March 2017

Adaptive Bitrate Streaming in Cloud Gaming

Lambert Wang
Worcester Polytechnic Institute

Matthew James Suarez
Worcester Polytechnic Institute

Robyn Angela Domanico
Worcester Polytechnic Institute

Adaptive Bitrate Streaming in Cloud Gaming

A Major Qualifying Project submitted to the Faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree in Bachelor of Science

Robyn Domanico

Matt Suarez

Lambert Wang

{rdomanico, mjsuarez, lwang5}@wpi.edu

Advised by Mark Claypool

claypool@cs.wpi.edu

March 19, 2017

Abstract

Cloud gaming streams games as video from a server directly to a client device making it susceptible to network congestion. Adaptive bitrate streaming estimates the bottleneck capacity of a network and sets appropriate encoding parameters to avoid exceeding the bandwidth of the connection. BBR is a congestion control algorithm as an alternative to current loss-based congestion control. We designed and implemented a bitrate adaptation heuristic based on BBR into GamingAnywhere, an open source cloud gaming platform. We conducted a user study and did objective analysis comparing the software with our modifications to the original. Through our user study we found that our adaptive system was less challenging for a player and improved player retention rates. From our objective tests, we also found that there was no statistically significant difference in visual quality or appearance of the tested game.

Acknowledgements

We would like to thank our MQP advisor, Professor Mark Claypool, for his guidance throughout our project. We would like to thank Kuan-Ta Chen and Chun-Ying Huang for creating the GamingAnywhere open source platform for us to modify. We would like to thank John Leveillee and Michael Voorhis for allowing us to borrow hardware and space to conduct our user studies. Lastly, we would like to thank everyone who volunteered to participate in our user study.

Contents

1 Introduction

Cloud gaming is an online gaming service in which someone plays a game streamed from a server directly to a client device such as a computer or console. The player's inputs are sent to the server in return, effectively playing the game in real time. Cloud gaming enables players to play games that would not normally run due to high hardware requirements on their device, lowering the barrier of entry for many higher end games. It also provides players with access to games they may not previously own, often offering a large variety of titles for a singular subscription fee. Despite these benefits, cloud gaming is more susceptible to variations in bitrate and network congestion than games played on traditional platforms. Compensating for congestion in this system is important to creating an enjoyable player experience.

Network congestion causes reduced quality when utilizing an online service as a result of the bandwidth capacity of the network being exceeded. When the amount of user traffic on a network increase beyond a certain point, the network is congested because the amount of bandwidth being used by the users exceeds the limit of the network. This limit is also known as the bottleneck capacity. Increased network delay and packet loss are some of the effects that may result from network congestion [38].

Lag is a common term in online gaming for the delay between a player's actions and the corresponding responses from the server. Contributors to lag may include high latency, processing time (either server or client-side), or screen lag which are all symptoms of network congestion [38]. One way to avoid network congestion in cloud gaming is to avoid exceeding the connection's bottleneck capacity.

Current cloud gaming services address this problem by offering adaptive bitrate streaming to players. Adaptive bitrate streaming is a technique implemented when streaming media, often video and audio, which allows a client to request content dynamically in response to real-time measurements of its own CPU and bandwidth. A client can determine its own bandwidth capacity and request from the server a media stream of a specific bitrate appropriate for the situation.

Many commercial implementations of adaptive bitrate streaming used in large content delivery networks (CDNs) where static content is pre-encoded and then made available to audiences with a variety of capacities. However, this differs from cloud gaming because

gaming is meant to be interactive and the video content stream is unique for each player. Adaptive bitrate streaming methods that work for video streaming incur delays that are too high to produce a good quality gaming experience over cloud gaming. The media stream for cloud gaming cannot be pre-processed and must be able to change its encoding parameters in real time.

The client must first be able to determine its own bandwidth capacity and send this information to the server which can then control amount of network traffic generated by the application. TCP (transport control protocol) is a major transport protocol which carries a majority of the Internet's traffic [28]. The specific implementation of TCP being used controls the throughput of an application over an Internet connection. Loss-based congestion control algorithms, such as that used in early implementations of TCP like Reno and Tahoe and also modern implementations like BIC and CUBIC, rely on measuring packet loss as an indicator of congestion [9].

Avoiding packet loss is necessary due to the interactive and responsive nature of cloud gaming. Cloud gaming platforms must use a bandwidth estimation technique that does not intentionally incur loss. One such algorithm is BBR, a congestion control algorithm introduced by Google [9].

GamingAnywhere is a popular option out of the few open source cloud gaming platforms available and has been used as a platform for academic research [11]. Currently, GamingAnywhere does not have a solution to account for variable network bandwidth. Objective analysis of GamingAnywhere has shown that video quality decreases significantly when the network bandwidth falls below 3 MB/s while streaming 720p 50fps video [10]. GamingAnywhere performs noticeably worse than existing proprietary platforms in that regard [11]. We designed and implemented a bitrate adaptation heuristic into GamingAnywhere in order to address this performance problem. Our bitrate adaptation extension selects an encoding bitrate that maximizes bandwidth usage while avoiding packet loss.

Overall, by minimizing the processing delay required by the bottleneck link to empty its queue, the algorithm allows for increased responsiveness and control in a reaction-intensive game. By using BBR to estimate bandwidth, the adaptive system avoids packet loss while making efficient use of the available network capacity. This improves performance by reducing the number of frames that drop between the client and server.

We conducted a user study of 26 volunteers in order to assess the effectiveness of our modification at improving a player's experience while using the platform. We also conducted objective tests on our system under varying bandwidths, measuring packet loss, peak signal to noise ratio (PSNR), and structural similarity index (SSIM) of the video stream.

Through our user study we found that our adaptive system made the exact same system conditions less challenging for a player and improved player retention rates. This results in an improved experience over the default configuration, based on statistically significant question responses. From our objective tests, we also found that there was no statistically significant difference in visual quality or appearance of the tested game. When combined with user study responses, our adaptive configuration provides unchanged or improved quality to the player experience.

Chapter 2 contains research and background information related to the fields of cloud gaming, multimedia streaming, and network congestion. In addition, it includes work related to evaluating and measuring the performance and quality of networked game clients and video streams. Chapter 3 describes the tools and methods executed in order to implement a bitrate adaptation extension to GamingAnywhere. The chapter also includes the process undertaken to evaluate our solution through both user studies and objective signal analysis. Chapter 4 contains our analysis of the data we collected from our study and from our objective testing. Chapter 5 is the final chapter of our report and includes limitations, discussion, and conclusions regarding our findings.

2 Background

In this chapter we summarize research, background information, and existing works related to our topics of cloud gaming, video streaming, and network congestion. The chapter begins by introducing the concept of cloud gaming and existing popular services providing cloud gaming. We then discuss video compression techniques and codecs. The next section provides an overview of existing implementations of network bandwidth estimation over TCP. The fourth section describes adaptive bitrate streaming and considerations required to implement it for our purposes. The chapter concludes with describing performance and effectiveness metrics for a cloud gaming platform.

2.1 Cloud Gaming

Cloud gaming is a type of online gaming that utilizes video streaming with the goal of providing users with the ability to play a game, often PC or console titles, through the cloud instead of directly from the user's computer. A server runs the game and streams video and audio of the game to a device running a thin client. The client presents the multimedia content to the user and stream user's input to the server for the game to be process. Figure 1 shows a simplified version of the data that a cloud gaming system transfers during operation.

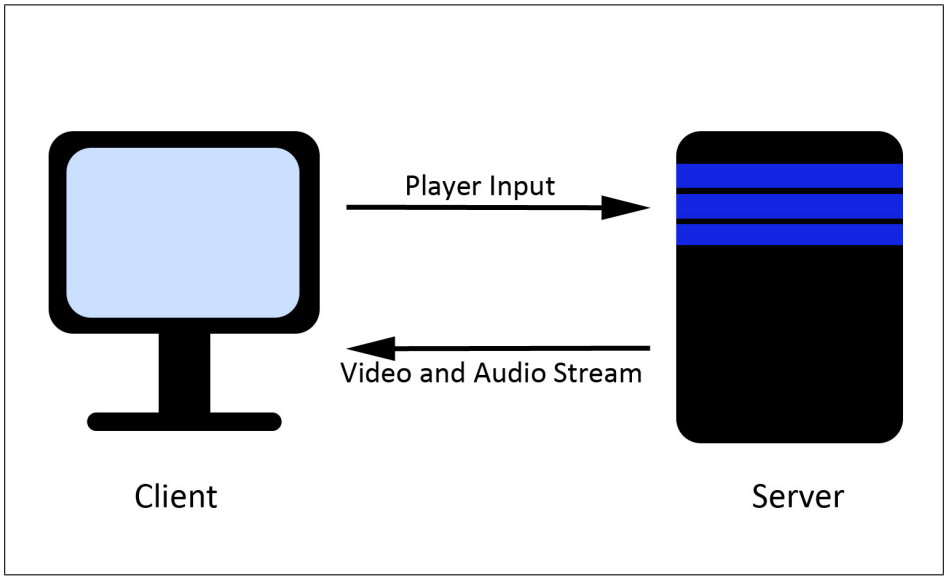

Figure 1: A simple cloud gaming system.

As a result, the hardware requirements (in terms of computational and graphics processing power) for the client device can be relatively low, often being lower than the requirements to run the game. This also allows a user to play a game without having to download, install, and run the game itself.

2.1.1 Cloud Gaming Platforms

Gaming Anywhere is one of many cloud gaming applications available; however, there are other platforms that are commercially available. Some companies have produced commercial cloud gaming platforms, usually tied to a user account or proprietary hardware. These services all have similarities in their requirements and are usually chosen based on the availability of games and the associated costs related to using the service.

2.1.2 GamingAnywhere

GamingAnywhere (GA) is an open source cloud gaming platform [11]. GA delivers a high quality gaming experience that would not concern players about installing and patching games or upgrading their current system hardware or software. GA allows players to play the latest computer games from anywhere at anytime through a resource-constrained platform.

The service's design choices are to be extensible, portable, configurable, and open. GA uses a modular design with platform independent components, meaning that individual components like codecs and network protocols are easy to modify or replace [19]. The service is also offered on mobile platforms, such as android, with support for other platforms available with added dependencies [7].

A user can provide a configuration file or use one of the given files in order to customize the experience. An individual can test and experiment with different settings to find the best configuration for running certain games on his or her hardware. These configuration files use short text based parameters that a basic text editor can set. Anyone can access, use, and modify the source code of GA free of charge under the BSD 3-clause license [21].

The resolution and frame-rate of GA are currently set based on either the original resolution and frame-rate of the server computer or a specified configuration parameter. The system uses avcodec, an open source audio video codec library [33], to encode the video and audio content. A user can stream any content, including games, available on the server computer.

GamingAnywhere has been tested to determine the average delays across different types of games. Figure 2 reports the average delays experienced while using GA. The x axis separates each bar in the graph by the streaming service being tested and the game played. The y axis measures the total delay in milliseconds, which includes both processing and playout delays. Processing delay is the difference between the time the server receives a command and the time it responds with a corresponding frame. Playout delay is the difference between the time the client receives the encoded form of a frame and the time the frame is decoded and presented. On the y axis, lower values are better.

This performance test measured delays using the games LEGO Batman: The Videogame, F.E.A.R. 2: Project Origin, and Warhammer 40,000: Dawn of War II. The study selected these games in order to measure performance across three popular genres of games: action adventure games, first-person shooter games, and real-time strategy games. On average, these games had delays of approximately 44 milliseconds [20]. The graphed results show that GA had low processing and playout delays compared to other cloud game systems. The figure also shows that playout delay is very small in comparison to processing delay, with processing delays making up most of the time taken in all platforms.

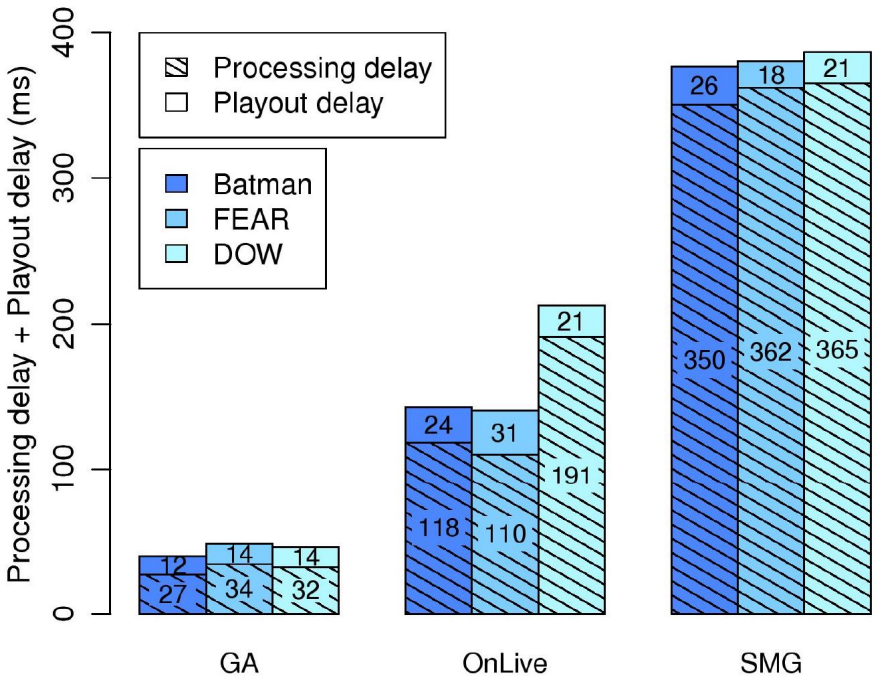

Figure 2: GA delays compared to other cloud gaming services [10].

GA has also been tested to determine video quality under variable network conditions. The x axis of Figure 3 compares each streaming service with a bar for each of the three games tested. The y axis shows the measured peak signal to noise ratio (PSNR), with higher values indicating a better score. The video quality does not suffer under network delays up to 600 ms; however, it is severely affected by higher amounts of packet loss and bandwidth below 3 MB/s. Higher frame-rates can also incur irregular playout during use. Overall, testing has shown that GamingAnywhere performs well when not under higher than normal rates of packet loss or connection speeds as shown in Figure 3 [10].

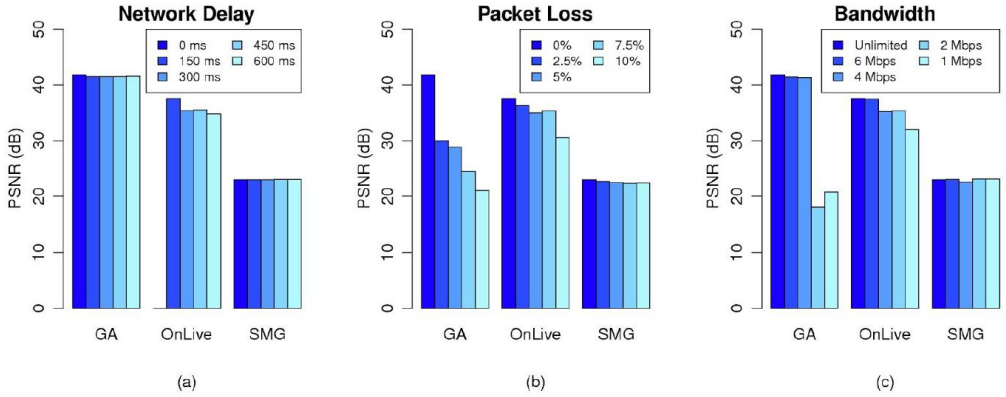

Figure 3: Results of image quality measured using PSNR under variable network conditions [10].

