducted and finalized by Interactive Qualifying Project Team (James Cannon, Saketh Dinasarapu, Ao Jiang, Hanzalah Qamar) [22].

Users were recruited from majors in computer science, robotics engineering, electrical and computer engineering, and interactive media and game development. A Calendly link was provided for scheduling participation times. Participants were offered credits for playtesting and a chance to win a $10 gift card either through high scores or a raffle. Reminders were sent on the day of their appointments to confirm attendance.

The data collection for the Zombiefield User Study included academic and personal information, gaming profiles, platform and connection types, human benchmark tests, and voluntary raffle entries. Below is a simplified summary of each section:

Academic and Personal Information: Collected participants' academic levels and ages to understand their gaming backgrounds. Information on gender and years of experience in FPS games was also gathered.

Gaming Profile: Participants detailed their experience with specific FPS games, playstyles (competitive or casual), and self-assessed skill levels.

Platform and Connection Type: Information on whether participants played on PCs, consoles, or other platforms was collected, along with their type of internet connection (Ethernet or Wi-Fi).

Human Benchmark Test: Participants completed a ten-round reaction time test, with results used to analyze FPS game performance.

Raffle Entry: Participants could enter a raffle for a $10 gift card by providing their email, which was kept confidential. The highest scorer in the study also received a $10 gift card, with no participant winning both.

Procedure: A training video [24] was created to provide detailed instructions for participants on playtesting procedures and gameplay. Each session began with participants agreeing to terms on an Institutional Review Board (IRB) consent form, followed by completing a demographic survey to collect information about their gaming background. Participants then took a human benchmark test to assess reflexes by clicking when the screen background changed. This was followed by a 4-minute practice round to familiarize them with the game mechanics. For the official gameplay, participants completed 16 rounds, each lasting 75 seconds, after which they rated their experience on a scale of 1 to 5 and noted if the conditions of that particular round were acceptable. Each session lasted 20 minutes. Participants were also encouraged to ask questions and provide feedback after each session to gather insights for improvement.

3.1.11 Hardware Used

The experimental system was equipped with top-tier hardware, including an RTX 2080 graphics card, Intel Core i9 11900K processor, and 32 GB of Corsair Vengeance DDR4 RAM. Storage was handled by a Samsung 970 EVO SSD, and the system was built on an ROG Strix Z590E motherboard. Input devices included a Logitech gaming keyboard and mouse, and visual output was provided by a Lenovo Legion 240 Hz monitor.

The input latency of the hardware was measured 10 times, yielding an average latency of 26.2 ms. The measurements ranged from a low of 20 ms to a high of 32 ms, with a standard deviation of 3.2 ms.

3.2: Last Stand – Two Player Study

The goal of this two player study is to study adaptive time delays effect on fairness and player experience. This study also evaluated all 3 hypotheses. A multiplayer game called Last Stand was built using Unity 2023 to support this study.

3.2.1 Overview

Last Stand has realistic graphics and fast-paced gunplay mechanics, which includes hitscan-based gunplay and positional audio. The game has a configurable experimental harness which was used to setup our experiment. It also has a logging system which gives round logs, per tick player logs and per bullet logs.

3.2.2 Gameplay

Players were pitted against each other in a 1v1 setting with rounds of deathmatch. Dying made the player's respawn at the most distant of six predetermined spawn points, chosen to be as far away from the other player as possible. On respawn, players health and ammo is replenished. There is also 1.5 seconds of invincibility period on respawn, which is 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 fire-rates, for our study we chose a semi-auto DMR with 11 bullets and unlimited magazines. A laser is projected from the muzzle of the gun for target assist.

The FPS Animation Framework [23] was used to animate the gun and the player’s avatar. Recoil, weapon sway, look and leaning, among others, are procedurally animated. All of these animations are replicated over the network.

For our study, the players were able to walk, sprint and lean left or right. Besides shooting, for gunplay, players are able to aim down sight and reload. Although the game auto reloads if the player runs out of ammo, the player can do a tactical reload as well, before the ammo runs out.

The gameplay takes place in an abandoned hanger, surrounded by large trees. Trees are strategically planted to provide cover for the players. There is foliage and terrain elevation for aesthetics as well. There are 6 spawn points surrounding the hanger and the players spawn at those points away from each other, after being eliminated.

The game has texts on for score, health, kills, deaths, ammo, rounds and duration. Yellow indicators light up when the player gets close to an opponent, marking the direction and proximity of the opponent. For example, a yellow bar with light up in the top, if the opponent is in front of the player. The yellow bar gets bigger and brighter as the player gets closer to the enemy. While being shot, the yellow markers turns red. So, if the player gets shot from left, a red bar is be shown at the left side of the screen. For shooting, a white hit marker is displayed on confirmed hit, yellow for headshots and red for kills. Figure 10 shows a screenshot of Last Stand, highlighting some of its UI elements stated above.

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

Figure 10: Last Stand screenshot.

A video of the game can be found here: https://www.youtube.com/watch?v=5AMLja4qZkI. The video demo illustrates two rounds of Last Stand. The demo starts by showing animations regarding movement and aiming. Each player can detect the proximity and direction of the other player by looking at the yellow indicators on the sides of the screen. If the indicator is on the top, it means the opponent is in front of the player, and similarly for both the sides as well. The indicator also grows larger as the player approaches the opponent. Starting from 0:28, the opponent's animations are displayed, indicating that animations are replicated across the network. At 0:44, the player secures a confirmed kill, increasing both the kill count and score. After that regular combat takes place, highlighting death events, respawns, and headshots. After the end of the round players are presented with two consecutive qualitative questions at 1:22. During round 2, regular combat is shown followed by the same questionnaire at the end.

3.2.3: Networking Implementation

Unity netcode was used to enable client server networking in Last Stand. The game runs as client-server mode, where the server controls the rounds and the configurations for those rounds. The server runs at a fixed ticket rate of 60hz. The client framerate is uncapped.

After basic networking we implemented ways of simulating latency conditions in game. This was achieved by delaying all player actions on the client side by a set latency value. The server can dynamically control how much latency each client experience. This simulated a fully server authoritative environment, for the client. This latency is added to all player actions in Last Stand by delaying all player inputs. Inputs are put inside a queue and executed in later ticks where it matches or exceeds its execution time. This procedure of adding input delay is identical to Zombiefield as shown in Figure 8.

3.2.4: Implementing Time Delay

3.2.4.1: Fixed Time Delay

The time delay compensating system has been implemented in Last Stand to improve fairness in gameplay. In Figure 11, player 1 has a network latency of 100ms, while player 2 has a network latency of 200ms. To mitigate the unfairness caused by these differences in latency, a time delay was added to all player inputs and then sent to the server. This makes the game simulate fully server authoritative player actions. The added delay equalizes the effective latency between both players. By doing so, both players have a level playing field with regards to network latency. In Last Stand, different latency can be set via the experimental configuration file, and the server propagates the configs to the clients.

