AN INVESTIGATION OF A HYBRID COMPUTATIONAL SYSTEM FOR CLOUD GAMING

A Thesis Submitted to the Committee on Graduate Studies in Partial Fulfillment of the Requirements for the Degree of Master of Science in the Faculty of Arts and Science

TRENT UNIVERSITY

Peterborough, Ontario, Canada

Applied Modelling & Quantitative Methods M.Sc. Graduate Program

September 2023

Abstract

An Investigation of a Hybrid Computational System for Cloud Gaming

Sean Baxter

Video games have always been intrinsically linked with the technology available for the progress of the medium. With improvements in technology correlating directly to improvements in video games, this has recently not been the case. One recent technology video games have not fully leveraged is Cloud technology. This Thesis investigates a potential solution for video games to leverage Cloud technology. The methodology compares the relative performance of a Local, Cloud and a proposed Hybrid Model of video games. We find when comparing the results of the relative performance of the Local, Cloud and Hybrid Models that there is potential in a Hybrid technology for increased performance in Cloud gaming as well as increasing stability in overall game play.

Keywords: streaming, cloud gaming, video game, cloud

Acknowledgments

I would like to thank Trent University for their continued support especially my thesis supervisor Dr. Richard Hurly and Brian Srivastava for their wonderful help and contributions to this thesis. I would also like to thank Professor Wesley Burr for his support in understanding probability distributions. I would also like to thank my family for their continued support throughout this entire process, especially while working on a thesis during the COVID pandemic. Without my family's support I would not have completed this thesis.

Table of Contents

List of Figures

List of Tables

Definitions

Bandwidth - The amount of data that can move through a network in a given time period. The throughput of a network,

Latency - The time between a action and response in a system.

Lossy compression - Compression style that the resulting data cannot be used to rebuild the original data. This style of compression allows for in general smaller resulting files but has a potential loss of data that cannot be restored.

Lossless compression - Compression style where the resulting compressed data can be used to rebuild the original uncompressed data set. This in general means that the compression cannot push file sizes down as far as lossy compression, but is very useful when the validity of data has to be preserved. (VP9 YouTube's compression allows for lossless)

Asset - Any piece of content used to assemble a game, including graphic models, images, textures, sound files, and code segments used to assemble and process a video game.

Thin Client – a low processing power computing device to input and display results with minimal processing abilities.

Thick Client – a high processing power computing device that can handle processing complex tasks.

Cloud Service – On-Demand Computer Service accessible through a client device

Cache – data storage location that allows for high-speed data retrieval

Cloud gaming - Using an on-demand cloud service to play Video Games

Online gaming - The act of playing a Video game that has an online component as a core design piece

Interarrival Time - The time between each arrival into the system and the next.

Chapter 1

Introduction

1.1 Introduction

Cloud Streaming has changed the way many people engage with digital media, allowing for users to access and interact with this media on a wide range of client devices as long as a network connection is available. This has shifted media consumption from physical media, and locally stored downloads, to remote access to Cloud storage for video, audio and image content. Video games have failed to leverage these new Cloud technologies because of inherent increases in network latency which results in an adverse effect on user enjoyment of video games. Cloud gaming without latency problems would open this activity to many new users. This thesis aims to investigate a new approach for Cloud gaming called Hybrid Cloud gaming.

This thesis explores the potential of a Hybrid Cloud gaming solution to improve the performance of Cloud gaming. We found that leveraging both Local and Cloud components for Cloud gaming improves performance and leads to a more consistent game experience. In addition, thesis adds to the field of Cloud gaming by proposing a new approach to distributing computational requirements between Local and Cloud processing. We use simulation to compare the Mean File Transfer times of a Local, Cloud and our proposed Hybrid model to find that a Hybrid solution can provide a suitable environment for interacting with video games while leading to improvements in input latency inherent in Cloud gaming systems.