2.1.3 PlayStation Now (Sony)

PlayStation Now (PS Now) is a service offered by Sony that allows users to play older PlayStation games from a PlayStation 4 or PC [4]. Users pay a monthly subscription fee of $20, or $45 for three months and are able to access any title in the PS Now library, which consists of hundreds of games from previous PlayStation consoles. PS Now requires a device that meets the minimum requirements for the PS Now client, and a DualShock 4 controller. The service also requires a minimum bandwidth of 5 MB/s in order to play games; if a user drops below this connection limit, the platform forces the user out until their connection improves [13].

When launching a game from PlayStation now, the initial load time averages between 30 to 60+ seconds. During this loading time the system measures bandwidth speeds to determine whether the user meets the connection minimum. PS Now encodes the game

content in the H.264 format and locks the stream parameters at 720p resolution with 60 frames per second. With the combined bandwidth limit and fixed resolution, frame drops present in game are more frequently due to the technology limitations of the older games as opposed to being a result of variance in the network connection [15].

2.1.4 GeForce Now (Nvidia)

GeForce Now is a cloud gaming solution offered by Nvidia [29]. The service offers a library of games for a monthly subscription fee of $8 in the United States. The service requires the use of Nvidia shield products, which include the Shield Portable, Shield Tablet K1, and Shield Android TV Console, each of which cost $199. Both the Shield Tablet and Shield TV Console also require the Shield controller.

Users can play almost every game in the GeForce Now library through the subscription price; however, the user must purchase newer games through GeForce Now in order play them through the service. GeForce Now requires a minimum bandwidth of 10 MB/s and a maximum ping of 60ms to an Nvidia data center in order to use the service. If the connection bandwidth exceeds this minimum, the platform may scale up the quality of the video stream sent to the user to match the higher connection speeds. If the user moves too far from the router or their connection speed drops below the required minimum GeForce Now will warn the user and then kick the user if their connection speed remains low [26].

The resolution and frame-rate can change depending on the connection speed of the user. The maximum resolution is 1080p at 60fps and the minimum connection speed at this resolution is 50 MB/s. GeForce Now encodes the game content in the H.264 format [39]. Significant external network usage can also introduce stutter and lag while playing, meaning that the network used must be stable and free of congestion in order to prevent stuttering.

2.1.5 Steam In-Home Streaming (Valve)

Steam In-Home Streaming also offers similar features to the previously mentioned services with some differing requirements. Users access Steam In-Home Streaming through the Steam client which allows users to stream any game in their steam library for no additional cost. Steam In-Home Streaming requires the user to own a computer that can act as a server on the same local network as their client machine. There is no minimum enforced bandwidth, but the server and client machines must be on the same local network. This

setup makes it possible to play games that are exclusive to one platform on another platform, i.e., through in-home streaming it is possible to play a Windows exclusive title on a Linux machine [35].

The minimum resolution and frame-rate are set by the server computer and can be manually adjusted in game. Steam In-Home Streaming encodes streamed content using the H.264 format. The in-home client can also stream non-Steam games. Windows computers can accomplish desktop streaming by using the alt-tab feature to switch the top-level window from Steam [3].

2.2 Compression

Video compression is a method of taking video data and reducing redundancy within the data so that it requires fewer bits to store and transmit. Most video compression algorithms use lossy compression which maintains an approximate representation of the content while discarding some of the video's information. Although lossless video compression codecs perform at a compression factor of 5-12, a typical MPEG-4 lossy compression video has a compression factor between 20 and 200 [36].

Some trade-offs to consider when configuring video compression include quality of the media, speed of compression and decompression, system requirements, and space required to store the compressed data. Highly compressing video may have undesirable consequences including reduced frame-rate and image degradation.

2.2.1 Intra-Frame Compression

Intra-frame compression is a technique that to reduces the file size of the image by exploiting redundancies between certain regions of data in the image [25]. H.264 video compression partitions each individual frame into rectangle-shaped units called macroblocks. The encoder may encode each macroblock in the frame predictably based on data values from previously encoded and neighboring macroblocks in the image. This way, each macroblock represents video data accurately without needing to fully express the data uniquely for each block [31].

2.2.2 Inter-Frame Compression

Inter-frame compression is a technique in video compression that reduces the size of a frame using information from neighboring frames. Macroblocks in a frame encoded predictably based on information from previous frames are P macroblocks. P frames or predictive frames refer to frames that contain P macroblocks. A decoder cannot decode these frames independently from other frames as some macroblocks encoded in these frames rely on other frames to represent their data.

In contrast, frames that contain no P macroblocks are I frames or Intra-coded frames [31]. I frames are independently decoded and do not rely on information from other frames in order to decode correctly.

2.2.3 Quality Compression

Quality scaling is one of the most obvious effects of compression that an end user will notice. Data compression is usually achieved in one of two ways: lossless and lossy [23]. When using lossy video compression, it is not possible to reproduce the original video with prefect accuracy. Video compressed in a lossy manner loses some amount of information regarding the original content. A higher compression ratio is by using lossy video compression over lossless compression and as a result each frame of video is smaller in size.

Adaptive bitrate streaming uses this feature of lossy compression to its advantage. When dynamically adjusting bitrate, the encoder may produce a lower bitrate video which allows the system to stream content over congested network.

2.2.4 Dithering

Dithering is a technique for reducing color depth by placing a small combination of pixels with different colors within a small neighborhood. The combination of colors looks like the original color when viewed from a distance [27]. Dithering changes the color of an area of pixels to match the colors of a predetermined palette in order to approximate the color of the original pixels. The image will still appear to have pixels of the original color depth by combining pixels of colors in the new color depth that, when perceived, look similar to the original pixels. As seen in Figure 4, the gradient appears to consist of a large variety of shades, but in actuality only consists of a total of 23 shades. When viewed up close, the

pixel dithering becomes apparent, but appears to blend when viewed from further away.

An issue with dithering is that it may not preserve the information from the original image. A viewer perceiving a dithered image at a close distance while comparing it to the original image may see a loss of detail and precision. In gaming, this can pose a particular issue with objects in the distance. The object might be too small to meaningfully appear on a dithered image and may be unnoticed by the player. Another issue that can arise is maintaining text readability. A common visual artifact produced by dithering is aliasing which often makes text difficult and uncomfortable to read [32].

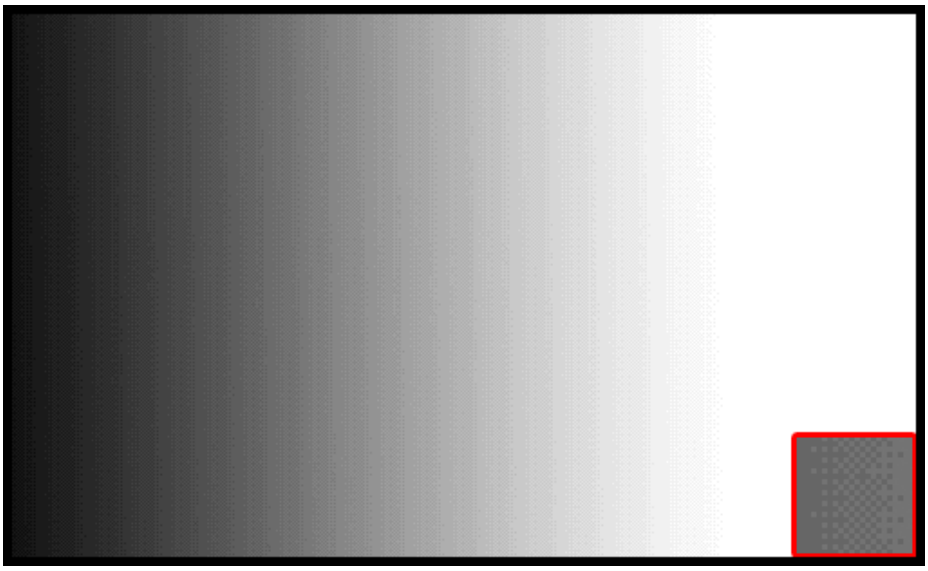

Figure 4: An example of dithering. The color depth of the image appears to be deeper than the actual color depth of the pixel representing the image.

2.2.5 H.264 Codec

H.264, also known as MPEG-4 AVC, is an industry standard for video compression [1]. This standard builds on concepts used by MPEG-2 and MPEG-4. H.264 offers a better compression efficiency. When comparing a video encoded by H.264 against a video encoded using MPEG-2 or -4, the H.264 compressed video will have a smaller file size despite having the same bit rate.

Each of these encoding methods produces an efficient, compact, and binary representation of the information [8]. The encoder processes a frame of video in units of macroblocks (By default a 16 x 16 area of displayed pixels). It forms a prediction of the macroblock

based on previously-coded data either from information in the current frame (intra-frame compression) or from information in other frames that have already been encoded (inter-frame compression). The encoder subtracts the prediction from the current macroblock to form a residual. The encoder stores or transmits the the encoded bit-stream.

2.3 Bandwidth Estimation

In digital communications uses bandwidth estimation to determine the amount of data a link or network path can deliver per unit of time [28]. Bandwidth estimation is important in adaptive bitrate streaming. In order to have an effective adaptive bitrate streaming algorithm, it must be able to estimate the available bandwidth to determine how to properly encode a content stream that a network and client can handle. The adaptive bitrate streaming algorithm requires an accurate estimation of the available bandwidth to be able to produce a content stream that provides the best quality given the current network capacity.

2.3.1 Bottlenecks

A bottleneck within a network is a link with the lowest capacity. The bottleneck link will saturate with data before other links. When that occurs, no additional data can flow through this link. Content streaming through a saturated link will not be able to increase its quality because its bandwidth usage exceeds the available capacity of the link.

2.3.2 Network Congestion Control

Congestion in relation to computer networking refers to when demand on a network is greater than the capacity of the network, often as a result of increased load. The measurable effects of network congestion include queuing delay, packet loss, and no longer being able to establish new connections. Network collapse occurs when the incoming traffic of a network exceeds the output bandwidth of the bottleneck. Communication on the network becomes severely limited or even halted in the presence of network collapse. Software which transmits data on the Internet requires congestion avoidance algorithms and bandwidth estimation techniques to avoid network congestion while utilizing as much of the network bandwidth as possible [38].

Content streaming platforms transmit data using the real time streaming protocol (RTSP), often implemented on top of UDP [30]. UDP does not innately implement con-

gestion control. Software utilizing a UDP connection must implement congestion avoidance independently. A streaming client, such as a cloud gaming platform, requires an end-to-end congestion control implementation. To select an algorithm to implement over UDP, we discuss five TCP congestion control algorithms: Tahoe, Reno, BIC, CUBIC, and BBR.

2.3.3 TCP Tahoe & Reno

TCP maintains a congestion window size (CWND) which limits how many unacknowledged packets the protocol transmits [17]. TCP Tahoe and its successor Reno behave similarly but differ in how they react to packet loss. These two algorithms use a strategy called slow-start. Slow-start initializes the CWND to 1, 2, or 10. The CWND will increase by one for each acknowledged packet. This causes the CWND to increase by a factor of 2 for each round-trip time. When the CWND reaches a certain slow-start threshold (SSThresh) the protocol will switch states to congestion avoidance.

In the congestion avoidance state, the algorithm will increase the CWND linearly with time, usually increasing the CWND by 1 for each RTT [5]. The algorithm detects loss when it receives three consecutive acknowledgments for the same packet. Tahoe will react to loss by setting CWND to 1 and setting the SSThresh to half of the current CWND and also switching states to slow-start. Reno differs as it will halve CWND, set SSThresh to the current CWND, and then enter a state called fast-recovery. In the fast-recovery state, the missing packet will be re-transmitted and then wait for the entire CWND to be acknowledged before switching to the congestion avoidance state [17].

2.3.4 TCP BIC & CUBIC

TCP BIC and CUBIC are algorithms introduced to improve TCP scalability over high-speed long distance networks [18]. These algorithms differ from previous congestion control implementations as their window size changes independently of RTT. BIC works by conducting a binary search for the available bandwidth. It increases the CWND to the midpoint between the size where TCP experienced packet loss, the maximum, and the largest size where TCP did not experience any packet loss, the minimum. The actual capacity of the network must be somewhere in between these two values.

BIC will grow the window using a logarithmic concave function. The window size will stay at the midpoint of the max and min at which point the algorithm will attempt to

detect packet loss. If the algorithm detects no loss, it will set the new minimum to the midpoint and proceed with the new range. If the algorithm detects loss, it will set the new maximum to the midpoint.

BIC was shown to perform well in high-speed networks; however, its growth function was too aggressive for low-latency or low-speed networks. CUBIC is as an improvement on BIC. CUBIC addresses the previously stated issue with BIC by using a cubic function to model window growth [18].

2.3.5 BBR

In 2016, Google introduces a new congestion control technique called BBR which is a congestion-based congestion control algorithm [9]. BBR is different from existing algorithms as it is not loss-based as in it can detect network congestion without incurring packet loss. BBR detects congestion by measuring differences between RTT and round-trip propagation delay (RTProp). When congestion occurs, the packet buffer of the bottleneck link begins to fill faster than the link can process packets.

This results in increased RTT but the bottleneck link does not drop packets until its buffer fills completely. When the buffer is not empty, the maximum delivery rate of the link will remain constant. This is the effective maximum throughput of the bottleneck, or the bottleneck bandwidth (BtlBw). If the system can detect the increase in RTT before the buffer fills then the user can decrease throughput to allow the buffer to drain to utilize all of the available capacity of the network without experiencing packet loss [9].

Loss-based congestion control methods will utilize the full bandwidth of the bottleneck at the cost of increased RTT and packet loss. BBR attempts to minimize RTT and packet loss while still utilizing all of the available capacity of the bottleneck. The rate at which BBR sends packets is a function of the BtlBw and RTProp. BBR will periodically probe the network for its capacity by periodically increasing the number of inflight packets and measuring any changes in RTT. If the RTT increases and BtlBw does not then link is beginning to experience congestion. BBR will then decrease its throughput in order to avoid eventual packet loss [9].

A system which streams interactive content over a network often does not re-transmit dropped or unacknowledged packets in order to to avoid added delay. In these cases it is crucial to avoid packet loss due to network congestion. Most current congestion control

algorithms detect congestion by measuring packet loss. BBR is able to detect network congestion without incurring packet loss. We chose to implement BBR for this reason.

2.4 Adaptive Bitrate Streaming

Adaptive bitrate streaming is a technique used when streaming multimedia content, such as video or audio, over computer networks in order to reduce buffering and improve video quality over low-end and high-end Internet connections.

2.4.1 General Concept

When a client is streaming video, parameters such as available bandwidth may change over a short amount of time. If the streaming platform did not adapt to handle these changes, then disruptions would occur in the video stream. These variances in network capacity may render a game unplayable for a user on a cloud gaming platform. To address this, the system can implement an adaptive bitrate streaming algorithm.

Either the client or the server can determine bandwidth capacity. If the server determines the bandwidth capacity then it can send an appropriate encoding to the client. Otherwise if the client determines the bandwidth capacity, it can send metadata regarding the connection quality to the server requesting an appropriate encoding. When the client detects that the bandwidth has decreased, it can request a change in video bitrate from the server. A server compatible with such an algorithm can encode content with various parameters specified by the client on the fly.

2.4.2 Comparison of Services

For our own purposes, we are most concerned with video streaming services that minimize encoding and decoding time of bit streams on the client and server as we are concerned with creating an adaptive bitrate streaming algorithm for GamingAnywhere. We do not plan to make modifications to the network transport layer. Additionally, the physical transport layer is out of our expertise and we do not want to make any changes to the transmission protocol. As such, it is valuable for us to look at other services that also use adaptive bitrate streaming algorithms in their own systems.

2.4.3 Considerations for Cloud Gaming

Encoded video content has two types: static and dynamic. For static content, the video stream has terminated and all content in the video has already been generated. The server can pre-encode all the content at various bitrates well ahead of when a client will request it. This is not applicable to cloud gaming as cloud gaming is an example of dynamic video content.