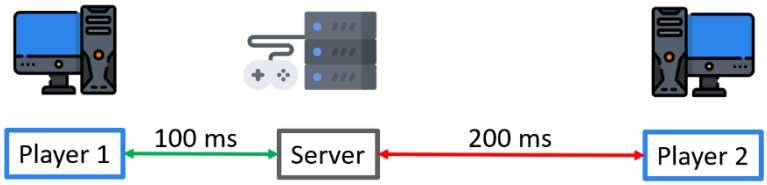


Figure 11: Server and client model without time delay.

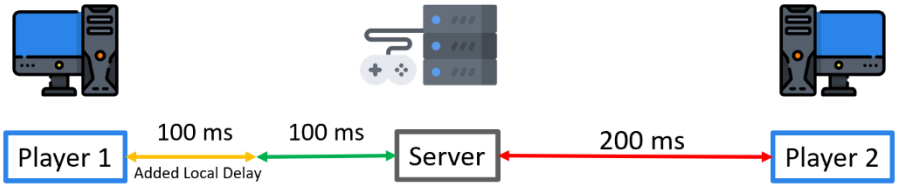


Figure 12: Server and client model with time delay.

3.2.4.2 Adaptive Time Delay

Adaptive time-delay is a technique that adjusts network delay during gameplay based on player conditions, such as network latency and interactions. It activates to balance gameplay when players interact and turns off when they do not, preserving a fair and responsive experience. For instance, if players can see each other and might interact (as shown in Figure 13), time delay is activated. Conversely, if an obstacle blocks interaction between players, the delay is not activated to avoid degrading the experience for the low latency player.

Figure 13: Players can interact (left); players cannot interact (right).

The adaptive time delay system in Last Stand is a way to make the game more fair, regardless of players network latency. The system was only activated when there were no obstacles between players, which is determined by shooting a ray between player 1 and player 2, and vice versa. If the ray does not intersect with any object in the game environment, the time delay system is activated. If the ray does intersect with an object, the delay is disabled to avoid affecting the player's experience. Unlike Zombiefield, instead of shooting a ray from the chest, Last Stand shoots a ray from the middle of one player's head to another. This ray is shot continuously, even when the players are not facing each other.

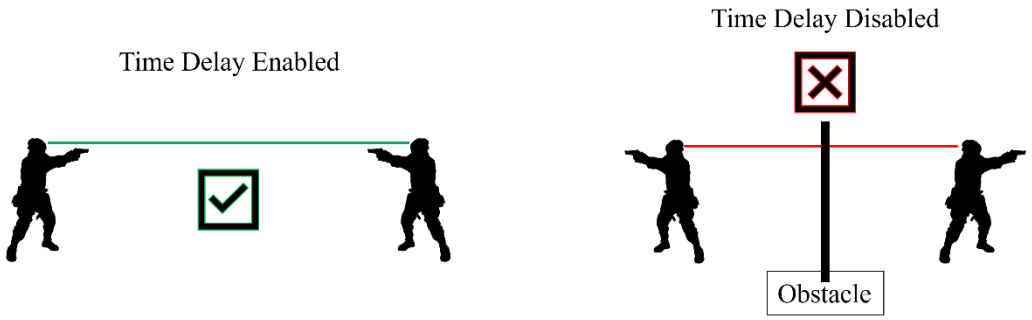


Figure 14: Ray shot from head for Time Delay Enabled (Left) versus Time Delay Disabled (Right).

3.2.4.3 Adaptive Time Delay Activation System

In Last Stand, the activation of the time delay system can be gradual or abrupt depending on the desired effect. Gradual or smooth adaptive time delay increases player latency slowly (within a few seconds), mitigating the effect of sudden change in latency. Abrupt activation, on the other hand, can be used to quickly introduce latency (within a few frames) into the game environment and maintain maximum fairness for players. Implementing different ways of activating adaptive time delay helped assess its effectiveness and impact on player experience and fairness.

All these activation effects were achieved by dynamically changing the player latency variable based on desired conditions, which effected the latency simulation for player input shown in Figure 6, similar to Zombiefield.

3.2.5 Log System

A logging system was developed to collect performance data in the back end as well as in game QoE survey answers. The logging system generated 3 files in each session for each client. They are:

- *PerTickLog* logs player positional, performance, latency, pose, animation and other metrics per tick as a row of data. This file has one row of data for every tick, thus making it the most comprehensive file.
- *RoundLog* logs the summery of each round including various performance metrics and the QoE answers. Each row of data is recorded after the end of each round. Most of the data analysis was done using this log file.
- *ShotLog* logs metrics for each bullet that's shot. Each row of this file is created whenever a shot is fired. It logs metrics such as what the bullet hit, where it was shot from, what was its destination, how far apart the players were during shot, how much it missed the other player by etc.

3.2.6 User Study

This user study provides an evaluation of the impact of time delay on the fairness of a first-person shooter game. The main comparison was between no compensation and adaptive time delay, with both fixed time delay and adaptive time delay being evaluated. Our user study had 1 practice round and 20 rounds with different sets of latency configurations.

Table 2: Experimental Parameters for Last Stand Clients.

Player 1				Player 2			
Base Latency	Max Latency	ATD Increase Rate	ATD Decrease Rate	Base Latency	Max Latency	ATD Increase Rate	ATD Decrease Rate
0	0	2000	2000	0	0	2000	2000
0	0	2000	2000	100	100	2000	2000
0	0	2000	2000	200	200	2000	2000
100	100	2000	2000	100	100	2000	2000
200	200	2000	2000	200	200	2000	2000
100	100	2000	2000	200	200	2000	2000
0	100	2000	2000	100	100	2000	2000
0	200	2000	2000	200	200	2000	2000
0	100	75	75	100	100	2000	2000
0	200	75	75	200	200	2000	2000
.	.	.	.	.	.	.	.
10 more configurations were used with Player 1 and Player 2 configuration values swapped							

Table 2 lists the experimental configurations that were used. Here the base latency indicates the latency the player experiences regardless of any compensation technique. So, when base and max latency are the same, it works as fixed time delay. For different base and max latency, adaptive time delay is activated. The rate indicates how fast the latency goes from base to max latency. For increase (or decrease) rate of 2000, frame time * 2000 is added to the base latency per game tick until the latency reaches max. For an increase/decrease rate of 75, the base latency reaches max much slower, while 2000 being instant. This gave us the opportunity to test both smooth adaptive time delay and abrupt adaptive time delay. The following configurations were used:

- Base: A condition with no latency for both players. (Row 1)
- Unfair Conditions: One player has no latency while the other has a fixed latency. (Row 2,3)
- Latency Conditions: Both players experience a fixed latency. (Row 4,5,6)
- Adaptive Time Delay Abrupt: Latency that adjusts abruptly during player interactions. (Row 7,8)
- Adaptive Time Delay Smooth: Latency that adjusts smoothly during player interactions. (Row 9,10)

These conditions were repeated twice for both players, with a practice round in the beginning, making a total of 21 rounds each lasting 80 seconds. After the round ended, players were asked two questions regarding their experience in that round. They are:

- Rate how laggy did you feel the round was - (1) Very Laggy to (5) Very Smooth – A slider was used.
- Was the round fair? (Yes/No)

The user study for Last Stand was approved by the IRB. Recruitment email was sent for players to sign up, in slotter, with their institutional email. This was later used to contact them and give players IMGD playtesting credit if asked for.

