Quality of Experience Evaluation for Buffer Sizing of Cloud-Based Game Streaming

A Major Qualifying Project submitted to the Faculty of WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the degree of Bachelor of Science

By
Brennan Aubuchon
Morgan Langstaff

Date:
25 March 2022

Report Submitted To:
Professor Mark Claypool
Worcester Polytechnic Institute

This report represents the work of WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the projects program at WPI, please see https://www.wpi.edu/academics/ugradstudies/project-learning.html
Abstract

Emerging cloud-based game streaming systems such as Google Stadia, NVidia GeForce Now, and Amazon Luna allow players to play games with only a lightweight computer and a good network connection. Cloud-based game streaming can benefit from carefully tuned playout buffers to provide players with smooth but interactive gameplay. Our project aimed to quantify the user quality for game conditions that might arise with different buffer sizes for cloud-based game streaming. We conducted a user study in which users played cloud-based games and rated their experience while we perturbed the network in order to simulate different buffer sizes and network connection qualities. Based on our experiments, we found that games that require quick and precise inputs are more negatively affected by the delay from a buffer than those that do not. Conversely, games that involve lots of camera movement and animations are more negatively affected by the visual jitter from a lack of a buffer than those that do not.
Acknowledgements

We would like to thank the following individuals for their help during this study, it could not be completed without them.

- Xiaokun Xu: Helped with background information and created the testbed and scripts to run the study
- Professor Mark Claypool: Advised the process of the study from start to finish
- Stadia Staff (Philip Lamoureux, Brian Clark, and Peter Djeu): Helped us setup and use the Stadia dev environment, provided feedback and valuable information
# Table of Contents

**Section 1 Introduction**  
1

**Section 2 Background**  
3  
Section 2.1 What is cloud gaming?  
3  
   Section 2.1.1 How does it work?  
3  
   Section 2.1.2 History  
4  
Section 2.2 What is a Buffer?  
5  
Section 2.3 How does Stadia use buffers?  
6  
   Section 2.3.1 Why it is important to measure the effects of buffers  
7  
Section 2.4 How we will use these ideas  
7

**Section 3 Related Work**  
8  
Section 3.1 Buffer sizes in Internet video streaming  
8  
Section 3.2 Buffers for cloud gaming based on video streaming  
8  
Section 3.3 QoE For MultiParty Video Conferencing  
9  
Section 3.4 Impact of Buffering and Interrupts on Streaming Video QoE  
10  
Section 3.5 An evaluation of QoE in Cloud Gaming  
10

**Section 4 Methodology**  
12  
Section 4.1 Custom Games  
12  
   Section 4.1.1 Whack-a-Goat: Delay Sensitive  
12  
   Section 4.1.2 Goat Runner: Interrupt Sensitive  
14  
Section 4.2 Commercial Games  
15  
   Section 4.2.1 Avicii Invector: Delay Sensitive  
15  
   Section 4.2.1 The Falconeer: Interrupt Sensitive  
16  
Section 4.3 Test Bed  
17  
Section 4.4 User Study  
19  
   Section 4.4.1 Iterations  
19  
   Section 4.4.2 Procedure  
20  
Section 4.5 Preliminary Tests  
21  
Section 4.6 Data Collection  
22  
Section 4.7 Recruitment  
23

**Section 5 Analysis**  
24
List of Figures and Tables

Figure 1: Whack-a-Goat Custom Game 12
Figure 2: Goat Runner Custom Game 14
Figure 3: Avicii Invector Commercial Game 15
Figure 4: The Falconeer Commercial Game 16
Figure 5: A Diagram of the Room Used in the User Study 17
Figure 6: Diagram of the Test Bed 18
Figure 7: Gender Demographics 25
Figure 8: Major Demographics 26
Figure 9: Delay vs QoE graphs w/ Linear Regression line 27
Figure 10: Delay vs Score graphs w/ Linear Regression line 28
Figure 11: Score vs QoE graphs w/ Linear Regression line 30
Figure 12: Jitter Magnitude vs QoE graphs w/ Linear Regression line 31
Figure 13: Jitter Magnitude vs Score graphs w/ Linear Regression line 33
Figure 14: Score vs QoE for Jitter Magnitude w/ Linear Regression line 34
Figure 15: Jitter Frequency vs QoE graphs w/ Linear Regression line 35
Figure 16: Jitter Frequency vs Score graphs w/ Linear Regression line 37
Figure 17: Score vs QoE for Jitter Frequency w/ Linear Regression line 38

Table 1: Best Case Averages 24
Table 2: Linear Regression for delay 27
Table 3: Linear Regression with delay 28
Table 4: Linear regression scores vs QoE with delay 30
Table 5: Linear Regression for jitter magnitude 32
Table 6: Linear Regression for scores with jitter magnitude 33
Table 7: Linear regression for scores vs QoE with jitter magnitude 34
Table 8: Linear regression for jitter frequency 36
Table 9: Linear regression for scores with jitter frequency 37
Table 10: Linear regression for scores vs QoE with jitter frequency 38
Section 1 Introduction

Assessing the quality of experience (QoE) for different types of games under different system conditions provides feedback for system and game developers on their products and ways they can improve. QoE is especially important for cloud-based game streaming, not only because it is an emerging market with relatively less feedback, but also because the cloud-server delivery means there are additional factors that can impact a player’s QoE.

Two of the major factors that impact QoE in cloud-based games are delays and interrupts. Both of these can negatively impact QoE. Delays in cloud-based game streaming tend to take the form of input delay. This specifically impacts games that are timing and precision sensitive. Input delay can cause a player to make mistakes which then causes frustration with the game and degraded QoE. Delay affects games that require precision and timing such as rhythm games.

Interrupts appear as pauses or stuttering in the video playout. Interrupts affect immersion games, like role-playing games or games with smooth camera and landscape movement. Both delays and interrupts can impact a game’s QoE which is why it is necessary to quantify their impact so an ideal buffer size can be created that results in minimal added delay and minimal interrupts. We hypothesize that not all types of games need the same buffer size, so being able to see how delays and interrupts differently impact games informs how that buffer size needs to be changed.

Our study aims to assess exactly how delay and interrupts impact QoE by conducting user studies. We manipulated both delay and interrupts while a user played through four different games, two created by us to test specific interactions and two commercially available games in Stadia’s store to test full games embedded with the specific interactions. The delay and interrupts were added into the system using a Raspberry Pi as a pseudo-router and scripts with NetEm commands that added in the specified delay or interrupts. To see how the different values
impact their experience, the participants filled out a QoE survey between rounds to assess their experience.

The study yielded results that align with our hypothesis. In general, delays and interrupts do negatively impact QoE. The higher the delay that was added the lower the QoE particularly for timing and precision games. The greater the interrupts in the playout the lower the QoE for games with continuous motion. It appears that the number of interrupts did not matter as much as the length of the interrupt, for the range of values tested. We also analyzed how delays and interrupts affected score and the relationship between QoE and score. As expected, when delays or interrupts were added into the system, scores went down. The relationship between QoE and score was positive; QoE values went up as score went up.