Dynamic video content is a continuous stream of video content often depicting events that are happening in real time. Having multiple encoding parameters of various bitrates available to the client is a computationally expensive operation for the server as it will require multiple encoders running simultaneously. Rather than make a large number of encoded video streams available to the client, the server can change encoding parameters in real-time. If the bitrate of the current encoding exceeds the measured bitrate of the network link, the system may reinitialize the encoder to meet the bandwidth requirements of the system. Additionally the server should allow the client to request a higher bitrate from the server if it can determine that the connection has more bandwidth available than the stream is using.

Modern implementations of adaptive bitrate streaming are almost exclusively over HTTP using TCP. This creates immense overhead as client to server communication needs to happen multiple times for each packet transmitted. Modern video games use UDP for their packet transportation protocol because acknowledgments for packets and re-transmission of packets are not necessary. The application can discard old packets as their information is no longer necessary.

2.5 Testing Metrics

When analyzing our final algorithm, there are two types of metrics: objective and subjective.

2.5.1 Objective Metrics

Emotions or opinions do not influence objective metrics; these kinds of metrics are measurable without user input. A mathematical metric measures an adaptive bitrate streaming algorithm's performance to produce concrete results. Two objective image quality measurements we will use for our testing are peak signal to noise ratio and structural similarity

index.

2.5.2 Peak Signal to noise Ratio

$$PSNR = 20\ log_{10}(\frac{MAX_f}{\sqrt{MSE}})$$

Figure 5: PSNR equation. MAX_f if the maximum signal value in the original image. Figure 6 shows the mathematical definition of MSE [2].

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}||f(i,\ j) - g(i,\ j)||^2$$

Figure 6: Mean Square Error equation. $f(x, y)$ is the matrix of channel values of the original image. $g(x, y)$ is the matrix of channel values of the degraded image. m is the number of rows of pixels in our image. n is the number of columns of pixel in our image [2].

Peak signal to noise ratio (PSNR) is a term that describes the ratio between the maximum value of a signal and the value of the distortion or noise affecting the quality of the representation of the signal [2]. A logarithmic decibel scale expresses PSNR in order to account for the high dynamic range of many signals. Video sources produced from different encoders or algorithms can be empirically compared to identify which particular algorithm produces a higher quality image. In the case of PSNR, image quality refers to how much the values of the pixels in the altered video differ from the source. As described by the equation in Figure 5, the mathematical model for PSNR represents signal noise with the mean squared error (MSE) between every pixel in the original image and the altered image. Figure 6 shows the calculation for MSE.

2.5.3 Structural Similarity Index

$$
\begin{aligned}
l(x,\ y) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \\
c(x,\ y) &= \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \\
s(x,\ y) &= \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}
\end{aligned}
$$

Figure 7: Luminance, contrast, and structure equations. Let μ_x, σ_x^2, σ_{xy} be the mean of x, variance of x, and covariance of x and y respectively. x and y are two signal sources, in our case, the original image and the degraded image [37]. The equations represented by C_1, C_2, and C_3 are shown in Figure 8

$$
\begin{aligned}
C_1 &= (K_1 L)^2 \\
C_2 &= (K_2 L)^2 \\
C_3 &= \frac{C_2}{2}
\end{aligned}
$$

Figure 8: C_1, C_2, and C_3 are all constants. L is the dynamic range of the pixel values(256^3 for 24 bit color images or 256 for each of 3 channels). $K_1 < 1$ and $K_2 < 1$ are two scalar constants [37].

$$
SSIM(x,\ y) = [l(x,\ y)]^{\alpha} \cdot [c(x,\ y)]^{\beta} \cdot [s(x,\ y)]^{\gamma}
$$

Figure 9: The SSIM equation. α, β, and γ are parameters specifying the importance of each of the three image components [37].

The structural similarity index (SSIM) is an image quality assessment based on the human perception rather than traditional techniques based on measuring error in raw channel values of an image [37]. SSIM and MSE accounts for certain image defects which human vision perceive as a degradation of image quality differently. For example, a blurred image will have SSIM rating consistent with human vision while an MSE technique will say the blurred image is close to the original [37]. SSIM evaluates perceived image quality if the images being compared by taking into consideration their luminance, contrast, and structure.

Figure 7 describes by the equations for luminance, contrast, and structure. Figure 8 shows the equation for SSIM which combines the images' luminance, contrast, and structure.

A sliding window calculates the SSIM indexing algorithm locally for each region of pixels. SSIM uses a mean of all the values calculated to evaluate the overall image quality.

2.5.4 Subjective Metrics

On the other hand, direct user responses determine subjective metrics. In our case, we will measure subjective results by having users test our system using different parameters and record their responses.

One big drawback of subjective testing is that it requires a large sample size in order to get an accurate evaluation of our system. Subjective testing can sometimes be time consuming. Performing the test for each individual subject can take a considerable amount of time. In addition, simply acquiring a group of participants for a study may take a long time.

2.5.5 Quality of Experience

The subjective metric we will use to evaluate the performance of our platform is quality of experience (QoE). QoE involves asking users directly how much they enjoyed a specific experience. In our case the user experience would be playing a game on our cloud gaming platform with variable network loads and congestion. The questions asked often are in the form of a short survey asked for responses in the form of true-false or a Likert scale (1 to 5 ratings) [22] [12].

3 Methodology

Our goal was to create an adaptive bitrate streaming algorithm within GamingAnywhere that provides an improvement in quality of experience to a user experiencing varying network bandwidths. In order to achieve this goal, we completed the following tasks:

1. We conducted research on modern bandwidth estimation and congestion control algorithms as well as video codecs and compression techniques (see chapter 2).

2. We created an adaptive bitrate streaming algorithm that can dynamically reconfigure the bitrate of the video encoding of the system using a bandwidth estimation technique based on BBR.

3. We tested our solution objectively using signal quality and image corruption metrics as well as subjectively by conducting a user study.

3.1 Software Implementation

3.1.1 Development Guidelines

The source code for GamingAnywhere is available on github.com under the BSD-3-Clause license [21]. We created a fork of the repository where we implemented our bitrate adaptation extension [14]. Any changes we made to the system were first created on our fork of the original repository. Pull requests were periodically created to introduce our features to the master version of GamingAnywhere after testing and code review.

Before we created the pull requests, we tested the system to ensure our changes to the system maintain platform compatibility. The source code must continue to compile successfully on the Windows and Linux operating systems. Our extension must not impede the base functionality of the software. We used the existing logging features in order to ensure correctness of our implementation throughout the development process. Code review ensured that any changes we made were properly documented and our code style was consistent with the existing code.

As of March 1, 2017, the only feature we implemented that we have merged to the original repository is the GA encoder-video reconfiguration extension.

3.1.2 GA Server Modules

GamingAnywhere adopts a modular design. A developer for GA can extend and modify features of the platform by modifying or creating modules for the system. Modules are generally threaded and communicate between each other via IO control (IOCTL) messages. Modules can only communicate to other modules on the same client or server. A specific controller that handles sending, receiving, and handling messages facilitates communication from the client to the server. Modules can queue messages to be sent by the controller. For the purposes of our project, we modified the `encoder-video` module which handles server-side initialization and execution of the video encoder.

3.1.3 GA encoder-video Reconfiguration

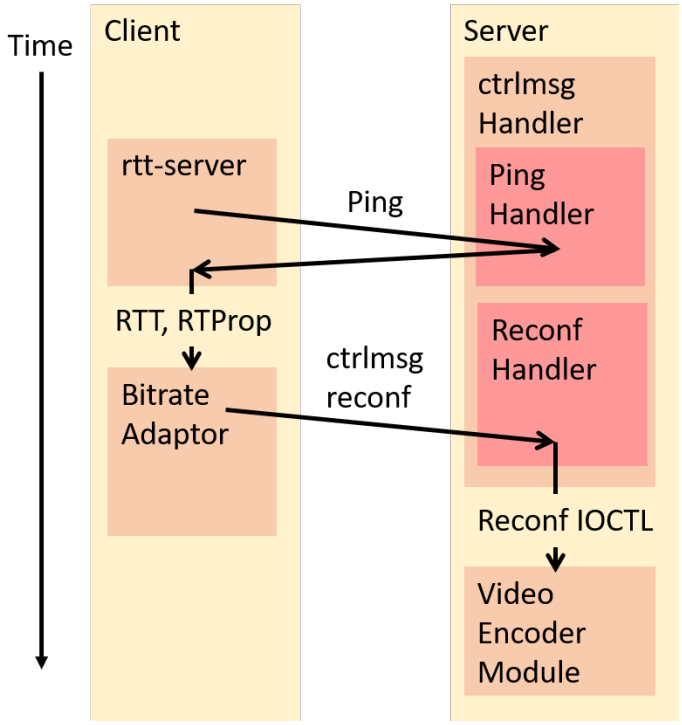

Figure 10: A high level overview of the GA IOCTL and ctrlmsg flow.

Figure 10 shows from a high level how the individual components involved in our extension communicate with each other. The `encoder-video` module of GA (located in `ga/module/encoder-video/encoder-video.cpp`) handles the encoding of frames provided by the video source module. The encoder module handles initialization, management, and

destruction of encoding contexts. In order to enable bandwidth adaptation, the encoder module must be able to dynamically produce videos of different bitrates from the same video source. In order to accomplish this, we extended the encoder module to handle the messages of the existing `GA_IOCTL_RECONFIGURE` type.

We modified the module to accept a reconfiguration message which allows the module to reinitialize the encoder context with different encoding parameters stored in the a `ga_ioctl_reconfigure_t` (defined in `ga/core/ga-module.h`) struct which included bitrate and frame-rate. The IOCTL reconfiguration message will never be sent to the encoder module unless the user enables our client-side bitrate adaptation extension by setting the `bitrate-adaptation` configuration flag to true in the client configuration file. This allows the original behavior of GA to remain the same because the default GA configuration files disabled the flag.

As shown in Figure 10, the client sends messages and commands to the server via system control messages (in the form of a `ctrlmsg_system_t` struct defined in `ga/core/ctrl-msg.h`). The only way for the client to communicate to the server is through control messages. We created a new ctrlmsg, the `ctrlmsg_system_reconfig_t`, to allow the client to send a reconfiguration message to the server which in turn handles those messages by converting the data in the ctrlmsg to reconfiguration parameters. We created the server reconfiguration message handler, `handle_reconfig` (defined in `ga/server/periodic/ga-server-periodic.cpp`), to handle system messages sent from the client and in turn send a IOCTL message to the `encoder-video` module requesting a reconfiguration of the encoder.

3.1.4 GA Client Extension

We created a new bitrate adaptation module and a round trip time (RTT) measurement module (located in `ga/client/bitrateadaptor.cpp` and `ga/client/rttserver.cpp`) on the client to measure RTT and throughput and then determine an appropriate bitrate for the video stream. The bitrate adaptation module runs on a separate thread on the client which will only initialize if the client configuration file has the appropriate flag. The module adds a packet-handler to the RTSP connection on the client which retrieves the size and time-stamp of received packets and measures the throughput of the video stream.

We created a new ctrlmsg, `ctrlmsg_system_ping_t`, to request a ping from the server in order to measure the RTT of the network connection. We created a new `handle_ping`

ctrlmsg handler on the server to respond to the pings sent from the client. The client bitrate adaptation module sends these messages and the rttserver on the client listens for responses to the message. The client rttserver measures the RTT from the server responses and stores the values on the client. The system calculates the round-trip propagation delay (RTProp) from finding the minimum within a sliding window of the stored RTT values.

The bitrate adaptation module uses measured RTT, RTProp, and actual throughput values in a heuristic based closely on the BBR congestion control algorithm (described in section 2.3.5) in order to estimate the actual bottleneck throughput of the network. The bitrate adaptation module sends reconfiguration messages to the server to reinitialize the video encoder with an appropriate bitrate based on the estimated throughput.

3.1.5 BBR State Overview

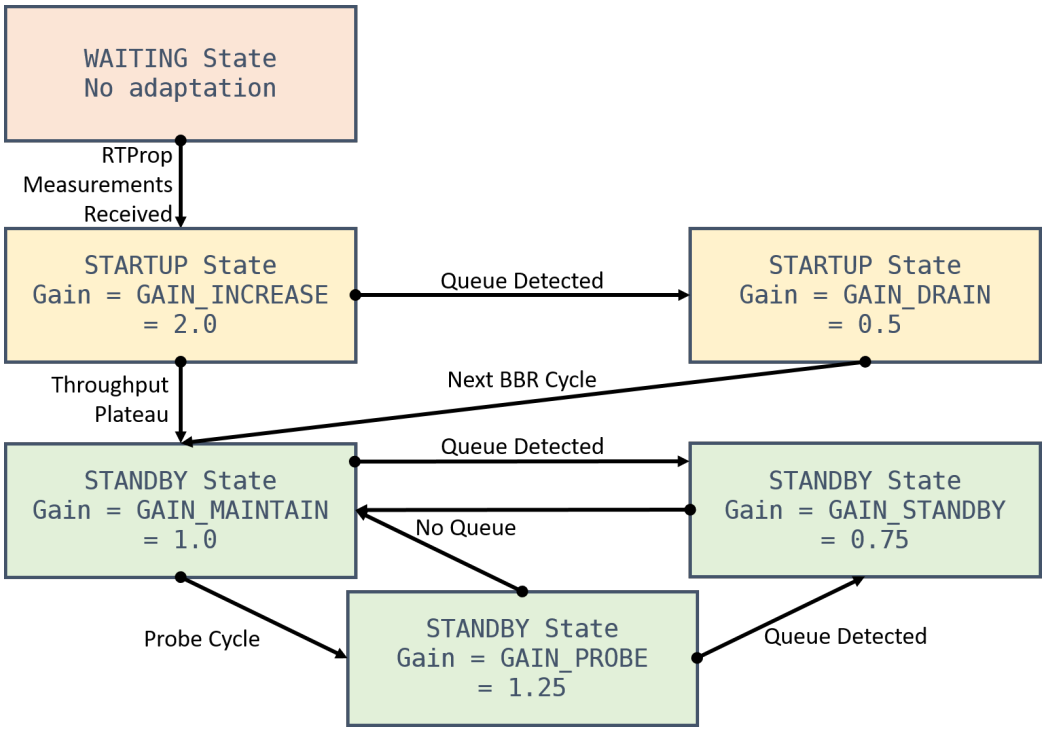

Figure 11: A diagram showing how different BBR states transition between each other and also the gain values within each state.

The bitrate adaptation module on the client functions by transitioning between various states (defined by the `bbr_state_s` struct in `ga/client/bitrateadaptor.cpp`) based on the BBR algorithm. Figure 11 shows how the system moves between different BBR states

and gain values. There are 3 states: waiting, startup, and standby. The state of the BBR algorithm changes periodically on a BBR state transition cycle. Pre-processor definitions, such as `BBR_CYCLE_DELAY`, located in the header file `ga/client/bitrateadaptor.h` define the time between BBR cycles and other specific timings.

The BBR state stores the measured characteristics of the network and determines how the client should request encoder bitrate changes from the server by setting the a gain rate. The gain rate determines the multiplication factor of the encoder bitrate between cycles. For example, a gain rate of 2.0 would double the encoder bitrate while a gain rate of 0.5 would halve the bitrate. We chose the gain rates based on values suggested in Google's original publication [9]. During every BBR cycle the module calls the `bbr_gain` function in `ga/client/bitrateadaptor.cpp` which uses the latest network performance measurements to determine which state to transition to in addition to setting the gain rate of the system.

3.1.6 BBR State Descriptions

The system always begins in the waiting state. While in this state, the module waits for the server and client to set up the required measurement tools for collecting information regarding RTT, RTProp, and throughput. Once the program can obtain values for those measurements, the client transitions to the startup state. While in the startup state, the gain rate is set to 2.0, doubling the bitrate every cycle.