Before the study, users were asked to read and sign the Informed Consent form. Then they were asked to fill out the demographics survey form where they were asked general demographic questions and questions related to their gaming/FPS skill level. All these questions were optional.

Afterwards they completed a "Reaction Time Test" where they reacted to change of color in the screen on a website. The proctor explained the game features, controls, and instructions to the players. Afterwards they played 21 rounds with each other, which lasted about 30 minutes.

3.2.15 Hardware Used

Both of the clients used a system was equipped with an GTX 1080 graphics card, Intel Core i7 8700K processor, and 64 GB of DDR4 RAM. An HP 1TB SSD was used for storage. Input devices included a generic Dell keyboard and Logitec g502 mouse. Visual output was provided by a Alienware 500Hz Gaming Monitor - AW2524HF. The monitor was set to 360 Hz during the entire study. A pair of headphones were also provided, but some users brought their own headphones. The input latency of the hardware was measured 10 times using NVIDIA Reflex Latency Analyzer on

Alienware AW2524HF, yielding an average latency of 31.27 ms. The measurements ranged from a low of 23.7 ms to a high of 36.3 ms, with a standard deviation of 3.5 ms. The mouse latency was 2.5 ms as reported by the NVIDIA App Beta v10.0.0.535.

4. Analysis

This chapter presents an analysis of user studies for Zombiefield and Last Stand, focusing on four primary aspects: demographic profiles of participants, the impact of adaptive time delay on responsiveness and fairness, comparison between different adaptation strategy, and a comparison with previous work.

4.1 Demographics

4.1.1 Zombiefield Single-Player User Study

For the Zombiefield study, 38 participants completed the study but 3 user sessions were discarded due to the incompleteness of the testing play or inactivity during the testing play. Table 3 summarizes the demographics of the users analyzed. Numeric values are means with standard deviation in parentheses. The average age was 20 (M=20.2, SD=3.8) years, ranging from 17 to 37 years and 36 of them were male with 6 female. General and FPS gaming skills were self-reported answers (1 low to 5 high); most were experienced gamers and many were experienced in FPS games. The average reaction time was measured in milliseconds and was generally fast, which is typical for experienced gamers.

Table 3: Zombiefield participant demographics.

User	Age (years)	Gender	General Gaming Skill (1-5)	FPS Gaming Skill (1-5)	Reaction Time (ms)
38	20.2 (3.8)	♂36 ♀2 O0	3.7 (0.9)	2.9 (1.1)	195.6 (24.1)

4.1.2 Last Stand Two-Player User Study

A total of 23 users participated in the Last Stand user study. There were a total of 14 sessions, in each 2 players played against each other and 5 sessions where there was only 1 participant and they played against the proctor. Table 4 summarizes the participants demographics. Numeric values are means with standard deviation in parentheses. The average age of the participants was about 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 1 non-binary. General and FPS gaming skills were self-reported answers (1 low to 5 high) – players were experienced gamers with many being experienced in FPS. Total hours of FPS played are also self-reported. Reaction time is in milliseconds. Participant reaction time are fast, on average, typical of experienced gamers.

Table 4: Last Stand participant demographics.

User	Age (years)	Gender	General Gaming Skill (1-5)	FPS Gaming Skill (1-5)	Total Hours of FPS Played	Reaction Time (ms)
23	24.6 (4.9)	♂16 ♀6 O1	3.5 (1.3)	2.7 (1.4)	838.8 (1204.02)	204.3 (39.7)

4.2 Responsiveness

4.2.1 Quality of Experience – Zombiefield

Figure 15 illustrates player QoE across latencies of 0 ms, 50 ms, 100 ms, and 150 ms for both fixed and adaptive time delays. 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 plot shows mean QoE value with a 95% confidence interval. The green line represents fixed time delay and the blue here represents adaptive time delay. Points are mean values and with 95% confidence interval. From the graph adaptive time delay provides better QoE, as per the mean values for latencies above 50 ms.

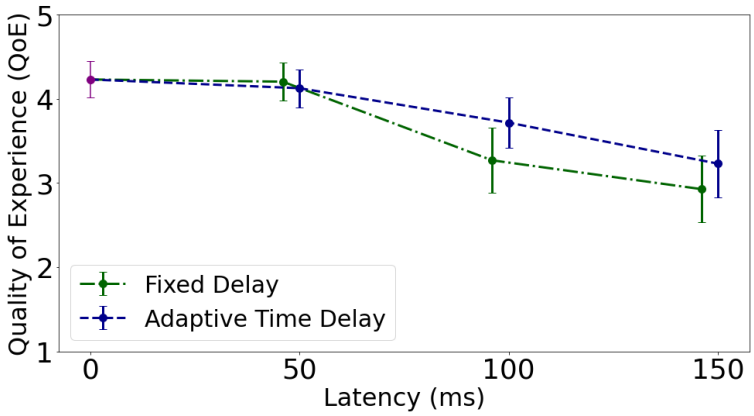


Figure 15: QoE comparison between adaptive time delay and fixed delay for Zombiefield.

Figure 16 shows the cumulative distribution function (CDF) of QoE. There are two lines in the graph: one for fixed time delay (green) and the other for adaptive time delay (blue). The green line is mostly to the right throughout the CDF range, which means player experience was better for adaptive time delay, across different simulated latency, compared to fixed time delay.