The rest of this paper is structured into different sections detailing key information and the process of the study. Chapter 2 gives background information on the cloud-based game streaming as well as defines terms that are relevant to our study. Chapter 3 has the related works that discusses previous studies and their relevance. Chapter 4 details the methodology for our study, including the games used and what participants specifically did. Chapter 5 describes the results from the study and analyzes what those results mean. Chapter 6 discusses possible next steps to be taken from this study using our current results. Chapter 7 concludes with a summary of the study and data analysis, including broader implications for buffer sizing.
Section 2 Background

This chapter gives insight into the technologies we use in our study. First, we give an introduction on cloud gaming as a whole, how it works, and why we are focusing on specifically Google Stadia. Second, we go into detail about what buffers, delay, and interrupts are, how they work in general as well as with Stadia, and how they contribute to Quality of Experience (QoE). Lastly, we discuss how we plan on using these concepts and what we aim to achieve at the end of this study.

Section 2.1 What is Cloud-based gaming streaming?

Traditional gaming is best explained by installing the game on a computer locally and utilizing the hardware performance of the computer. The way a player experiences traditional gaming greatly depends on the capabilities of the hardware. However, for cloud gaming, the game engine processing happens on powerful servers which are normally located in data centers. Cloud gaming servers send the game frames rendered as a video stream to any computer connected over the Internet. The user is able to play the game by installing a client program (i.e. an application) or simply using a browser, which is able to access servers where the game is running. The client renders and plays frames as a video playout and their inputs are sent back to the server to then be sent back to the client to show the results of their inputs.

Section 2.1.1 How does Cloud-gaming work?

Cloud gaming is not a new technology. Over the past decade, technologies that support cloud gaming have been well developed. For example, video streaming services like Netflix, Youtube, and remote desktops operate similarly to cloud gaming. The differences are video streaming services do not have frequent interactions and clients only receive the information passively. Remote desktops behave almost the same as cloud gaming since servers send the
video stream of the desktop and it receives input from the client (Cai, Yu, & Zhou 2004). But cloud gaming has more critical requirements for graphical quality and responsiveness, therefore, needs more bandwidth and lower network latency.

In general, cloud gaming stores and executes games remotely on a provider’s dedicated servers and sends the video stream to client’s devices via the client software or a browser. The client software or browser processes the player’s inputs and sends them back to the server to execute in games.

The advantage of cloud gaming is that players only need lightweight clients without a full game install and updates. Cloud gaming also can be available in a wide variety of devices, including smartphones, PCs, and cloud gaming consoles like Google Chromecast. However, cloud gaming demands a high-speed and low latency connection to the Internet because of high bit rates required for the video streaming and the added round-trip latency for all player actions. High latency or low capacity connection to the internet could cause the game to be unplayable.

**Section 2.1.2 History of Cloud-based game streaming**

The first commercial attempt of cloud gaming was made by G-Cluster which introduced its cloud gaming product in 2000 (W3, 2022). But at that time the infrastructure and Internet connections could not support the high bitrate demands. In 2009, a notable commercial cloud gaming service, called OnLive, was revealed and officially launched in 2010 with the sale of its OnLive microconsole (W5, 2022). Its subscription price model did not get widely supported by publishers. At the same time, another company, Gaikai, started its game streaming service (W4, 2022). Instead of streaming full games, Gaikai did streaming game demos, making the service a form of game advertising and got far more support from publishers. Both Gaikai and OnLive were acquired by Sony Corporation and used as the foundation for PlayStation Now.
Current mature cloud gaming services include NVidia GeForce Now, Microsoft XCloud, Sony PlayStation Now, Amazon Luna, and Google Stadia, which is used for this project. Because of the rapid development of infrastructure and Internet technology, these cloud gaming services are able to offer players good qualities of experience with higher graphic quality and better responsiveness, making cloud gaming increasingly popular.

Section 2.2 What is a buffer?

Client side buffers can help with any compensation needed for a user’s internet connection. In general, a buffer is a pre-allocated section of memory used to temporarily store information. For video streaming specifically, a client-side buffer holds frames from the video being streamed that have yet to be displayed. The idea here is that network connections are not always stable, in fact there is almost always some kind of network variance at play no matter how strong one's connection may be. Thus, while streaming a video without a buffer, the viewer may experience issues with the playback of the video. Anything from some jitter to outright interrupts in the video may occur. A buffer can help reduce frame jitter. The buffer stores the frames of the video, and as the video is being played it sends out the frames to be played while receiving in the frames for it to store. The buffer can hold anywhere from a few seconds to several minutes worth of frames, and thus even if the network connection dips in quality for a short time there are still frames of the video to play until that connection is restored.

Finding the right buffer size is important. Very large buffers can prevent any interrupts from occurring once a video starts playing. However, the drawback is the delay it adds to start play and to all user interaction. In order to shorten this time, the buffer could be made to be very small. This is what causes the interrupts. If network connection gets disturbed for even a small
amount of time, the buffer will not have enough frames to continue to play the video uninterrupted, and thus it will get stuck until the buffer fills again.

All video streaming services from YouTube to Netflix use client-side buffers to improve playback quality. The size used for the buffers varies from service to service, but all generally work the same way. When a video is started, the buffer begins to fill with the frames of the video. Not all of these frames are an actual full image though, some of them are just information about what has changed since the last frame.

Section 2.3 How does Stadia use buffers?

Stadia leverages buffers in a similar way to streaming services since it basically fuses a video feed from the server and streams it to the client over the Internet and the buffers that Stadia uses holds the frames of the game that need to be played out. However, because Stadia is interactive and not just a video being played out, factors like input and response need to be taken into account which is somewhat related to how video conferencing works. Video conferencing is another form of interactive media that has to take in real-time input and output. Buffering and interrupts would have a similar impact in this space as they would to Stadia and would be noticed in a similar way by the user. In a video conference, buffering happens more during the call rather than before since it does not have anything to load before the call happens, but it can be perceived by the viewer with low quality of video, low framerate, and other impacts to the video feed. These are similar to how they would be perceived in Stadia.

The way buffers work in Stadia before starting gameplay is similar to how a streaming video works, holding as many frames as it needs to before launching, which can be shown by a loading screen. However, if it needs to buffer while gameplay is being played out this can lead to interrupts. Interrupts in gameplay degrade player quality of experience (QoE). For example, in a
platformer where the screen is moving with the player, interrupts can cause a user to miss a jump or, for a rhythm game that depends a lot on timing, buffer delay can cause a user to not hit the button in time and lose that perfect combo.

**Section 2.3.1 Why it is important to measure the effects of buffers**

In order to account for delay-interrupt tradeoffs, the client and the server are constantly communicating back and forth in case the user has a change in Internet speed and they need to change the buffer size to accommodate a lower speed. However, this can create negative impacts on the user’s experience. When the user’s network bitrate drops and Stadia has to accommodate this, one of the methods it uses is to lower the stream quality but it is still able to keep up with the user’s inputs (Carrascosa & Bellalta, 2020). If the user’s bitrate drops further, then they may start to experience lag and framerate drop, which can cause them to stop playing. Visually, in bad Internet situations, users may experience low quality video, low framerate so it looks stuttery, and there may even be times when the framerate drops and then picks up which to the user would look like the video slowing down and then speeding up as if fast forwarding. All of these can lead to a negative QoE, which is why it is important to measure these factors so that they can be used to adjust buffer sizes appropriately.

**Section 2.4 How we will use these ideas**

Buffers are a way streaming services compensate for bad connections and to smooth out playback. In our case, it is used for streaming cloud games. Our study looks at the effects that delay and interrupts have on gameplay without a buffer present. This will give insight into how these factors truly impact QoE. This can be used in further work to see how the values differ when a buffer is added to smooth out the delay and interrupts, which will be discussed in a later section.
Section 3 Related Work

This section describes previous studies done that are relevant to our study. There have been studies done on how buffering works in traditional video streaming. However, there have not been many on Cloud-based game streaming but they use similar technologies. The previous studies provide us with tactics we use in our own study.