The startup state is left when one of two conditions is met. The first condition being when our function detects a plateau in the measured throughput indicated by the measured throughout from the latest cycle not being at least 1.25 times greater than the either of the previous two throughput values measured. The second condition being when our system detects a queue on the bottleneck link indicated by RTT measurement being significantly greater than the system's RTProp value. The startup state then transitions to the standby state. When the startup state transitions due to detecting a queue, the gain rate is set to 0.5 for one cycle in order to allow the bottleneck link to drain its queue before continuing.

The standby state is the state that the system will remain in for the remainder of its lifetime. While in this state, the module will periodically increase its encoder bitrate to probe for additional network capacity. When the state is in standby and there is no queue detected the gain rate is set to 1.0 to maintain the current encoder bitrate. When the system probes for additional bandwidth, it will set the gain rate to 1.25 for one cycle. On

the next BBR cycle, if the system detects no queue, the gain rate will be set to 1.0 and will maintain the current bandwidth. Whenever the algorithm detects a queue, indicated by the difference between the latest RTT measurement and the RTProp being greater than the queue threshold, the system will always immediately set the gain to 0.75 to drain the queue created.

3.1.7 BBR State Testing

Throughout development of our bitrate adaptation extension, we used a UNIX traffic control tool (TC, described later) to ensure that our bandwidth estimation heuristic was working as intended. We added logging to the client to record time-stamped values for RTT, RTProp, and requested encoder bitrate. We used this logging throughout development to ensure that the BBR state transitions functioned as intended. We calculated queue threshold by adding a threshold difference to the RTProp (5ms in this case).

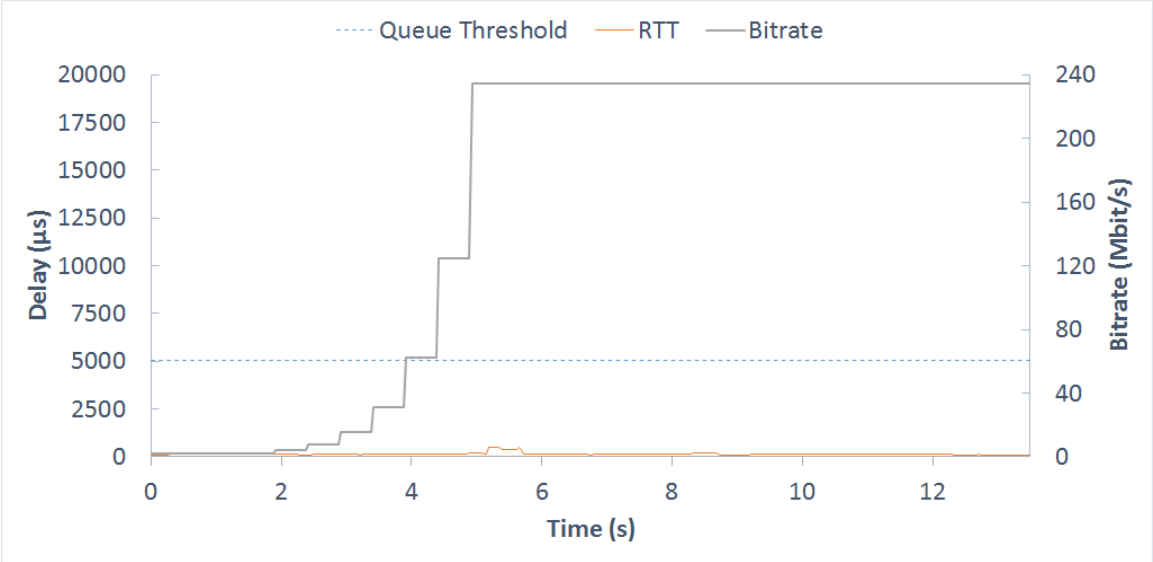

Figure 12: The behavior of our bitrate adaptation module in the startup state on an congestion-free LAN with no traffic control enabled. The queue threshold is the threshold for the RTT for when a the algorithm should detect a queue on the bottleneck link. The bitrate, represented on the secondary axis, is the bitrate sent from the client to the server to reconfigure the encoder parameters.

The startup behavior of BBR uses a binary search to quickly reach the initial maximum capacity of the network. In order to test whether we implemented the startup state correctly, we graphed the recorded values of our algorithm on a congestion-free LAN connection. As

shown in Figure 12, our algorithm in the startup state is capable of quickly reaching the maximum capacity of the network by doubling the encoder bitrate after every BBR cycle.

Note that the system does not increase the encoder bitrate above 30 MB/s despite having virtually unlimited bandwidth. This is an artificial maximum enforced by our system. We determined that a bitrate of 30 MB/s is enough to encode 4K video at 24fps with a motion rank of 2 [6]. Motion rank refers to the amount of movement present in the encoded content with a higher number representing more motion.

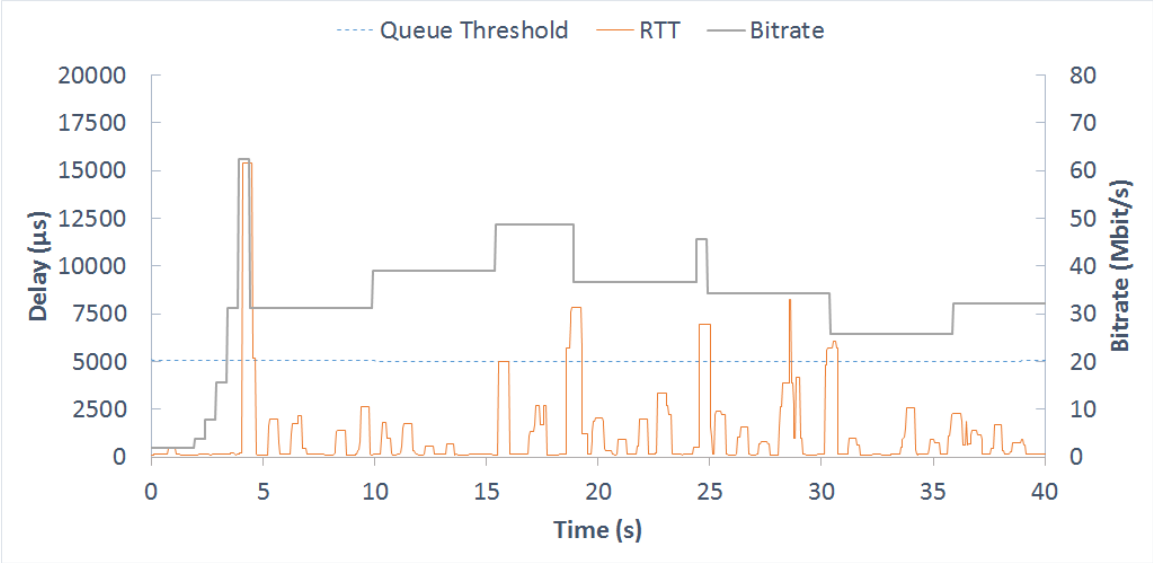

Figure 13: The behavior of our bitrate adaptation module on a LAN with a 40 Mbit (5 MB/s) TBF enabled. The TBF used a latency parameter of 100ms to to specify the bucket size.

The probe behavior of BBR periodically increases the bandwidth of sent data of the system in order to determine if the system may utilize additional network capacity. In order to test whether we implemented the probe correctly, we graphed the recorded values of our algorithm in on an artificially congested LAN connection. In Figure 13, our BBR implementation is shown to be able to maintain a constant bitrate in a consistently congested network.

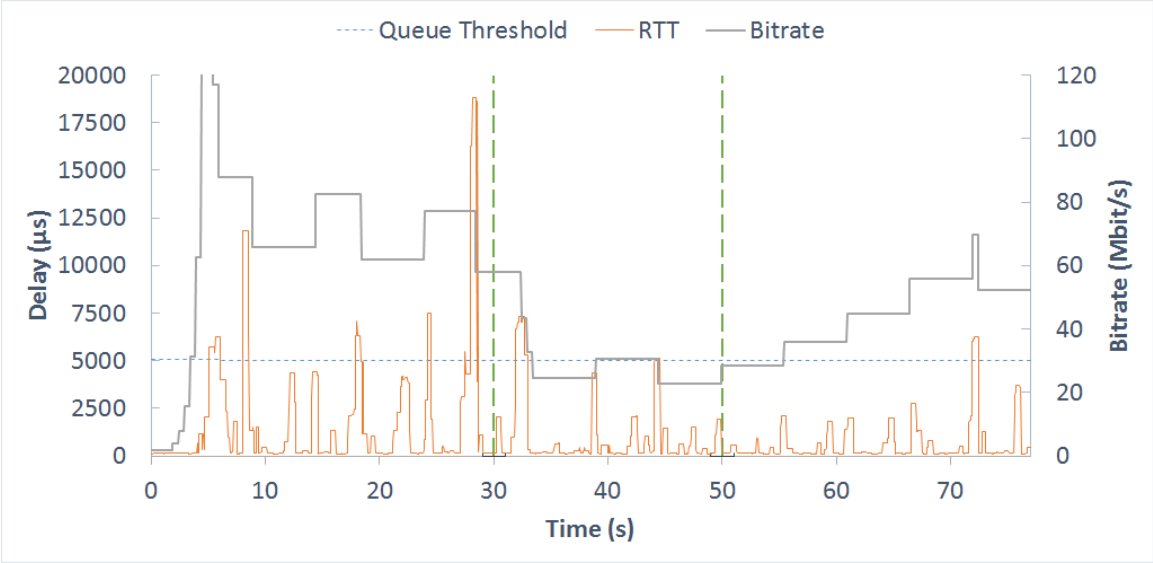

Figure 14: The behavior of our bitrate adaptation module on a LAN with changing network conditions. Initially we enabled a TBF of 80 Mbit (10 MB/s). At T=30s we changed the TBF to 40 (5 MB/s). At T=50s we changed the TBF to 80 Mbit (10MB/s). The TBF used a latency parameter of 100ms to to specify the bucket size.

In order to determine the probe state behavior correctly adapts to increases and decreases in the network capacity, we used a changing TBF pattern as described in Figure 14. The algorithm correctly leaves the startup state and enters the standby state when it detects the first bottleneck queue. After the algorithm leaves the startup state, the encoder bitrate hovers around a value of 80 Mbit from T = 10s to T = 30s.

The algorithm will attempt to probe setting the bitrate of the encoder above 80 Mbit. Whenever the measured RTT exceeds the queue threshold, the encoder bitrate drops below 80 Mbit. After T = 30s, we see in the figure that the encoder bitrate quickly falls to below 40 Mbit. After T = 50s, the algorithm slowly probes for bandwidth. Additional motion on the video source may have resulted in a briefly increased RTT causing the bitrate drop after T = 70s. As shown in these figures the algorithm reacts as expected when experiencing packet queuing in the bottleneck as indicated by when the measured RTT is greater than the queue threshold.

3.2 Testing Overview

To analyze the effectiveness of adaptive bitrate streaming on cloud gaming to improving players' quality of experience, we evaluated whether or not our algorithm has any noticeable improvement over the current implementation. We conducted a user study in which test subjects used the system in a controlled environment and recorded feedback regarding their perceived quality of experience. We collected results detailing the performance of GA with our modification and also the default implementation of GamingAnywhere. This allows us to form a subjective evaluation of the effectiveness of our product. In addition, we also conducted objective measurements of video and signal quality. We compared frames of the original video source to frames received by the client using PSNR and SSIM.

3.2.1 Setup

We created a clean and controlled environment in which both the client and server communicated over a wired LAN to ensure no external sources affected the network conditions. We made sure that the hardware was consistent throughout testing and that the hardware itself was not a bottleneck of our software.

3.2.2 Network Shaper

A network shaper creates a virtual layer between software and the network card that can intentionally drop, delay, or throttle the data or packets written to a connection. Using a network shaper allows for easily controlled testing at specified levels of network throughout while maintaining constant values for other attributes. This can allow us to ensure that any variations in the behavior of the program only appear from variance in a specific characteristic of the network. A network shaper can reveal flaws in a system that performs well in ideal conditions and struggles when exposed to poor network conditions.

We needed to be able to measure how the system performs while under a controlled conditions that model different yet realistic networks. We controlled the network behavior using a network shaper. We used the UNIX traffic control (TC) tool to artificially simulate a congested network in a predictable manner. We used TC to create a token bucket filter (TBF) which restricts the throughput of a network interface on the server machine.

3.3 User Study

We initially planned to conduct a study a minimum of 30 individuals recruited from the WPI computer science and interactive media and game development undergraduate and graduate mailing aliases. Throughout our eight day period of user studies we ran our study with 26 volunteers.

3.3.1 Game Selection

In order to evaluate the performance of our system, we chose to have our volunteer test subjects play a specific game through the platform. We had a set of requirements and selection criteria for the game that we chose. The TC tool we chose to use runs only on Unix based systems so the game must be compatible with Linux. The game must be playable single-player and in an offline setting. Another requirement was that it must have low hardware requirements order to reduce stress on our hardware so that the only bottlenecks in the system are the ones we artificially introduce. The game must have a low learning curve so that subjects who have never played the game or subjects who do not have extensive video game experience can understand and use the controls with ease. Lastly, the game needs some moderate level of graphical fidelity so that quality differences between different encoding bitrates were noticeable.

The game that we chose for our study was Race The Sun [24], by Flippfly LLC. Race The Sun is an arcade style racing game where the objective is to stay alive as long as possible by avoiding obstacles and collecting energy boosts. Race The Sun satisfies all of the requirements stated above.

3.3.2 Study Procedure

We asked the test subjects to use our modified system and also the original system while we simulated various network conditions on the server. Before the study began we briefed our subjects on our study and asked each one to sign an informed consent agreement. Before the individual test trials began, we asked each subject to complete an entry survey which asked for data regarding the subject's demographic and video game experience. We asked each test subject to play nine short 45 second trials of the game. The first trial was an unrecorded practice run to allow the player to be familiar with the controls and objective

of the game. We recorded each of the 8 subsequent trials for our research purposes.

#	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8
1	Adap. Hill	GA Valley	GA High	GA Hill	Adap. High	GA Low	Adap. Valley	Adap. Low
2	Adap. Low	GA Hill	Adap. Hill	Adap. High	Adap. Valley	GA High	GA Valley	GA Low
3	Adap. Hill	GA High	Adap. High	Adap. Low	Adap. Valley	GA Hill	GA Valley	GA Low
4	Adap. Valley	Adap. Low	GA Valley	GA Hill	Adap. High	GA Low	GA High	Adap. Hill
5	Adap. Hill	Adap. High	GA High	GA Low	Adap. Valley	GA Valley	GA Hill	Adap. Low
6	GA Hill	Adap. Hill	GA Valley	GA Low	Adap. Valley	Adap. Low	GA High	Adap. High
7	Adap. High	GA Low	Adap. Valley	GA Valley	GA High	Adap. Low	Adap. Hill	GA Hill
8	Adap. High	Adap. Valley	Adap. Low	GA Hill	GA High	GA Valley	GA Low	Adap. Hill
9	GA High	GA Low	Adap. Hill	Adap. High	GA Hill	Adap. Low	Adap. Valley	GA Valley
10	Adap. High	GA Valley	GA Low	Adap. Low	GA High	GA Hill	Adap. Hill	Adap. Valley
11	Adap. Valley	GA Valley	Adap. Hill	GA Low	GA Hill	GA High	Adap. High	Adap. Low
12	GA Valley	Adap. Hill	Adap. Valley	GA Hill	Adap. Low	GA High	GA Low	Adap. High
13	GA Valley	Adap. Low	Adap. Valley	Adap. High	GA High	GA Hill	GA Low	Adap. Hill
14	Adap. Hill	Adap. Low	GA High	GA Valley	GA Hill	GA Low	Adap. Valley	Adap. High
15	GA Valley	GA High	Adap. Valley	GA Low	GA Hill	Adap. Low	Adap. High	Adap. Hill
16	Adap. High	GA Hill	Adap. Hill	GA Valley	GA Low	GA High	Adap. Valley	Adap. Low
17	Adap. Valley	GA High	GA Valley	Adap. High	GA Hill	Adap. Hill	GA Low	Adap. Low
18	Adap. Hill	GA High	GA Valley	GA Hill	GA Low	Adap. High	Adap. Valley	Adap. Low
19	GA High	GA Valley	Adap. Low	GA Low	Adap. Hill	GA Hill	Adap. High	Adap. Valley
20	Adap. Valley	GA Hill	GA Low	Adap. Hill	Adap. High	GA Valley	Adap. Low	GA High
21	Adap. Low	Adap. High	GA Valley	Adap. Hill	Adap. Valley	GA High	GA Hill	GA Low
22	GA Hill	Adap. High	Adap. Valley	Adap. Low	GA Valley	Adap. Hill	GA High	GA Low
23	GA Valley	GA Low	Adap. Low	Adap. High	GA High	GA Hill	Adap. Hill	Adap. Valley
24	Adap. High	Adap. Low	GA Low	Adap. Valley	GA Valley	GA High	Adap. Hill	GA Hill
25	GA Hill	GA Valley	Adap. Hill	Adap. Low	Adap. Valley	Adap. High	GA Low	GA High
26	GA Hill	Adap. Hill	Adap. High	GA High	GA Low	Adap. Valley	Adap. Low	GA Valley

Figure 15: A table showing the trial configuration for each of our individual test subjects (identified by number).

