March 2013

Predicting the Perceived Quality of a First Person Shooter Game: the Team Fortress 2 T-Model

David Thomas Dwyer
Worcester Polytechnic Institute

Eric M. Finn
Worcester Polytechnic Institute

Project number: MLC-LG12

Predicting the Perceived Quality of a First Person Shooter Game: the Team Fortress 2 T-Model

A Major Qualifying Project Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements

for the Degree of Bachelor of Science

by:

Eric Finn

David T. Dwyer

Date: March 12, 2013

Approved:

Professor Mark L. Claypool

Abstract

This paper describes the development of a model which quantifies the effects of latency on a player’s perceived game quality in the networked first person shooter game Team Fortress 2, in an attempt to replicate and extend previous research done on other games. We conducted a study to measure the subjective quality of gameplay under various induced latency conditions. We determined that latency had a significant effect on a player's perceived quality of the game, yet did not significantly affect player performance. Using these data, we constructed the Team Fortress 2 G-model to predict the mean opinion score of Team Fortress 2 game based on known network conditions.

Keywords: games, network, G-Model, Team Fortress 2, user study

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Internet games are a popular way for Internet users to spend their time, and have been getting even more popular over the years. As these games rise in popularity, it is increasingly important to develop an understanding of how network conditions affect them. Game developers have many decisions to make when creating their products, like how to minimize network latency using latency compensation algorithms while keeping the game entertaining. Internet games have this extra consideration concerned with how to handle and compensate for latency. The importance of latency compensation for internet games can be better understood through studies like this one.

One of the topics for which there is not enough data for is how network conditions affect the playability of Internet games. Games are less playable as network latencies increase (Beigbeder, Coughlan, Lusher, Plunkett, Claypool, & Agu, 2004), but exactly how much worse the gameplay gets is not accurately modeled. Thus, it would be useful to have a better understanding of the effects of latency on Internet games, like Team Fortress 2. More information on how network conditions impact user experience can aid game developers in focusing their efforts on compensating for issues which are most important to their users.

There has been some previous research done in quantifying the effect of network conditions on the playability of networked games (Wattimena, Kooij, Van Vugt, & Ahmed, 2006). Some of this research has been focused on constructing a mathematical model which predicts players' opinion of gameplay quality based on various metrics of network performance. A study involving the first person shooter game Quake IV created a model for predicting the mean opinion score of a Quake IV game given the values of delay and jitter. This model was called the G-model (Wattimena, Kooij, Van Vugt, & Ahmed, 2006). Most previous research on the effects of network conditions on games has been done on games in which the primary in-game interactions were between players. The study involving the game Warcraft III involved players commanding armies to destroy one another under induced latency conditions (Sheldon, Girard, Borg, Claypool, & Agu, 2003). A research study involving the game Unreal Tournament 2003 focused on observing how latency affected player movement as well as various precise and imprecise weapons (Beigbeder, Coughlan, Lusher, Plunkett, Claypool, & Agu, 2004). The intent of this study is to continue and expand upon the research of previous studies of latency effects on first person shooters and to create a new model using a newer first person shooter.

In order to study how latency affects player performance and perceived game quality, we chose a game that allows us to create a model similar to the Quake IV G-model, to provide a more up to date model for a more modern first person shooter. The game we chose was Team Fortress 2. We have conducted a user study using Team Fortress 2 in which we controlled network conditions and measure the players' opinions of gameplay quality. Delay and jitter values were controlled with the Linux kernel component named netem, on a Linux PC running our Team Fortress 2 server. Information on player performance and scoring along with gameplay style choices were recorded so that we could obtain a better understanding of how players are affected by and react to changes in network conditions. Data was also collected from users. Test subjects provided age, education, and gender demographics and also provided an opinion score and which classes that they played for each test round.

There were a total of 31 participants in this study. Nearly all had played first person shooters before, as well as played Team Fortress 2 specifically. The participants provided an average skill score of three on a scale of one to five, with five being the most skilled at first person shooters. The majority of the participants were between the ages of 18 and 21. Analysis of the data provided by these participants revealed that the users were aware of the induced latency and scored lower mean opinion scores for higher values of induced delay and jitter. Participant’s combat performance, however, appeared to be mostly unaffected by the latency conditions induced in testing. While Team Fortress 2 allows the players to choose from a number of different classes in a round, the data collected on participant’s class choices was not able to determine whether or not class selection was influenced by latency conditions.

Chapter 2 describes studies and research papers that are related to our study, focusing primarily on the effects of latency on online games. Chapter 3 provides the methodology of our study, including the hardware and software used, information about the tests performed, and details on our methods of data collection. Chapter 4 consists of our results and analysis of the data collected from testing, using the data to create our model and various performance graphs. Chapter 5 summarizes the results of our data, our model, and our findings. Chapter 6, future work, describes proposed studies that could extend this study by further developing work from this study or researched areas not covered by this study.

2 Related Work

This chapter examines current research regarding online computer games and makes comparisons to this project. The first section is a description and a comparison of the G and E models as related to latency metrics. The second section inspects previous projects done on the effects of latency on the real time strategy game Warcraft III, and another on Unreal Tournament 2003. The third part is a comparison of the latency algorithms currently used by various games. The fourth part is some background information on the Source engine, which is used by Team Fortress 2, along with many other popular games, such as the Quake series.

2.1 The E and G models: A Model for Sound, a Model for Games

This section describes two models that predict mean opinion scores based on values of delay and jitter. Mean opinion scores are the average score on a scale of one to five provided by a user under a specific amount of delay and jitter. A score of one is the lowest score, indicating bad quality, whereas a score of five indicates that the quality of the service was good. The E-model is used to predict the mean opinion score of VoIP audio transmissions given values of jitter and delay (Bergstra & Middelburg, 2003). The G-model is used to predict the mean opinion score of the game Quake IV given the values of jitter and delay (Wattimena, Kooij, Van Vugt, & Ahmed, 2006).

The E-Model:

The E-model assesses the combined effects of varying transmission parameters that affect the conversation quality of narrow band telephony (Bergstra & Middelburg, 2003). The principle of the E-model is based on the assumptions that transmission impairments can be transformed into psychological factors and that these factors are additive. The primary output of the E-model is a transmission rating factor R (Ding & Goubran, 2003):

$$R = R_o - I_s - I_d - I_e + A$$

where R_o represents the basic signal-to-noise ratio, I_s represents the impairments occurring simultaneously with the voice signal, I_d represents the impairments caused by delay, and I_e represents the impairments caused by low bit rate codecs. The advantage factor A can be used for compensation when there are other advantages of access to the user. R can be transformed into a MOS scale by (Ding & Goubran, 2003):

$$\begin{array}{ll} \text{MOS} = 1 & \text{if } R < 0 \\ \text{MOS} = 1 + 0.035R + R(R-60)(100-R) \times 7\times10^{-6} & \text{if } 0<R<100 \\ \text{MOS} = 4.5 & \text{if } R > 100 \end{array}$$

The G-Model:

The G-Model was created after the E-Model. The G-Model's purpose is to describe the perceived effects of latency on Quake IV. The network impairment variable X is defined as follows:

$$X = 0.104 \times ping\_average + jitter\_average.$$

The authors calculated ping_average by sending 100 standard pings first from the client to the server, and then the server to the client. The average value over these 200 pings gives one ping_average value in milliseconds for each scenario. Secondly, a series of 300 UDP packets were sent from client to server and vice versa. The UDP packet size including UDP and IP header was 100 bytes. Each UDP packet was sent 50 ms after the other. At the receiving side the transmission time of each UDP packet from sender to receiver was logged. The transmission time of the fastest UDP packet was shifted to 0 ms, after which all shifted transmission times were averaged. This was done for both directions and the average of these was called jitter_average. The predicted mean opinion score, defined in the introduction, is then generated as follows (Wattimena, Kooij, Van Vugt, & Ahmed, 2006):

$$G = -0.00000587 \times X^{3} + 0.00139 \times X^{2} - 0.114 \times X + 4.37$$

Comparing the E and G Models

To compare the two models, we calculated the mean opinion scores for one way delay values between 0 and 160 milliseconds for both models. Figure 1 describes the mean opinion scores for the E-model and G-model, in relation to one way delay values, as shown here:

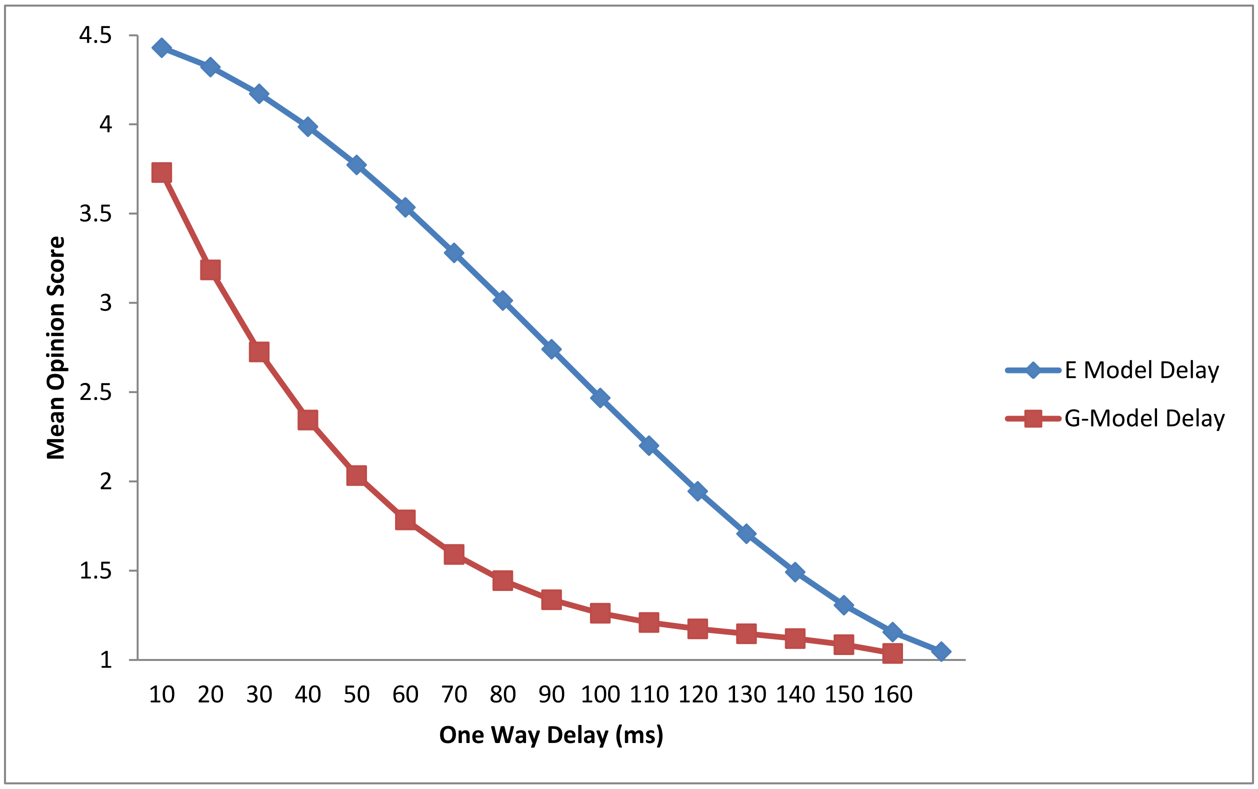


Figure 1: E and G-Model Predicted MOS vs. One Way Delay

2.2 Latency effects on Games

A study related to ours researched the effects of latency on the real time strategy game Warcraft III (Sheldon, Girard, Borg, Claypool, & Agu, 2003). This MQP project was motivated by a desire to research the effects of latency on a real time strategy game, as opposed to the more commonplace first person shooter analysis. They performed research using the game Warcraft III under varying amounts of jitter and latency. Through their experimentation, they discovered that the effect of latency on the game of Warcraft III was negligible. Because of the nature of real time strategy games, as compared to first person shooters, even latencies as high as 800ms had little effect on gameplay quality. Latency only affects a small subset of units with time sensitive abilities.

The second study related to ours researched the effects of delay, jitter and packet loss on a first person shooter, namely Unreal Tournament 2003 (Beigbeder, Coughlan, Lusher, Plunkett, Claypool, & Agu, 2004). Since precision aiming, character movement, and bullet speeds all rely heavily on constant, accurate information from client to server and back, first person shooter computer games are highly sensitive to network latency. In this MQP project, the researchers studied player performance in Unreal Tournament 2003 under varying amounts of latency and packet loss. They broke down the game into specific components for testing, such as straight movement, aiming, and a combination of both. They first concluded that neither latency nor packet loss has any affect on straight movement. Their next movement test was an obstacle course that the player had to move through. It was found that packet loss had little effect on a player's performance times in the course. However, when latency was introduced, a player's time to complete the course increased. Finally, they found that for even moderately small amounts of latency (up to 200ms), any game situation with a precision weapon present (such as a sniper rifle) were extremely affected by latency. When players were allowed to use imprecise weapons (such as a rocket launcher) the effect of latency on game performance was slightly less pronounced. After breaking the game down into specific components and running tests under controlled network conditions, they determined that packet loss and latency levels likely to be encountered in real world settings do not drastically impact player performance. They found that packet loss especially has no significant effect on overall gameplay.

2.3 Latency Compensation Algorithms

Latency compensation algorithms attempt to mitigate the effects of latency on players, helping either compensate for latency or making it appear as if there is none. The following methods described have all been used in various games, with the pros and cons described in Table 1.

Table 1: Pros and Cons of Latency Compensation Algorithms

Name	Pros	Cons
Extrapolation	The client makes a prediction based on recent server information, and can then render this assumed future state.	Player's movements are non-deterministic and subject to high jerk. This may be made worse by unrealistic physics common in some games.
Interpolation	Displays an object's location accurately by moving objects in the past.	Lost packet requires Extrapolation or stopping at the most recent update until another packet arrives, causing objects to stutter.
Presentation (Time) Delay	For latencies less than the presentation delay threshold, consistency among players (in regards to the game state they see) is excellent. In the case of fixed, known delays it should be perfect.	The user is still delayed from seeing input feedback. This largely disqualifies the technique for physically realistic games with fast dynamics.
Time Warp	The server presents a more accurate game state.	Could cause an action (like player death) to seemingly be delayed, even if it was accurate in the past.

The first form of latency compensation is called *Extrapolation*. In Extrapolation, game objects are simulated forward in time from the last known location, direction, and velocity (Bernier, 2001). The client draws the player at that extrapolated position and the local player can aim right at the other player. There are two different forms of extrapolation named *Dead Reckoning* and *Short Circuiting*. Dead Reckoning is a common name for a client extrapolating its game state based on predictions it makes about what the future state of the game will be. It should be noted that the server does not necessarily accept all predicted states. This means that the prediction made by the client is not always right, causing the client to temporarily display false information. Short Circuiting takes this a step further, sending the "extrapolated position" directly back to the client's state, thereby having the client in a "future state" that is not necessarily accurate (but tries to be), in an attempt to not make the user wait one full round-trip time for their input to be seen in the game state (Buchheit, 2004). Short circuiting is analogous to *Client Side Prediction*. Client Side Prediction generally involves the client and server sharing some similar code, such as the code for player movement. The client can then “predict” what the server reports after the send/recieve delay, since it uses the same code to determine the player’s movement (Bernier, 2001).

Another form of latency compensation is *Interpolation*. Interpolation can be viewed as always moving objects somewhat in the past with respect to the last valid position received for the object. As frames are rendered, the position is interpolated between the last updated position and the position before that (alternatively, the last render position). As the object just gets to the last updated position, a new update is received from the server and the object starts moving toward this new position. This method is helpful for reducing the observed effects of latency, which include characters skipping around on the screen (Bernier, 2001).

There are two other forms of latency compensation that deserve notice. The first is *Presentation Delay*, also known as *Time Delay*. This technique is similar in some ways to the buffering that is used to improve playback on streaming media. The basic concept of the technique is that when the game client receives an update from the server it delays displaying it to the user for a fixed period of time. In this way, all users whose connections with the server have a latency less than or equal to the induced delay are able to play the game on the same level as all other users. The other method is a form of interpolation named *Time Warp*, used by some Steam games in the Source engine. In Time Warp, the server can go back in time (perhaps by storing a number of previous game states) and perform an action that would have taken place at that time (like player A shooting player B) and check the outcome of the action then. The server can use Time Warp to compensate for various users' differing latencies (Bernier, 2001).

