The Effects of Latency and Jitter on a First Person Shooter: Team Fortress 2

An Interactive Qualifying Project Report

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements

for graduation

by:

Adam Myers

Joseph Blackman

Stephen Lafortune

Project Advisors:

Professor Mark Claypool

Professor David Finkel

Abstract

This paper describes the effects of latency and jitter on the enjoyment and performance of playing the online multiplayer first person shooter game Team Fortress 2. We ran a study to measure the performance of the player and the enjoyment of the game under various latency and jitter conditions. We analyzed data gathered from the game and questionnaires to find correlations for player performance or enjoyment at latency and high jitter versus just high latency. Our results suggest that higher latency makes a difference in gameplay, but there is no correlation for players performing better or worse with latency and jitter compared to just high latency.

Table of Contents

1 Introduction…………………………………………………………… 3
2 Background…………………………………………………………… 6
2.1 Basics of TF2……………………………………………….. 6
2.2 Latency and Jitter…………………………………………… 11
3 Related Work…………………………………………………………. 13
3.1 Previous Observations of Latency and Jitter……………….……. 13
4 Methodology…………………………………………………………. 16
4.1 General Setup………………………………………………. 16
4.2 Client Configuration………………………………………... 17
4.3 Server Configuration……………………………………….. 18
4.4 Setup of the Study………………………………………….. 19
4.5 Participants………………………………………………..... 21
4.6 Running the Sessions………………………………………. 21
4.7 Collecting Data……………………………………………... 22
5 Results………………………………………………………………... 23
5.1 Demographics………………………………………………. 23
5.2 Observations and Graphs - Game Data……………………..... 24
5.3 Observations and Graphs - Questionnaire Data………………. 30
6 Conclusion………………………………………………………….... 35
Works Cited…………………………………………………………. 37
Appendix A………………………………………………………….. 39
Appendix B………………………………………………………….. 40

1 Introduction

People enjoy playing video games together. Due to the widespread accessibility of high speed Internet, online multiplayer games are a popular medium for gamers. Online multiplayer games can be an enjoyable experience, however some of the time that experience can be degraded due to latency. Latency is the amount of time it takes for the input of an action to be received, and will cause the outcome of actions in online multiplayer games to be delayed. Jitter, a variation in latency, causes gameplay to feel inconsistent. In an optimal setting latency and jitter would not be observable and gamers could have a consistent experience. However, given the number of factors that can cause latency and jitter, such as one's Internet provider and whether he or she is using a wireless connection, this is not always the case.

Online multiplayer games can take different forms, from Real Time Strategy (RTS) games, to Role Playing Games (RPG), to First Person Shooter (FPS) games. The impact of latency and jitter differs from game to game, but it has been shown that latency and jitter may have different effects on gameplay (M. Claypool and K. Claypool, 2006). In a study on player perception of latency and jitter, subjects played a simple third person game in which they followed a line and jumped across platforms under different conditions of latency and jitter (Normoyle, 2014). It was determined that following the line was more affected by latency and jumping across platforms was more impacted by jitter. This study addressed the topic of latency and jitter across two different actions, however the range of conditions was too large and the actions too basic to determine when latency and when jitter start to have an effect.

One of those forms of online multiplayer games that latency and jitter has an effect on is the First Person Shooter. FPS games are games in which the character is controlled from a first-person perspective. First-person actions, such as aiming and shooting, are heavily influenced by latency and jitter (M. Claypool and K. Claypool, 2006). In a multiplayer environment, it is not very enjoyable to keep shooting and missing due to latency, and it can have a negative influence on the player's performance. Team Fortress 2 (TF2), an online multiplayer FPS game developed by Valve Software in 2007, is an FPS game with different classes and weapons. The variety of actions from the different classes create a study opportunity as their weapons each have unique properties. The rocket launcher which damages enemies in an area could be affected by latency and jitter differently than the shotgun which requires the player to directly hit the enemy to do any damage.

This report explores the effects of latency and jitter on the enjoyment and performance of playing TF2. It covers how TF2 is played and the important aspects of the game such as the different classes and how hit registration works. Chapter 2 goes into more detail about the causes and effects of latency and jitter. Works related to our study are explored in Chapter 3 to show what is already known about latency and jitter affecting online multiplayer games. Chapter 4 describes our study where we had three human participants play against the TF2 bots over seven rounds of different latency and jitter conditions. In between rounds, we had the players fill out a questionnaire asking them for their thoughts on their performance and feelings through questions on difficulty, enjoyability, and accuracy. We also collected data on player performance such as kills, deaths, accuracy, and damage taken. The results in Chapter 5 of our study showed no correlation of players performing better or worse with latency and high jitter compared to just high latency, but did suggest that higher latency makes a difference in gameplay. Chapter 6 follows up with possible explanations for our lack of any significant trends, and how future work can approach the topic of latency and jitter.

2 Background

This chapter describes the details of Team Fortress 2 (TF2) along with latency and jitter. The first section goes over the basics of TF2, describing its features and the game mode used for our study. The second section describes what latency and jitter are, how one can simulate latency and jitter with TF2, and how TF2 compensates for latency and jitter by default.

2.1 Basics of TF2

Team Fortress 2 is a First Person Shooter (FPS) in which players choose the Red or Blue team. They then fight in moderately large (6-12 players) teams on various maps, with varying objectives. One objective is Capture the Flag, where teams try to retrieve each other's flag and bring it to back to their base. Another objective is King of the Hill, where teams fight over control of a shared point. Other objectives include Control Point, where teams take control of each other's points, and Payload, where one team supports a moving objective.

When a player dies in TF2, he or she respawns after a certain amount of time. In TF2, respawning occurs in "waves" – players respawn in groups, every few seconds. This encourages team-based play, since it is easier to group up with teammates if you respawn with them. Every respawn, players are allowed to change among one of nine classes. Most classes have around the same health and movement speed, and differentiate with their weapons. For a detailed list of each class, its respective role, and its movement speed and health numbers, see Table 2.1. Each class serves different roles, and choosing the correct class composition for a team is an important part of the game's strategy.

Class	**Speed**	**Health**	**Primary role**
Scout	133%	125	Flanking, capturing objectives
Soldier	80%	200	Area denial with a Rocket Launcher
Pyro	100%	175	Area denial with a Flamethrower
Demoman	93%	175	Explosive traps, rolling grenades
Heavy	77%	300	Defensive line with a Minigun
Engineer	100%	125	Defensive turrets, health and ammo dispensers
Medic	107%	150	Healing, can build up an invulnerability charge
Sniper	100%	125	Long-range kills, can one-shot any class in the head
Spy	100%	125	Turns invisible, can one-shot with a stab to the back

Table 2.1: The classes of TF2 (Classes, 2015)

There are two kinds of attacks in TF2. Weapons which fire bullets are handled as "hitscan" – they scan in a straight line, then hit the first player or object in a line. These weapons include shotguns, sniper rifles, and pistols. The Scout, Heavy, Engineer, Sniper, and Spy have only hitscan weapons as displayed in Table 2.2. The second kind of weapon is a projectile, in which case the server creates a new entity for the projectile, tracks its progress, and takes action when it collides with a surface or player. Projectile weapons include rocket launchers, grenade launchers, and crossbows. Table 2.2 shows how only the Demoman, Soldier, Medic, and Pyro have at least one projectile weapon.

Class	**Weapon**	**Type**	**Weapon**	**Type**
Scout	Scattergun	Hitscan	Pistol	Hitscan
Soldier	Rocket Launcher	Projectile	Shotgun	Hitscan
Pyro	Flamethrower	Projectile	Shotgun	Hitscan
Demoman	Grenade Launcher	Projectile	Stickybomb Launcher	Projectile
Heavy	Minigun	Hitscan	Shotgun	Hitscan
Engineer	Shotgun	Hitscan	Pistol	Hitscan
Medic	Syringe Gun	Projectile	Medi Gun	Hitscan
Sniper	Sniper Rifle	Hitscan	Submachine Gun	Hitscan
Spy	Revolver	Hitscan		

Table 2.2: List of ranged weapons and their type for each class (List of Weapons, 2015)

The game mode our study is based around is called King of the Hill. In this mode both teams fight over control of a central point. To capture the point, players of one team, and only that team, must be standing on it. Once the point is captured, the capturing team's timer starts counting down from the set amount of time. If they lose control of the point, their timer stops, and the other team's timer starts. Once a team's timer reaches zero and no members of the opposing team are on the point that team wins. The team in control of the point has twice as long respawn times, giving the team not in control a better chance at capturing the point after killing members of the controlling team. The map we chose, Viaduct, was fairly simple, and had an easily-accessed high ground to serve as a defensive position when controlling the point as seen in Figure 2.1 by area labeled as the point. King of the Hill matches tend to be fast paced, with a simple objective of standing on a point and keeping enemies off that point.