During the recorded trials, a subject would play the game through one of two GA configurations with one of four network behavior simulations running on the server. Each individual test subject will play one of the eight combinations of these parameters exactly once. Figure 15 shows how we randomized the ordering of the trial settings for each participant. At the end of each play segment, we asked the test subject to complete a questionnaire asking questions regarding their experience with the latest segment of gameplay. We recorded the player's score at the end of each play segment.

3.3.3 Survey Question Selection

We created surveys to record our test subjects' responses to using either system. The survey we created is shown in appendix A. Our pre-trial survey includes only basic and unobtrusive questions regarding demographic and background. Our survey questions included both multiple choice questions and questions using a Likert scale. This enables us to easily

convert responses into numerical data for a straightforward analysis. In addition, these questions are quick to answer in between trials so that we can ensure our studies do not run later than their scheduled end time.

We chose our questions to answer three primary research goals:

1. Does our modified system mitigate the effects of additional lag due to network congestion better than the unmodified system?

2. Does our modified system provide an adequate level of graphical fidelity given the available network bandwidth?

3. How much user frustration does our modified system induce compared to the unmodified system?

3.3.4 Trial Configuration

We conducted all of our user study trials using a class-based traffic controller which created a TBF on top of a 40ms latency delay. The default TBF throughput value chosen for the study was 70 Mbit (8.75 MB/s). We selected this throughput value as we did not expect our program to exceed that throughput during testing. We ran the first practice trial for each subject under these conditions. We chose the specific values for the artificial latency and congestion for the recorded trials to emulate a poor broadband network connection in the US [34].

We chose four network condition scenarios based on work by Zink et al [40]. The four network conditions we chose were high, low, hill, and valley. Each network condition scenario determined the throughput of the TBF over a 45 second period, the same as the length of each individual gameplay segment of our study.

The high scenario modeled a high-throughput network of 24 Mbit (3 MB/s) for 45 seconds without changing its parameters. The low scenario modeled a low-throughput network, or a congested network, of 12 Mbit (1.5 MB/s) for 45 seconds without changing its parameters. The hill scenario modeled a low-throughput network for 15 seconds, then changed to a high-throughput network for 15 seconds before changing back to a low throughput network for the final 15 seconds. We intended this scenario to simulate a initially congested network that would then become less congested in order to test how the system reacts to an increase in available bandwidth. The valley scenario modeled a high-throughput network

for 15 seconds, then changed to a low-throughput network for 15 seconds before changing back to a high-throughput network for the final 15 seconds. We intended this scenario to simulate a initially congestion-free network that would then become congested in order to test how the system reacts to an increase in network congestion or a decrease in available bandwidth.

3.4 Objective Testing

Similar to our subjective test, we used a network shaper to control network congestion to test our modified version of GA with the bitrate adaptation module and the original version of GA. The streaming platform will produce some amount of data loss (either due to dropped packets in the network or inherent loss due to the nature of video encoding) when sending the video stream to the client.

3.4.1 PSNR and SSIM Metrics

For our assessment, we used a tool called VQMT (Video Quality Measurement Tool) [16], which is a program for computing objective video quality metrics. This tool allows us to compare two video streams stored in the YUV color space using several different metrics including PSNR and SSIM. It stores the comparison calculation results of each frame between the two videos in a CSV file.

We recorded 6 pairs of videos of Race the Sun. Each video pair consisted of a server recording of the video source and a client recording of the received frames. We tested both the adaptive and non-adaptive GA configurations on two different network shapes. These consisted of 3 settings: no traffic control, high bandwidth, and low bandwidth. We saved each of these videos as YUV file types. In addition, loss and throughput rates were recorded for each stream.

Using VQMT, we calculated the PSNR and SSIM taking individual frames from the video source, or server, and comparing it to frames captured from the client. We stored these values in four different CSV files (2 SSIM data files on high- and low-bandwidth settings and 2 PSNR data files on high- and low-bandwidth settings). We used a Python script to easily compute the average PSNR and SSIM between each frame.

4 Results and Analysis

In this chapter, we will talk about the results from both our user testing and our objective testing. We begin this chapter with an analysis of our user study data. We made several graphs and describe their significance, important features, and compare them with other findings. We then discuss our objective findings. Included in the objective findings are data about packet loss, SSIM values, and PSNR values for video at different configurations and bandwidth limitations.

4.1 User Study Results

Our user study obtained results from 26 participants. For the results of our data, we have assumed that the data collected is normally distributed. Out of all 26 users, 25 were male. Additionally, 24 our of 26 users were ages 18-22.

4.1.1 Perceived Challenge

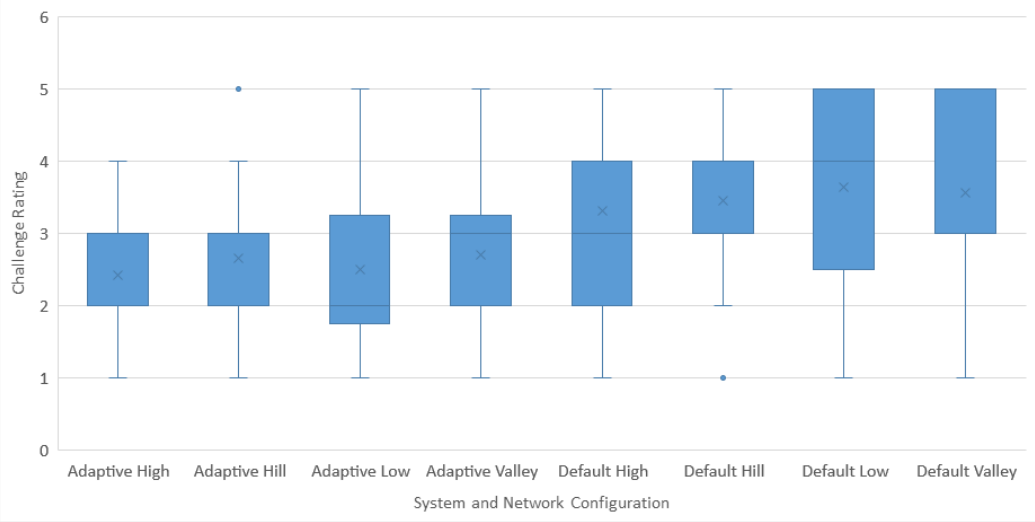

Figure 16: Box-and-whisker plot of user responses for question "How challenging was this run of the game?"

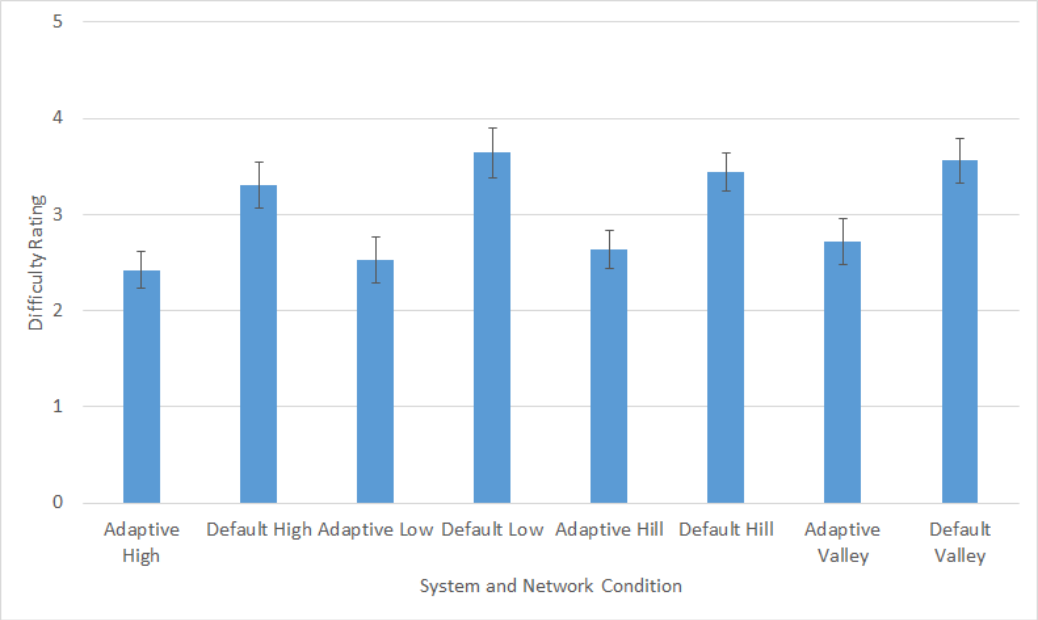

Figure 17: Average of user responses for question "How challenging was this this run of the game?" The error bars represent standard error.

Based on the information in Figure 16 and Figure 17, users rated the adaptive configuration as less challenging than the default configuration. Overall averages between the two configurations were about approximately 1 point apart from each other on a scale of 1 to 5, although almost all configurations except for adaptive high and adaptive hill were given a rating of 5 by at least one user. Conversely, all system configurations received a rating of 1 by at least one user. This indicates that the adaptive configuration introduced less difficulty in overall gameplay than the default during user trials. The time of the trial generates the seed for the level difficulty which may have affected some users' results. Race the Sun generates a new seed for the level each day, meaning that user trials across different days would have different level layouts.

4.1.2 Difficulty of Character Movement

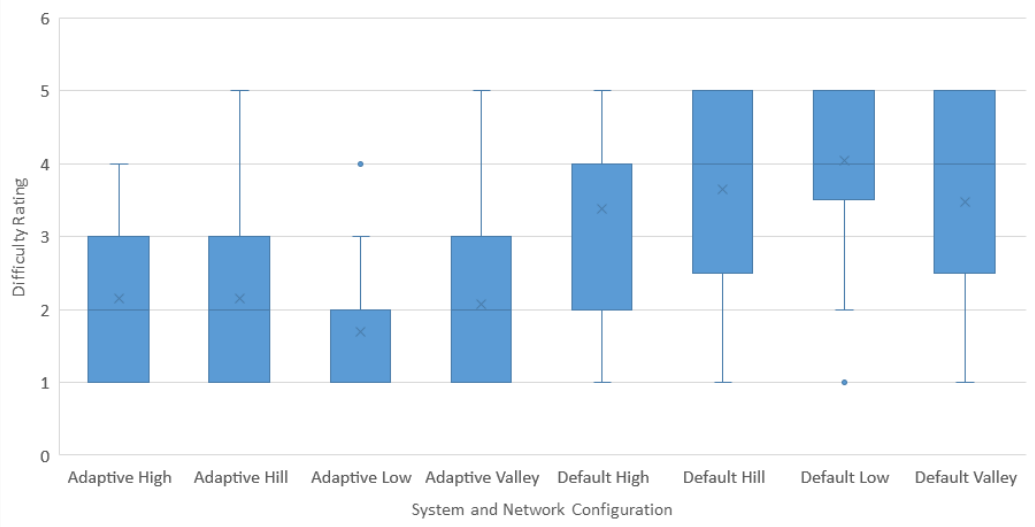

Figure 18: Box-and-whisker plot of user responses for question "How difficult was controlling the movement of your character?"

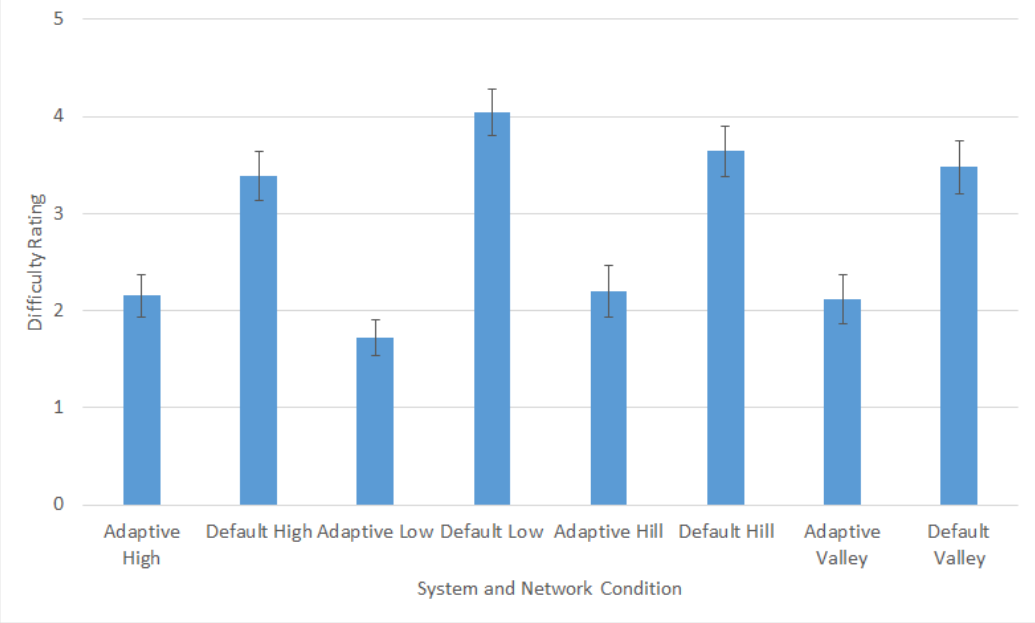

Figure 19: Average of user responses for question "How difficult was controlling the movement of your character?" The error bars represent standard error.

Based on the information in Figure 18 and 19, on average users rated the adaptive configuration as less challenging than the default configuration. Differences between averages

for default and adaptive were more pronounced compared to question one from the survey. All adaptive systems averaged low on difficulty. Only hill and valley network configurations receiving any ratings of 5. All default systems had ratings at all values, but received more 3, 4, and 5 level ratings. Unlike the first question, various level seeds likely would not impact results for this question, as character movement is not dependent on the level seed.