2.4 The Source Engine

The Source engine is a popular game engine that has been used for many games over years (Category:Source engine games, 2010). It was used in the popular game called Half-Life (Half Life), which included a mod called Team Fortress (Team Fortress), the predecessor of Team Fortress 2 (Team Fortress 2). In the Half-Life version of the Source engine, the client sends information to the server in the form of a data structure, which includes information such as interpolation time, directional velocities, and the frame time. The client uses these fields to describe where a player is, and then sends the information back to the server, and waits for a response. To overcome latency, Half-Life uses Client Side Prediction. A system where movements and weapon effects are predicted client-side is a system that the Quake 3 engine, also based off of the Source engine, supports. The Source Engine documentation states that it is up to the individual game designers to decide on which, if any, latency compensation methods are used in their games (Bernier, 2001).

3 Methodology

We conducted a user study of the playability of Team Fortress 2 under different network conditions. To do this, we set up and configured a computer to run the game server software and computers to run the game clients. Then, we solicited volunteers from the student body at Worcester Polytechnic Institute. We had the volunteers play TF2 for an hour, giving us feedback every time the game round ended as to the quality of gameplay of the round. We used the responses given as well as the messages written to server logs to determine mean opinion scores, players' performance, and which character classes players played for each round.

In order to do this, we utilized a Linux computer running Ubuntu 12.10 and the Source Dedicated Server version 50 to host the game of Team Fortress 2 (TF2) and netem to simulate the different network conditions. The server was on the same local area network as the client PC's running Worcester Polytechnic Institute's version of Windows 7 as of January 2013, Steam, and the TF2 game client released October 10, 2007. Additionally, we had a Windows PC running the TF2 client which was kept in "spectator" mode so that we could view the match in real-time and record it without disrupting the players. Unlike players, spectators do not affect game state and therefore only receive information from the server. As such, spectators are always viewing a copy of the game state that is consistent with the server's state, rather than a client's game state, even though they are viewing it after the server is.

The recommended system requirements for TF2 are "Pentium 4 processor (3.0GHz, or better), 1GB RAM, DirectX® 9 level Graphics Card" [9]. The minimum system requirements are even lower. We used a computer lab on campus called the Zoo Lab that had divided rooms of

PCs that met the requirements. The computers reported that they were equipped with Intel Core i7-3770 processors running at 3.40 GHz, 12 GB of RAM, and AMD Radeon HD 7700 video cards with 3794 MB of dedicated graphics memory.

For computing of statistics, we saved the logs for each match, which provided detailed information on the performance of individual players on the server once the information is extracted and parsed. The server logs contained information about player kills and deaths, as well as classes played, weapons used, and healing and damage done during a round.

To determine the game settings to use in the test, we had to consider multiple factors. Match time is how long the players spend playing under a single test situation. A match time that is too short may not allow the players enough time to develop an opinion on the network conditions and frustrate the players by frequently interrupting gameplay, while one that is too long could prevent us from running through as many test situations as we may like. The game mode and map needed to be chosen so that players were able to have enough interaction during the match time to form an opinion of the gameplay quality, while keeping them interested through a large number of tests. The respawn time (the maximum delay between a player dying and them being alive again) also needed to be chosen to prevent the players from becoming disinterested by waiting to respawn for too long while still being long enough to prevent a stalemate situation caused by death not impacting a team’s effectiveness significantly enough. With these requirements, it was determined that the map Viaduct with game mode King of the Hill was the best choice for testing. Match times for King of the Hill were between 3 and 6 minutes normally, with occasional overtimes of 0.5-1 minutes. The default respawn time after death was used.

All test subjects played on the same team, in a cooperative match against bots. We configured the server so that bots would also automatically fill in empty slots on the test subject's team. The bots were set to the "normal" difficulty level. There were 8 players on each team. All classes were allowed, but subjects were told not to equip any items or modifications that might be unlocked so as to keep the classes all at their defaults for testing. All subjects were provided with the same computers, keyboards, mice, and a set of headphones.

Additionally, we also disabled the scoreboard in the players' game clients, which show the network delay between client and server. We also disabled in-game VoIP, as the game's lag compensation would not apply to VoIP, so audio quality of VoIP may have impacted the MOS. Since the subjects were on the same team and in the same room, electronic forms of communication were not needed for testing.

We generated a set of test scenarios to run, each scenario having a different set of values for various test parameters in order to test varying levels of latency on gameplay. Delay, jitter, and packet loss were controlled by netem (Linux Foundation, 2009). Netem delayed each packet by a random amount based on a normal distribution and was configured to not allow packets to be reordered. We created scenarios that had mean delays of 0ms, 100ms, 250ms, and 500ms, with standard deviations that were 0%, 10%, and 20% of the mean. We then randomized the order of the scenarios, repeated the worst-case scenario at the beginning of each test to give players a chance to warm up and calibrate their perception of lag. We then discarded the scenario with a mean delay of 100ms and a standard deviation of 0ms, as we did not have enough time per test to run each scenario. The test scenarios, in the order that they were run for each test, were as shown in Table 2.

Table 2: Test scenarios

Test	Mean	Standard Deviation
(Discarded) 1	500 ms	100 ms
2	100 ms	20 ms
3	25 ms	0 ms
4	250 ms	50 ms
5	0 ms	0 ms
6	500 ms	100 ms
7	100 ms	10 ms
8	250 ms	25 ms
9	500 ms	0 ms
10	500 ms	50 ms

The first test was a control case where the subjects were told that they will experience the highest latency that they might encounter during testing. In each subsequent test scenario run, the subjects played the game for a set period of time without knowing the level of latency that was being induced for that scenario. At the end of the time period, the game was stopped and the players were asked to provide their opinion of game quality on a scale of 1 to 5 and to indicate the classes that they played that round. They had previously been told at the start of the test, "The first question on the score sheet, quality of gameplay, is about the quality of gameplay in terms of how latency (or lag) influences it. The worst rating, 1, indicates a round in which the game is almost unplayable due to lag, while the best rating, 5, indicates a round in which the lag has no negative impact on gameplay." These opinions were used to create a Mean Opinion Score for each test scenario. Additionally, the scores of each player were captured at the end of each test scenario and an in-game recording was taken of each test scenario. See Appendix A for the questionnaires that were provided to participants.

To solicit participants, we sent an email to the "students" mailing list at Worcester Polytechnic Institute, which broadcast the request for participants to all undergraduate and graduate students. The email sent read,

Would you like a chance to play Team Fortress 2?
What if you could help SCIENCE while doing it?
Want a chance to win a $10 Dunkin Donuts gift card?
We are running an MQP and need participants to play TF2 and answer questions about the gameplay experience.

Go to http://tinyurl.com/tf2mqp to sign up for a time to participate. Email ericfinn@wpi.edu if you have questions or comments.

We scheduled when volunteers would participate by asking them to indicate which times they were available on a web form, then collecting responses and giving as many volunteers as possible time slots that they indicated they were available for. Due to the number of volunteers, most time slots filled up and we were not able to schedule all volunteers for a session.

4 Results

This chapter describes the data collected during the course of the experiment and the models created based on the data. We collected data on demographics, network conditions, player performance and behavior, and participant opinion of gameplay under different network conditions. We then investigated different aspects of the data, described later in the chapter, and created two possible models to predict mean opinion score.

4.1 Demographics

A total of 31 people participated in the study. 29 participants identified as male, while 3 identified as female. Three of the participants had not played Team Fortress 2 before, and one of those participants had not played any first-person shooter games before. When participants were asked to provide a rating of how skilled they rate themselves at first-person shooter games, most participants gave themselves a rating of three on a scale from one to five. Participants were aged 18 to 30, with more participants in younger age categories. The information on how many participants fall under each group is shown in Figure 3.