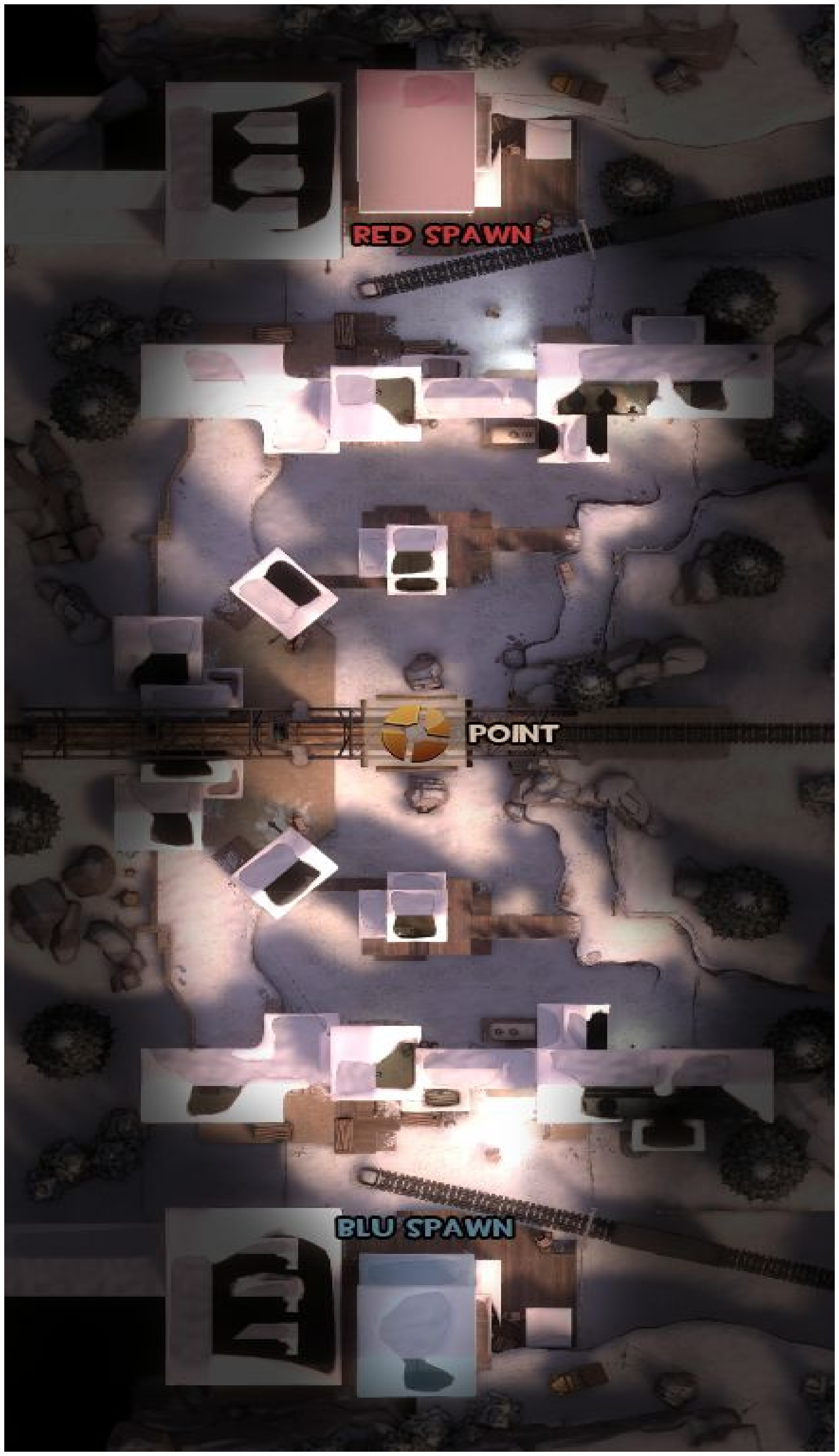

Figure 2.1: Top-down view of the map Viaduct with labeled spawns and capture point (Viaduct, 2016)

2.2 Latency and Jitter

In online multiplayer games units of data called packets contain useful information are routed between an origin and a destination. Latency is defined as the amount of time it takes one of these packets to travel from a source to a destination. Jitter is the range in network latency that is caused by changes in the state of the network (Kurose, 2009). Both latency and jitter can have a serious effect on online video games. When an online multiplayer game requires the player to perform an action that requires accuracy, a high precision action, jitter can result in the player performing the action inaccurately. An example of this would be a sniper trying to make a headshot. When the player has to perform an action in a short time period, a tight deadline action, latency can make the player miss the time window to succeed in the action. One example of this would be a player walking past a large fast moving fan. However, there are games that have many low precision, loose deadline actions such as Real Time Strategy games. Most first and third person online multiplayer games have some actions which require precision or have a tight deadline, so latency and jitter will cause them to be hard to perform (M. Claypool and K. Claypool, 2010).

A game server hosts the resources and information needed by the player's client to play with others. The location of this server, the Internet connection of the player, and the player's computer all can determine how much latency and jitter he or she may experience. In order to simulate playing Team Fortress 2 (TF2) with high latency and jitter, the network emulation tool NetEm (Linux Foundation, 2009) can add a fixed amount of delay to the outgoing packets of the server, resulting in a latency that is the base latency plus the added delay. Variation can also be added in the artificially induced latency, thereby simulating jitter. Given only a delay and variation, NetEm defaults to using a uniform distribution, which is not representative of typical network delay (Linux Foundation, 2009). NetEm can be run on the server machine or on a separate machine as a bridge, passing traffic between the server and the clients.

Much like other online multiplayer games, Team Fortress 2 sends data in packets. Since the server only sends data every so often, there is a question of what should the client display before it receives its next packet. The client can choose to extrapolate, or to assume that objects will continue moving in a straight line. This is risky, because if a player were to take a sharp turn, the client would have to correct quickly. TF2 takes a different approach: instead of rendering the most recent data from the server, the client instead displays old information (by default, 100 ms). This allows the client to interpolate, or to display object locations between two packets of data from the server, since the latest packet from the server is beyond what is currently being displayed (Valve Developer Community, 2014). The TF2 server also has a method for dealing with latency. Since bullets travel instantaneously in the game, it would make no sense to have to lead a target. In that spirit, when the server registers a client firing a shot, it calculates the client's latency, and moves all of the players back an equal amount of time. It then checks to see if the shot hits, and reports back to the client if it made a successful hit. This method is known as time warp, and only applies to the hitscan weapons in the game. (Valve Developer Community, 2014).

3 Related Work

This chapter describes research concerning online multiplayer games and makes comparisons to this project. It details what other studies have done in testing the effects of latency and jitter on online multiplayer games, and whether or not we followed suit and why.

3.1 Previous Observations of Latency and Jitter

One of the goals for our study was to see how latency and jitter changed the players' feelings while playing the game. The sensations of excitement, challenge, and boredom are some examples that are all part of the player experience in a game. Rahul Amin held a study that looked into the quality of the experience a player had when playing an online video game, Call of Duty: Modern Warfare 2. Their study sought to find out a subject's quality of experience by developing a mean opinion score (MOS) metric. The study ran multiple matches of the game under different latency and jitter conditions with one player competing against another player for the most kills in ten minutes. Afterward, surveys were given to the subjects asking questions about their experience, like how satisfied and how frustrated they were, on a scale from one to five. An MOS was then created from the average of these answers for each subject. It concluded that neither latency nor jitter had any significant effect on the MOS.

Measuring MOS can provide an overall idea of how the player's experience was affected by latency and jitter. The problem with measuring MOS was that it may assume each measure had equal importance. Due to this potential problem, we decided not to measure MOS from our results and instead looked at individual responses to questions about the gameplay.

In addition to experience, we wanted to evaluate the quality of the player's performance. Normoyle et al. used a platform based game to test a player's skill under various levels of latency with no jitter and various levels of jitter with a fixed latency. It found that only a latency as high as 500 milliseconds (ms) changed the player's performance significantly, while a 200 ms latency with 150 ms jitter changed the players' performance in the game even more. It also found that a player's ability to control his or her character improved with practice at high latencies with no jitter, while this did not happen when playing under any amount of jitter.

This study brought up our main reason for evaluating player performance. Its results suggested that latency with no jitter in games hardly affected player performance until reaching an extreme like 500 ms. In contrast, a low latency with jitter affected player performance even more than a high latency like 500 ms. We ran our test sessions in Team Fortress 2 (TF2) under latencies with no jitter and under latencies with jitter to investigate whether it is better for players to play at a high constant latency rather than an moderate latency that has jitter.

A study that used Quake 3 (Bredel and Fidler, 2010) got different results from the platform game study, where high latency proved more significant in the performance of bots while a lower latency with jitter proved insignificant on the performance of bots. Another study which used TF2 (Finn and Dwyr, 2013) found insignificant effects on player performance from both high latencies with no jitter and lower latencies with jitter. We could not find a way to apply latency and jitter to the bots in TF2, and for that reason did not run tests with bots versus bots.

Our tests involved players versus bots so we could get more data through questionnaires like Finn and Dwyr's Team Fortress 2 study. Their study had eight players per trial and all nine classes available for participants to play, while ours had three players per trial and only three classes: Soldier, Demoman, and Scout. We limited our players per trial and class options to help prevent team size and class choices from masking the association we wanted to find between latency and jitter and player performance and experience. If there are a large number of players on a team, it is more likely that one player's performance is good or bad not because of latency or jitter, but because of the other team members. If a few players are very good, said player would have fewer deaths than expected because those players killed all the bots. While if a few players are very bad, said player would have more deaths because he or she has to face multiple bots at once since his or her allies were killed quickly or are unable to get kills. And the other classes available, like Engineer and Medic, are not the best evaluators of performance. The Engineer primarily uses a turret that automatically targets and kills for him and is not affected by latency or jitter because the turret is handled by the server. The Medic just heals allies and rarely goes to kill enemies. Soldier, Demoman, and Scout all require the player to actively move, aim, and shoot at targets and are therefore much better evaluators of performance.