4.1.3 Perceived Graphical Quality

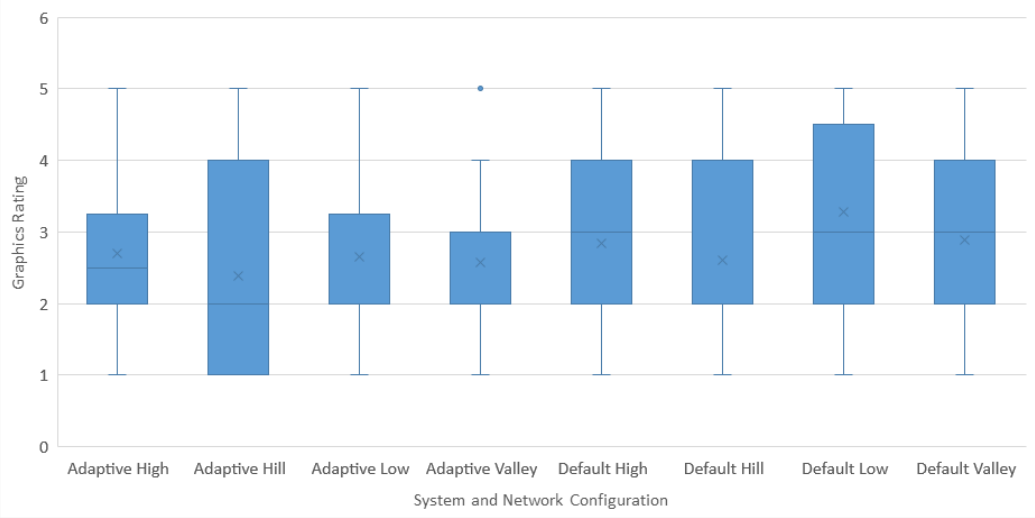

Figure 20: Box-and-whisker plot of user responses for question "How did the game look?"

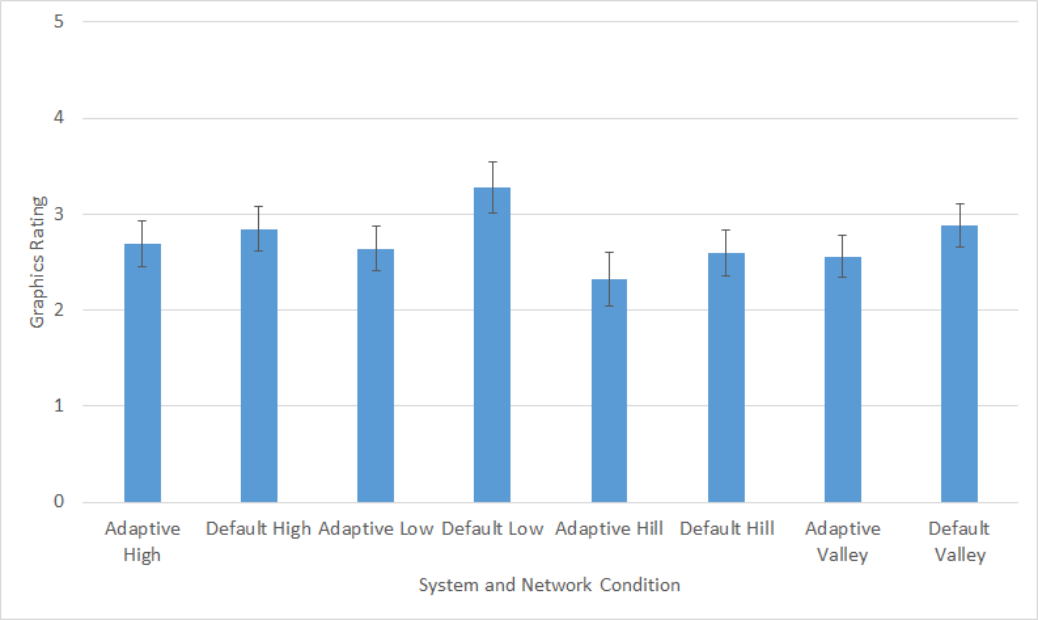

Figure 21: Average of user responses for question "How did the game look?" The error bars represent standard error.

Based on the information in Figure 20 and 21, on average users rated the default configuration as looking clearer than the adaptive configuration. While the default configuration rated higher on average, average user ratings were within the error margins of the average default ratings, as shown in Figure 21. All adaptive systems had very similar ratings, with the exception of adaptive hill, which had a higher number of ratings lower than 2. All default systems had similar ratings, with the exception of default low, which had a slightly higher average. Similar to question 2, level seeding is unlikely to have impacted the results of the question.

4.1.4 Difficulty of Discerning Objects

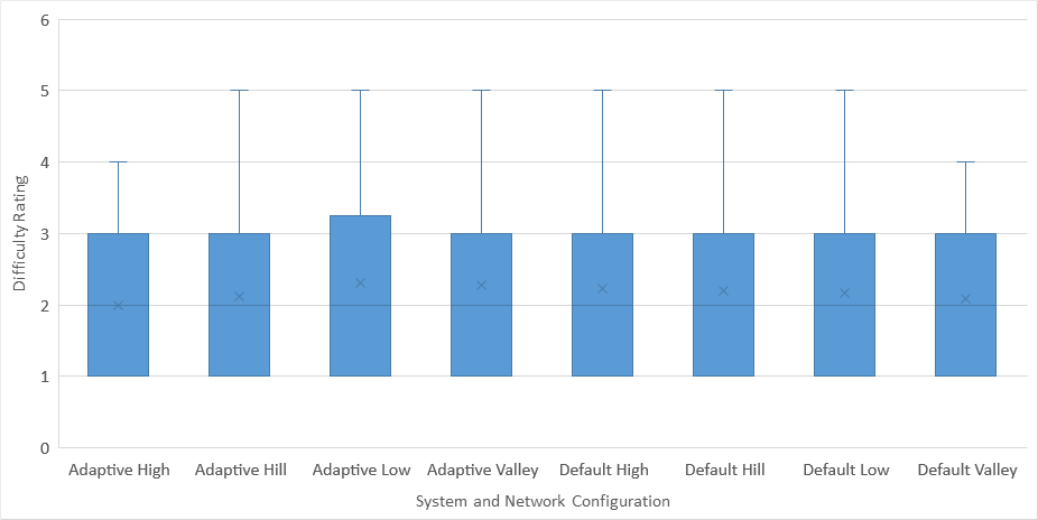

Figure 22: Box-and-whisker plot of user responses for question "How difficult was it to distinguish obstacles?"

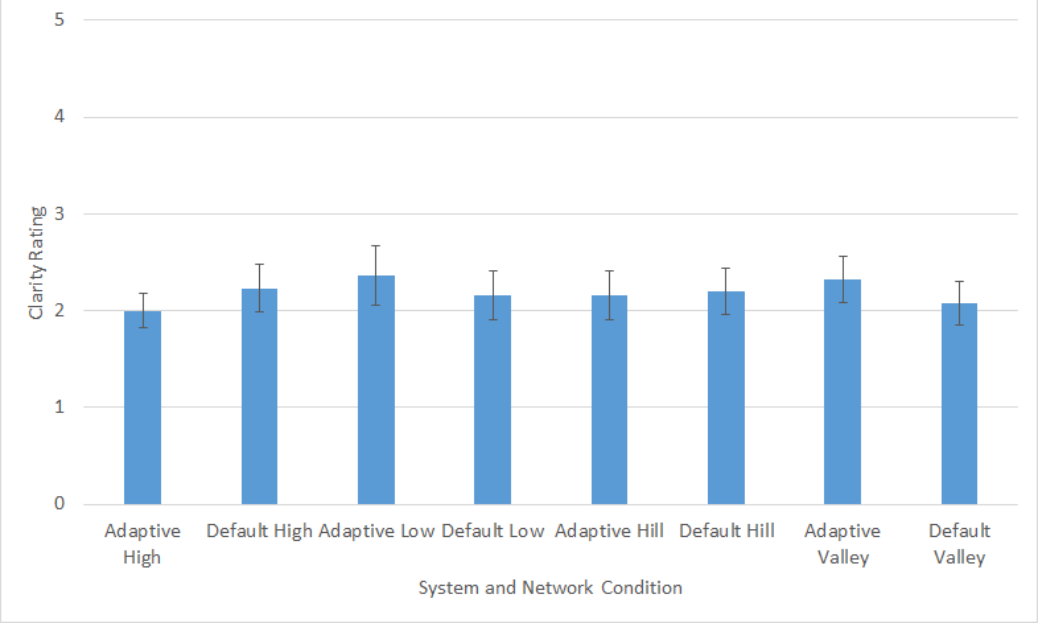

Figure 23: Average of user responses for question "How difficult was it to distinguish obstacles?" The error bars represent standard error.

Based on the information in Figure 22 and 23, users have almost identical ratings for both adaptive and default configurations. On all configurations, users rated the difficulty of

distinguishing obstacles relatively low. All configurations except for adaptive high and default valley received ratings at all values. This indicates that our system did not improve obstacle clarity, but maintained parity with the default system. The level seed does not affect a player's ability to distinguish obstacles.

4.1.5 Player Satisfaction

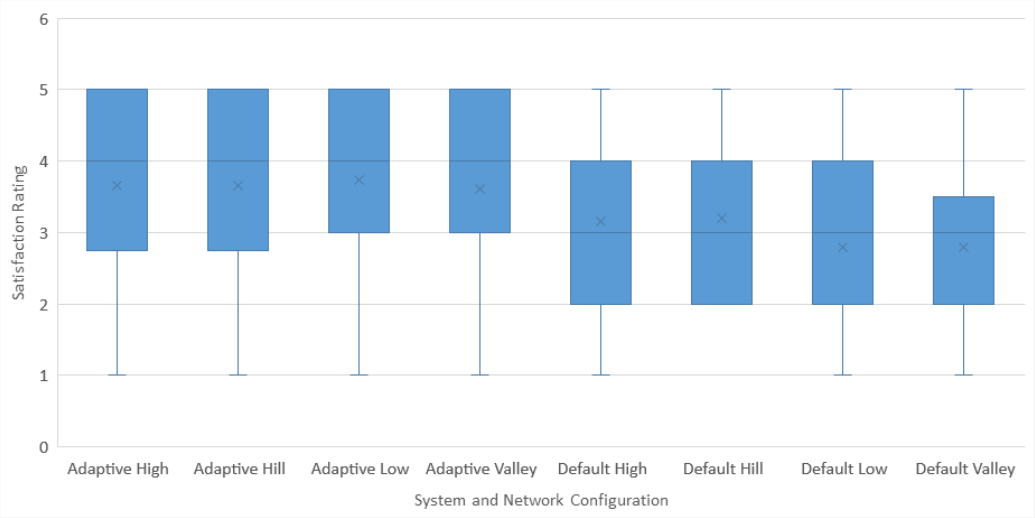

Figure 24: Box-and-whisker plot of user responses for question "How frustrated were you after this round?"

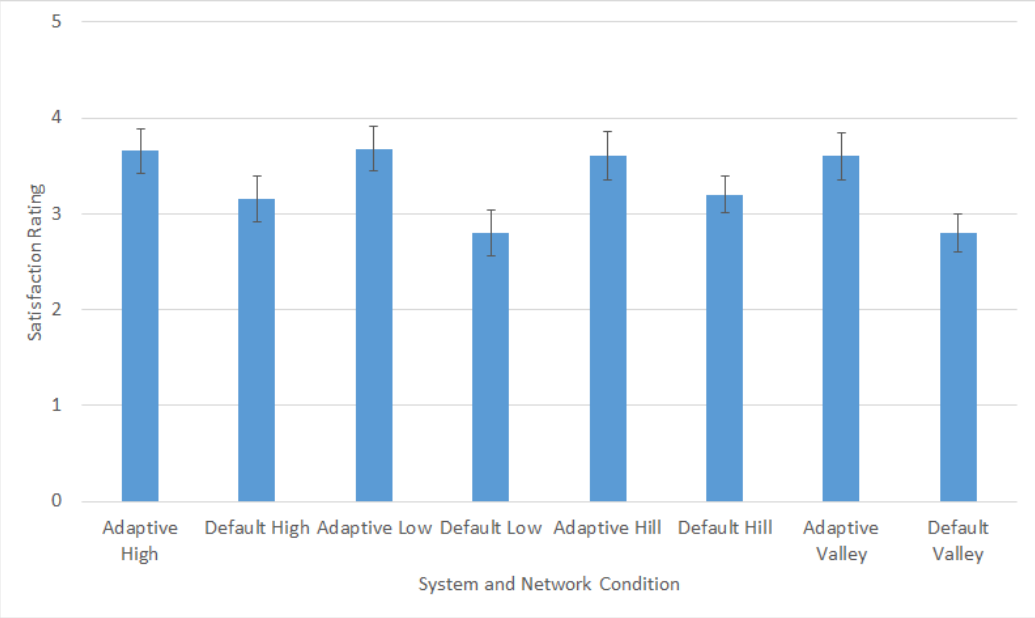

Figure 25: Average of user responses for question "How frustrated were you after this round?" The error bars represent standard error.

Based on the information in Figure 24 and 25, users on average rated higher levels of satisfaction with adaptive systems over default systems. Adaptive systems rated very close on average to a score of 3.5 out of 5, with approximately half of all ratings being a 4 or 5. Default systems averaged closer to a score of 3 out of 5, but had fewer ratings of 5. When compared with results from the first four survey questions, lower challenge ratings can correlate to higher satisfaction ratings for the adaptive system, despite slightly lower graphical ratings and similar ability to distinguish obstacles. Level seeding may have some effect on user satisfaction, as a more difficult seed compared to another user could make a user's trials more difficult and frustrating.

4.1.6 Player Retention

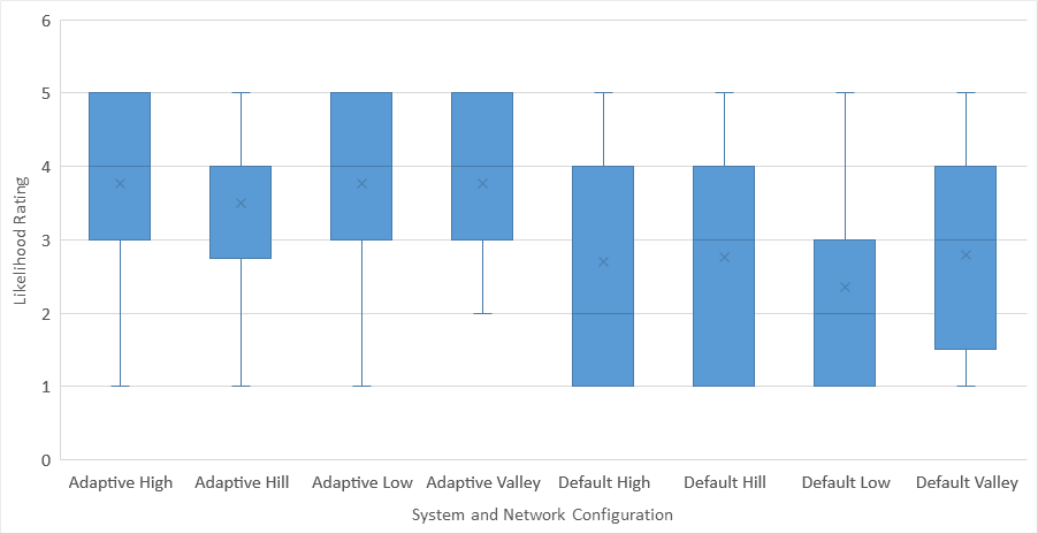

Figure 26: Box-and-whisker plot of user responses for question "Based soley on this round, how likely would you keep playing the game?"