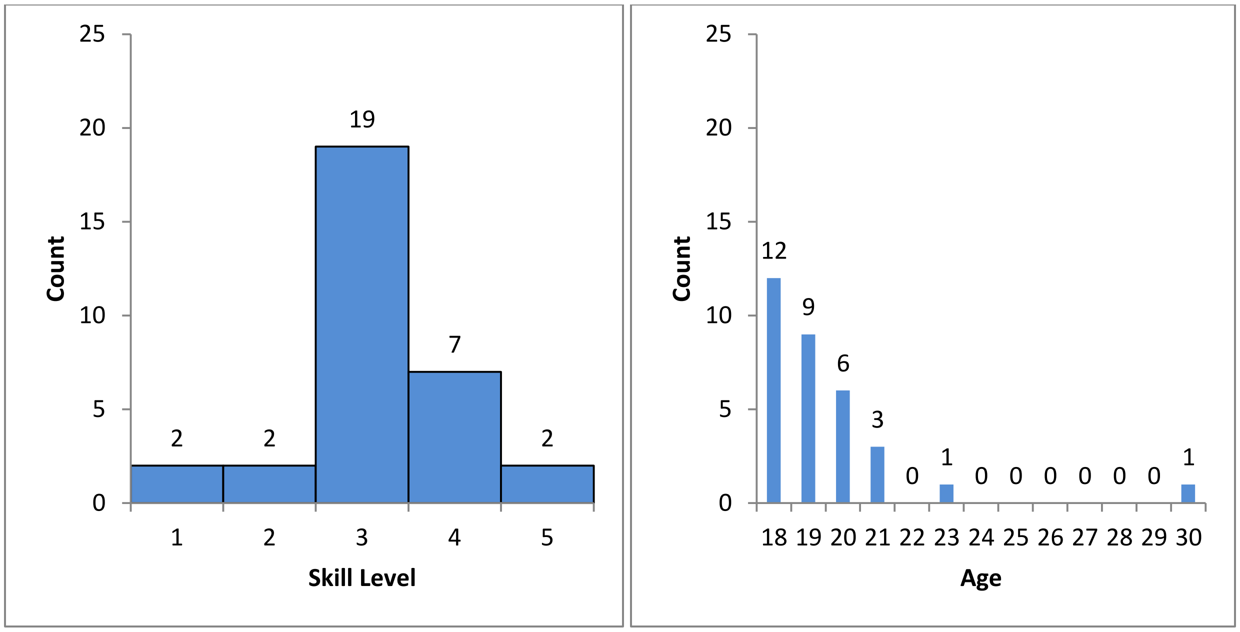


Figure 2: Skill and Age Distribution of All Participants.

Unfortunately, not all of the participants' data could be used. The one participant who had not played any first-person shooter games before did not perform any actions that would be influenced by latency for the majority of that participant's time during the study. Another participant started playing partway through a session. However, some participants stayed for multiple sessions, including the participant who started partway through a session. Overall, although there were 31 participants, there were 35 usable participant-sessions due to the participants who stayed for two sessions. These 35 participant-sessions constitute the "final set" of data that were used.

Of the final set, 2 participant-sessions had female participants and 33 participant-sessions had male participants. All but one of the participant-sessions had a participant who had played Team Fortress 2 before, and all of the participant-sessions had a participant who had played first-person shooter games before. The trends in skill levels and ages for the final set of participant-sessions were similar to those of the set of all participants.

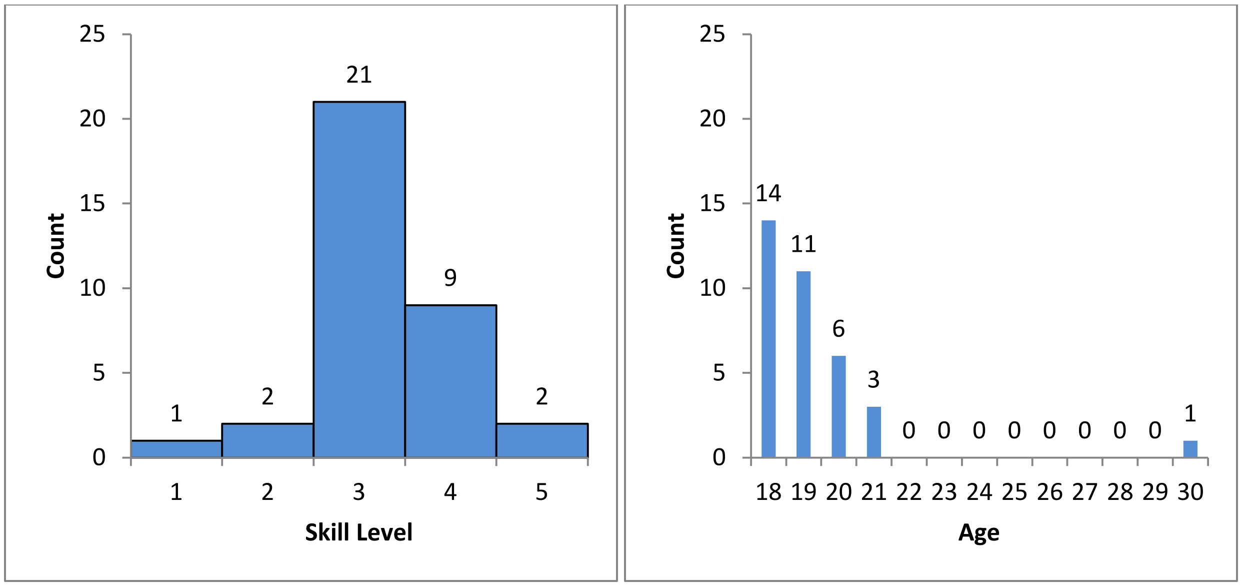


Figure 3: Skill and Age Distribution of All Used Person-Sessions.

4.2 Network Conditions and Mean Opinion Score

Table 3 contains the mean opinion scores and 95% confidence intervals for each scenario. Network conditions had an observable effect on the mean opinion score of the gameplay quality. At 95% confidence, there is a significant difference between mean opinion scores in the scenario with an average delay of 500ms and standard deviation of 100ms and the two other scenarios with an average delay of 500ms and standard deviations of 0ms and 50ms. Additionally, there is a significant difference between the mean opinion scores in the scenario with no delay or jitter and the scenario with an average delay of 250ms and a standard deviation of 25ms.

Table 3: Mean Opinion Scores with 95% Confidence Intervals

	Mean 0ms Delay	Mean 100ms Delay	Mean 250ms Delay	Mean 500ms Delay
Stdev 0% of Mean	4.15 ± 0.31		3.53 ± 0.34	3.58 ± 0.42
Stdev 10% of Mean		3.85 ± 0.33	3.54 ± 0.34	3.00 ± 0.45
Stdev 20% of Mean		3.79 ± 0.26	3.26 ± 0.33	2.06 ± 0.30

Figures 5 and 6 show mean opinion scores of different scenarios. In Figure 5, the scenarios are placed based on the mean delay present in the scenario, and are colored by the amount of jitter in the scenario, relative to delay. In Figure 6, the scenarios are placed based on the amount of jitter present in the scenario and are colored based on the mean delay for the scenario. In both graphs, the error bars are 95% confidence intervals. Figures 5 and 6 show that as both jitter and mean delay are increased, mean opinion score also decreases.