4 Methodology

Our user study of Team Fortress 2 (TF2) was conducted under various network conditions of latency and jitter. As an overview, we set up and configured one computer to run the game server, and had three other computers each run the game client. The network conditions were added using the network emulation tool NetEm, which was run on the server computer. The clients were set up with a file containing the game configuration and placed on the computers used for the study. Once we determined the setup was sufficient we ran two test studies to confirm the settings and make and tweaks. Then we recruited volunteers, scheduled play sessions, and ran our study. This section provides details on this methodology.

4.1 General Setup

We used a computer lab on campus, called the Zoo Lab, which had divided rooms of computers. Each computer had the same specifications, keyboard and mouse. The test subjects used the computers in the lab as the clients, and one of the computers on the local network acted as a host of the TF2 server in which the clients connected to in order to play. The study was conducted across multiple TF2 updates starting with the update on November 18th, 2015, and finishing with the update on December 7th, 2015. These updates generally contained bug fixes, none of which impacted the study. In order to delay the packets, NetEm (Linux Foundation, 2009) was run on the same machine that was hosting the TF2 server. The specifications of the client computers were:

- 24” 1920 x 1200 Monitor
- An Intel Core i7-4790 processor running at 4.0 GHz
- 16 GB of RAM

- A dedicated GeForce GTX 960 graphics card, with 4017 MB of video memory
- Windows 7 64-bit

The specifications of the computer hosting the server was:

- An Intel Core i7-3770 running at 3.40 GHz
- 12 GB of RAM
- A dedicated AMD Radeon HD 7700 graphics card, with 3794 MB of video memory
- Ubuntu Server 14.04

These surpass the recommended system requirements for TF2 which are (Steam Store, 2016):

- A Pentium 4 processor (3.0GHz, or better)
- 1GB of RAM
- A DirectX 9 level Graphics Card
- Windows 7 (32/64-bit)

4.2 Client Configuration

The Team Fortress 2 (TF2) client was configured with specific settings to provide a controlled game environment for our study. Teams were preconfigured so that only three classes were made available for selection, each player choosing a different class. Since the Zoo Lab does not provide headphones, we disabled the in-game audio and enabled a setting which makes damage dealt appear over enemy players. We also disabled random critical hits and random damage spread for the sake of consistency. There were a few other smaller settings that we changed in order to make playing more straightforward for the participant. Fast weapon switching was enabled to make it easier for the player to switch weapons by reducing the number of key presses needed. We also turned on automatic reloading, a feature that causes weapons to automatically reload when out of ammo and lacks any disadvantage to manually reloading. In-game popups, in-game chat, and bots taunting after kills were turned off because they were unnecessary distractions. We configured the game's launch options so that it skipped the intro video and clients automatically connected to our server.

4.3 Server Configuration

The TF2 server incorporated the various network conditions through the use of NetEm (See Section 2.2). When adding latency and jitter, NetEm set a constant amount of delay, and then added or subtracted a random amount following a normal distribution. For example, when we wanted to have a range of latency from 80 to 100 milliseconds (ms), Netem would be set to 90 ms latency and 10 ms jitter. The queue of packets was set to prevent packet reordering caused by jitter, and high-resolution timers were turned on in the Linux kernel to give NetEm better granularity with delaying packets (Linux Foundation, 2009).

We were not able to get all the game statistics we needed through the TF2 itself, so we installed and used a modding platform called SourceMod. SourceMod was used for two plugins, one was SupStats2 (MedicStats Sourcemod Plugin, 2015) which provided additional logging information, such as health pickups, accuracy, and more precise damage values. The other was a purpose-built plugin which provided a number of features, including disabling players' melee weapons, pausing the game after each round, and limiting which classes were selectable by each player.

There are also several settings related to lag compensation that we disabled. Interpolation, in which the clients receive older information as opposed to the most recent information from the server, was turned off. Time warp, a server-side lag compensation method that looks through the previous server states to see what should be happening the exact moment a player took an action, was also turned off. We elected to leave weapon prediction and movement prediction enabled. These two effects mean that when a player clicks or pushes a movement key, they see an immediate shot being fired or themselves moving, even if the server has not yet received the information of it happening. If the server reports a conflict, such as a player moving into a wall, the client will update to match the server's version of events. However, when we tested the game with movement prediction turned off, we found the game to be hard to play at 100 ms (milliseconds) latency and unplayable at 200 ms latency, so we left these settings on. By turning off the lag compensation settings that we could we were able to focus on the different generated latency and jitter conditions.

4.4 Setup of the Study

Game settings for the test were chosen based on multiple factors. We selected teams of three since we felt that was the smallest size that still provided active gameplay. In addition, when one player died it did not mean the team lost control of the control point, there were usually fights still going on with the other two players. Team Fortress 2 (TF2) has over 220 weapons (Team Fortress 2 wiki, 2016) that can completely change how a class plays. We therefore chose to use only the default primary and secondary weapon for each class, leaving the melee weapon out. The three classes we chose were Scout, Soldier, and Demoman, since they were the most straightforward to play. These classes had a balanced distribution of the two weapon types, hitscan and projectile. Hitscan weapons scan in a line for the first surface or player and register that as a hit, while projectile weapons create an entity on the server when fired and track when it collides with a player or surface. The Scout having two hitscan weapons, the Soldier having a hitscan weapon and a projectile weapon, and the Demoman having two projectile weapons created a balanced total of three of each weapon type. Based on these factors and our playtesting, we chose the game mode King of the Hill and the map Viaduct. To win King of the Hill, a team needed to hold the objective for a total of three minutes. When a team stands on a point that the enemy controls they start to capture it, and once fully captured the timer starts to tick down. Because teams only had to hold the point for three minutes, rounds rarely exceeded six minutes. The map selected was small and had fast respawn times to encourage players to fight more frequently. It is also relatively simple, with few side-routes or niches. The top down view of the map can be seen in Figure 2.1.

Each session had three volunteers on the same team play a cooperative match against bots. Bots in TF2 have four difficulty levels, easy, normal, hard, and expert. When we ran our pilot study we initially set the difficulty level to hard, however the volunteers struggled to compete and as a result we lowered the difficulty to normal. Both teams had exactly one Scout, one Demoman, and one Soldier. This decision was made because of the strengths and weaknesses each class has, and resulted in more varied fights.

Our network conditions for each round were picked at random from our set of conditions, seen in Table 4.1. These values were chosen based on our own testing and what previous studies had covered. For example, a previous study found that values over 300 milliseconds caused a severe drop in player performance (Normoyle, 2014). In our study we decided to be more conservative with latency. For our jitter values, we picked values that caused gameplay to become less smooth, however following our conservative latency values we made sure the jitter values were not overbearing.

Max Latency (ms)	0	100	200	100	200	100	200
Jitter (ms)	0	0	0	20	20	40	40
Resulting Range of Latency (ms)	0	100	200	80-100	180-200	60-100	160-200

Table 4.1: Latency and Jitter Combinations

4.5 Participants

Volunteers of the study consisted of the student population at Worcester Polytechnic Institute, recruited through email and WPI's Social Sciences Participant Pool. The participant pool contained students in psychology courses who were able to participate in our study to receive credit.

4.6 Running the Sessions

At the beginning of the study the participants filled out a pre-survey to provide demographics info, as well as indicating how often they play different categories of video games (See Figure 4.2). Steam is required to play Team Fortress 2 (TF2), so we provided one of three steam accounts, each created specifically for the study, to each participant in order to play. We read through a script that explained the game mode, the controls, the classes available, and the location of health and ammo pickups to the volunteers (See Appendix A). They then played seven rounds of TF2, each set to a different latency and jitter condition (See Table 4.1). After each round participants were asked to give feedback by answering a questionnaire (See Appendix B). Once the questionnaires were completed we collected them, set new latency and jitter values, and then started the next round.

4.7 Collecting Data

In addition to the questionnaires filled out during the study, we collected data through the Team Fortress 2 log files on how well the players performed. Matches were also recorded using the built-in game “demo” system and can be played back if needed for reference. We gathered our data by writing down the questionnaire responses on a spreadsheet, and writing a script which analyzed all of the log files. After analyzing the log files the script matched the participant number to the class he or she was playing, as well as matched the rounds to the randomly selected latency and jitter. The script then generated various graphs based on the data. Using these graphs, we then examined the question on whether it is better to play with low latency and jitter or high latency and no jitter.

5 Results

This chapter analyzes the demographics of those who participated in our study, and the data that we collected during the study that was the closest to being significant.

5.1 Demographics

In total there were 42 people who participated in the study. 31% of participants identified as female, while the other 69% identified as male. The information on the experience and age of the participants is shown in Figure 5.1. 21.4% of the participants had played Team Fortress 2 (TF2) before. 50% of the participants had never played TF2, but had played another First Person Shooter (FPS), 16.7% had never played a FPS, but had played another video game, and 11.9% had never played video games before. The age ranged from 18 to 24.