Video games are an interactive medium whereby a player, or players, use a piece of software to accomplish either a user-generated or design-determined goal. Video games have always been reliant on advancements in hardware to allow for new experiences and improved quality in the medium, as technology improved, so did the quality and complexity of video games. With early video games, advances in technology removed design limits and allowed for more complex games to be produced. For this reason, the development of video games as a medium has been strongly tied to advances in computer hardware, most notably computer graphics. While video games have continued this connection with each new advancement, attempts to utilize Cloud technology have had limited success. One example of this is the company OnLive which failed to gain a large market share and closed down in 2015 [1].

Multimedia streaming is the process of sending a constant series of packets along a network that creates a multimedia file that can be played on the client device without requiring the installation or direct storage of the file. Streaming services have changed the landscape of media consumption, from *Netflix* [2] changing the way people experience and consume movies and television shows, to *Spotify* [3] becoming one of the most common ways for users to consume music. This shift toward consuming media through streaming services has fundamentally changed the landscape of media production. *YouTube* [4] continues to be the largest force in streaming, and recently the shift towards live streaming, specifically on websites such as *Twitch* [5], has allowed users to both watch and provide live streaming content [6] [7].

Video game live streams have thrived with these new technologies, as many streamers make a living by playing popular video games through websites such as *Twitch*, which can be watched by thousands of people. Live stream technology has had a profound effect on how users consume the medium in a variety of different contexts. Specifically for video gamers, live streaming has allowed users to showcase their gameplay, and the increased popularity of live streaming has changed how games are presented to the public. This has also had a large effect on how games are advertised [8]. Previously games were advertised very traditionally, relying on static print, video, and website-based advertisements similar to how films and television series are advertised. Meanwhile live streaming games allows users to see gameplay in real-time, creating an environment where live streamers act as advertisers for the video games they play while receiving revenue from live streaming.

Even though video game live streaming has a large cultural impact on how people engage with video games, Cloud gaming technology has failed to penetrate the gaming zeitgeist. This failure to gain traction comes from two important factors; Cloud gaming has failed to attract existing users, because of its performance failures on the technical side caused by latency [9]. Due to these limitations, the medium of video games has not achieved the same in utilizing streaming technologies for Cloud gaming as *Netflix* and *Spotify* have achieved for televisual and music media respectively.

To properly understand video games and the way that Cloud technologies can interact with this medium it is important to explore previous technologies. In this

chapter, I will introduce the concepts of Local Gaming, Cloud gaming as well as introduce my proposed solution of Hybrid Gaming.

1.2 Local Gaming

Video games started as experiments by computer scientists to see what they could create using then-current technologies [10]. One of the first commercially released video games was Atari's *Pong* in 1972, which was released on what is widely accepted as the first video game home console the Brown Box [11]. The Brown Box had moderate success but did not gain large traction in households, primarily due to its high price point. Following this, Atari began to focus on arcade cabinets with one of their largest successes being *Space Invaders* in 1978, along with other breakouts such as *Pac-Man*, developed in 1980 by NAMCO, and *Donkey Kong*, developed in 1981 by Nintendo [11]. One interesting aspect of the arcade cabinet technology is that each cabinet was built for one specific purpose. A *Space Invaders* cabinet would only play *Space Invaders*, while a *Donkey Kong* cabinet would only play *Donkey Kong*. This meant that improvements in video game performance and graphics were intrinsically tied to the hardware in the cabinet. A substantial amount of work on video games during this time was focused on the improvement of cabinet boards to allow for more complex and interesting arcade games.

Following the arcade boom, work began to generalize video game hardware allowing for cabinets to swap games, rather than needing complete cabinet replacements with each new game release [12]. This occurred with home video consoles

such as the Nintendo Entertainment System (NintendoCo. Ltd), originally known as the Family Computer (Famicom) in Japan, and was initially released in 1983 in Japan, and subsequently released in 1985 in North America [11]. At the same time, arcade cabinets allowing for swapping games began to be developed and released, such as the Neo Geo Multi Video System (MVS) developed by SNK Corporation in 1990. [13]

Video games were still limited by the technologies available, meaning that any improvements in hardware allowed for large technical developments in video game complexity and graphics. This was particularly noticeable with the subsequent release of new home game consoles such as the Nintendo 64 (NintendoCo. Ltd) in 1996. These new consoles permitted video games to make noticeable leaps in quality, particularly with 3D video games becoming the new standard on home consoles. While home consoles were improving, arcade cabinets still had the edge in terms of hardware performance, which resulted in the most visually impressive games of the era [10]. Looking at game consoles on the market today, processing power has improved to such a degree that hardware is no longer a limiting factor. We no longer see large visual improvements in games coming from hardware improvements but instead, see improvements in the form of software changes in design philosophy and innovative ideas built upon a mostly plateaued hardware environment.