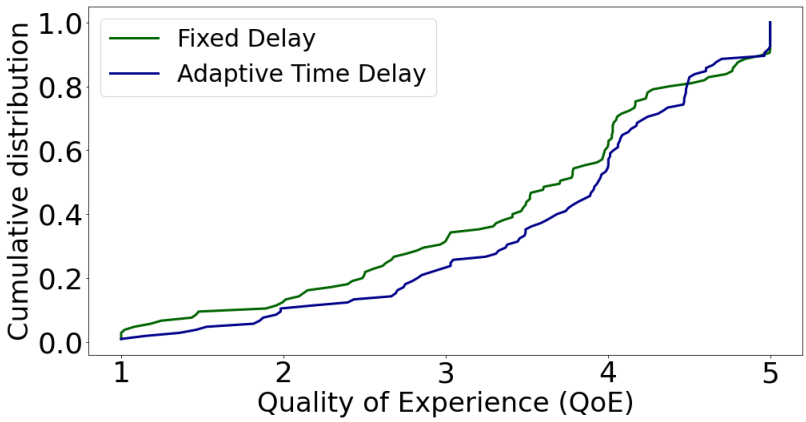


Figure 16: CDF of QoE for adaptive time delay and fixed delay.

4.2.2 Quality of Experience – Last Stand

Figure 17 shows player reported quality of experience (QoE) for no latency, fixed latency and adaptive time delay. The axes of the data presented are the same as Figure 15 but refer to the Last Stand game. When the players had adaptive time delay, they reported a better QoE compared to fixed time delay especially at the higher latency ranges. This suggests that adaptive time delay enhances the overall player experience.

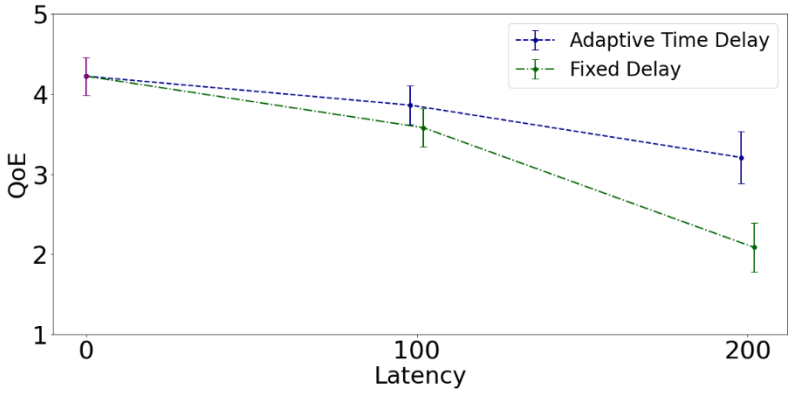


Figure 17: QoE comparison between adaptive time delay and fixed delay for Last Stand.

4.3 Fairness

For Last Stand, we can evaluate fairness between the participants. All the results in the subsections are from the Last Stand user study.

4.3.1 Accuracy

In Figure 18, player accuracy is shown as a mean with a 95% confidence interval. The X axis is latency and the Y axis is accuracy as percentage. Here On left, unfair conditions are shown, where the blue line represents players with no delay

and the green line represents players with fixed delay. The delay for the fixed latency player is shown on X axis. On the right adaptive time delay is shown as blue compared to players with fixed delay in green, for the same latencies.

In the case where one player had no latency and the other had a fixed 200 ms of latency, there is a noticeable difference in accuracy. The player with higher latency had significantly lower accuracy, indicating that the game was less fair for them as they struggled to aim accurately at their opponent.

The graph on the right, compares players with adaptive time delay to those with fixed delay. The accuracy difference between these two groups is small, indicating that adaptive time delay helps make the game fairer for the players.

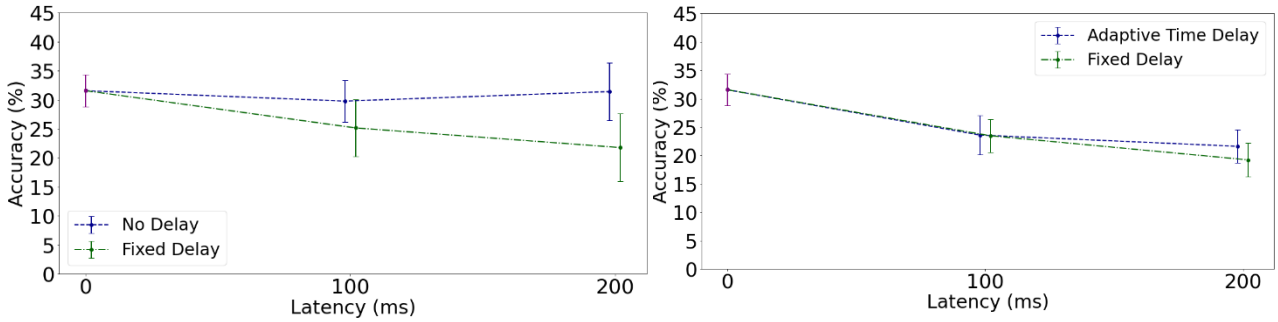


Figure 18: Accuracy for no delay versus fixed delay (left) and adaptive time delay versus fixed delay (right).

4.3.2 Score

Figure 19 shows results for similar conditions as Figure 18, but for score instead of accuracy. There is a consistent trend when comparing scenarios with no latency (left) and adaptive time delay on the right, versus fixed latency. The left graph shows that when players with no latency competed against players with 100 ms and 200 ms of latency, those with higher latency scored significantly lower. On the contrary the graph on the right indicates that adaptive time delay improves fairness in gameplay, as evidenced by the more balanced scores among players.

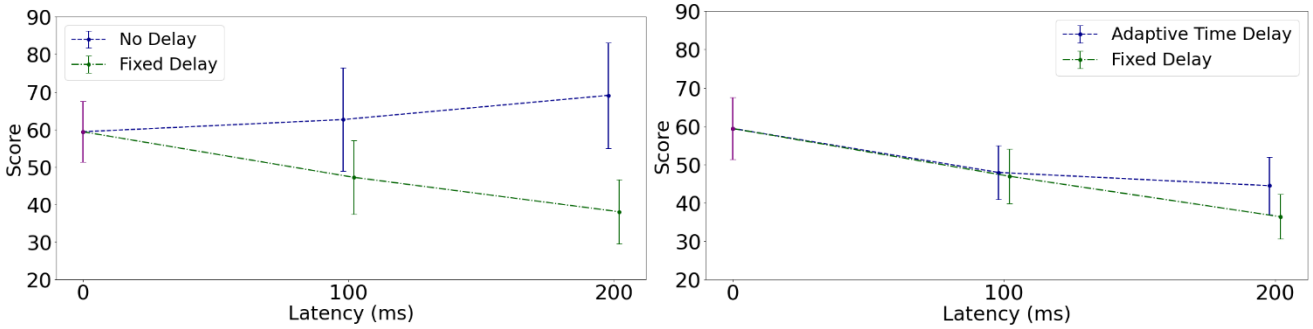


Figure 19: Score for no delay versus fixed delay (left) and adaptive time delay versus fixed delay (right).