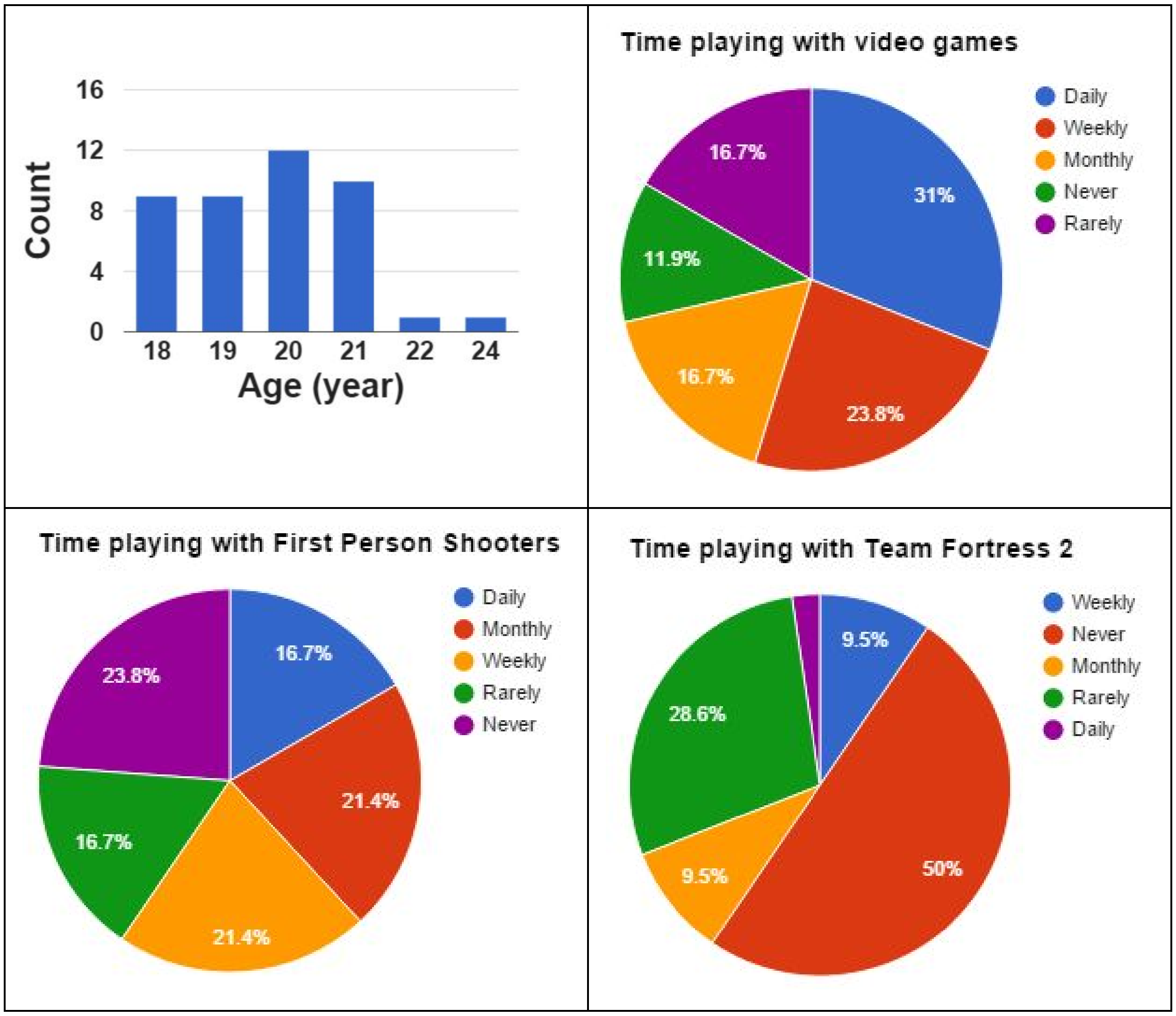

Figure 5.1: Age and Experience Distributions of Participants

5.2 Observations and Graphs - Game Data

One noticeable correlation in our data can be seen in Figure 5.2, in a box and whisker plot. A box and whisker plot is a concise way of visualizing the spread of the data. The black lines represent the maximum and minimum values, while the red line represents the median value. The blue box contains all values that are in the middle 50%, as defined by medians. The top blue line is the median of all the values between the red line and the top black line, and the bottom blue line is the median of all values between the red line and the bottom black line.

The median (red line) for kills is overall increasing with each round, but the range (black lines) and the interquartile range (blue box) generally vary. Players' kills did not trend with the latency or jitter in a round. As can be seen in Figure 5.2, players gained more kills as the rounds progressed, so they were improving.

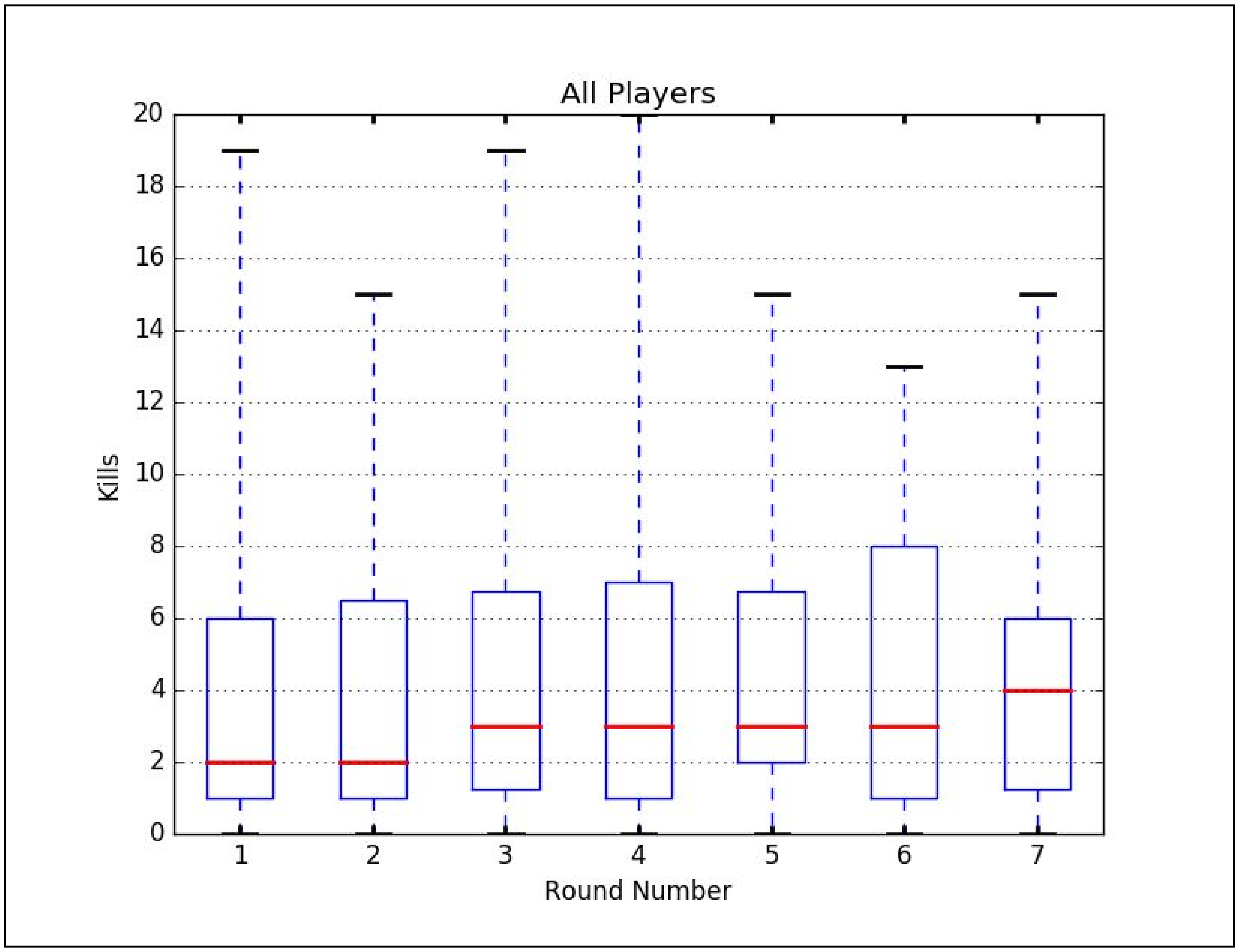

Figure 5.2: Kills per Round for All Players

Accuracy describes the percentage of times a weapon was fired and hit an enemy. We graphed accuracy at 100 ms (milliseconds) latency and varying jitter values, and separated out each player skill group. We chose 100 ms over 200 ms because the graphs were more suggestive.

As can be seen in Figure 5.3, players who were more experienced with the game performed more consistently, but were on occasion less accurate than less experienced players. That being said, there is no clear trend with players getting more or less accurate as jitter increases.