Section 3.1 Buffer sizes in Internet video streaming

Ozbek and Tunali look to find the effect client-side buffer size has on perceived video quality, and what size works the best (Ozbek & Tunali, 2006). They ran 100 experiments, each lasting 10 minutes, with different combinations of WAN/LAN, turning their bandwidth estimator on or off, and using five different buffer sizes. They measured the average kb/s, the number of changes in kb/s, the average frame discard level, and the number of changes in the frame discard level. After analyzing all of the data, they found that if bandwidth could not be estimated, a buffer of size 5 works the best (they do not specify the size of the buffer exactly, but simply refer to them as sizes 1, 2, 3, 4, 5 with 5 being the largest buffer size). If bandwidth can be estimated, they suggest a buffer size of 5 for high-bandwidth, and a buffer size of 3 for low-bandwidth.

Section 3.2 Buffers for cloud gaming based on video streaming

Since cloud based gaming is largely a new field, there have yet to be many research papers about some of the more niche topics within it (such as client-side buffers). However, He, Liu, and Yuchen, they do explore the area of buffers in cloud gaming(He, Liu, & Yuchen, 2014). Specifically, they propose an algorithm that would estimate the status of the client-side buffer, use that to calculate the urgency for the packets, which can then be used to improve packet scheduling. In simpler terms, the algorithm should enhance playback continuity. To test their algorithm, they pit it against 4 other similar algorithms and measure which of them performs the
best. Their metrics for determining this are received Peak Signal to Noise Ratio (PSNR) and playback continuity. Their proposed algorithm is found to outperform the others in both of these categories. Although this paper does not cover buffer sizing specifically, it is still a relevant study done within the same field as our paper.

Section 3.3 QoE For Multi-Party Video Conferencing

In the Background chapter, we mentioned how videoconferencing is similar to cloud gaming in the way it has to work in real time with streaming and inputs. Schmitt, Redi, Bulterman, and Cesar conducted a study to see how changes in a system affect multiparty video conferencing (Schmitt, Redi, Bulterman, & Cesar, 2017). For this study, they gathered groups who were familiar with each other and had them build legos over a video conference, one had to show/give instructions while the other built. To assess how changes in the system affected their QoE they encoded different bitrates to mimic different Internet speeds. Lower encoding (lower internet speeds) caused the video quality to degrade and even lead to more robotic sounding speech. When this occurred, they discovered that older users got more frustrated and rated a lower QoE whereas younger users adapted and their QoE was less affected. They determined that QoE was influenced by engagement and enjoyment of the task. The study also determined that when trying to accurately estimate QoE of participants, system factors alone could be taken into account and that there needs to be an understanding of what the users are using it for to balance the quality. This study shows how network latency can affect QoE in interactive media. They implement similar strategies that we would like to implement in our own study, such as controlling internet issues, for this study they encoded bit rates to simulate bad internet speeds, for our study we will do something similar but to simulate delays in the system. Even though this was done with video conferencing and not cloud gaming, they are still in a similar space
with needing to work in real time to convey important info. For example, build instructions to each other versus a user’s button input that needs to be sent to the server.

**Section 3.4 Impact of Buffering and Interrupts on Streaming Video QoE**

Allard, Roskuski, and Claypool seek to put an actual number to how QoE is affected when a user is faced with buffering before a video starts compared to interrupts during playback of said video (Allard, Roskuski, & Claypool, 2020). To accomplish this, they gathered short (~30 second) clips from seventeen various videos and “baked in” buffering and interrupts of 2, 4, 8, and 16 seconds. The interrupts were evenly spaced though the videos and lasted exactly one second. They then gathered a total of thirty-seven participants to watch the videos and rate each one, both in how annoyed they were with the quality, as well as their opinion on the content of the video. Ultimately, they found that interrupts were perceived as roughly two times as annoying as buffering. In our own paper, we hope to run a similar experiment, except with input delay versus interrupts or jitter.

**Section 3.5 An evaluation of QoE in Cloud Gaming**

Jarschel, Schlosser, Scheuring, and Hoßfeld emulate a cloud gaming environment to test how packet loss and delay affect user’s QoE (Jarschel, Schlosser, Scheuring, & Hoßfeld, 2011). They created different scenarios to test varying delays and packet loss. They test no delays and no loss as a control, delays without loss, loss without delays and then both. They aimed to answer the question of which has a bigger impact on QoE. Their test group consisted of 14 year old and older casual/leisure gamers. They set up their cloud gaming emulator using a Playstation 3 as the data center that sends its stream to a Spawn Box that is put through an internet emulator using NetEm and then to the client. Users played an omnipresent game (Pro-Evolution Soccer), a 3rd person perspective game (Final Fantasy XIII), and a first person perspective game (Gran
Turismo HD), these were chosen based on their different gameplay types, slow to fast paced. The tests revealed that depending on how fast paced the game is, users perceived either delay or packet loss to have more of an impact on their QoE. Delay was more of an impact for fast paced games, whereas for medium paced games packet loss has more of an impact, and slow paced games did not show either having more of an impact. They also determined that fast paced games are more delay sensitive than slower paced games and users do not perceive packet loss as much as delay since they are not paying as much attention to their surroundings and don’t notice a small frame drop. This study is very similar to the one we will conduct but instead of creating a cloud server to use, we are using Google Stadia. We will also be using NetEm to emulate different network settings like adding delay and packet loss. Our study will aim to answer similar questions but more specified to Google Stadia.
Section 4 Methodology

In order to determine how delay and interrupts impact a user’s Quality of Experience (QoE), we created a methodology in which human participants play both custom games and the commercial games.

Section 4.1 Custom Games

As mentioned previously, two games were created using Unity. These two games aim to be able to test the edges of the delay and interrupts spectrums by isolating specific interactions, something difficult to do in commercial games.

Section 4.1.1 Whack-a-Goat: Delay Sensitive

Figure 1: Whack-a-Goat Custom Game

The first game is called Whack-a-Goat which has a similar idea to the game Whack-a-Mole (W7, 2022). Figure 1 displays the look and layout of Whack-a-Goat. The goal of the game is to gain points by selecting the green button on the screen. Only one button turns green randomly, but not where the cursor is currently, and not four spaces (i.e. in the opposite corner, if
the cursor is currently in one of the corners) away from the cursor. The participant uses the arrow keys to maneuver a cursor around the grid and the space bar to hit the button.

When the participant hits the green button, an image of a goat appears on the button and a bleat sound plays. If the participant continues to hit successive green buttons in a row they start a streak and it increases the score with a multiplier. If they hit a red button, an image of a goat also appears and a goat scream sound will play. This causes the streak to be lost and the score to decrease. If the participant does not hit any buttons, it counts as a miss and has the same outcome as if they hit a red button.