4.4 Comparison of Adaptation Strategies

In both of our studies we had different types of adaptive time delay based on how fast latency was introduced in the game. The following results show comparison between these adaptation strategies.

4.4.1 Zombiefield

Figure 20 is similar to Figure 15, but instead of just showing fixed time delay and adaptive time delay, all the other types of adaptive time delay results are also shown here. Smooth adaptive time delay provides the best QoE, especially above 50 ms of latency.

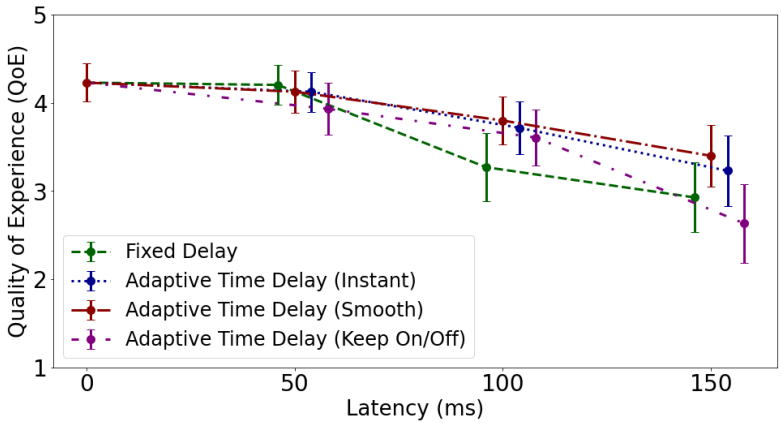


Figure 20: QoE for Adaptive time delay smooth, instant and keep on/off with fixed delay.

4.4.2 Last Stand

Figure 21 compares the QoE between adaptive abrupt and adaptive smooth with fixed time delay. At a latency of 200 ms, the adaptive smooth setting provides better QoE than adaptive abrupt. However, the difference between the two is less noticeable at 100 ms. This suggests that for higher latencies, smooth adaptive time delay may be preferable over abrupt adaptive time delay.

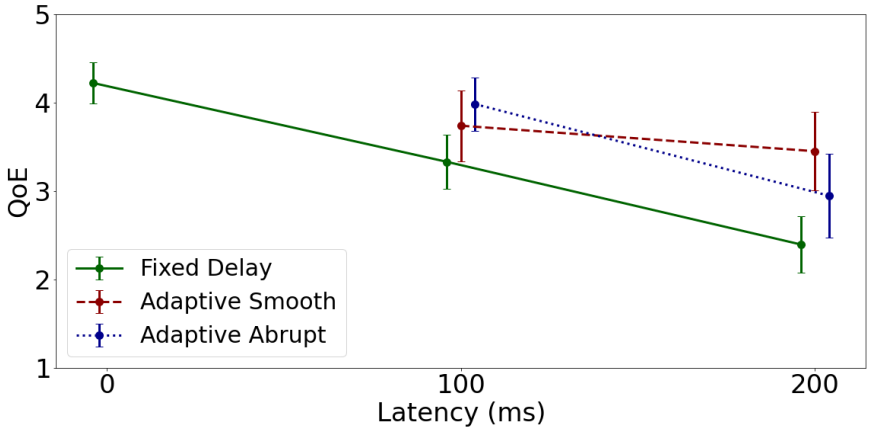


Figure 21: QoE for adaptive time delay abrupt and smooth with fixed delay.

Figure 22 shows the performance scores under the same conditions. Here, players show improved performance with adaptive smooth at 200 ms, compared to fixed time delay, whereas the performance with adaptive abrupt remains largely

unchanged compared to fixed time delay. Overall, while adaptive smooth may slightly enhance QoE, it may have reduced game fairness. This could be attributed to the adaptive smooth mechanism, where the adjustment of shot timings allows low latency players to not fully match the latency of the high latency players.

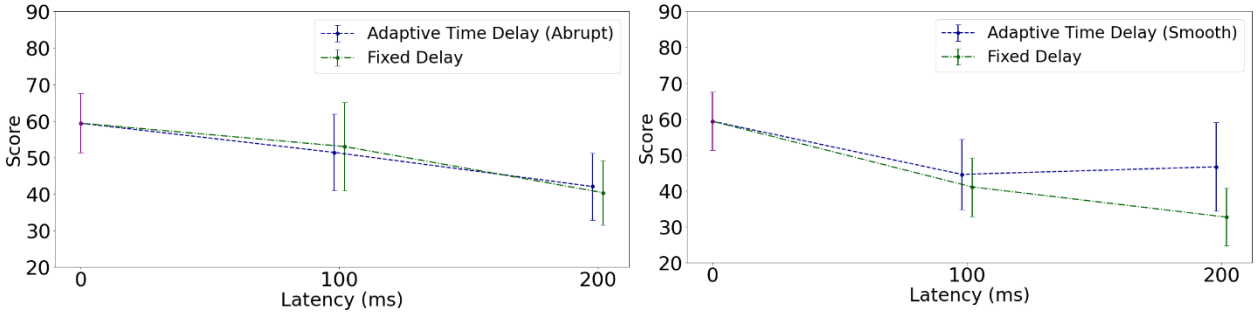


Figure 22: Score for adaptive time delay abrupt and smooth with fixed time delay.

4.5 Comparison with Previous Work

Our findings related to fairness in Last Stand can be directly compared with a study done by Zander et al. studying fixed time delay [16]. They added outgoing delay to a game server using a third-party software called Self-Adjusting Game Lagging Utility (SAGLU). When SAGLU is enabled, it simulates regular time delay by adding a desired amount of delay to the server. They used bots in Quake 2 and ran simulations with and without their software. One portion of their study covered accessing fairness between the bots with SAGLU enabled and disabled.

Figure 23 shows their graph on the left, which has SAGLU disabled. The X axis show delay the bots experienced and Y axis shows mean kill rate of the bots. The plot shows mean kill rate value with 99% confidence interval. The red line represents the bots when they had no latency, and the blue line represents bot with a fixed latency, that is shown on the X axis. On the right is our graph which is identical to Figure 22 (left) but scaled to 400 ms on the X axis. For no delay and fixed delay conditions, both studies show increased unfairness due to performance differences as the latency goes