Video game software is one of the most complex and computationally taxing processes that can be run on a modern consumer-grade computer [14]. A complex series of tasks, simulations and rendering pipelines are necessary to produce a final useable result to display to the user. Contemporary video games often require

specialized hardware either in the form of personal computers being entirely built for the sole purpose of playing video games, or home consoles such as the Sony PlayStation and Microsoft Xbox series. As video gaming continues to grow, more people are investigating what makes a Local system work and how to best design a Local video gaming experience. One of the key issues is in performance, specifically reducing total system latency to create the best possible experience for a user.

Local video gaming systems have multiple delay points that factor into the overall system latency: input latency, simulation latency, render latency and display latency. Figure 1.1 illustrates total system latency as outlined in this section.

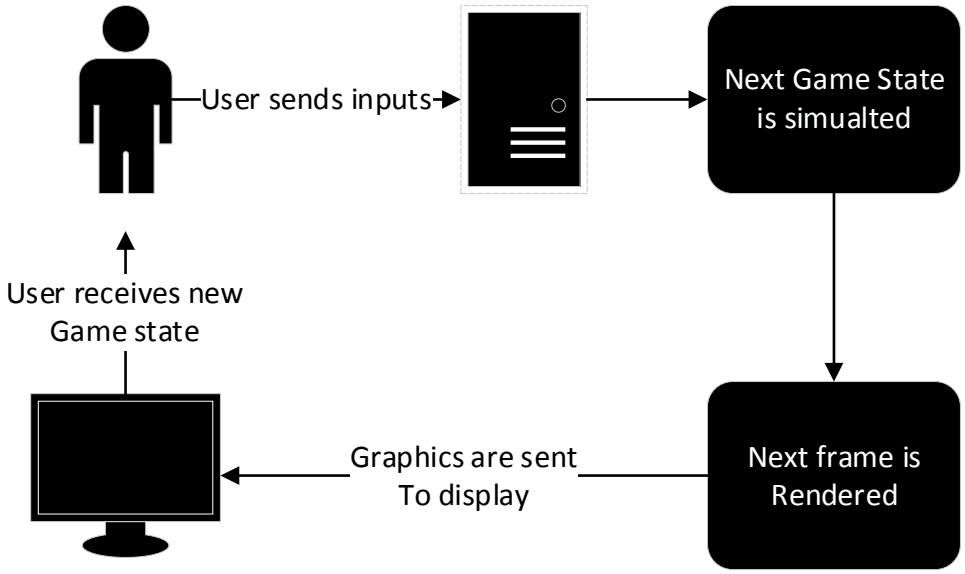

Figure 1.1 Layout of Pieces of Total System Latency in a Local System

Input latency is the time it takes for a gaming controller to receive an input and transfer it to the Local system [15]. Input latency varies depending on the type of

controller, with typical controller latencies currently are around 24ms [16]. For this thesis, a more in-depth discussion of controller latency, and how controller latency changes depending on various factors, is not relevant. Thus, we will assume that the average controller latency for our simulation is 24ms. The Local system then checks to see if new inputs, based on a set time interval, have arrived and proceeds to collect these. [17] [16]. The Local system then uses any input that has arrived, as well as the current game state, to simulate the next game state. This time makes up simulation latency and is variable from game to game. Once the new game state has been simulated, the system then renders that game stage into computer graphics through a graphics pipeline, this makes up render latency. Once rendering has been completed the finished result is passed to the display. The display then also has slight latency inherent in transferring the graphics data from the Local console to the display and updating the display to match the newest render, which makes up display latency. All of these forms of latency sum together to create total overall Local system latency.