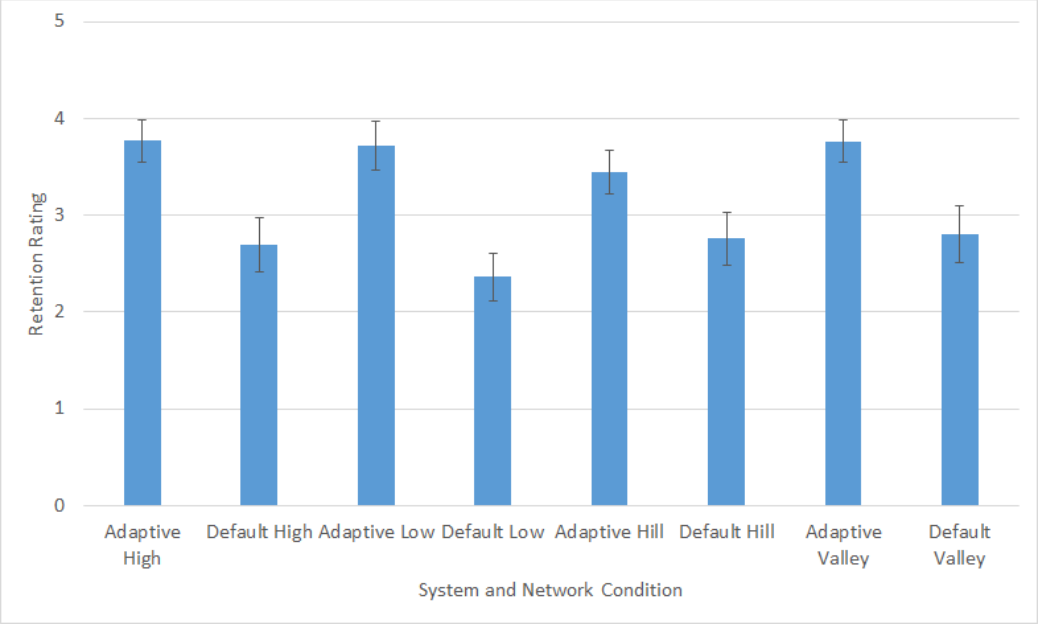

Figure 27: Average of user responses for question "Based soley on this round, how likely would you keep playing the game?" The error bars represent standard error.

Based on the information in Figure 26 and 27, users on average overwhelmingly rated higher chances of playing the game again on adaptive systems compared to default systems.

The differences between adaptive low and default low had the greatest difference, with a separation of approximately 1.5 points. Adaptive systems overall primarily received ratings ranging from 3 to 5 points, with some ratings a low as 1. Default systems however were primarily rated from 1 to 4 points, with the exception of default low which received a rating of mostly 3 or lower. Similar to question 5, level seeding may have some effect on user retention when compared to difficulty and frustration ratings.

4.1.7 Player Effort

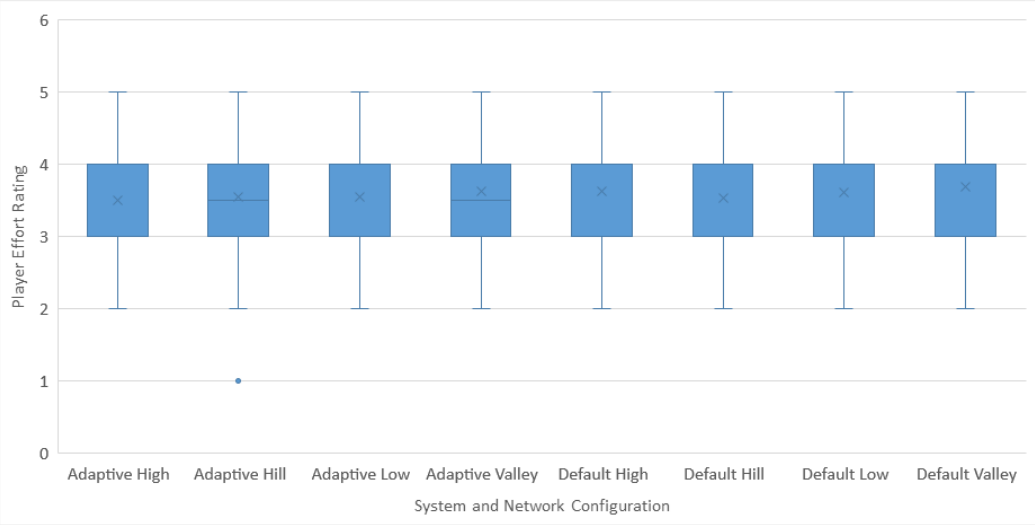

Figure 28: Box-and-whisker plot of user responses for question "How much effort did you put into the game during this past round?"

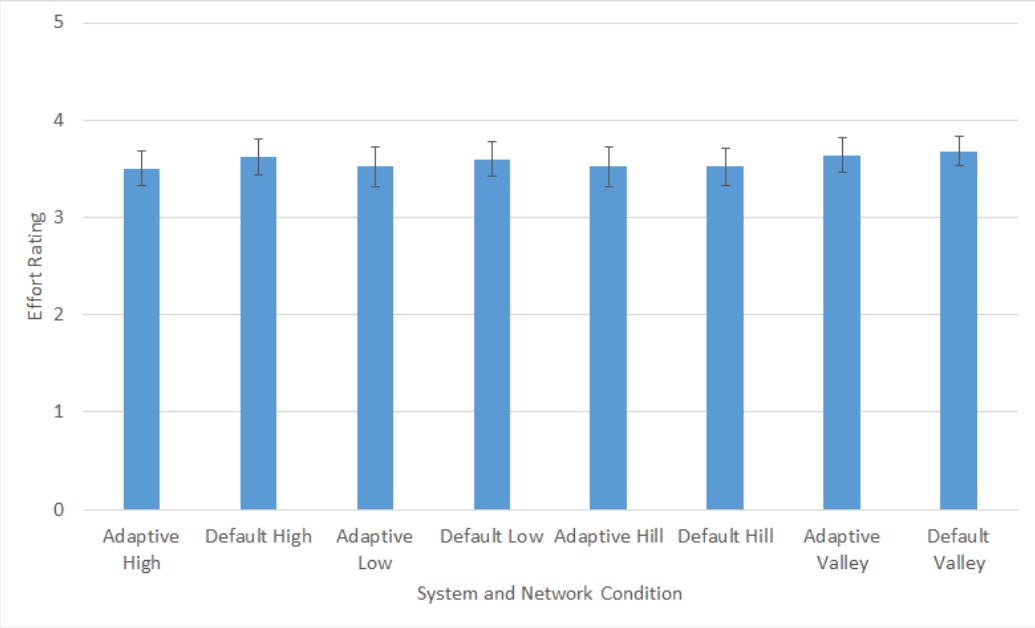

Figure 29: Average of user responses for question "How much effort did you put into the game during this past round?" The error bars represent standard error.

Based on the information in Figure 28 and 29, user effort ratings were almost identical across both adaptive and default systems. All averages are within error ranges for each network condition, indicating that the system and network condition did not have any effect on user effort. User responses on average rated an effort level of around 3.5, with some outlier surveys rating some or all trials either very high or very low. Some users reported a maximum effort level of 5 on all trials, while some users reported minimum effort levels for all trials. Level seeding has no apparent effect on user effort levels.

4.1.8 Significance Testing

System and Network Condition	How challenging was this run of the game?	How difficult was controlling the movement of your character?	How did the game look?	How difficult was it to distinguish obstacles?	How frustrated were you after this round?	Based soley on this round, how likely would you keep playing the game?	How much effort did you put into the game this past round?
High	**0.00342394**	**0.000737745**	0.615224703	0.326891913	0.172779668	**7.71053E-05**	0.416074077
Low	**0.002929591**	**2.04065E-09**	**0.046479507**	0.533068997	**0.006647962**	**6.55697E-05**	0.663916078
Hill	**0.005865324**	**0.001317617**	0.355708318	0.857057561	0.210528712	**0.013948313**	1
Valley	**0.019350236**	**0.00062473**	0.275986884	0.265248773	**0.034047207**	**0.007541578**	0.769807738

Figure 30: P-values from each network condition. We calculated values with a null hypothesis of zero mean difference. P-values less than 0.05 are shown in bold.

Figure 30 displays the P-values calculated from a paired two sample for means t-Test. We created each set of values using the adaptive and default survey results from each question, after which we took the two-tailed P-value. We used a significance level of 0.05 for all tests.

From these results, we found that the questions about graphical fidelity, distinguishing obstacles, and user effort had high P-values, which fail to reject our null hypothesis. Out of these three questions, the only result that was significant with a value of $P = 0.046$ was user responses on graphical fidelity for the low network setting. These results indicate that user responses on graphical fidelity and levels of effort are more subjective, and may vary widely between users, providing less significant data on the differences between the adaptive and default configurations.

Aside from the listed questions, most questions had very low P-values indicating significance, with the exception of user frustration. User frustration had significant values for low and valley network configurations, but failed to reject the null hypothesis for high and hill configurations. This suggests that user frustration depends on whether a user used an adaptive or non-adaptive system, but cannot be fully proven from the current data given.

4.2 Objective Testing Results

We calculated our objective testing results from video recordings that we took of Race the Sun. We recorded both the client and server on both adaptive and non-adaptive GA configurations and for each of these, we ran tests on three different bandwidth limitations.

4.2.1 Packet Loss and Effective Data Rates

GA Config	Default	Adaptive	Default	Adaptive	Default	Adaptive	Default	Adaptive
Scenario	No TC	No TC	High	High	Low	Low	Very Low	Very Low
TBF	None	None	32Mbit	32Mbit	4Mbit	4Mbit	1Mbit	1Mbit
Packets Sent	3245	11357	3245	5308	3859	3637	18960	5989
Packets Lost	0	0	0	0	527	0	13899	16
Duration (s)	9.89	9.91	10.15	9.89	13.06	26.49	64.20	123.72
Received (KB)	4450	15983	4431	7384	4526	4684	6478	6369
Loss Rate	0.00%	0.00%	0.00%	0.00%	13.66%	0.00%	73.31%	0.27%
Rate (Mbit)	3.51	12.60	3.41	5.83	2.71	1.38	0.79	0.40

Figure 31: Table showing packet loss rate and effective data rates of adaptive and non-adaptive GA over various network conditions. We calculated the "Rate" column by dividing the amount of data received by the duration of the stream.

Loss-rate was also recorded for each configuration on four bandwidth settings: unlimited, high, low, and very low. The default encoder bitrate used for the non-adaptive GA configuration was 3 Mbits. The server parses data collected from periodic net-reports returned from the client regarding the RTSP connection. The duration between net-reports is intermittent and is sometimes based on the amount of data successfully received. Figure 31 shows a table containing the collected data. The data for different streams were recorded over time intervals of varying sizes.

With both no traffic control and with high bandwidth, the our adaptive and the original system both incurred no packet loss. In both cases, the adaptive system successfully transmitted data at a higher rate than the non-adaptive version. This means that, given excess bandwidth, our adaptive solution is successfully able to detect excess bandwidth available to use and set an appropriate encoding rate to use it.

On the low bandwidth scenario, the adaptive system continues to incur no packet loss while the default system experiences a packet loss rate of 13.66%. Despite that the default software transmits data at a rate nearly twice that of the adaptive system (2.71 Mbit/s vs 1.38 Mbit/s), the frame loss causes the client unable to decode many of the frames. The seemingly low encoder rate set by our adaptive solution ensures that the stream does not drop a single packet and as a result the client is able to render every single frame of the original content.

The trend continues in the very low bandwidth scenario. In the very low bandwidth scenario, the adaptive system begins to experience packet loss. The loss rate is 0.27%. The system only dropped at most 16 frames over 123 seconds meaning that client would be unable to display one frame every 7.5 seconds. Again, the default system transmits data at a rate nearly twice that of the adaptive system (0.79 Mbit/s vs 0.40 Mbit/s). However, the packet loss was 73.31% meaning that client was unable to render many frames from the source video.

4.2.2 Objective Quality Results

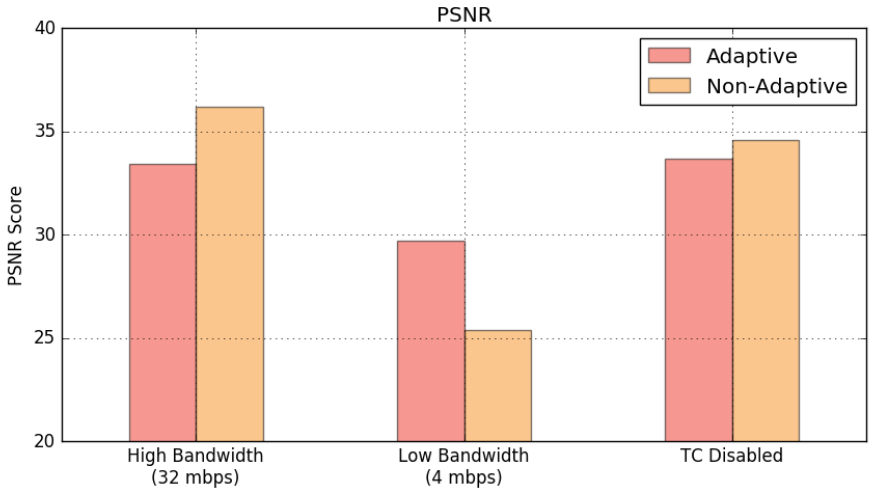

Figure 32: Comparing PSNR calculations between client and server of adaptive and non-adaptive GA at low and high bandwidth. PSNR is measured in decibels with higher being better.

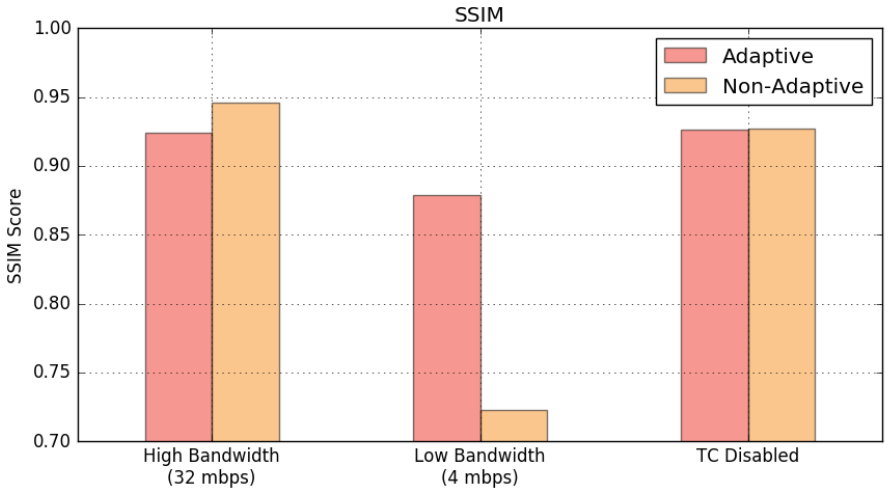

Figure 33: Comparing SSIM calculations between client and server of adaptive and non-adaptive GA at low and high bandwidth. An SSIM index of 1.0 indicates that the two signals are identical.

Based on the information in Figure 32 and Figure 33, GamingAnywhere suffers from signal corruption and quality loss in response to reduced bandwidth. We can see that the adaptive configuration of GA has higher PSNR values and SSIM values than the non-adaptive config-

uration in poor network conditions. It is interesting to note that the adaptive configuration of GA has a lower PSNR score and SSIM score than the non-adaptive configuration at a high bandwidth. When we disabled the TC network shaping tool the two configurations of GA scored almost identically in both objective measures.

In the high bandwidth environment, the difference in PSNR between the adaptive and non-adaptive configurations of GA is approximately 3.7 dB and the difference in SSIM is 0.22. Although the intent for the bandwidth was to be suitable for both adaptive and non-adaptive configurations, the adaptive configuration still outperformed the non-adaptive configuration by a sizable amount. When we disabled traffic control, the two configurations scored almost identically. It is possible that the high bandwidth of 32 MB/s was still not enough bandwidth and this may have caused the adaptive configuration to reconfigure the encoder to a lower bitrate, reducing objective measures. The GA encoder is set to encode video at 24 MB/s and so at 32 MB/s, the expected result was that the adaptive configuration would perform almost if not equal to the default GA configuration.