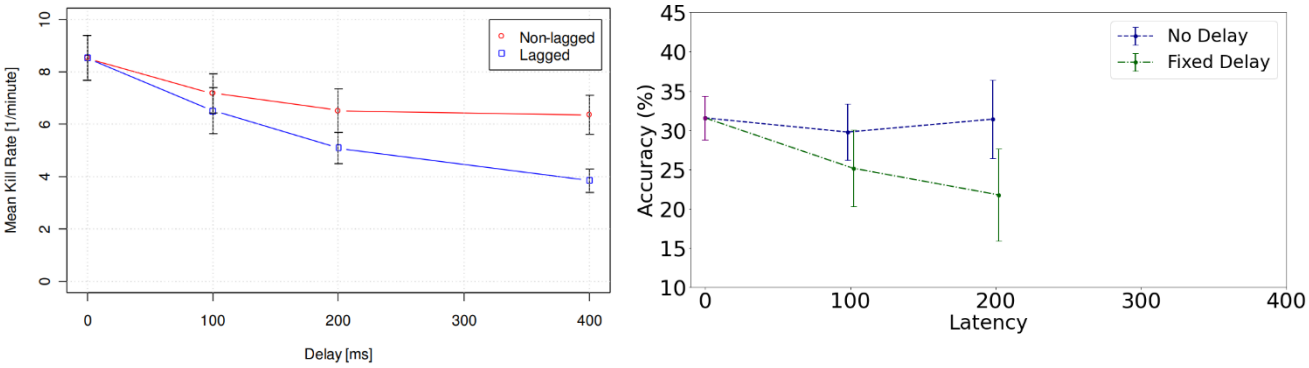


Figure 23: Performance comparison between no delay versus fixed delay for Zandar et al. (left) and Last Stand (Right).

Figure 24 is similar to Figure 23 but with regular time delay enabled. For the graph on the left, it was achieved by enabling SAGLU and for Last Stand (right) it was achieved by making both players experience the same latency all the time. Both studies show an increase in fairness as the performance difference diminished.

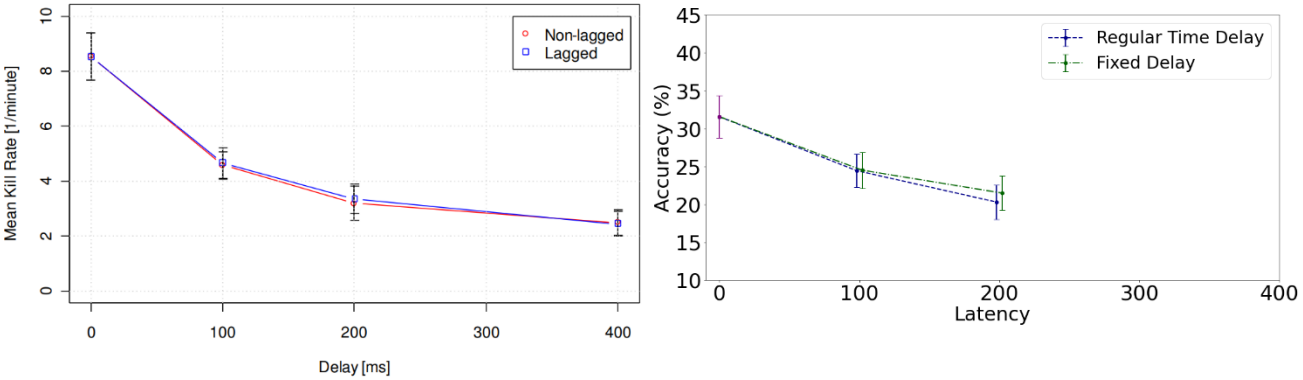


Figure 24: Performance comparison between regular time delay versus fixed delay for Zandar et al. (left) and Last Stand (Right).

4.6 Summary

Based on the results adaptive time delay improved player experience compared to fixed time delay in two FPS games, Zombiefield and Last Stand for latencies above 50 ms. In Last Stand, adaptive time delay also improved fairness. Smooth adaptive time delay, which gradually introduces delays, enhanced both the quality of experience and fairness compared to fixed time delay and other types of adaptive time delay. This suggests that adaptive time delay, especially when applied smoothly, can make games more enjoyable compared to fixed time delay, and more fair compared to no latency compensation.

4.7 Limitations

One of the main limitations of our study is that we couldn't test our technique in scenarios where more than two players are on the map. In large commercial game modes like conquest, battle royale, and team deathmatch, which involve many players, adaptive time delay might need to be managed differently. Network conditions used in both studies were simulated, and we did not implement network jitter, packet loss or dynamically changing latency. In Last Stand, there were instances where the players had a gap in their FPS skill level, which caused unfairness. Our studies used a certain type of weapon where in typical FPS games, users have multiple choice of weapons (ie. snipers, rocket launchers, throwables, crossbow etc.). Moreover, we also had only one type of map designed which only promoted close combat battles.

5. 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 latency to the player with lower latency which equalizes latency amongst the players. This can make the game fairer but it degrades the responsiveness of the low latency player due to added latency.

We propose adaptive time delay with adds latency to the low latency player only when they have a chance of interacting with each other. This has the potential to make the game more responsive for the low latency player when they are not interacting with the other player and make the game fairer when they are interacting with others.

A 38 person study was conducted with single player Zombiefield to evaluate adaptive time delay with different amount of latency as well as comparison with fixed time delay. Another 23 person study was conducted with two player Last Stand to evaluate impact of adaptive time delay on player experience and fairness. During the study, players played multiple rounds, with multiple conditions and latency. After each round they were asked questions regarding their experience in that round, and their performance was also logged by both games in the back end. Analysis of our result suggests that adaptive time delay can make the overall experience for the player better for both studies over fixed time delay. Performance analysis from Last Stand user study show improved fairness when adaptive time delay was used over no latency. We also evaluated different type of adaptive time delay based on their activation time i.e. smooth, instant and keep on/off. After analyzing the result, we found that smooth adaptive time delay performs slightly better than the other techniques, in terms of QoE.

6. Future Work

Adaptive time delay in both studies showed promising results. However, its effectiveness can be further evaluated. This section describes potential work that can be done in the future to further assess its effectiveness. Future work is split into 3 subsections:

- Short term - immediate parts of the work that could be started.
- Medium term - work over a several months that continues and expands the study.
- Long term - extensive work that aligns with the study's goals but likely needs a new study.

6.1 Short Term

In our studies we used 3 types of adaptive time delay based on how fast the latency is added to the players actions. Results suggest that adaptive smooth provides better QoE than the other methods. In the single player study, adaptive smooth goes from 0 to target latency over 2 seconds. This is true for Last Stand as well, as it

takes a few seconds to go from base to target. We can further evaluate this adaptation strategy by testing over various time ranges and to fine tune the adaptation. Adaptive smooth can also be paired with keep on/off, where latency is added to the player smoothly and keep the latency for a fixed amount of time, before changing again. Our implementation of adaptive smooth is linear. We could use a more complex algorithm i.e., a non linear function, increasing or decreasing the latency.

6.2 Medium Term

In both of our studies, we shot only one ray to detect the visibility to the other player. This detection method is very efficient, but it may have issues depending upon the terrain and player orientation. For example, if the opponent is partially obstructed, the ray may be blocked, disabling adaptive time delay. In future work, this detection method can be further improved by shooting multiple rays, to cover the entirety of the opponent.