Local gaming continues to be the platform of choice for the majority of video game players. The strength in Local gaming is improved performance and response time compared to Cloud options however its weakness is the necessity of continual upgrading as well as being a stationary option. In the next section, I will discuss Cloud gaming and the advantages and disadvantages of a Cloud environment.

1.3 Cloud Gaming

Cloud gaming allows for video games to be run on a Cloud architecture permitting users to access video games that they do not have installed on their clients, such as Local desktops, laptops, or game consoles. It also allows for streaming to thin clients, such as smartphones, tablets, or low-end personal computers [18]. One such example is a Chrome Book that otherwise would not have the processing power required to properly display certain video games.

Cloud services, such as OnLive, were early examples of Cloud technologies being used to deliver video games to users. In 2019 developer *Google* released *Google Stadia* as the most recent Cloud gaming service to deliver video game content. [19] [20]. While media streaming, for music and video, has reached a large audience, it is a passive service as the user is simply consuming content. The key difference between streaming media and Cloud gaming is that the user is an active participant in Cloud gaming streams.

With music and video, the content is sent through a network to the user and displayed once it arrives. Through buffering content in advance, interruptions to the stream can be hidden from the user, allowing the stream to run almost seamlessly [21]. For a Cloud gaming experience to properly process the next result, it requires continuous inputs from the user. Therefore, the typical method in the streaming of content, such as buffering the data for upcoming content in advance, will not work. This is because the expected content in a Cloud gaming session is dependent on user input,

and the content cannot be predetermined as it is in a video or audio stream. When listening to a song on *Spotify*, you are a passive consumer of the audio content. When playing a Cloud game stream, you are an active participant in the creation of the next moment of content. Figure 1.2 illustrates a general overview of how Cloud gaming services work.