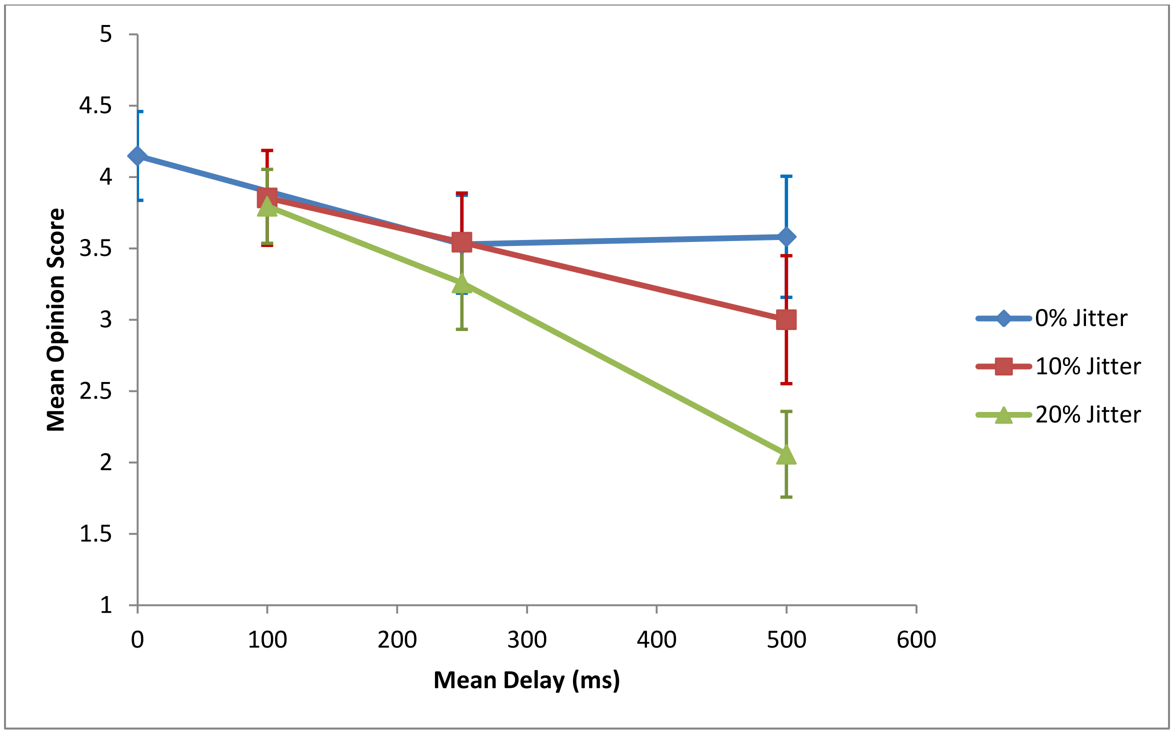


Figure 4: Mean +Opinion Score vs. Delay for Different Jitter Values.

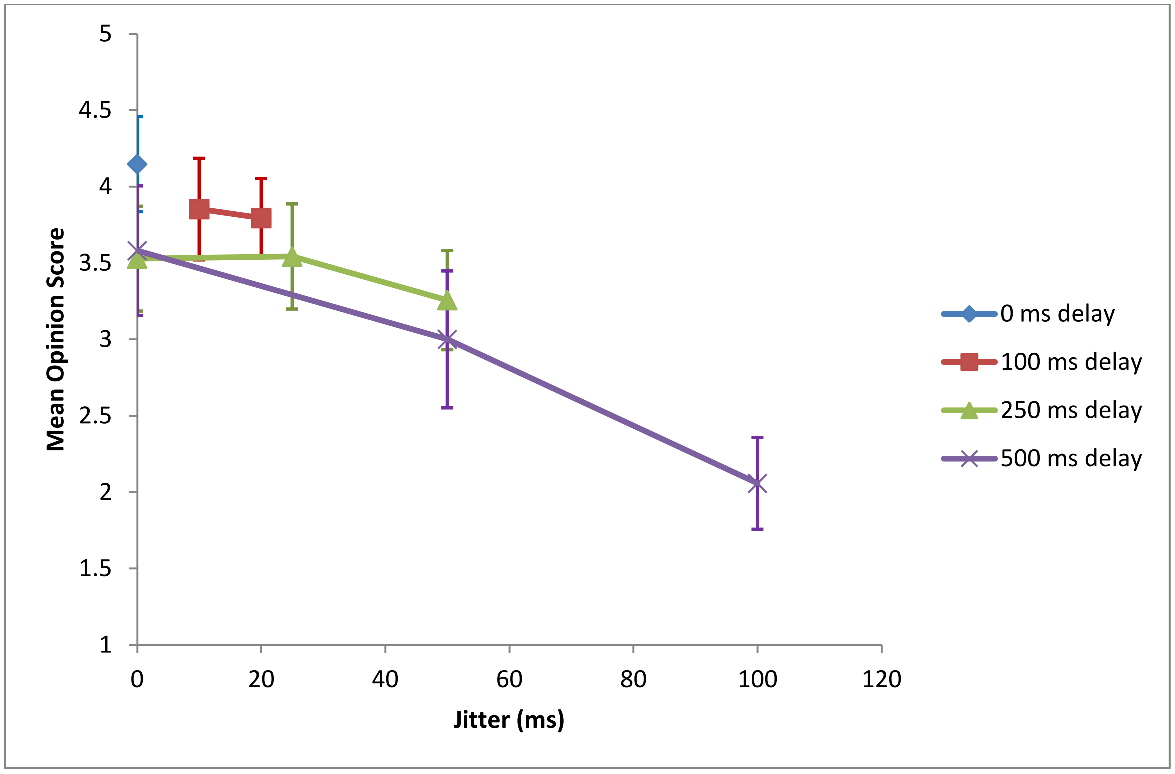


Figure 5: Mean opinion score vs. jitter for different delay values.

4.3 Skill, Performance, and Opinion

A few questions we sought to answer were what effect latency has on game performance, how players' skill levels effect their performance and opinions, and how performance and gameplay quality opinions are related. In order to help with this, we created a script for parsing Team Fortress 2 server logs and extracting player performance data. From this, we determined the number of kills and kill assists each player had during each round, and calculated a combat performance score by adding half of the player's kill assists to the player's kills. This is part of how score is calculated in-game, without any points due to construction of objects, destruction of objects, or completing objectives. This calculated combat score (CCS) is a measure of just how players interact with opponents in the game world. Figure 7 shows the mean Calculated Combat Score for participants in the different scenarios. The error bars are 95% confidence intervals. As can be seen in the graph, based on the 95% confidence intervals, there is not a significant difference between any of the scenarios. Therefore, the network conditions did not have a significant effect on player performance.

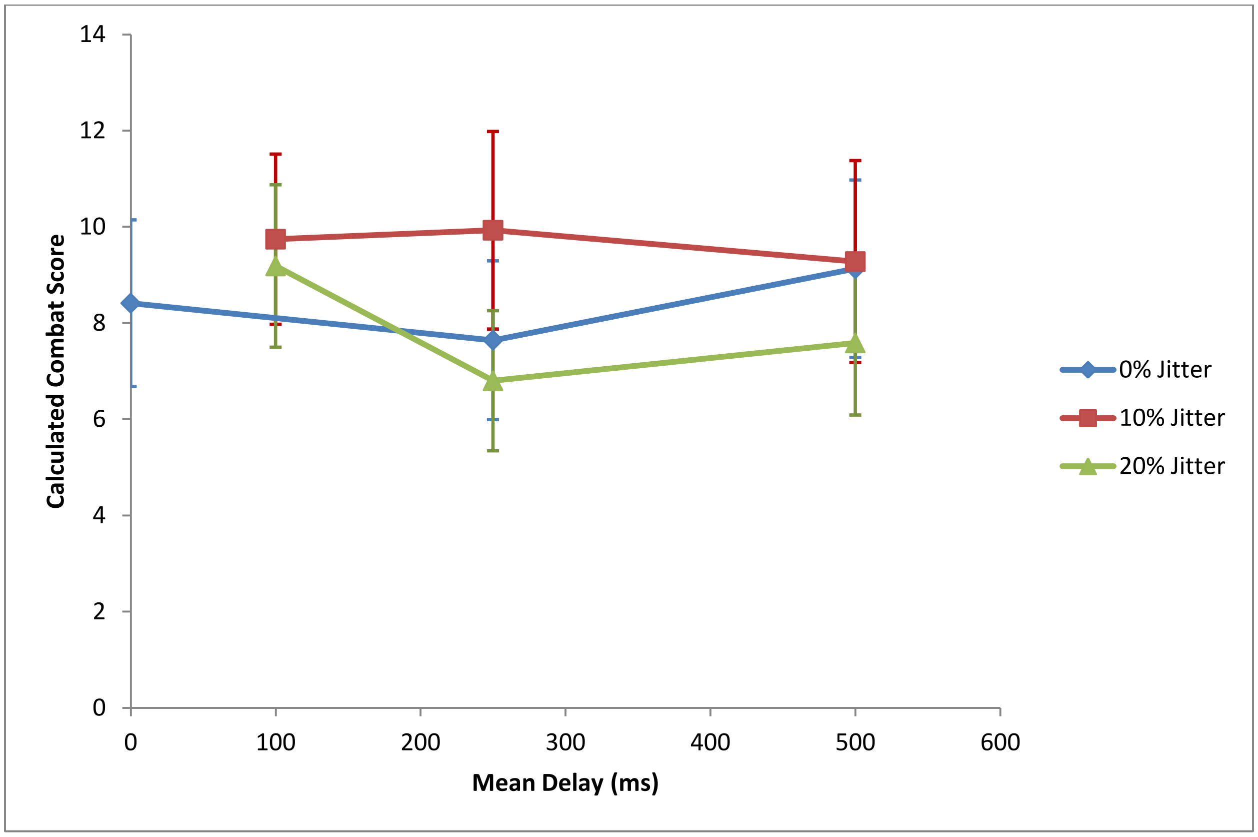


Figure 6: Mean calculated combat score vs. mean delay for different jitter values.