We wanted this game to be delay sensitive and interrupt insensitive. In order to accomplish this we tried to limit the amount of camera movement and animations, so that interrupts in the playback would be less perceptible. To this end we made the camera completely stationary, and cursor movement instantaneous, meaning there would be no animations to depict losing quality due to jitter. To make the game more delay sensitive, we made sure that the amount of time the user had to hit the button was just enough to usually reach it in time, but not quite enough if a sizable amount of delay were added. We included the cursor to help visualize this. As opposed to having a button on the keyboard correspond to each button on the screen, being able to see the cursor slightly lagging behind their inputs has intended to enhance the presence of delay.
Section 4.1.2 Goat Runner: Interrupt Sensitive

The second custom game is called Goat Runner and is in the style of an endless runner game (W2, 2022). Figure 2 displays the look and layout of this game. The participant uses the left and right arrow keys to move the goat between 3 lanes, moving left and right respectively. The up and down arrow keys switch the goat’s color between red and blue. If the goat runs into an obstacle of the same color, particle effects appear, and points are added to the score depending on the current streak of obstacles destroyed. However, if the goat runs into an obstacle of the other color, it flashes white and loses its color and the score decreases.

We wanted this game to be interrupt sensitive and delay insensitive. In order to achieve interrupt sensitivity, we made sure that the camera was moving at all times. We also made an animation for the goat running, and included the moving particle effects when an obstacle is destroyed. All of these hopefully made any interrupts in the playback more apparent. To reduce delay sensitivity, we tried to make the game less fast-paced than Whack-a-Goat by having the
goat’s speed be relatively slow. We also chose to make the obstacles a source of points, instead of something to avoid, to minimize the feeling of needing to react quickly and replace it with having the player strategize to gain more points.

Section 4.2 Commercial Games

We chose two commercial games, Avicii Invector (Wired Productions, Hello There AB, 2019) and The Falconeer (Wired Productions, 2020), from the Google Stadia Pro library to be able to have a better idea of how delay and interrupts affect a commercial game.

Section 4.2.1 Avicii Invector: Delay Sensitive

Figure 3: Avicii Invector Commercial Game

The first commercial game we chose is Avicii Invector, shown in Figure 3. Avicii Invector is a rhythm game (W1, 2022) so it is sensitive to delay. However, it does also have continuous motion which makes it interrupt sensitive. As with most rhythm games, the goal is to press the right buttons at the right time to the beat of the music. The more buttons correctly hit in
a row, the more the score increases, and the faster the pace gets. Participants use a keyboard to play, along with headphones so they can hear the song they are playing.

**Section 4.2.1 The Falconeer: Interrupt Sensitive**

![Figure 4: The Falconeer Commercial Game](image)

The second commercial game we chose is The Falconeer, shown in Figure 4. The Falconeer is a free roaming flying game with mini objectives (W6, 2022). The continuous motion of the wing flying animation and the background going past makes it interrupt sensitive. In the game sections we use, the game has minimal, if any, timing related interactions which makes it delay insensitive. We ask participants to fly around the tutorial level, going to various locations and back again. Participants use a keyboard to play.
Section 4.3 Test Bed

*Figure 5: A Diagram of the Room Used in the User Study*

The room which was used for the user study is pictured above in Figure 5. The participants sat at the dark blue desk on the left. They played all of the games using the monitor, keyboard, and mouse marked with “1” in the figure above, and used the monitor, keyboard, and mouse marked with “2” for filling out the quality of experience survey. Additionally, the commercial Stadia games took some additional time to load compared to the custom games, so
those would be preloaded on the computer connected to monitor 3 while the proctor explained the instructions for the game.

Figure 6: Diagram of the Test Bed

Figure 6 depicts the layout of the testbed and how each component is connected. For the study, we have one Raspberry Pi set up to act as a router (Router 1). This router runs on Linux OS version 20.04 and is used to add in the delay and jitter to the system. The user study computer (client) runs on an Intel(R) Core(TM) i7-4790K CPU @ 4.00 GHz, Nvidia GeForce GTX 1080 GPU with 8 GB RAM, 16.0 GB RAM, and Windows Version 10 Enterprise. The monitor used to run the games is 24.5”, 1080p resolution, 240Hz refresh rate, and 1ms response time. The keyboard and mouse we used was the Logitech g910 Orion Spectrum and the Logitech g502 wired. The headphones used were a pair of Sony headphones outputting 24 bit 48kHz audio. The games were run on Chrome version 98.0.4758.102 (64 bit). The commercial games (server) used an undisclosed Stadia version between the days of February 9th 2022 to February 22nd 2022. The custom games were made using Unity version 2020.3.9f1 and ran on Stadia’s test portal with SDK version 1.68.0.23560 and platform core 1.74.0.25274. The delay and jitter are added into the system through the Linux router by running their corresponding scripts and changing the values. For delay, this means changing how many milliseconds of delay using the
NetEm. For example, the command “sudo tc qdisc add dev eth0 root netem delay 1ms” would add 1ms of delay into the system. For jitter, there are two settings: frequency and magnitude. For jitter magnitude, we changed how many milliseconds we wanted the interrupt to last for. Jitter magnitude is added into the system using the NetEm command “sudo tc qdisc add dev eth0 root netem delay 0.1ms 25.5ms distribution f100 rate 1000mbit”, where 25.5ms is the magnitude that will be changed and f100 is a set frequency. The jitter frequency was determined by the number of initial zeroes used in a NetEm script. The zeros are translated to an amount of interrupts per second with more zeroes meaning fewer interrupts per second. We computed the average of how often interrupts occurred with 400, 250, and 100 zeros, which came out to about 0.57 interrupts/sec, 0.99 interrupts/sec, and 2.12 interrupts/sec respectively. Jitter frequency is added into the system using the NetEm command “sudo tc qdisc add dev eth0 root netem delay 0.1ms 50ms distribution f250 rate 1000mbit”, where 50ms is a set magnitude and f250 is frequency that can be changed.

Section 4.4 User Study

Each participant played all four games. The order in which they played varied from participant to participant. They came to the test lab one at a time with a proctor in the room to help ensure the study ran smoothly. They were placed in front of a computer with the games set up and were given instructions on how to play. Each participant was given a chance to be acquainted with the controls before data gathering began.

Section 4.4.1 Iterations

Each participant had a different amount of delay and/or jitter added, with jitter having both frequency and magnitude changed. Before each study, the script chose a random value for each, within a range:
● Delay: 0ms-100ms
● Jitter Frequency: 400 zeros-100 zeros (0.5s interrupts/sec-2.2 interrupts/sec)
● Jitter Magnitude: 40ms-100ms.

The participants played through the rounds with these values being put in a random order each time. The different iteration versions are laid out below:

1. Version 1 (1 total) : no delay added, no jitter added
2. Version 2 (5 total): delay added, no jitter added
3. Version 3 (5 total): no delay added, jitter frequency added, jitter magnitude constant at 100ms
4. Version 4 (5 total): no delay added, jitter frequency constant at 100 zeros (2.2 interrupts/sec), jitter magnitude added
5. Version 5 (1 total): delay added, jitter frequency added, jitter magnitude added

**Section 4.4.2 Procedure**

1. Participant signs the consent form
2. Participant starts up the first game (this is different for each participant)
3. Participant plays a trial round, 30s, of game
4. Participant plays through the different iterations in a random order (30s each)
   a. Takes QoE survey between each round
5. Repeat step 3-4 with the rest of the games

The study took about an hour of the participants’ time, sometimes longer if something went wrong. Participants were emailed a link to a $15 Amazon gift card after completing the experiment. They would also be provided an IMGD playtesting credit if they needed one.
Section 4.5 Preliminary Tests

We conducted some preliminary tests to test the ranges. One concern we had with the study was with the delay range. At high delays, Stadia would kick us out of the game. After conversing with the team from Stadia we learned that 80ms of delay is about the peak of the range that they test. We used 80ms as a starting point and increased the delay by 5ms during gameplay until we got kicked out. We also recorded when we got warnings from Stadia about the conditions.