Our multiplayer study using Last Stand has a 1v1 deathmatch format. Larger game modes such as conquest, battle royal, team deathmatch and rush involve more than two players. In large game modes, shooting multiple rays between the players over a large map, could cause performance issues. In these scenarios, setting an area of interest for the player could allow for efficient adaptive time delay detection only when an opponent enters this zone. We can also try out different weapon types, such as snipers, rocket launchers or throwables. When long range weapons such as snipers or any weapon with a large scope is in use, the players' area of interest could be expanded so it covers a larger area for detection.

6.3 Long Term

We used two FPS games to evaluate the effectiveness of adaptive time delay, in our studies. Our long-term goal can be to test the effectiveness of adaptive time delay for other genres of multiplayer games, such as real time strategy, racing, multiplayer online battle arena and action/adventure. As latency affects different game genres in different ways, figuring out different ways of implementing adaptive time delay effectively could be a possible future work.

We did not use any other latency compensation techniques in our studies. Usually most of the multiplayer games use client-side prediction for movement, aiming and various other player actions. Adaptive time delay could be combined with client-side prediction to enhance player experience. It can also be combined with interpolation, time-warp and potentially other latency compensation techniques to further improve player experience and fairness.

7. References

[1] Marko Dimitrievski. 2023. Gaming statistics – 2023. Last Accessed 26th April 2024. https://truelist.co/blog/gaming-statistics/

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

[3] Shengmei Liu, Mark Claypool, Atsuo Kuwahara, Jamie Sherman, and James J Scovell. 2021. Lower is Better? The Effects of Local Latencies on Competitive First-Person Shooter Game Players. In Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 326, 1–12. https://doi.org/10.1145/3411764.3445245.

[4] Zenja Ivkovic, Ian Stavness, Carl Gutwin, and Steven Sutcliffe. 2015. Quantifying and Mitigating the Negative Effects of Local Latencies on Aiming in 3D Shooter Games. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15). Association for Computing Machinery, New York, NY, USA, 135–144. https://doi.org/10.1145/2702123.2702432

[5] Kjetil Raaen and Andreas Petlund. 2015. How Much Delay is There Really in Current Games? In Proceedings of the 6th ACM Multimedia Systems Conference (MMSys '15). Association for Computing Machinery, New York, NY, USA, 89–92. https://doi.org/10.1145/2713168.2713188

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

[7] Wai-Kiu Lee and Rocky Chang. 2015. Evaluation of Lag-related Configurations in First-person Shooter Games. In Proceedings of the ACM Workshop on Network and System Support for Games (NetGames'15). IEEE, Los Alamitos, CA.

[8] L. F. Kawabata de Almeida and A. S. Felinto. 2018. Evaluation of the Motion-Aware Adaptive Dead Reckoning Technique under Different Network Latencies Applied in Multiplayer Games. In proceedings of 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames). Foz do Iguacu, Brazil, pp. 137-13709, https://doi.org/10.1109/SBGAMES.2018.00025.

[9] C. Gao, H. Shen and M. A. Babar. 2016. Concealing Jitter in Multi-Player Online Games through Predictive Behaviour Modeling. In proceedings of IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD). Nanchang, China, pp. 62-67, https://doi.org/10.1109/CSCWD.2016.7565964.

[10] Yahyavi, A., Huguenin, K. & Kemme, B. Interest Modeling in Games: The Case of Dead Reckoning. Multimedia Systems 19, 255–270 (2013). https://doi.org/10.1007/s00530-012-0275-z

[11] Yi Zhang, Ling Chen, and Gencai Chen. 2006. Globally Synchronized Dead-reckoning with Local Lag for Continuous Distributed Multiplayer Games. In Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games (NetGames '06). Association for Computing Machinery, New York, NY, USA, 7–es. https://doi.org/10.1145/1230040.1230071

[12] Dingliang Liang and Paul Boustead. 2006. Using Local Lag and Timewarp to Improve Performance for Real Life Multiplayer Online Games. In Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games (NetGames '06). Association for Computing Machinery, New York, NY, USA, 37–es. https://doi.org/10.1145/1230040.1230047

[13] Robert Salay and Mark Claypool. 2020. A Comparison of Automatic versus Manual World Alteration for Network Game Latency Compensation. In Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '20). Association for Computing Machinery, New York, NY, USA, 355–359. https://doi.org/10.1145/3383668.3419910

[14] Saeed Shafiee Sabet, Steven Schmidt, Saman Zadtootaghaj, Babak Naderi, Carsten Griwodz, and Sebastian Möller. 2020. A Latency Compensation Technique Based on Game Characteristics to Mitigate the Influence of Delay on Cloud Gaming Quality of Experience. In Proceedings of the 11th ACM Multimedia Systems Conference (MMSys '20). Association for Computing Machinery, New York, NY, USA, 15–25. https://doi.org/10.1145/3339825.3391855

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

[16] A. Kaiser, N. Achir and K. Boussetta, "Improving Energy Efficiency and Gameplay Fairness for Time-Sensitive Multiplayers Games in MANET," 2010 IEEE International Conference on Communications Workshops, Cape Town, South Africa, 2010, pp. 1-5, https://doi.org/10.1109/ICCW.2010.5503904.

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

[18] Doowon Paik, Chung-Ha Yun, and Jooyeon Hwang. 2008. Effective Message Synchronization Methods for Multiplayer Online Games with Maps. Computer Human Behavior 24, 6, 2477–2485. https://doi.org/10.1016/j.chb.2008.03.004.

[19] Jeremy Brun, Farzad Safaei, and Paul Boustead. 2006. Managing Latency and Fairness in Networked Games. Communications of the ACM 49, 11, 46–51. https://doi.org/10.1145/1167838.1167861.

[20] Cheryl Savery, T. C. Nicholas Graham, and Carl Gutwin. 2010. The Human Factors of Consistency Maintenance in Multiplayer Computer Games. In Proceedings of the ACM International Conference on Supporting Group Work (GROUP '10). Association for Computing Machinery, New York, NY, USA, 187–196. https://doi.org/10.1145/1880071.1880103.

[21] A. Le and Y. E. Liu, Fairness in Multi-Player Online Games on Deadline-Based Networks, 2007 4th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, pp. 670-675, https://doi.org/10.1109/CCNC.2007.137.