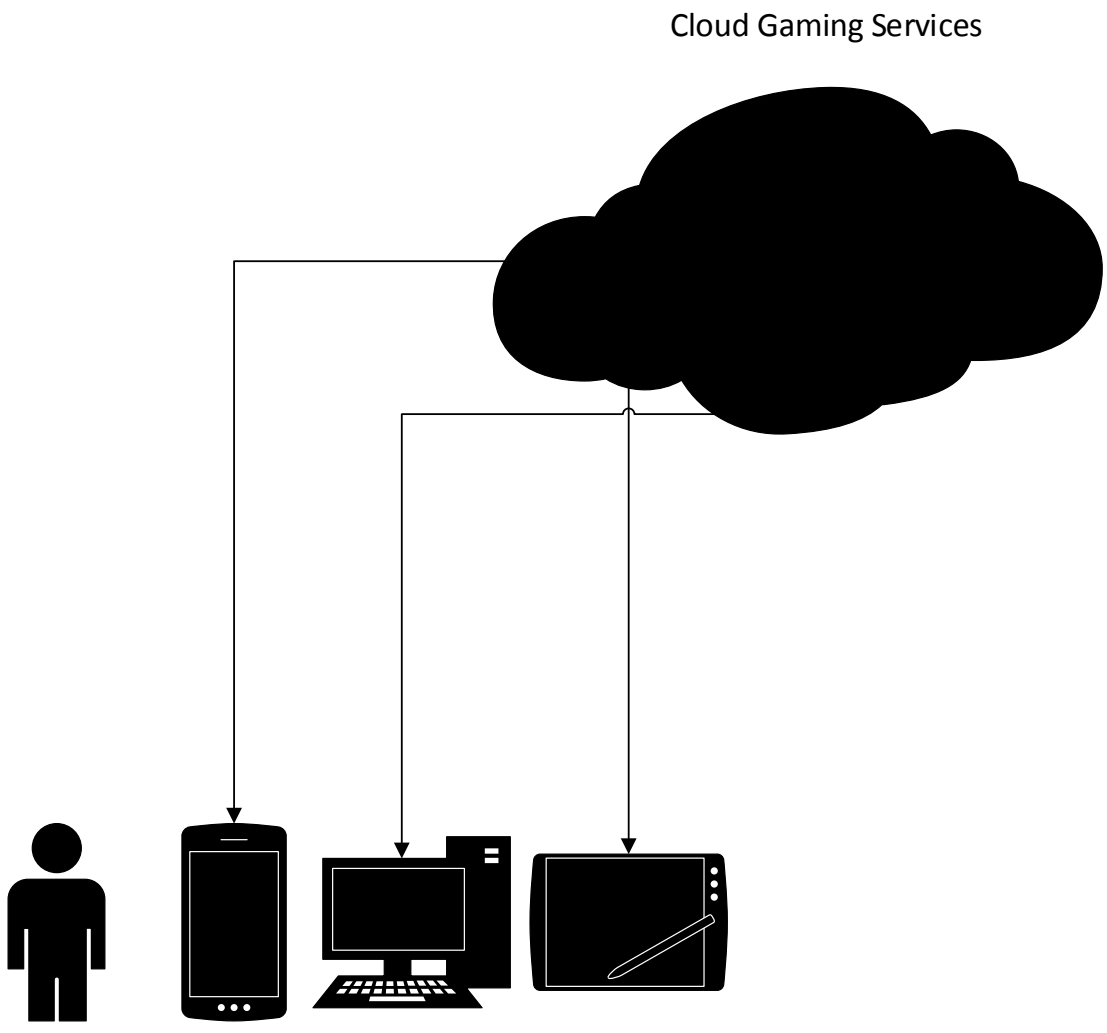

Figure 1.2 Overview of Cloud Gaming

Cloud gaming like all Cloud services has the advantage of only requiring a connection from the user to the Cloud storage location. As shown in Figure 1.2 a user can use any of multiple potential thin clients to establish a connection from the Cloud service and initiate a piece of media, in this case, a video game, through a Cloud stream and interact and use that media without having to load or process any of the media states on their device, instead downloading the result of the game.

Current Cloud gaming services have an unavoidable increase in input delay, which is caused by the requirement for inputs to be sent through a network and the result being sent back across the network. These input delays, inherent in the network connection in Cloud gaming, cause the responses to a user's input to feel sluggish, reducing engagement, and thus, user enjoyment since the experience no longer feels responsive. The addition of sending information through a network nearly doubles input delays, potentially adding 100ms of delay [14] compared to average controller input delays which are around 24ms as mentioned earlier in this section.

The point at which input delay is noticeable varies depending on both the video game and the user. The game types that demand the fastest response times are First-Person Shooters (FPS), Racing, Rhythm, and Fighting Games, with FPS and Racing games expecting 100ms or faster response times [22]. Other games such as turn-based and omnipresent management games are much less affected by latencies and perform well even with large delays. Figure 1.3 illustrates where additional latency is added to a Cloud gaming experience by the network.