The Falconeer:

- 80ms of delay before booting the game: warning before the game started and during gameplay
- 100ms during gameplay: warning
- 110ms during gameplay: warning and kicked off ~1 min into game play
- Starting with none -> putting 80ms once game was booted: warning ~1.5 min in, kicked off ~2min in
- Starting with none-> putting 100ms once game was booted: warning and kicked off (was not timed)

Avicii Invctor (we also tested what would happen if we went down by 5ms to see where the warnings started):

- 80ms before booting game: warning ~1min in
- 110ms during gameplay: warning ~1min in and kicked off ~2min in
- 75ms during gameplay: warning
● 70ms during gameplay: warning and kicked out (this is the only time this occurred, was not a concern)

● None -> 100ms: kicked out regularly

We decided on the ranges for delay based on the results of this testing. The warnings were recorded so we would know how often they may occur and where on the screen so we could cover it with tape. We determined that 100ms was a good upper bound since it mainly gave a warning and did not kick us out. We did notice that if we started the game with no delay added then added a lot of delay, Stadia would kick us out quicker. We decided that given the nature of the rounds of the study, that this was not a major concern. We did a small run through of the study to determine if the jumping of values would kick us out and they did not.

Section 4.6 Data Collection

Before starting the study, participants were asked to fill out a demographic survey to understand their background. During the study, their inputs and interactions were logged and accessed at the end of the playthrough with a script. Their scores in the games were also recorded, except in The Falconeer. The Falconeer does not have a progress bar or any score indication in the level so the data for this game is mostly dependent on the QoE survey results. The other games have a way for us to track progress and difference between each round. Whack-a-Goat and Goat Runner both have set round times in which we logged the scores at the end. Avicii Invector shows a constant update progress and score that we took a screenshot of at the end of the round and manually entered in. In between rounds the participant was asked to fill out another survey about their experience during the previous round. Both surveys can be found in Appendix A and Appendix B, respectively.
Section 4.7 Recruitment

Recruitment was done by sending an email to multiple mailing lists on campus. See Appendix C for the recruitment email that was used. Upon clicking the link to participate, the user was taken to our demographic survey. While we wanted to gather as many participants as possible, we also wanted to be prepared in case there are simply too many participants that sign up. Therefore, having them fill out the demographic survey beforehand allowed us to select our study pool with diverse participants who play video games regularly. Once we chose someone to enter the study, we allowed them to schedule a time to come in and participate. Included in this email was our consent form, which we required them to sign before they could participate. A copy of this consent form can be found in Appendix D.
Section 5 Analysis

This section analyzes the data received from the user studies conducted. Before the studies were conducted, we made some hypotheses on how the perturbations would impact the QoE. We have a total of four hypotheses, one for each game:

1. Whack-a-Goat: QoE ↓ Delay ↑, interrupts will have little to no effect
2. Goat Runner: QoE ↓ Interrupts ↑, delay will have little to no effect
3. Avicii Invector: QoE ↓ Delay ↑, QoE ↓ Interrupts moderate ↑
4. The Falconeer: QoE ↓ Interrupts ↑, delay will have little to no effect

We will refer back to these hypotheses when analyzing our data.

In Table 1, we have also included average best case QoE values and score values for each game. By “best case” we are referring to rounds without perturbation. We did not have any metric with which to judge performance in The Falconeer, so it does not have a score value.

<table>
<thead>
<tr>
<th>Game</th>
<th>Best Case QoE</th>
<th>Best Case Score (avg.)</th>
<th>Best Case Score (median)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicii Invector</td>
<td>4.35</td>
<td>886</td>
<td>988</td>
</tr>
<tr>
<td>The Falconeer</td>
<td>4.08</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>4.04</td>
<td>174</td>
<td>8566</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>4.05</td>
<td>170</td>
<td>10810</td>
</tr>
</tbody>
</table>

Table 1: Best Case Averages

Section 5.1 Demographics

We sent out the demographic survey as a screening survey to the Computer Science (CS) Department and the Interactive Media and Game Development (IMGD) Program. We received a total of 76 responses on that screening survey. From there, we selected participants who had
between 5 and 20 hours of gameplay a week. We initially chose this range to get people who had a fair amount of experience in the gaming world. In order to get a few more participants, we then invited more people by extending the range to anyone above 3 hours of gameplay a week. With both rounds of invitations we had a total of 33 participants for our user study. The age range of the participants was 18-29 years old with average and median of 19 years old. Participants played an average of 13.76 hours a week of games. There were 22 people who had no cloud gaming experience and 12 that had some. The participants had a large range of the types of games that they played from adventure/role-playing games to rhythm/music games, with 93% playing more than one type of game.

Figure 7: Gender Demographics
Figures 7 and 8 show the gender and major demographics of the participants we had. Most participants were male with a 26:6 ratio with one participant preferring not to say. Most of the participants we had were CS and IMGD majors with one Aerospace major.

Section 5.2 Delay Analysis

The graphs for the delay analysis show delay that was added, not the total delay in the system. The base delay for our client to stadia servers was around 12-13ms. We discuss how delay affected the QoE and how it affected the score of the games that we were able to gather scores for.
Section 5.2.1 Delay versus QoE

![Graphs showing the relationship between delay and QoE for various games.](image)

**Figure 9: Delay vs QoE graphs w/ Linear Regression line**

<table>
<thead>
<tr>
<th>Game</th>
<th>Slope</th>
<th>Y-Intercept</th>
<th>R² value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicii Invector</td>
<td>-0.007</td>
<td>4.36</td>
<td>0.062</td>
<td>0.001</td>
</tr>
<tr>
<td>The Falconeer</td>
<td>-0.0014</td>
<td>4.12</td>
<td>0.002</td>
<td>0.53</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>-0.011</td>
<td>3.76</td>
<td>0.085</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>-0.0039</td>
<td>4.02</td>
<td>0.018</td>
<td>0.088</td>
</tr>
</tbody>
</table>

**Table 2: Linear Regression for delay**

Figure 9 displays the graphs generated from the QoE surveys and from the perturbations of the delay for each user and game. Table 2 displays the R² values for each game. The graphs show that delay had a rather significant impact on the QoE for Whack-A-Goat, as well as Avicii Invector. Each shows a clear downward trend as delay worsens, while The Falconeer and Goat Runner see much less of an impact.
This implies that games that require precise, time-sensitive inputs (high precision and tight deadline) are most affected by delay. This would imply that games with high precision and tight deadlines are impacted much more by the presence of a buffer.

**Section 5.2.2 Delay versus Score**

We first discuss the custom games each round as 30 second and with scores taken from the log.