[22] James Cannon, Saketh Dinasarapu, Ao Jiang, Hanzalah Qamar. Effects of Adaptive Time Delay in First Person Shooter Games, *Interactive Qualifying Project*, Worcester Polytechnic Institute. (Advisor Mark Claypool)

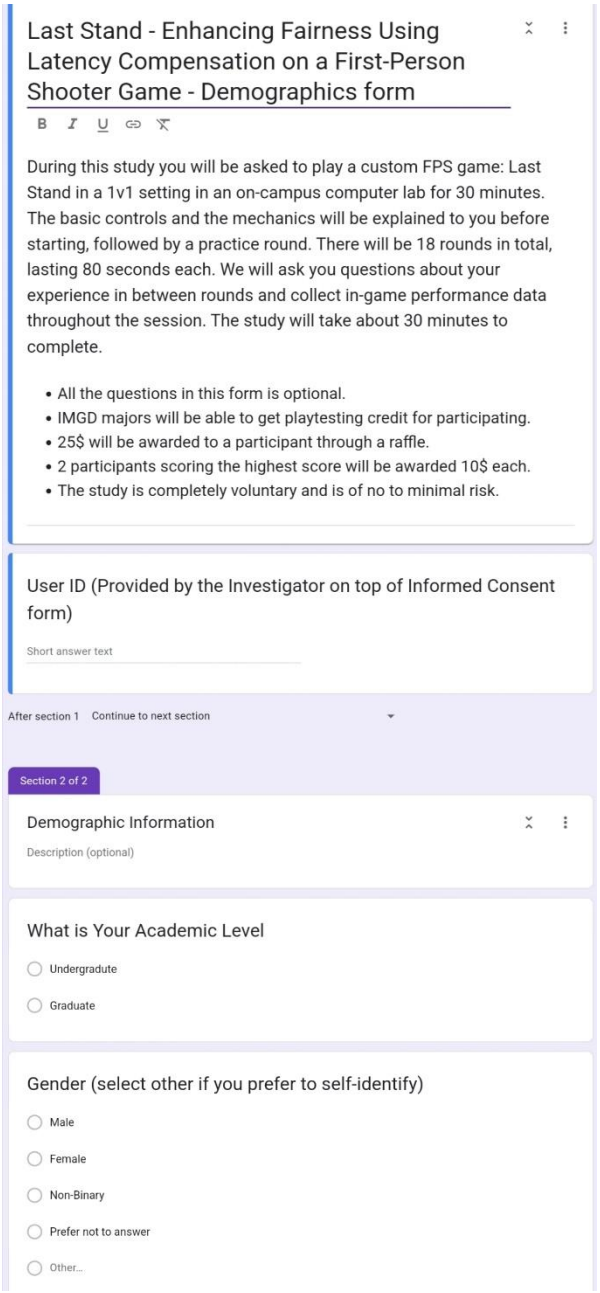
[23] Kinamation. FPS Animation Framework. Unity Asset Store. Last Accessed 29th March 2024. https://assetstore.unity.com/packages/tools/animation/fps-animation-framework.

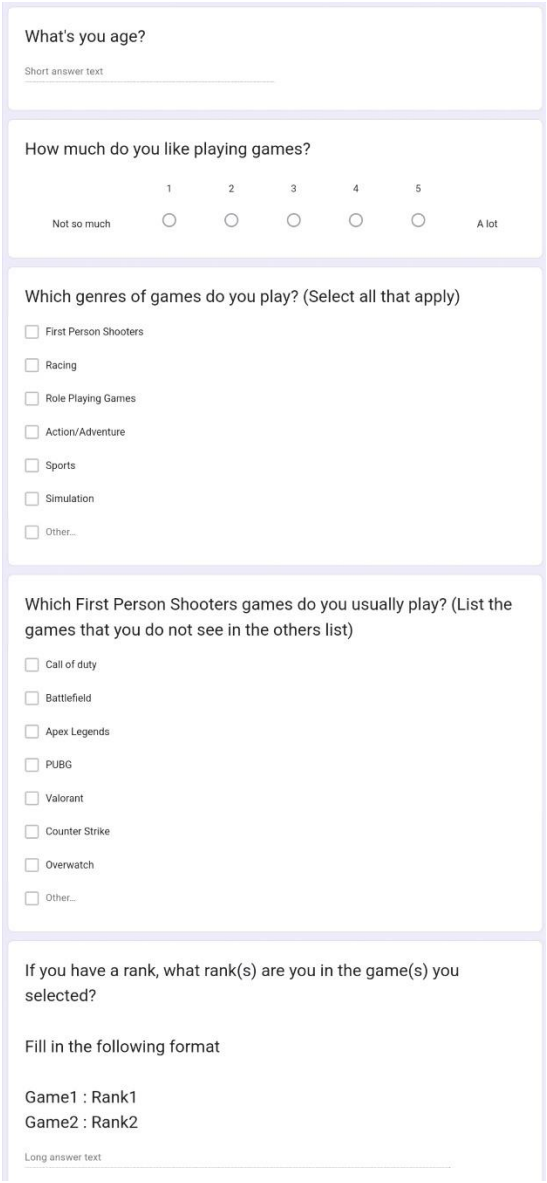
[24] Jiang, A., & Qamar, H. (2023, November). IQP testing video. YouTube. https://youtu.be/ouSVDV2TYQk?si=4b9GIHB-DZUeitqs.

[25] Gamers Decide. 2023. Top 7 FPS Esports Games That Are Popular. Last Accessed 10th October 2024. https://www.gamersdecide.com/articles/fps-esports-games

8. Appendices

8.1 Demographic Survey





8.2 IRB Approval Letter

WORCESTER POLYTECHNIC INSTITUTE

100 INSTITUTE ROAD, WORCESTER MA 01609 USA

Institutional Review Board

FWA #00030698 - HHS #00007374

Notification of IRB Approval

Date:	16-Feb-2024
PI:	Mark L Claypool
Protocol Number:	IRB-24-0432
Protocol Title:	Enhancing Fairness Using Adaptive Time-Delay on a First-Person Shooter Game
Approved Study Personnel:	Tokey, Samin Shahriar S~Claypool, Mark L~
Effective Date:	16-Feb-2024
Exemption Category:	3
Sponsor*:	

The WPI Institutional Review Board (IRB) has reviewed the materials submitted with regard to the above-mentioned protocol.We have determined that this research is exempt from further IRB review under 45 CFR § 46.104 (d). For a detailed description of the categories of exempt research, please refer to the IRB website.

The study is approved indefinitely unless terminated sooner (in writing) by yourself or the WPI IRB. Amendments or changes to the research that might alter this specific approval must be submitted to the WPI IRB for review and may require a full IRB application in order for the research to continue. You are also required to report any adverse events with regard to your study subjects or their data.

Changes to the research which might affect its exempt status must be submitted to the WPI IRB for review and approval before such changes are put into practice. A full IRB application may be required in order for the research to continue.

Please contact the IRB at irb@wpi.edu if you have any questions.

*if blank, the IRB has not reviewed any funding proposal for this protocol