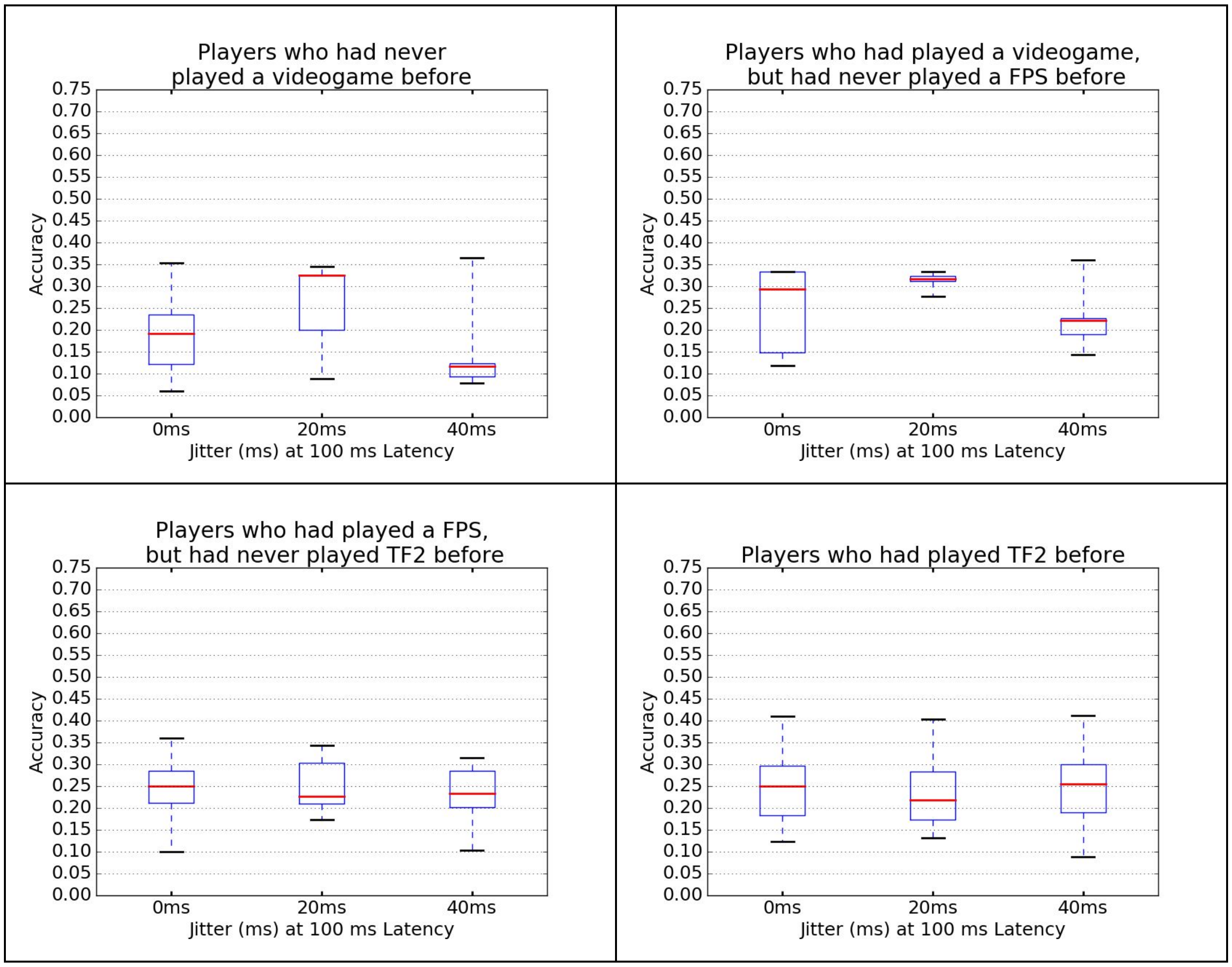

Figure 5.3: Accuracy at 100 ms latency and a variety of jitter values. For each graph the box and whisker plot on the left represents 100 milliseconds of latency and 0 milliseconds of jitter, or high latency and no jitter. The plot on the right represents 80 milliseconds of latency and 40 milliseconds of jitter, or moderate latency and high jitter.

As can be seen in Figure 5.4, players with more experience in FPS games, especially Team Fortress 2 (TF2), dealt more and took less damage. This means those players were more effective, since they are damaging the enemy team more than the enemy team is damaging them.

This metric is more descriptive than analyzing kills and deaths, since often an enemy will die due to damage from multiple sources. There is still no significant trend as jitter increases.

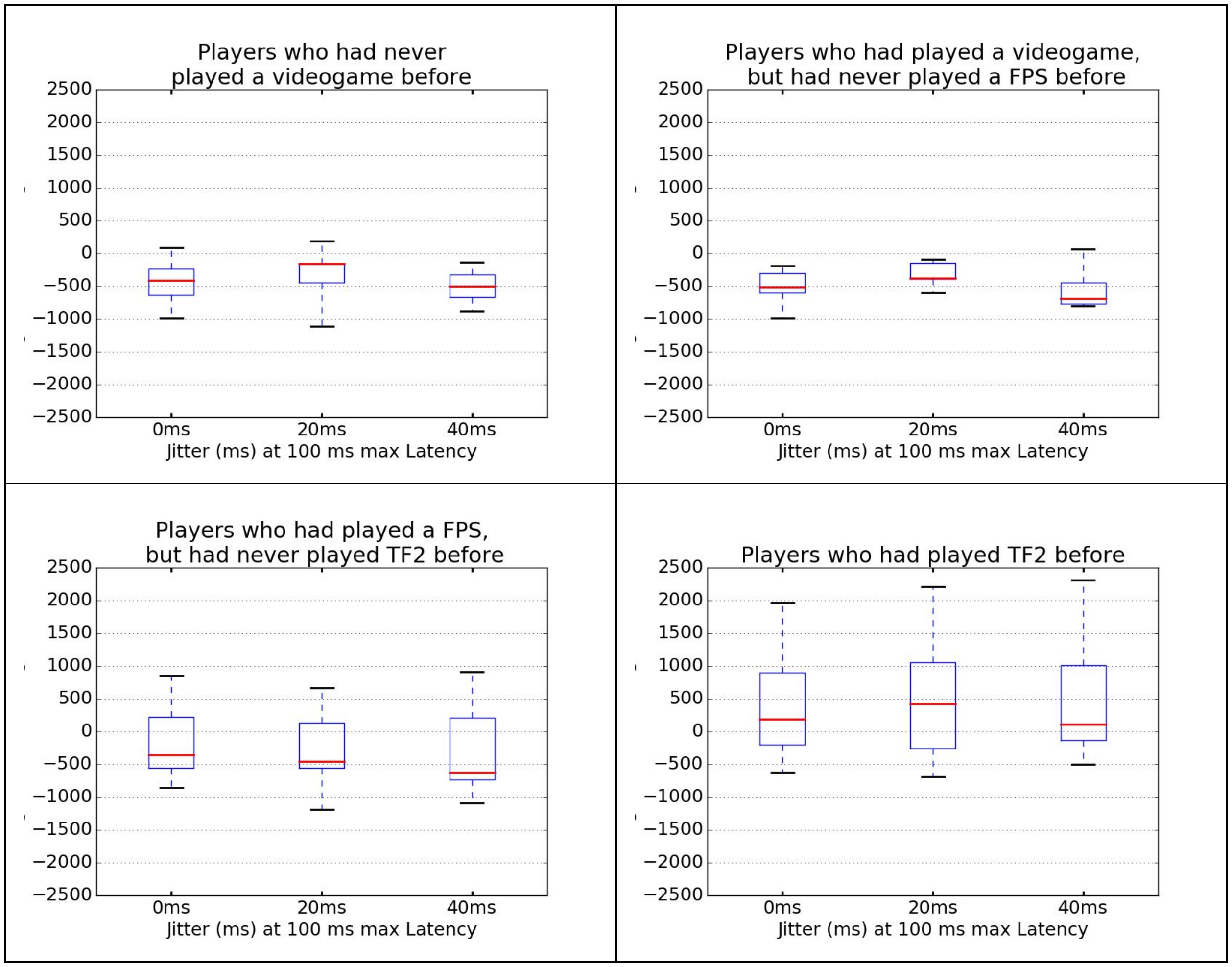

Figure 5.4: Damage Dealt minus Damage Taken at 100 ms maximum latency and a variety of jitter values. For each graph the box and whisker plot on the left represents 100 milliseconds of latency and 0 milliseconds of jitter, or high latency and no jitter. The plot on the right represents 80 milliseconds of latency and 40 milliseconds of jitter, or moderate latency and high jitter.

There are two kinds of weapons in Team Fortress 2. Projectile weapons, like the Rocket Launcher, fire projectiles. When these projectiles collide with a player, a wall, or the ground, they explode, and deal damage to any nearby enemies. These projectiles are handled on the server, and their movement is reported to the client. However, this projectile is not created until the request from the client reaches the server. The amount of time between a player clicking their mouse and the projectile appearing on screen increases with latency.

The second type of weapon is a hitscan weapon, like the Shotgun. These weapons deal damage instantaneously after they fire, although the time between a player clicking and the damage being dealt is still affected by latency. Clients will “predict” the shot being fired, and display to the user the animation of the shot firing before it happens. This helps to smooth out player experience and make their weapon feel more responsive.

Figure 5.5 and 5.6 show the different amounts of kills with projectile and bullet weapons respectively. More experienced players tended to get more kills, but it is also quite noticeable that more kills were made with projectile weapons than with bullet weapons. This may be in part due to the fact that the Demoman, who only has projectile weapons, consistently outperformed the other two classes.