There are a few possible explanations for this phenomenon. One that we investigated is the possibility that the players compensated for the change in latency by switching character class in the game. Different classes are designed for different play styles, and according to some of the participants, some classes are more suited for playing with lag than others. Figure 8 shows the average portion of a round spent as each class in scenarios with different amounts of delay for scenarios where the jitter is 20% of the mean delay. The error bars are 95% confidence intervals. As can be seen in the graph, there was not a significant change in how much time people spent playing a class due to delay.

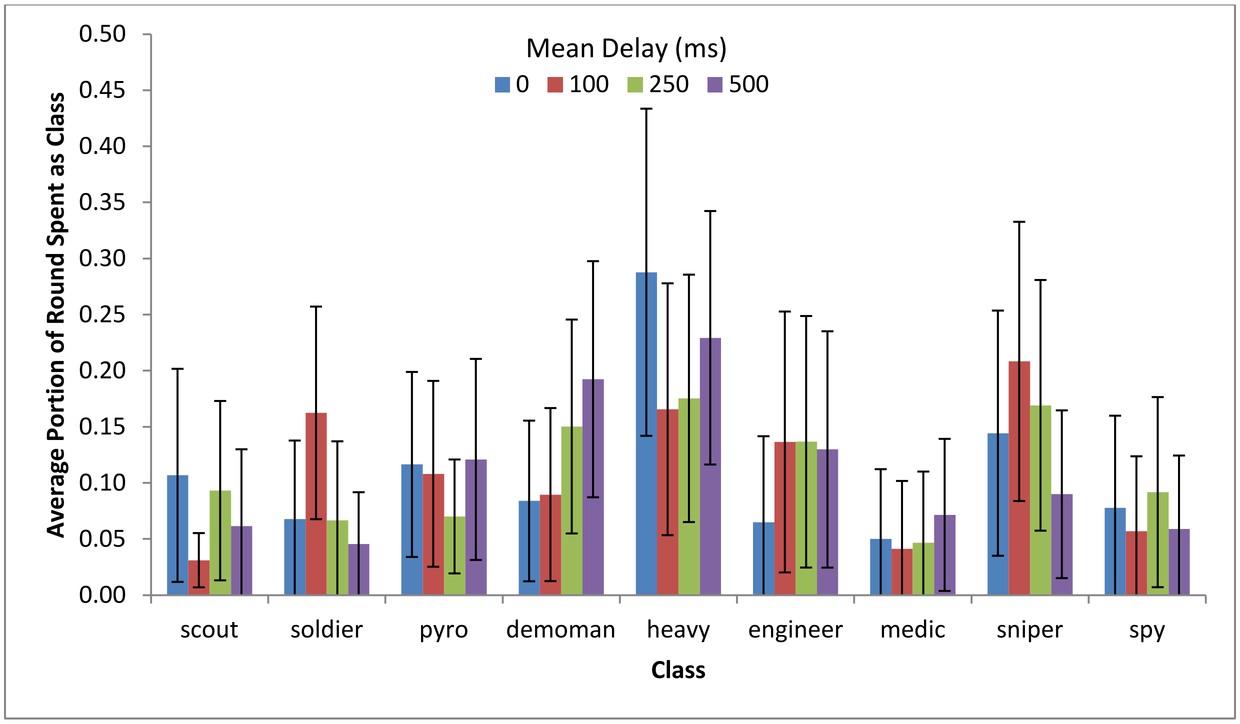


Figure 7: Portion of round spent as classes in different delay scenarios.

Since the participants did not, as a whole, select different character classes based on the network conditions, the selection of character class to compensate for lag cannot explain the lack of effect of lag on performance. One possible alternate explanation is that the lag compensation built into the Team Fortress 2 engine is able to eliminate any significant disadvantage due to poor network conditions. Another possible explanation is that since killing the opposing team members is not a primary objective of the game mode, the participants were not focusing on combat, so combat performance is not a good indicator of the effect of lag on the players. The other possibility is that the time a player has spent playing influences the player's performance, and, since the different scenarios happened in a set order, the order of the scenarios could confound the possible relationship between network conditions and player performance.

Figure 9 shows the relationships between combat performance and skill and mean opinion score. There is a significant difference between the average calculated combat score for the player with a skill level of 1 and the players of all other skill levels. Additionally, there is a significant difference between the average combat scores when players rated rounds 1 or 2 and when they rated rounds 4 or 5.

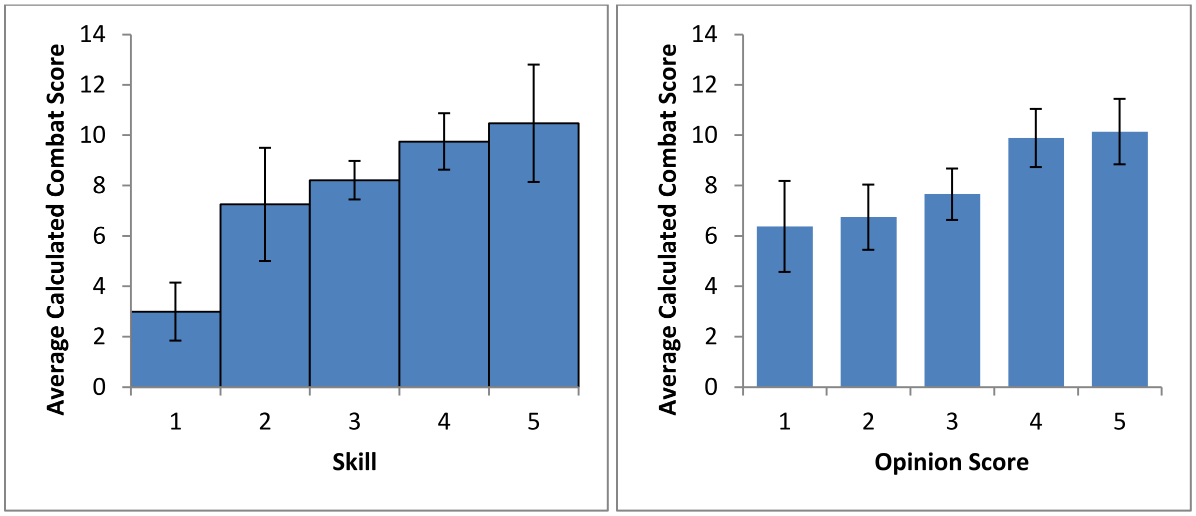


Figure 8: Average calculated combat score vs. skill and mean opinion score.

4.4 Models

Based on the experimental data, we created two models using Eureqa Formulize (Nutonian) to predict mean opinion scores given delay and jitter. The first model,

$$score = 4.15 - 0.0012delay - 0.013jitter$$

has a mean absolute error of 0.082. The second, slightly more complex model,

$$score = 3.98 - 0.0013delay - 0.00013jitter^2$$

has a mean absolute error of 0.075. In both equations above, delay and jitter are measured in milliseconds, delay is the mean delay, and jitter is the standard deviation in a normal model describing the distribution of delay values. The following graphs, Figure 10, show these models plotted along with the mean opinion scores of the scenarios we tested. Since the first model has an acceptable fit to the experimental results and it is simpler than the second model, we choose it as the Team Fortress 2 G-model.