In the low bandwidth environment, this difference in PSNR is approximately 4.3 dB. This tells that, objectively, the adaptive bitrate GA shows the user a higher quality video stream than the non-adaptive version. The adaptive bitrate GA produces less signal noise when transmitting the video than the non-adaptive configuration. This may be due to the adaptive configuration actively modifying the video encoder to ensure that the video is within the bandwidth and and avoiding video corruption artifacts due to packet loss on the network. The non-adaptive configuration of GA however is still trying to squeeze the default bitrate-encoding of the video through the limited bandwidth thus creating noise in the image on the client and giving it a lower PSNR and SSIM value.

5 Conclusions

Adaptive bitrate streaming is a technique that allows for quality content delivery under variable network conditions. Since cloud gaming requires the constant, reliable delivery of quality data, adaptive bitrate streaming naturally appeals as a solution. Since each packet sent in a cloud gaming environment is important to the play-ability of the game, it would be ideal that a congestion control algorithm that minimizes loss be used to manage the data stream. BBR is such an algorithm.

BBR avoids packet loss while making efficient use of available network capacity its secondary objective. In our user test, we found that bitrate adaptation made the same exact system conditions less challenging for testers to play. By minimizing the processing delay required by the bottleneck link to empty its queue, the algorithm is able to improve game responsiveness and level of control in a reaction-intensive game. From our user study, we were able to conclude that our bitrate adaptation extension improved the retention rate of users for the game. However, we were unable to draw conclusions about the other questions asked on our survey including user frustration and user effort based on high p-values.

We were able to conclude from our objective testing that our bitrate adaptation extension improved the quality of a video stream at reduced bandwidth. Under high bandwidth or idea conditions, our adaptive configuration took advantage of the available bandwidth and sent more data. We believe this because when we tested both configurations of GA without any bandwidth limitations, the two video streams produced nearly identical results.

6 Future Work

This section discusses different areas of work that others can expand upon to give better results for our implementation of BBR on GA.

6.1 Temporal or Spatial Scaling

Our system currently only scales the quality of the stream in response to bandwidth information. We modified the encoder-video module to accept frame-rate and video resolution parameters for reconfiguration. However, our bitrate adaptation heuristic only changes the bitrate parameter in the reconfiguration message. A contributor could apply temporal or spatial scaling to our system in order to improve user experience for specific applications.

6.2 Different Games

We chose Race the Sun because it fit certain criteria that would be particularly impacted by packet loss due to network congestion. However, we only tested on this one game. An area for future study is to test the algorithm on other types of games to diversify the results from this single game.

One suggestion is a first-person shooter (FPS) game. An FPS game requires that a player be able to aim and shoot accurately and such an action can be severely limited by lag and screen tearing. Testing BBR on such a game would give results that could potentially be relevant to a lot of other FPS games.

6.3 Control and Input Types

The controls tested in our study only involved the 'left' and 'right' arrow keys to maneuver the craft and 'return' to pause the game. Future work could involve testing other games with control schemes that use a larger variety of keys on the keyboard. Another suggestion is to test games that require mouse input. Games based on mouse input that suffer from latency could have different effects on a user than a keyboard.

6.4 Volunteer Diversity

A final suggestion to expand on this study is to test the bitrate adaptation algorithm on a more diverse set of subjects. Our test subject pool mainly consisted of males aged 18-22 with only 3 users falling outside of this category. Additionally, a large portion of these users were Computer Science majors or Interactive Media and Game Development majors. These two majors, in particular, involve spending a lot of time on a computer and so these users may be more familiar with computers and games than the average person. Future work could involve testing the algorithm on a larger, more diverse population to get more meaningful and reliable data.

References

[1] What is H.264. 2010. www.h264info.com/h264.html.

[2] Peak Signal-to-Noise Ratio as an Image Quality Metric. National Instruments, 2013.

[3] Is it possible to stream non Steam games? :: Steam In-Home Streaming, January 2015. steamcommunity.com/groups/homestream/discussions/0/630802978857564429/.

[4] PlayStation Now PS Now Subscription for PS3 Games. Sony Interactive Entertainment LLC, 2016. www.playstation.com/en-us/explore/playstationnow/.

[5] Mark Allman, Sally Floyd, and Craig Partridge. Increasing TCP's initial window. Technical report, 2002.

[6] Kush Amerasinghe. H.264 For the Rest of Us, 2009.

[7] Beyer, J. and Varbelow, R. Stream-A-Game: An Open-Source Mobile Cloud Gaming Platform. In Network and Systems Support for Games. IEEE, 2015.

[8] T. Bryant, 2013. U.S. Patent Application No. 13/863,732.

[9] Cardwell, Neal and Cheng, Yuchung and Gunn, C Stephen and Yeganeh, Soheil Hassas and Jacobson, Van. BBR: Congestion-Based Congestion Control. *Queue*, 14(5):50, 2016.

[10] K. Chen. Experiment Setup. GamingAnywhere.org, 2013. gaminganywhere.org/perf.html.

[11] Chen, K. GamingAnywhere - An Open Source Cloud Gaming System, 2013. gaminganywhere.org/.

[12] Dabrowski, R. and Manuel, C. and Smieja, R. *The Effects of Latency on Player Performance and Experience in a Cloud Gaming System*. PhD thesis, Worcester Polytechnic Institute, 2014.

[13] R. Devine. PlayStation Now games on PC are terrific... when they actually work. September 2016. www.windowscentral.com/playstation-now-pc-terrific-when-it-actually-works.

[14] Domanico, Robyn and Suarez, Matthew and Wang, Lambert. magellantoo/GamingAnywhere. github.com/magellantoo/gaminganywhere, 2016.

[15] J. Donnelly. PlayStation Now on PC isn't quite Netflix for games, but it comes close. August 2016. www.pcgamer.com/playstation-now-on-pc-isnt-quite-netflix-for-games-but-it-comes-close.

[16] Dr. Dmitriy Vatolin. *MSU Video Quality Measurement Tool*. Michigan State University Graphics & Media Lab, 2008.

[17] Kevin Fall and Sally Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *ACM SIGCOMM Computer Communication Review*, 26(3):5–21, 1996.

[18] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[19] C. Y. Huang, K. Chen, D. Y. Chen, H. J. Hsu, and C. H. Hsu. GamingAnywhere: The first open source cloud gaming system. ACM Transactions on Multimedia Computing, Communications, and Applications, 2013.

[20] C. Y. Huang, C. H. Hsu, Y. C. Chang, and K. T. Chen. GamingAnywhere: an Open Cloud Gaming System. In Proceedings of the 4th ACM multimedia systems conference, February 2013.

[21] Huang, C. GamingAnywhere. github.com/chunying/gaminganywhere, 2016.

[22] Jarschel, M. and Schlosser, D. and Scheuring, S. and Hossfeld, T. Gaming in the clouds: QoE and the users' perspective. Master's thesis, January 2012.

[23] Mahesh Kamat, Roopa Kulkarni, and Prashant Patavardhan. Analysis of lossy video compression technique. 2015.

[24] Flippfly LLC. Race The Sun. flippfly.com/racethesun/, 2013.

[25] Madhavan. Video Group of MSU Graphics and Media Lab. May 2007. videocodecs.blogspot.com/2007/05/image-coding-fundamentals_08.html.

[26] J. Newman. GeForce Now review: Nvidia's 'Netflix for PC games' is impressive, but impractical. 2013. www.pcworld.com/article/2991944/software-games/geforce-now-review-nvidias-netflix-for-pc-games-is-impressive-but-impractical.html.

[27] T. Q. Nguyen, J. Kay, and J. Pasquale. Fast Source-based Dithering for Networked Digital Video. Master's thesis, University of California, San Diego, April 1994. http://cseweb.ucsd.edu/ pasquale/Papers/spie94.pdf.

[28] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy. Bandwidth estimation: Metrics, measurement techniques, and tools. Master's thesis, Georgia Institute of Technology, 2003.

[29] S. Sarkar. GeForce Now, Nvidia's 'Netflix for games,' expands with Sega and Warner Bros. March 2016. www.polygon.com/2016/3/15/11222650/nvidia-geforce-now-sega-warner-bros-developer-program.

[30] Henning Schulzrinne. Real time streaming protocol (rtsp). 1998.

[31] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. Master's thesis, IEEE, September 2007. http://ip.hhi.de/imagecom_G1/assets/pdfs/Overview_SVC_IEEE07.pdf.

[32] J. E. Sheedy, M. V. Subbaram, A. B. Zimmerman, and J. R. Hayes. Text Legibility and the Letter Superiority Effect. Master's thesis, The Ohio State University, 2005.

[33] The FFmpeg developers. Libavcodec. Technical report, 2016. www.ffmpeg.org/libavcodec.html.

[34] Jon Thompson, Jennifer Sun, Richard Mller, Mathias Sintorn, and Geoff Huston. Akamais [State of the Internet] Q3 2016 report, 2016.

[35] Valve. Steam Support. Valve Corporation, 2015. support.steampowered.com/kb_article.php?ref=3629-RIAV-1617.

[36] D. Vatolin and S. Grishin. MSU Lossless Video Codecs Comparison. March 2007. www.compression.ru/video/codec_comparison/lossless_codecs_en.html.

[37] Wang, Z. and Simoncelli, E. P. and Bovik, A. C. Multi-Scale Structural Similarity for Image Quality Assessment. Master's thesis, November 2003.

[38] Michael Welzl. *Network congestion control: managing internet traffic.* John Wiley & Sons, 2005.

[39] R. Whitwam. Testing: Nvidia's GeForce Now Game Streaming Service. November 2015. www.tested.com/tech/551016-testing-nvidia-geforce-now-game-streaming-service/.

[40] Zink, Michael and Künzel, Oliver and Schmitt, Jens and Steinmetz, Ralf. Subjective impression of variations in layer encoded videos. In *international Workshop on Quality of Service*, pages 137–154. Springer, 2003.

Appendices

A User Study Survey

Latency and Cloud Gaming: Adapting Google's BBR

GamingAnywhere is an open-source cloud gaming platform. Cloud gaming is a way to deliver a high-quality gaming experience by running games on a powerful cloud server while gamers interact with the game on a thin client. The thin clients are lightweight and can be ported to resource-constrained platforms such as mobile devices.

GamingAnywhere does not have a feature that allows itself to adapt well to a changing network connection. We implemented an adaptive bitrate streaming algorithm to counter this. The purpose of this study is to see how the algorithm affects user performance in games and to learn how users feel when play the games. We would appreciate your help and input in our efforts.

Pre-Survey Questions

1. How often do you play video games?
   Daily Weekly Monthly Rarely Never

2. How many hours per week do you play video games? _____
3. Would you classify yourself as a gamer?
   (No) 1 2 3 4 5 (Yes)
4. Have you played Race the Sun before?
   Yes No
5. How would you rate yourself at games that require quick reactions?
   (Less skilled) 1 2 3 4 5 (More skilled)
6. How important is graphical fidelity to you when playing games?
   (Less important) 1 2 3 4 5 (More important)

7. (Optional) What is your age? _____

8. (Optional) What is your gender? _____
9. If you wish to be entered into our raffle, please write your email. We will only use this information to contact you regarding the raffle.

   _____

Post-Run 1 Questions

1. How challenging was this run of the game?
   (Very Easy) 1 2 3 4 5 (Very Challenging)
2. How difficult was controlling the movement of your character?
   (Very Easy) 1 2 3 4 5 (Very Challenging)
3. How did the game look?
   (Very blurry) 1 2 3 4 5 (Very clear)
4. How difficult was it to distinguish obstacles?
   (Very Easy) 1 2 3 4 5 (Very Challenging)
5. How frustrated were you after this round?
   (Very frustrated) 1 2 3 4 5 (Very content)
6. Based solely on this round, how likely would you keep playing the game ?
   (Very unlikely) 1 2 3 4 5 (Very likely)
7. How much effort did you put into the game during this past round?
   (Minimal) 1 2 3 4 5 (Maximum)

Post-Run 2 Questions

8. How challenging was this run of the game?
   (Very Easy) 1 2 3 4 5 (Very Challenging)
9. How difficult was controlling the movement of your character?
   (Very Easy) 1 2 3 4 5 (Very Challenging)
10. How did the game look?
    (Very blurry) 1 2 3 4 5 (Very clear)
11. How difficult was it to distinguish obstacles?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
12. How frustrated were you after this round?
    (Very frustrated) 1 2 3 4 5 (Very content)
13. Based solely on this round, how likely would you keep playing the game ?
    (Very unlikely) 1 2 3 4 5 (Very likely)
14. How much effort did you put into the game during this past round?
    (Minimal) 1 2 3 4 5 (Maximum)

Post-Run 3 Questions

15. How challenging was this run of the game?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
16. How difficult was controlling the movement of your character?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
17. How did the game look?
    (Very blurry) 1 2 3 4 5 (Very clear)
18. How difficult was it to distinguish obstacles?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
19. How frustrated were you after this round?
    (Very frustrated) 1 2 3 4 5 (Very content)
20. Based solely on this round, how likely would you keep playing the game ?
    (Very unlikely) 1 2 3 4 5 (Very likely)
21. How much effort did you put into the game during this past round?
    (Minimal) 1 2 3 4 5 (Maximum)

Post-Run 4 Questions

22. How challenging was this run of the game?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
23. How difficult was controlling the movement of your character?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
24. How did the game look?
    (Very blurry) 1 2 3 4 5 (Very clear)
25. How difficult was it to distinguish obstacles?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
26. How frustrated were you after this round?
    (Very frustrated) 1 2 3 4 5 (Very content)
27. Based solely on this round, how likely would you keep playing the game ?
    (Very unlikely) 1 2 3 4 5 (Very likely)
28. How much effort did you put into the game during this past round?
    (Minimal) 1 2 3 4 5 (Maximum)

Post-Run 5 Questions

29. How challenging was this run of the game?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
30. How difficult was controlling the movement of your character?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
31. How did the game look?
    (Very blurry) 1 2 3 4 5 (Very clear)
32. How difficult was it to distinguish obstacles?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
33. How frustrated were you after this round?
    (Very frustrated) 1 2 3 4 5 (Very content)
34. Based solely on this round, how likely would you keep playing the game ?
    (Very unlikely) 1 2 3 4 5 (Very likely)
35. How much effort did you put into the game during this past round?
    (Minimal) 1 2 3 4 5 (Maximum)

Post-Run 6 Questions

36. How challenging was this run of the game?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
37. How difficult was controlling the movement of your character?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
38. How did the game look?
    (Very blurry) 1 2 3 4 5 (Very clear)
39. How difficult was it to distinguish obstacles?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
40. How frustrated were you after this round?
    (Very frustrated) 1 2 3 4 5 (Very content)
41. Based solely on this round, how likely would you keep playing the game ?
    (Very unlikely) 1 2 3 4 5 (Very likely)
42. How much effort did you put into the game during this past round?
    (Minimal) 1 2 3 4 5 (Maximum)

Post-Run 7 Questions

43. How challenging was this run of the game?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
44. How difficult was controlling the movement of your character?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
45. How did the game look?
    (Very blurry) 1 2 3 4 5 (Very clear)
46. How difficult was it to distinguish obstacles?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
47. How frustrated were you after this round?
    (Very frustrated) 1 2 3 4 5 (Very content)
48. Based solely on this round, how likely would you keep playing the game ?
    (Very unlikely) 1 2 3 4 5 (Very likely)
49. How much effort did you put into the game during this past round?
    (Minimal) 1 2 3 4 5 (Maximum)

Post-Run 8 Questions

50. How challenging was this run of the game?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
51. How difficult was controlling the movement of your character?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
52. How did the game look?
    (Very blurry) 1 2 3 4 5 (Very clear)
53. How difficult was it to distinguish obstacles?
    (Very Easy) 1 2 3 4 5 (Very Challenging)
54. How frustrated were you after this round?
    (Very frustrated) 1 2 3 4 5 (Very content)
55. Based solely on this round, how likely would you keep playing the game ?
    (Very unlikely) 1 2 3 4 5 (Very likely)
56. How much effort did you put into the game during this past round?
    (Minimal) 1 2 3 4 5 (Maximum)