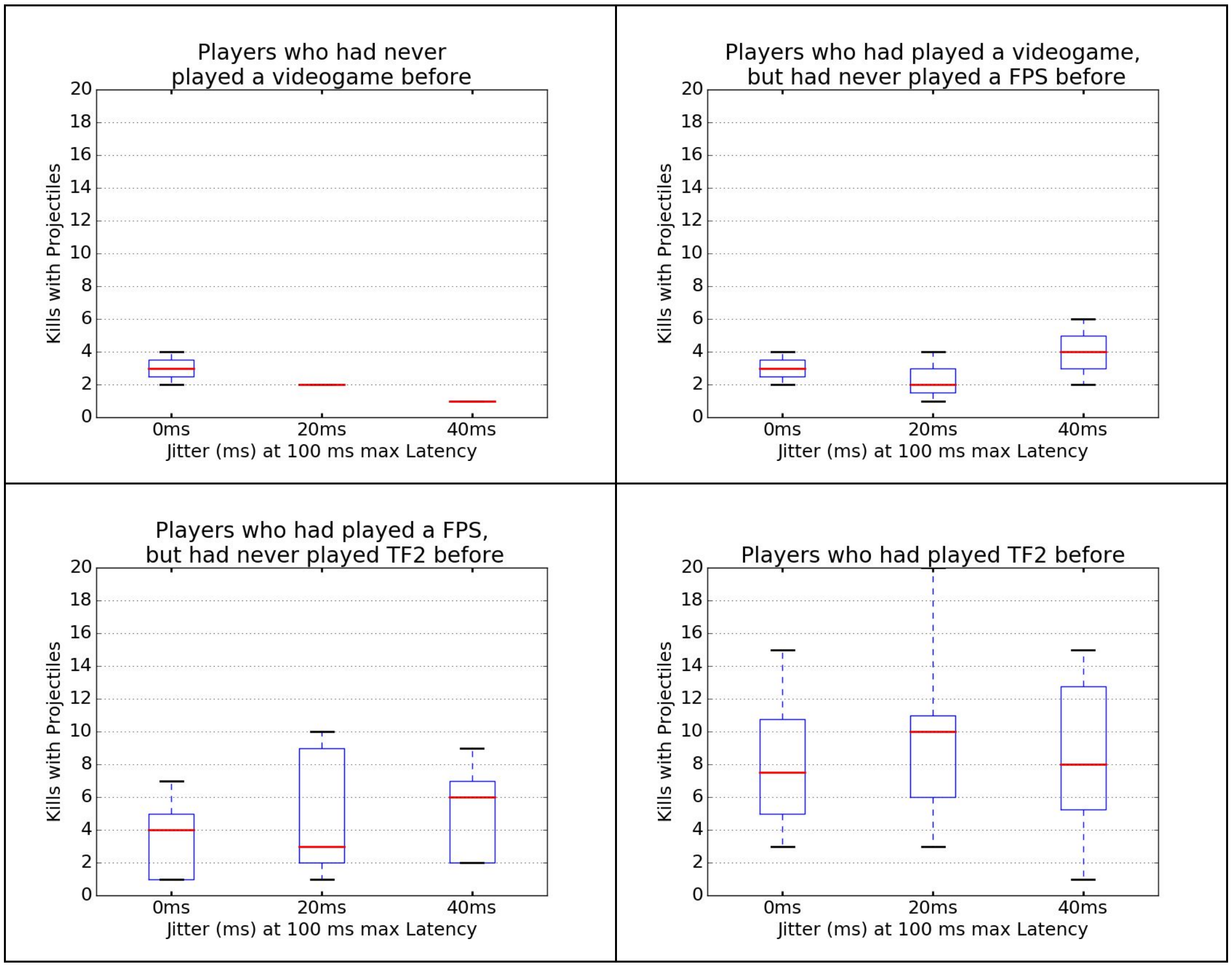

Figure 5.5: Projectile weapon kills. Kill counts are plotted at 100ms maximum latency and a variety of jitter values. For each graph the box and whisker plot on the left represents 100 milliseconds of latency and 0 milliseconds of jitter, or high latency and no jitter. The plot on the right represents 80 milliseconds of latency and 40 milliseconds of jitter, or moderate latency and high jitter.

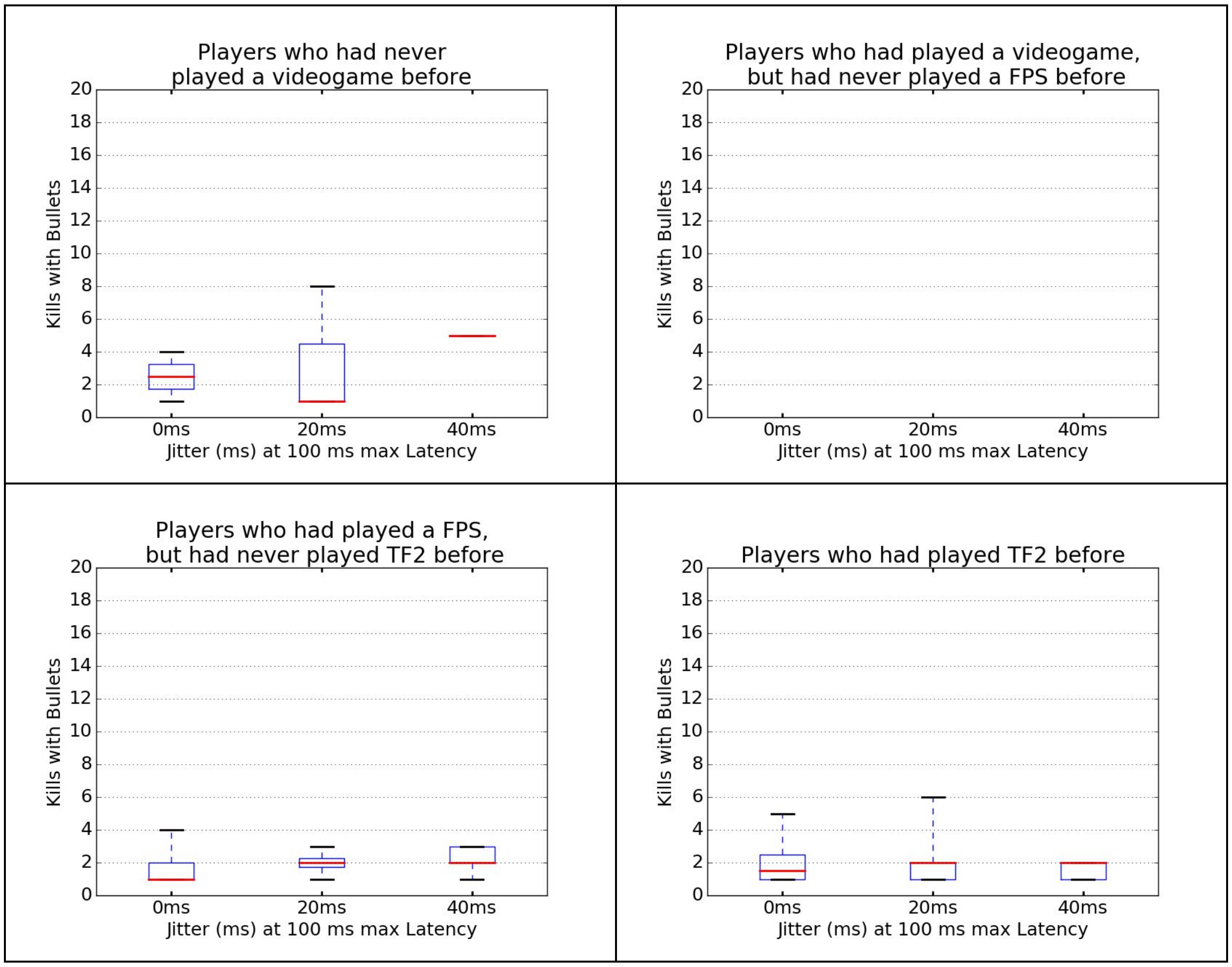

Figure 5.6: Bullet weapon kills. Kill counts are plotted at 100 ms maximum latency and a variety of jitter values. For each graph the box and whisker plot on the left represents 100 milliseconds of latency and 0 milliseconds of jitter, or high latency and no jitter. The plot on the right represents 80 milliseconds of latency and 40 milliseconds of jitter, or moderate latency and high jitter. Note that the Kills with Bullets for Gamers graph is empty because no players in that category ever played Scout, and the few that played Soldier never got a kill with the Shotgun.

5.3 Observations and Graphs - Questionnaire Data

Participants also filled out a survey at the end of each round. One question on this survey was “How challenging was it to play this round?” The responses to these questions can be found in Figure 5.7. The data is plotted in a Probability Distribution Function, where the height of the graph at a given response is the percentage of players who gave that response. In general, players reported the rounds as being more challenging as latency increased. Unsurprisingly, players with more experience with TF2 or FPS games in general tended to rate the rounds as less challenging.