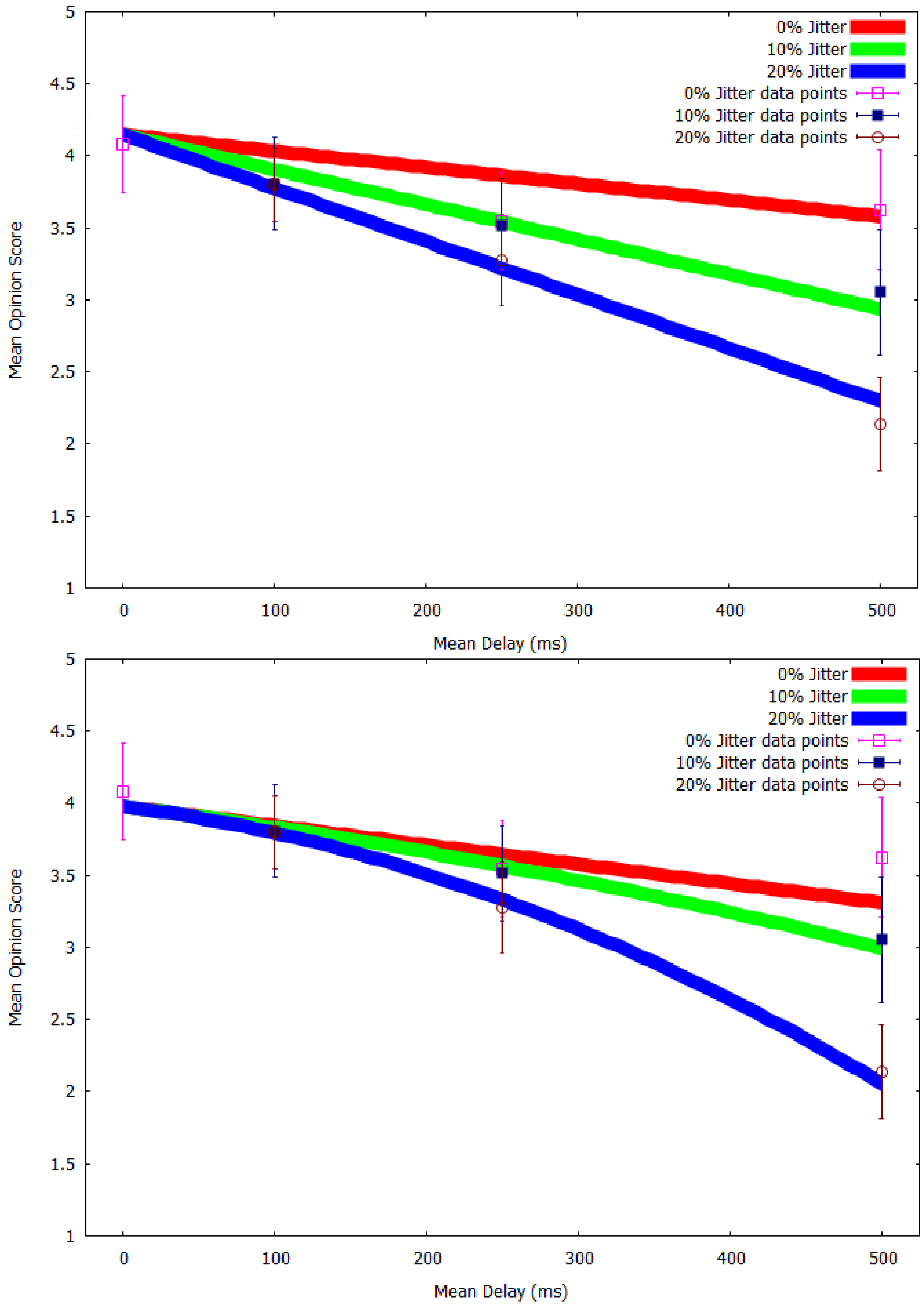


Figure 9: The two candidate G-Models, first on top and second on bottom.

This model differs from the Quake IV G-model (Wattimena, Kooij, Van Vugt, & Ahmed, 2006) in multiple ways. First, the Quake IV G-model introduced jitter using a pareto distribution as opposed to a normal distribution. Therefore, a comparison between the Quake IV G-model and the Team Fortress 2 G-model cannot be made if jitter is considered. However, the authors of the Quake IV G-model use an experimentally determined ping average. This means that the Quake IV G-model and the Team Fortress 2 G-model can be compared as long as there is no jitter. Figure 11 is a graph of the Quake IV and Team Fortress 2 G-models along with the data points for 0% jitter from this study. It is evident that these two G-models are not similar, and additionally the Quake IV G-model does not fit the experimental data of this study.

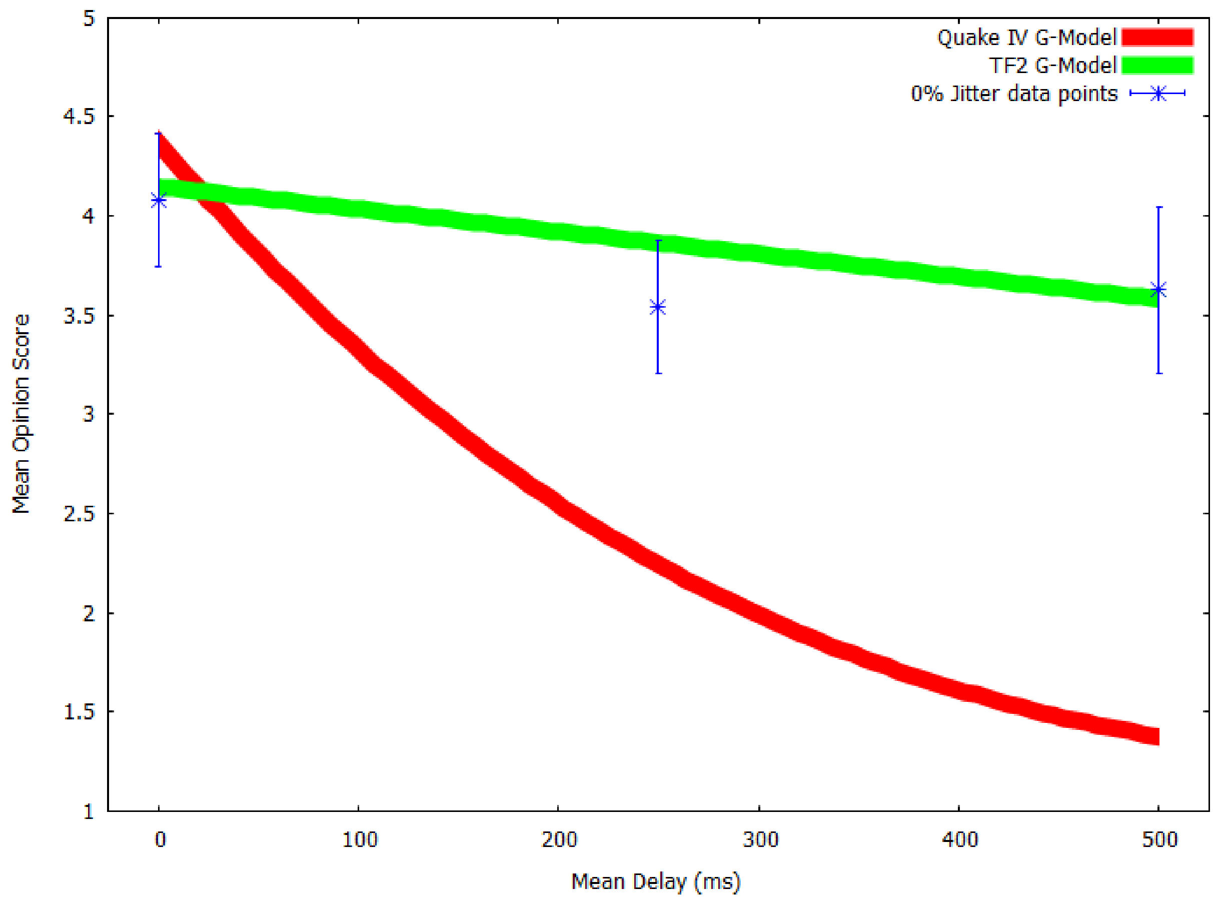


Figure 10: Comparison of Quake IV and TF2 G-models.

The fact that these two models are not compatible is indicative of the differences in the games used as well as differences in the experimental designs of the studies which led to these models. The games run on different engines, and as a result, likely have different built-in latency compensation methods. The Quake IV study had 6 participants all playing a game mode called “Free-For-All”, in which players score points by “killing” other players’ characters, and the first to reach a certain number of points wins the round. This means that the players in the Quake IV study were primarily engaging with each other over the network connection, rather than primarily engaging with the server’s bots and the virtual world, as was the case with this study. This suggests that G-models may not be generalizible to other games or game modes, and may differ for interactions that take place between multiple clients with a server in between and between a client and a server.

5 Conclusion

The ability to predict the perceived mean quality of game play given particular network conditions can be useful for game developers and network researchers. With this information, game designers seeking to improve latency in their games can understand what acceptable levels of latency are for users. When designers create new latency algorithms, data from studies like this one can provide latency bounds for testing those algorithms. High traffic game servers or clusters of servers can use this data to help determine whether upgrades to their infrastructure are needed due to excessive latency.