![Figure 10: Delay vs Score graphs w/ Linear Regression line](image)

<table>
<thead>
<tr>
<th>Game</th>
<th>Slope</th>
<th>Y-Intercept</th>
<th>R(^2) value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicci Invector</td>
<td>-5.52</td>
<td>1018.45</td>
<td>0.22</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>-0.4572</td>
<td>166.36</td>
<td>0.049</td>
<td>0.005</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>-29.57</td>
<td>9684.71</td>
<td>0.056</td>
<td>0.002</td>
</tr>
</tbody>
</table>

*Table 3: Linear Regression with delay*

As the Figure 10 and Table 3 above illustrate, scores decreased as delay increased. This is what we expected to happen, especially with a very time-sensitive game like Whack-A-Goat. However, it appears that the scores for Goat Runner were similarly impacted, despite having a
much lower drop in QoE when delay was added. This suggests that despite delay having a similar impact on performance, it was much less noticeable, or at least much less annoying, in Goat Runner than in Whack-A-Goat. This could be for a number of reasons: perhaps Goat Runner’s slower pace made the players care less about their score, or perhaps the animation and particle effects were enough to keep the QoE up even when they got a lower score.

Now we will discuss Avicii Invector, which we manually entered data for using screenshots captured at the very end of each round. This linear regression line is quite clear in displaying just how much of an impact delay had on a user’s performance. The average score with 100ms of delay added is half of that with little to no delay added. We would have thought that such a drastic drop in performance would have made users more inclined to rate their QoE lower, and while that certainly was the case, it was not to the extent that we had thought it would be. Once again, this could be due to the game having other qualities that the user’s found charming, making their drop in score less harmful to their experience.
Section 5.2.3 Score versus QoE with Delay added

Figure 11: Score vs QoE graphs w/ Linear Regression line

<table>
<thead>
<tr>
<th></th>
<th>Slope</th>
<th>Y-Intercept</th>
<th>$R^2$ value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicii Invector</td>
<td>0.001</td>
<td>3.29</td>
<td>0.17</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>0.0072</td>
<td>2.15</td>
<td>0.15</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>8.618E-5</td>
<td>3.12</td>
<td>0.14</td>
<td>&lt;0.0001</td>
</tr>
</tbody>
</table>

Table 4: Linear regression scores vs QoE with delay

Figure 11 displays the graphs generated from the QoE survey and the scores gathered from the Avicii Invector screenshots and custom games’ logs. As shown, there is a clear upward trend, implying that as the participants score improved, so did their QoE. This is to be somewhat expected as people generally have a better time experiencing things when they are rewarded in some way, like getting a better score. It appears that Whack-a-Goat has a much steeper trend line than Goat Runner, this could be that since Whack-a-Goat is more time and reaction based it has a higher reward when getting a higher score. This would also add to the notion that delay would significantly impact their QoE since it impacts the score quite a bit as shown in the earlier data.
Similarly with the custom games, the Avicii Invector graph shows a clear upwards trend with QoE improving as score improves. Avicii Invector is similar to Whack-a-Goat because of it’s timing and reaction sensitivity, but shows a more similar trend to Goat Runner. This could be since it has so much continuous motion and animation it is more similar to Goat Runner than Whack-a-Goat than we had initially intended.

Section 5.3 Jitter Magnitude Analysis

The jitter magnitude was one of the ways we could add interrupts into the system, this would be how long an interrupt was. We kept a constant high frequency value. Just like with the delay analysis, we will discuss how the magnitude impacted QoE and score.

Section 5.3.1 Jitter Magnitude versus QoE

Figure 12: Jitter Magnitude vs QoE graphs w/ Linear Regression line
Table 5: Linear Regression for jitter magnitude

Figure 12 shows the graphs generated from the QoE survey and the jitter magnitude values. Table 5 shows the $R^2$ values for each game. There is clearly a downward trend in these graphs. Each of the games, with the exception of Whack-A-Goat, clearly get worse as the jitter magnitude gets worse.

This would imply that games involving a lot of smooth camera motion, animations, particle effects, etc. are much more sensitive to changes in jitter than games without. This would also imply that these games would be the most susceptible to a user receiving a poorer quality of experience due to a bad internet connection.

<table>
<thead>
<tr>
<th>Game</th>
<th>Slope</th>
<th>Y-Intercept</th>
<th>$R^2$ value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicii Invector</td>
<td>-0.017</td>
<td>3.82</td>
<td>0.074</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>The Falconeer</td>
<td>-0.012</td>
<td>3.89</td>
<td>0.045</td>
<td>0.007</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>-0.0047</td>
<td>4.27</td>
<td>0.011</td>
<td>0.2</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>-0.016</td>
<td>3.87</td>
<td>0.079</td>
<td>&lt;0.0001</td>
</tr>
</tbody>
</table>
Section 5.3.2 Jitter Magnitude versus Score

Figure 13: Jitter Magnitude vs Score graphs w/ Linear Regression line

<table>
<thead>
<tr>
<th></th>
<th>Slope</th>
<th>Y-Intercept</th>
<th>$R^2$ value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicii Invector</td>
<td>-5.62</td>
<td>905.43</td>
<td>0.095</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>0.0924</td>
<td>169.12</td>
<td>0.001</td>
<td>0.705</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>-36.44</td>
<td>9472.4</td>
<td>0.035</td>
<td><strong>0.016</strong></td>
</tr>
</tbody>
</table>

Table 6: Linear Regression for scores with jitter magnitude

Figure 13 shows the users’ score values versus how much jitter was added when specifically changing the magnitude. The score values for Whack-A-Goat are, for all intents and purposes, flat as jitter magnitude is increased. This is also reflected in the QoE versus jitter magnitude graph, as Whack-A-Goat only has a very slight downward trend there as well. The graph for Goat Runner on the other hand shows a clear downward trend in score as jitter magnitude is increased. This drop in performance just about matches what we expected to see when looking at the QoE versus jitter magnitude graph, as Goat Runner saw a clear downward trend there as well.
Avicii Invector also displays a clear downward trend in score as jitter magnitude is increased, even more so than Goat Runner. This was also as we expected, as the QoE for Avicii Invector dropped significantly with rising jitter magnitude values.

Section 5.3.3 Score versus QoE with Jitter Magnitude changed

Figure 14: Score vs QoE for Jitter Magnitude w/ Linear Regression line

<table>
<thead>
<tr>
<th></th>
<th>Slope</th>
<th>Y-Intercept</th>
<th>R² value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicii Invector</td>
<td>0.0012</td>
<td>2.42</td>
<td>0.128</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>0.0022</td>
<td>3.74</td>
<td>0.021</td>
<td>0.07</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>5.793E-5</td>
<td>2.88</td>
<td>0.038</td>
<td>0.012</td>
</tr>
</tbody>
</table>

Table 7: Linear regression for scores vs QoE with jitter magnitude

Figure 14 displays the graphs generated from the QoE survey and score data acquired from the Avicii Invector screenshots and custom games’ logs. There is still a clear upwards trend, however it is not as steep as it was with delay. The Avicii Invector graph shows a clearer upwards trend of how score impacted QoE. Based on the graphs previously discussed, jitter magnitude did have a larger impact on Avicii Invector than we initially thought it would and
therefore has a larger impact on the Score versus QoE relationship than delay did. For Whack-a-Goat the slope is not that steep, which is to be somewhat expected since jitter does not impact the QoE or score as much as delay did, shown by the previous figures. Goat Runner has a steeper slope, which can be attributed to how jitter magnitude affected QoE and score individually. This still implies that as the score improved so did QoE even when jitter magnitude was changed.

Section 5.4 Jitter Frequency Analysis

Jitter frequency was the other way we evaluated interrupts in the system, specifically the frequency of interrupts per second with a constant magnitude of 100ms. Again, we will be looking at how jitter frequency affected QoE and score.