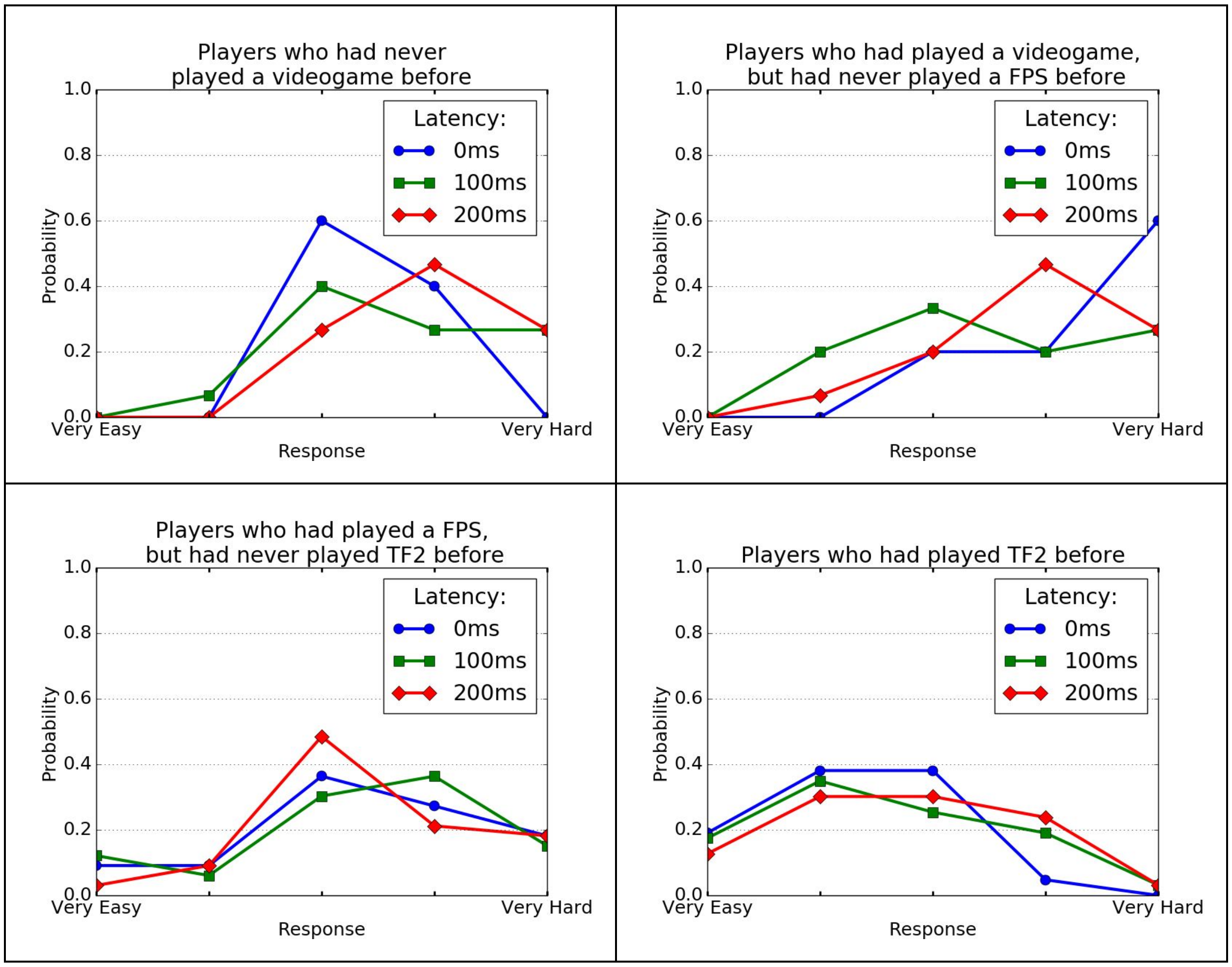

Figure 5.7: Probability Distribution Function for participant reports of challenge, plotted against latency.

Similarly, players were asked how difficult was it to aim each round. When jitter increases, enemy movements appear less smooth, since the client's connection to the server is not consistent. It is therefore expected that players would feel less accurate accuracy when jitter increases. As can be seen in Figure 5.8, there is no such correlation.

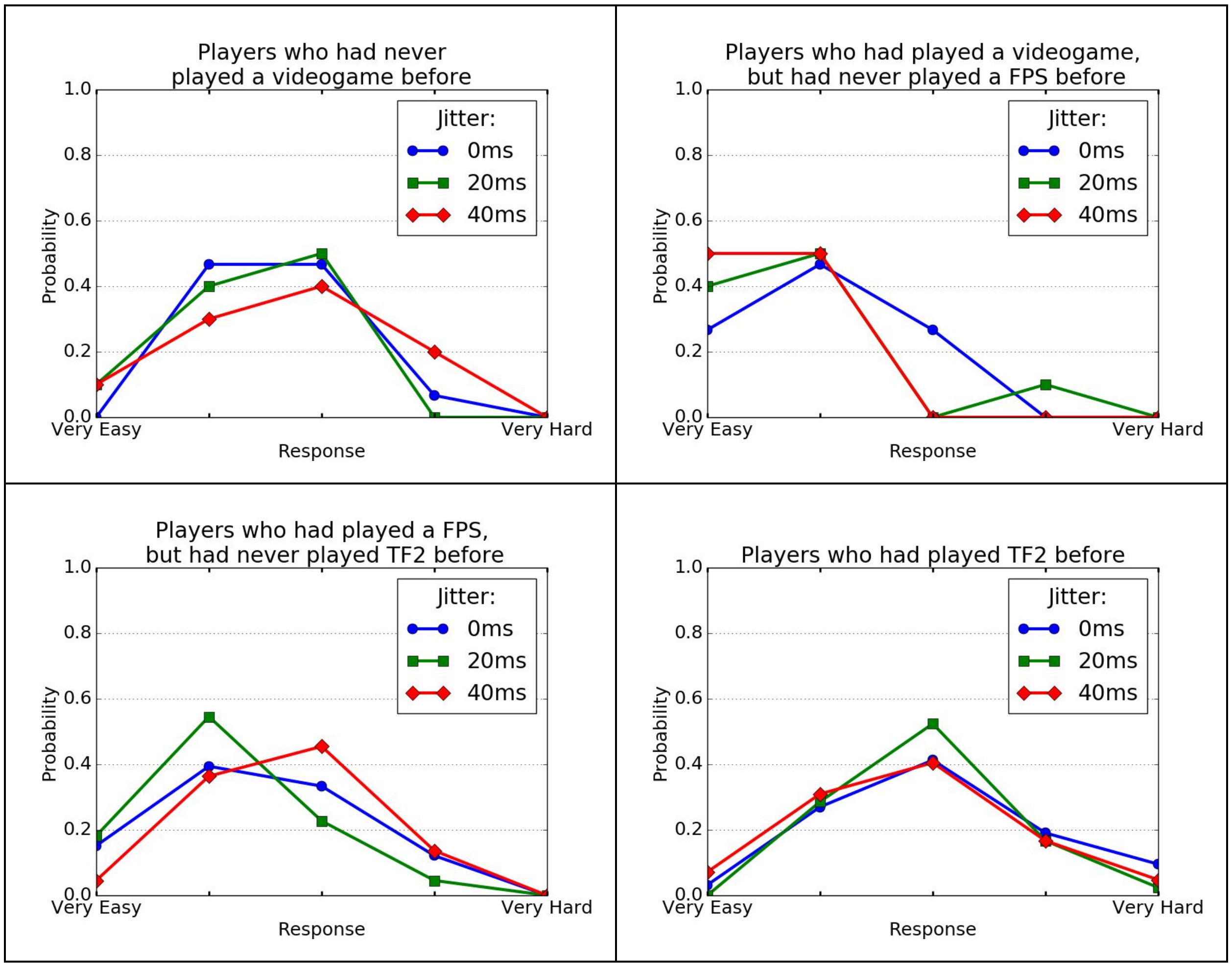

Figure 5.8: Probability Distribution Function for participant reports of Difficulty Aiming, plotted against jitter.

Players were also asked to rate how accurate they felt their shots were each round (Figure 5.9). In this graph, there is a suggestive trend between increased jitter and decreased accuracy, and once again, players with more experience reported overall that they felt more accurate.