In this study, we investigated the effects of latency on the first person shooter game Team Fortress 2 in terms of user’s perceived quality. We designed a number of tests which induced various amounts of delay and jitter to a LAN server running Team Fortress 2. We collected user opinion scores, combat performance data, and class selections for analysis. We used these data to determine the effects on latency on game play and developed a model for predicting the mean opinion score of a Team Fortress 2 game given the conditions of jitter and delay.

Overall, delay values between zero and a few hundred milliseconds were considered playable by users, whereas latency values above half a second were considered unplayable. Our model shows that between zero and two hundred milliseconds of delay, the predicted mean opinion score is between the best possible score of five and three. At values above two hundred milliseconds but below half a second, the model predicts a mean opinion score between three and two. Opinion scores over half a second are between two and one, and remain at the minimum score of one as latency increases beyond half a second. In regards to jitter, the model predicts a mean opinion score of between five and three for jitter values between zero and fifty milliseconds. For jitter values between fifty and one hundred milliseconds, the predicted mean opinion scores are between three and two. Jitter values over one hundred milliseconds are between two and one, and remain at the minimum score of one as jitter values continue to increase beyond one hundred milliseconds. These results fit the intuition that first person shooters rely on quick reactions and high precision, so even a second of delay can result in a lost fight. Our model was created from user data using the map Viaduct and game mode King of the Hill in the game Team Fortress 2. Thus, this model is valid for scenarios using those parameters. While the applicability of our model to different game scenarios has not been established, this model predicts the mean opinion score that users can be expected to provide, for a round of the above game mode, given the values of the network conditions delay and jitter.

Delay and jitter surprisingly did not appear to have a significant effect on players' combat performances. This could be attributed to test subjects having experience with latency and possessing knowledge of how to compensate for that latency. Another explanation could be that certain classes performed better under high latency conditions, but our data did not support that theory either way, with two classes in particular being played the most in all latency conditions.

Based on our results, latency did have an effect on a user's perceived quality of game play, which was the main focus of this study and of our model. User average mean opinion scores declined as values of latency and jitter increased. In scenarios with none or small latency values users gave higher scores, whereas at very high latency values the average scores were close to our unplayable score of one.

6 Future Work

Based on our findings, a next step could be to test a larger subset of subjects from different demographics. Since the majority of our test subjects were college gamers of similar age and gender, it would be prudent to explore older and/or younger demographic subjects, especially females, and generate test data from this larger subset.

Another step could be to test different game modes and maps to see how latency effects various game modes differently and be able to generate different models from these different game modes and make comparisons to the original test study. Comparing mean opinion scores of different game modes at the same levels of latency to see which game modes are more effected by latency than others could be of interest. One possible field to collect on and analyze is indirect game objects. For example, the number of flag captures under different latency conditions would be an indirect game object that could be analyzed. Also, data from studies described above could be used with data from this study to create a more generalized model usable for any Team Fortress 2 game mode and map, rather than separate models specifically for certain maps and/or game types.

Works Cited

Category:Source engine games. (2010 , February 5). Retrieved March 12, 2013, from wikipedia: http://en.wikipedia.org/wiki/Category:Source_engine_games

Beigbeder, T. P., Coughlan, R. J., Lusher, C. C., Plunkett, J. J., Claypool, M. L., & Agu, E. O. (2004). *The Effects of Packet Loss and Latency on Player Performance in Unreal Tournament 2003.* MQP Worcester, Massachussetts: Worcester Polytechnic Institute.

Bergstra, J. A., & Middelburg, C. A. (2003). *ITU-T Recommendation G. 107: The E-Model, a computational model for use in transmission planning.* International Telecommunication Union.

Bernier, Y. W. (2001). *Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization*. Retrieved October 30, 2012, from Valve Developer Community:https://developer.valvesoftware.com/wiki/Latency_Compensating_Methods_in_Client/Server_In-game_Protocol_Design_and_Optimization

Buchheit, R. F. (2004). *Delay Compensation in Networked Computer Games.* Cleveland: Case Western Reserve University.

Ding, L., & Goubran, R. A. (2003). *Speech Quality Prediciton in VoIP Using the Extended E-Model.* Retrieved January 24, 2013, from Department of Systems and Computer Engineering, Carleton University: http://voip.koe.pl/docs/Pliki/0021.pdf

Half Life. (n.d.). Retrieved March 11, 2013, from Steam Store: http://store.steampowered.com/app/70/

Linux Foundation. (2009, November 19). *Netem*. Retrieved March 11, 2013, from http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

Nussbaum, L., & Richard, O. (2009). A comparative study of network link emulators. *Proceedings of the 2009 Spring Simulation Multiconference* (p. 85). San Diego: Society for Computer Simulation International.

Nutonian. (n.d.). *Eureqa Formulize*. Retrieved March 11, 2013, from http://formulize.nutonian.com/

Sheldon, N., Girard, E., Borg, S., Claypool, M., & Agu, E. (2003). The effect of latency on user performance in Warcraft III. *Proceedings of the 2nd workshop on network and system support for games* (pp. 3-14). Redwood City, CA: ACM.

Team Fortress. (n.d.). Retrieved March 11, 2013, from Steam Store: http://store.steampowered.com/app/20/

Team Fortress 2. (n.d.). Retrieved February 20, 2013, from Steam Store: http://store.steampowered.com/app/440

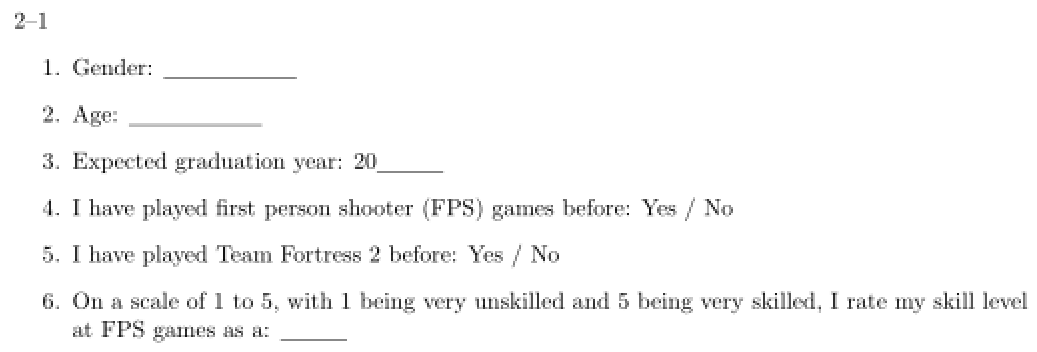
TronPaul. (2012, October 10). *tf2-loggr*. Retrieved February 13, 2013, from GitHub: https://github.com/TronPaul/tf2-loggr

Valve Corporation. (n.d.). *Steam Store*. Retrieved from Team Fortress 2: http://store.steampowered.com/app/440

Wattimena, A. F., Kooij, R. E., Van Vugt, J. M., & Ahmed, O. K. (2006). Predicting the perceived quality of a first person shooter: the Quake IV G-model. *5th ACM SIGCOMM workshop on network and system support for games* (p. 42). ACM.

Appendix A

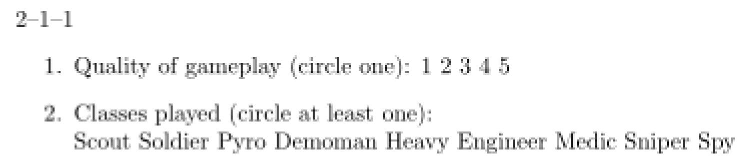
Example questionnaire provided to each participant:



2–1

1. Gender: ___________
2. Age: ___________
3. Expected graduation year: 20_____
4. I have played first person shooter (FPS) games before: Yes / No
5. I have played Team Fortress 2 before: Yes / No
6. On a scale of 1 to 5, with 1 being very unskilled and 5 being very skilled, I rate my skill level at FPS games as a: _____

Example score sheet, of which 10 were provided to each participant (one for each scenario):



2–1–1

1. Quality of gameplay (circle one): 1 2 3 4 5
2. Classes played (circle at least one):
   Scout Soldier Pyro Demoman Heavy Engineer Medic Sniper Spy