Section 5.4.1 Jitter Frequency versus QoE

![Jitter Frequency vs QoE graphs w/ Linear Regression line](image)
Table 8: Linear regression for jitter frequency

Figure 15 shows the graphs generated from the QoE survey and the jitter frequency values. Table 8 shows the $R^2$ values for each game. Based on the graphs, it appears as if jitter frequency does not have much of an effect as it increases. However, compared to the average rating of each game without any perturbation, we can see that all games (with the exception of Whack-A-Goat) have much lower ratings with jitter included. The little change could be because the “low” value we selected for testing jitter frequency was too high. Therefore, when rating their quality of experience, the participants did not see much difference between the low and high values. This could also be interpreted as the amount of interrupts in a game are not as impactful as the length of interrupts, given that jitter magnitude appears to have had more of an impact on the participants QoE.
Section 5.4.2 Jitter Frequency versus Score

As Figure 16 shows above, performance in both custom games was largely unaffected by increases in jitter frequency. Surprisingly, Whack-A-Goat does have a noticeable, albeit slight, downward trend. Oddly enough this is also supported by the QoE versus jitter frequency graph, with Whack-A-Goat having the most noticeable downward trend there as well. We are unsure what the cause for this could be. It is possible that at high jitter frequency values the amount of interrupts became enough to cause some input delay, which would explain the drop in performance and thus QoE. As for Goat Runner, score is entirely unaffected by increases in jitter frequency, as was suggested by the QoE graph.

Table 9: Linear regression for scores with jitter frequency

<table>
<thead>
<tr>
<th>Game</th>
<th>Slope</th>
<th>Y-Intercept</th>
<th>R^2 value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicii Invector</td>
<td>-0.6067</td>
<td>742.97</td>
<td>0.026</td>
<td>0.039</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>-0.0982</td>
<td>184.52</td>
<td>0.023</td>
<td>0.061</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>0.9027</td>
<td>8416.26</td>
<td>0.000</td>
<td>0.783</td>
</tr>
</tbody>
</table>
The scores for Avicii Invector do appear to go down as jitter frequency is increased. While we expected some drop in the score, the amount that it dropped is more than what we had anticipated when looking at the QoE versus jitter frequency graph. As with that graph, Figure 16 shows that the average score even with low jitter frequency added is much lower than the best case average. The increase in jitter frequency makes the score drop further still.

**Section 5.4.3 Score versus QoE with Jitter Frequency changed**

![Graphs](image)

**Figure 17: Score vs QoE for Jitter Frequency w/ Linear Regression line**

<table>
<thead>
<tr>
<th></th>
<th>Slope</th>
<th>Y-Intercept</th>
<th>R² value</th>
<th>P value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avicii Invector</td>
<td>0.0013</td>
<td>2.23</td>
<td>0.15</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Whack-A-Goat</td>
<td>0.0024</td>
<td>3.64</td>
<td>0.03</td>
<td>0.032</td>
</tr>
<tr>
<td>Goat Runner</td>
<td>3.77E-5</td>
<td>2.97</td>
<td>0.017</td>
<td>0.10</td>
</tr>
</tbody>
</table>

**Table 10: Linear regression for scores vs QoE with jitter frequency**

Figure 17 shows the graphs generated from the QoE survey and final scores gathered from the screenshots and logs. The Avicii Invector graph shows a clear upwards trend, with a much steeper slope than those of the custom games, meaning that the score when frequency of
interrupts are added impacted the QoE more so in Aviciii Invctor than in the custom games. This behavior does match what was expected since the graphs for QoE and score individually showed downwards trends. Again, Aviciii Invctor was more interrupt sensitive than we initially hoped. The graph displays upward trends for both Whack-a-Goat and Goat Runner, however Whack-a-Goat appears to have a steeper slope. Based on the previous graphs with how jitter frequency impacted QoE and score individually, this isn’t surprising, however given that the nature of Whack-a-Goat was to be interrupt insensitive it is difficult to place a reason as to why jitter frequency impacted it more than Goat Runner.

**Section 5.5 Analysis Summary**

The user study data mostly support our hypotheses. For Whack-a-Goat, there was a clear negative relationship between QoE and delay (H1). However, as mentioned previously, it was impacted by jitter frequency more than we had initially thought since it was not supposed to be interrupt sensitive at all and Goat Runner was supposed to be more sensitive.

Goat Runner was more sensitive to the magnitude of the interrupts which does support our initial hypothesis that QoE would be more impacted by the interrupts (H2). However, it appeared as though the length of interrupts mattered more than the frequency for Goat Runner.

Aviciii Invctor also showed a clear negative relationship between delay and QoE, but also showed a much larger negative relationship between jitter magnitude and QoE (H3). We hypothesized that interrupts would have some effect since there was a lot of continuous motion and animation, but we did not think it would have impacted more than delay did. Jitter frequency did have some impact, but not as much as magnitude.

The Falconeer showed the relationships we expected, with there being a negative relationship between jitter magnitude and QoE, again jitter frequency having much less of an
impact. There were not any unexpected cases with The Falconeer. The major conclusions to
draw from how the perturbations impacted QoE is that the length of an interrupt impacts the QoE
more so than the frequency of them and that there is mostly a negative relationship between
them.

The score data for each game all showed a positive relationship, opposite to their
respective individual QoE and score graphs. This makes sense as whole, that QoE would
improve as score improves, it is just a matter of how much a perturbation impacts the gameplay
as a whole.
Section 6 Future Work

There are additional areas of future work in cloud based game streaming. Some future work to directly follow this study could include a more in-depth analysis of the data we gathered. We only had the time to make a linear regression line for each of our graphs, but the relationship between QoE and delay & jitter may be more complicated. Using a sliding window approach, where the average value is taken for a small range of values and then the range is moved over by one and the process is repeated, could provide a better representation for the QoE. If given the time, we would likely do this with a window of one quarter to one fifth of the total range of the data, and make new graphs to match the old ones with points plotted at each average for each window.

There was a ping script run for each round to record the actual delay experienced. Unfortunately, depending upon how quickly the user acted, the ping time might not have accurately captured the perturbed network condition. Future work is to examine the ping files in detail and, where possible, extract and analyze QoE versus the actual delays experienced. Similarly, future work is to analyze the QoE versus actual frame display times experienced, as recorded by presentmon.

Additional future work includes simulating wifi and buffer sizing. These tests could provide data that is closer to what might be experienced in the “real world” while still being in a controlled environment. By simulating wifi instead of just using wifi, we can have complete control of connection quality without needing to worry about the volatility of an actual connection. This could be done using two Raspberry Pis, where one acts as the router, and the other acts as a “wifi” connection. A computer could be connected to the second pi (via ethernet), and then connect to a local server to simulate connecting to the Internet over wifi. As for
simulating buffer sizing, this would once again involve the two Raspberry Pis, with the first stimulating network perturbation in the form of jitter, and the second simulating a jitter buffer by adding delay and smoothing out the incoming jitter.

Longer term future work might involve recreating this study with some different features. For instance, tests using simulated wifi and buffer sizing could provide useful QoE data about how much a buffer could help improve the quality of experience for someone with a poor connection, depending on how bad the connection is and how large the buffer is (i.e. how much delay gets added). A re-creation of the study could also be made to include a more diverse selection of games. Including more games in the study would provide a better picture of which types of games are affected the most by certain aspects of perturbation.
Section 7 Conclusion

Measuring QoE is important, especially in the space of cloud-based game streaming. There have been similar studies done in the space of normal video streaming but not with cloud-based game streaming. Studies conducted on QoE in video-conferencing are the closest studies to cloud-based game streaming because they are producing playback and accepting inputs in real time, like how a cloud-based game would. The goal of this study was to assess how delays and interrupts affected QoE specifically for cloud-based game streaming, the results would then be used to inform how to size buffers for different games.