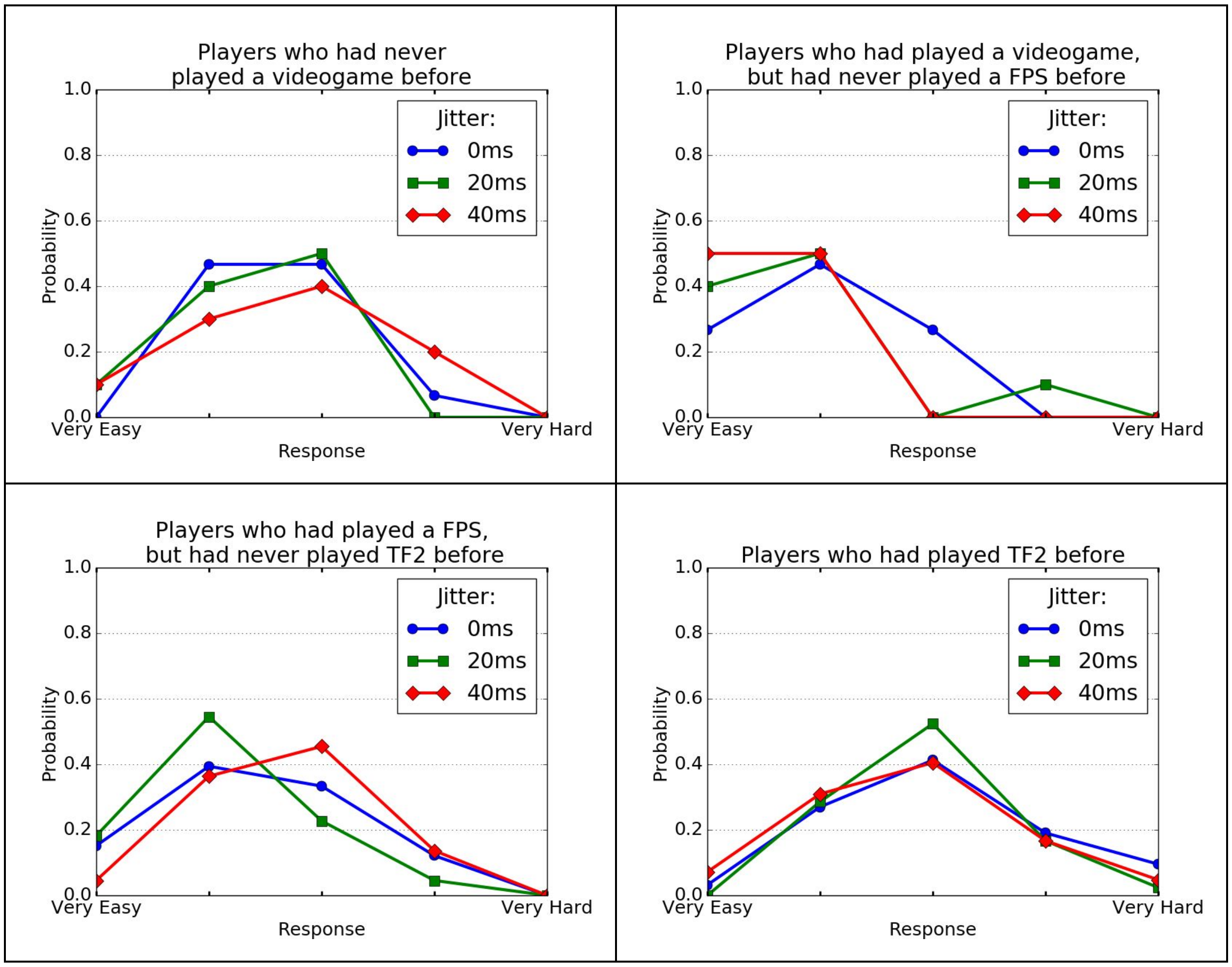

Figure 5.9: Probability Distribution Function for participant reports of accuracy.

When looking at high latency and no jitter and a moderate latency and high jitter together, no questionnaire graphs showed any suggestive trends or correlations.

6 Conclusion

Latency is the delay of information between the client and the server, and jitter is the range of variation in that delay. Our study analyzed whether it is preferable to have a high latency and no jitter, or a moderate latency and high jitter. We chose to study this through Team Fortress 2, a First Person Shooter game which has a variety of classes, each with a variety of weapons. We selected three classes which would be most impacted by increased latency and jitter, and conducted play sessions with three participants against three bots. The map and game mode were selected for simplicity and speed, and participants played seven rounds at a variety of latency and jitter values.

In the end, our study failed to see any significant trends. There are a number of potential reasons for this, but one of the key concerns is player skill. Certain players had to be taught the controls, and these players had a clear learning curve affecting their play. Many of them failed to leave the starting area for the first round, and would often move and then aim, which made them an easy target for the bots while they were aiming. An interesting line of future research would be to only study players with previous First Person Shooter game experience. These players got similar numbers of kills under the same latency and jitter values, and were also more willing to talk to and help their teammates.

Another choice we made was to run this study with groups of three. This helped keep players interested, so they maintained a consistent level of performance throughout the study. However, it meant that a player who was struggling could be helped by a teammate, who might give them tips which would change their performance during the study. A future study would likely benefit from running one-on-one games with only one class. The best choice for a single class study would be either the Scout or the Soldier. The Scout plays very quickly and would see strong effects from increased jitter, since the enemy movement would feel less smooth. The Soldier's rocket launcher deals damage over an area, and would see less of an effect from jitter. However, it would see a stronger effect from latency, since the firing of the rockets is handled by the server.

Another interesting line of research would be to try a different game mode. We selected King of the Hill, which was reasonably fast paced. Participants played seven rounds, each of which took three to six minutes, but each study ended up taking around 50 minutes, due to setup and teardown time. Another possibility is a custom game mode called MGE Mod, which features an even smaller map with a zero second respawn time, would increase the combat speed, reduce the learning curve effect, and help keep the study time low.

Latency and jitter are serious problems in online games. Light takes 66 milliseconds to reach halfway around the world, so it is not possible to remove latency entirely. Jitter, on the other hand, is often caused by inconsistent behavior of individual routers. This study looked into the possibility of countering jitter by increasing the latency and slowing down all received packets to a fixed response time. While this study did not find any significant trends in the data, participants' questionnaire responses suggest that players perceived themselves playing worse when jitter was increased. If players are willing to suffer increased latency to reduce this jitter, then the game itself can delay information coming from the server to reach a constant latency, and allow players a much smoother experience with no jitter.

Works Cited

Amin, Rahul et al. (2013). Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2. *Proceedings of the 15th international conference on Human-Computer Interaction: users and contexts of use* (pp. 97-106). Berlin, Heidelberg: Springer-Verlag

Bredel, Michael, and Fidler, Markus. (2010). A Measurement Study regarding Quality of Service and Its Impact on Multiplayer Games. *Proceedings of the 9th Annual Workshop on Network and Systems Support for Games*. Piscataway, NJ: IEEE.

Classes. (n.d.). Retrieved October 10, 2015, from wiki.teamfortress.com: https://wiki.teamfortress.com/wiki/Classes

Claypool, Mark and Claypool, Kajal. (2006). Latency and player actions in online games. *Communications of the ACM - Entertainment networking* (pp 40-45). New York, NY: ACM

Claypool, Mark and Claypool, Kajal. (2010). Latency Can Kill: Precision and Deadline in Online Games. *Proceedings of the first annual ACM SIGMM conference on Multimedia systems* (pp 215-222). New York, NY: ACM.

Finn, Eric and Dwyer, David. (2013). *Predicting the Perceived Quality of a First Person Shooter Game: The Team Fortress 2 T-Model*.

Jurgelionis, Audrius, et al. (2011). An Empirical Study of NetEm Network Emulation Functionalities. *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference* (pp 1-6). Maui, HI: IEEE.

Kurose, James F. (2009). *Computer Networking: A Top-Down Approach*. Boston, MA: Pearson. Print.

Linux Foundation. (2009, November 19). *Netem*. Retrieved October 3, 2015, from Linux Foundation: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

List of Weapons. (n.d.). Retrieved October 10, 2015 from wiki.teamfortress.com: https://wiki.teamfortress.com/wiki/List_of_weapons

MedicStats Sourcemod Plugin. (n.d.). Retrieved October 7, 2015, from Teamfortress.tv: http://www.teamfortress.tv/13598

Normoyle, Aline et al. (2014). Player perception of delays and jitter in character responsiveness. *Proceedings of the ACM Symposium on Applied Perception* (pp. 117-124). New York, NY: ACM.

Team Fortress 2. (n.d.). Retrieved October 9, 2015, from Steam Store: http://store.steampowered.com/app/440

Source Multiplayer Networking. (n.d.). Retrieved October 10, 2015, from Valve Developer Community: https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking

Viaduct. (n.d.). Retrieved February 28, 2016, from wiki.teamfortress.com: https://wiki.teamfortress.com/wiki/Viaduct

Appendix A

Debriefing Script

Thank you for participating in this study, The Effects of Latency and Jitter on Team Fortress 2. In online multiplayer games, data containing useful information are routed between an origin and a destination. These units of data are called packets, and the time it takes from them to get from server to client is called latency. Jitter is the range in latencies. The purpose of this study is to see the effects latency and jitter have on user performance and enjoyment of first person shooter games, like Team Fortress 2. Our hypothesis is that user performance and enjoyment will be higher at high latencies with no jitter compared to lower latencies with jitter. This study is important because if our hypothesis is correct, a method of implementing a constant higher latency could be used to fix these user performance and enjoyment issues in games like Team Fortress 2. In this study we asked participants to play seven rounds of Team Fortress 2 against a team of bots and fill out questionnaires each round asking about their performance and enjoyment that round. Each round we changed the latency and jitter in a random order so that the participants could not predict the treatments. We will use the data from the questionnaires and the data the server collects from the gameplay sessions to test whether or not our hypothesis is correct. Previous research has shown no significant results for Team Fortress 2 and similar online multiplayer first person shooter games.

Again, thank you for your participation in our research. Please do not divulge the purpose of this experiment to other people as this is an on-going experiment! If you want to see some of our references, or have any questions and comments, you can ask me now or you can contact me at aemyers@wpi.edu at a later date. You may also contact Joseph Blackman at jctblackman@wpi.edu or Stephen Lafortune at srlafortune@wpi.edu. If you would like to receive a copy of the results we can email them to you at the end of the study.

Appendix B

TF2 IQP Questionnaire

Pre-Survey:

Age: ______

Gender: __________

How often do you play videogames in general? (circle one)

Never	Rarely	Monthly	Weekly	Daily

How often do you play First-Person Shooters? (circle one)

Never	Rarely	Monthly	Weekly	Daily

How often do you play Team Fortress 2? (circle one)

Never	Rarely	Monthly	Weekly	Daily

Round 1:

Which class was the most difficult to fight against this round? (circle one)

Scout	Demoman	Soldier	I don't know

How challenging was it to play this round? (circle one)

Very easy				Very hard
1	2	3	4	5

How enjoyable was it to play this round? (circle one)

Not very enjoyable				Very enjoyable
1	2	3	4	5

How difficult was it to move in this round? (circle one)

Very easy				Very hard
1	2	3	4	5

How difficult was it to aim in this round? (circle one)

Very easy				Very hard
1	2	3	4	5

How accurate were your shots in this round? (circle one)

Very inaccurate				Very accurate
1	2	3	4	5

State how much you agree or disagree with the following statements.

I felt tired playing: (circle one)

Strongly Agree				Strongly Disagree
1	2	3	4	5

I got bored: (circle one)

Strongly Agree				Strongly Disagree
1	2	3	4	5

I really got into the game: (circle one)

Strongly Agree				Strongly Disagree
1	2	3	4	5