The study gathered participants that had a broad experience with different types of games. These users then played through four different games, two were made on Unity to test specific actions and the other two were games commercially available on Stadia’s store to test the interactions in a full fledged game. Each user played 18 total rounds per game, including a practice round, with varying delay and interrupts, in the form of magnitude and frequency, added in. In between each round, they filled out a QoE survey that asked them to rate their experience for that round. After the user studies concluded, we combined the data from all the users to create graphs.

We analyzed the relationship between the perturbations and QoE, QoE and score, and score and perturbation. The main relationship we wanted to analyze was the perturbations and QoE inform how the buffer size should be changed for different types of games. The studies gathered data that mostly aligned with the hypotheses that were made at the beginning. The Delay versus QoE graphs showed us that the delay sensitive games, Whack-a-Goat and Avicii Invector, had degraded QoEs when delay was added into the system. This affirmed that delay sensitive games would not do well with a large buffer or would be more sensitive to the presence
of one. Jitter was split between magnitude and frequency. The Jitter Magnitude versus QoE graphs showed that the interrupt sensitive games, Goat Runner and The Falconeer, had degraded QoEs when there were large interrupts in the playout. The Jitter Frequency versus QoE also degraded QoEs, however not as much as jitter magnitude. We came to the conclusion that users care more about length of interrupt versus amount of interrupts for the range of values tested. The relationships between QoE and score and score and perturbations show that as QoE went up so did score and as perturbations were added into the system, score went down.

By showing how delay and interrupts impact a user’s QoE, we can surmise how the presence of a buffer may improve that QoE. In the case of a delay sensitive game, like a rhythm game, the buffer can not be too big or else it will start to more negatively impact QoE. Timing matters in rhythm games, a buffer that is large adds to delay which, as shown by the studies, degrades QoE. A game that is precision and timing based benefits from a smaller buffer size that does not add in a lot of delay but still helps with smooth playout if there is a change in their network connection.

However, in the case of a game that is not delay sensitive, like an exploration game, a smaller buffer size may lead to more visual interrupts in the middle of gameplay. This negatively impacts QoE especially when continuous motion and that movement is captivating the player. In this case, a larger buffer size would be better. They can size their buffers based on how delay and interrupts impact specific types of games. The results of this study can be used as a reference for cloud gaming companies like Google Stadia to help inform them of how adding in a buffer may impact their users’ QoEs.
References


10.1109/ICME.2014.6890204


https://doi.org/10.1631/jzus.2006.AS0112


10.1109/TCE.2016.7838089


10.1109/INFOCOMWKSHPS51825.2021.9484481
Appendices

Appendix A: Demographic Survey

1. What is your age?
   a. (enter a number)

2. What is your gender?
   a. Male
   b. Female
   c. Non-Binary
   d. Prefer not to say

3. What is your major?
   a. (enter text)

4. How many hours do you spend playing video games in a week?
   a. (enter a number)

5. What types of games do you typically play? (check all that apply)
   a. First-Person Shooters (FPS)
   b. Adventure/Story-Driven
   c. Role-Playing Games (RPG)
   d. Real-Time Strategy (RTS)
   e. Multiplayer Online Battle Arena (MOBA)
   f. Puzzle Games
   g. Simulation/Sandbox
   h. Casual/Mobile Games
   i. Music or Rhythm
j. Other (enter text)

6. How much experience do you have with cloud-based game streaming (e.g., Stadia, GeForce Now, Playstation Now, Luna)?
   a. None
   b. Some
   c. A lot

Appendix B: Quality of Experience Survey

1. How was your experience in the previous round? (floating point number) Experience Ratings: 1.0 - Bad, 2.0 - Poor, 3.0 - Fair, 4.0 - Good, 5.0 - Excellent
   a. (enter a number)

2. Was the quality acceptable? (i.e. would you continue to play longer with the current quality?)
   a. Yes
   b. No

Appendix C: Recruitment Email

Hello everyone!

As part of our MQP, we are looking for participants to take part in a user study that explores how buffer sizing can impact the user’s experience with cloud-based game streaming.
The study should take about an hour per person, and involve playing four different games and answering survey questions as you play.

You will be compensated a $15 Amazon gift card for your time.

Additionally, participation can count as an IMGD playtesting credit, if you need it.

Participation is voluntary, you may leave at any time. The study does include a game with many bright flashing lights, so if you are prone to epileptic seizures please do not participate. Apart from this, there is no/minimal risk for this user study.

If you are interested in participating, click the link here: [link goes here]

If you have any further questions, feel free to reach out to Brennan (bjaubuchon@wpi.edu) or Morgan (mlangstaff@wpi.edu).

tl;dr Play games, get paid: [link goes here]

Sincerely,

[team member who sends the email]

Appendix D: Consent Form

Informed Consent Agreement for Participation in a Research Study
Investigator: Brennan Aubuchon and Morgan Langstaff

Contact Information: bjaubuchon@wpi.edu, mlangstaff@wpi.edu

Title of Research Study: Quality of Experience Evaluation for Buffer Sizing of Cloud Based Gaming

Sponsor: Google

Introduction
You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of the study:

This study aims to find the optimal buffer size(s) that should be used to maximize the quality of experience of cloud-based games.

Procedures to be followed:

We will ask you to sit at a computer and play four cloud-based games for approximately an hour total. Two of these games will be of our own design, and the other two will be commercial games. The games will come in short 30 second rounds, and each round we will alter the delay and jitter experienced while playing the game to simulate changing the buffer size. At the end of each round we ask that you fill out a short two question survey about how you felt the quality of
experience was. After 10-15 minutes with each game, we will switch to another game to remain on time.

**Risks to study participants:**

One of the games features many bright, fast moving colors. **If you are prone to epileptic seizures, please do not participate in this study.** Additionally, while we will be taking precautions to mitigate the spread of COVID-19, there is still the risk of it spreading during one of our tests.

**Benefits to research participants and others:**

There are no benefits for participating in this study. **For details on compensation, please see the Cost/Payment section.**

**Record keeping and confidentiality:**

In addition to your survey responses, we will also be logging gameplay-related inputs, as well as measures of performance (score, number of hits & misses, etc.). We will not be recording any gameplay footage.

Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators, the sponsor or its designee and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to confidential data that identify you by name. Any publication or
presentation of the data will not identify you.

Compensation or treatment in the event of injury:

You do not give up any of your legal rights by signing this statement.

Cost/Payment:

You will be compensated a $15 Amazon gift card for your time. Additionally, you will receive an IMGD playtesting credit if you request it.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact:

Brennan Aubuchon or Morgan Langstaff (information listed above).

IRB Manager: Ruth McKeogh, Tel. 508 831-6699, Email: irb@wpi.edu

Human Protection Administrator: Gabriel Johnson, Tel. 508-831-4989, Email: gjohnson@wpi.edu

Your participation in this research is voluntary.

Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.
By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

________________________
Date: ________________
Study Participant Signature

________________________
Study Participant Name (Please print)

______________________________
Date: ____________ Signature of Person who explained this study

Appendix E: Perturbation versus QoE Box Plots