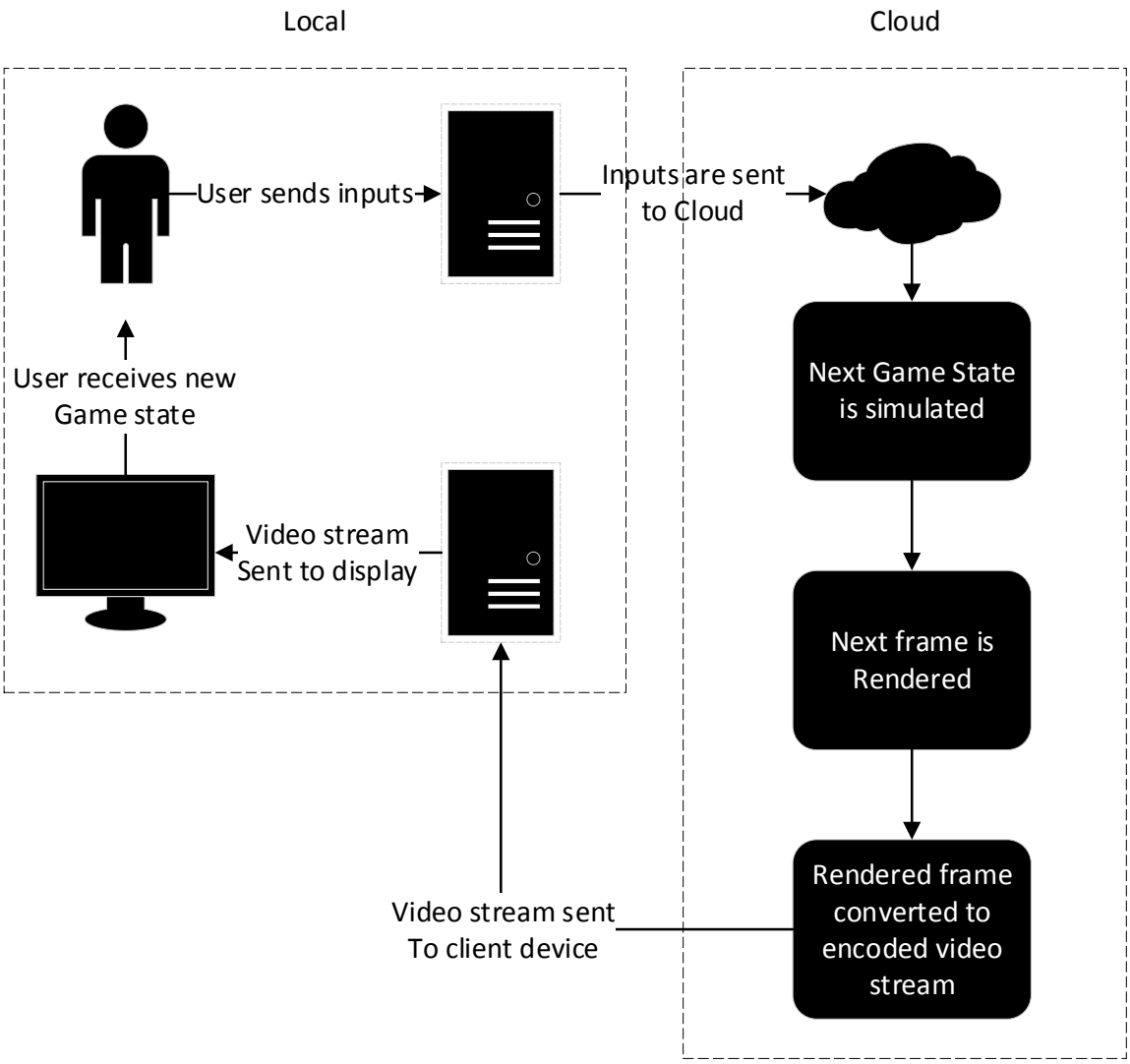

Figure 1 3 Example of contributions to latency in Cloud gaming

The point at which most of the additional latency is added is in the transfer between Local and Cloud components. Specifically, when inputs are sent through a network from a Local client to the Cloud process and when the video stream is sent back through the network from the Cloud process to the Local client. A small additional piece of overhead can be found in the encoding of the video stream.

While Cloud gaming provides several useful features, such as avoiding installation time and allowing users to play on any client, the weaknesses in Cloud gaming, such as increased latency and high network traffic, make Cloud gaming a less attractive option to many users. In the next section, I will explore the proposed solution of Hybrid gaming which aims to use components of both Cloud and Local gaming to improve user experience.

1.4 Hybrid Gaming

The purpose of this thesis is to examine methods to reduce or bypass the delay inherent in the modern Cloud gaming system's use of networks by dividing responsibilities between the Cloud and the Local device. We believe that this approach can circumvent the extra delay by having critical processes handled locally, and non-critical processes handled in the Cloud. This does limit the ability for the Hybrid gaming method to be used on thin clients, as the user would require a thick client to leverage the Hybrid Cloud system. Given that modern 'thin' clients have significantly greater local processing abilities, this is less of a barrier for general users.

A Hybrid system would in essence be a Local and Cloud system working together to share the load, with the time-sensitive critical process working locally and non-critical processes working in the Cloud. By sharing these responsibilities, the Hybrid system attempts to gain the advantages of both the Local and Cloud paradigms while minimizing their weaknesses. Figure 1.4 illustrates the proposed simulation and its components.

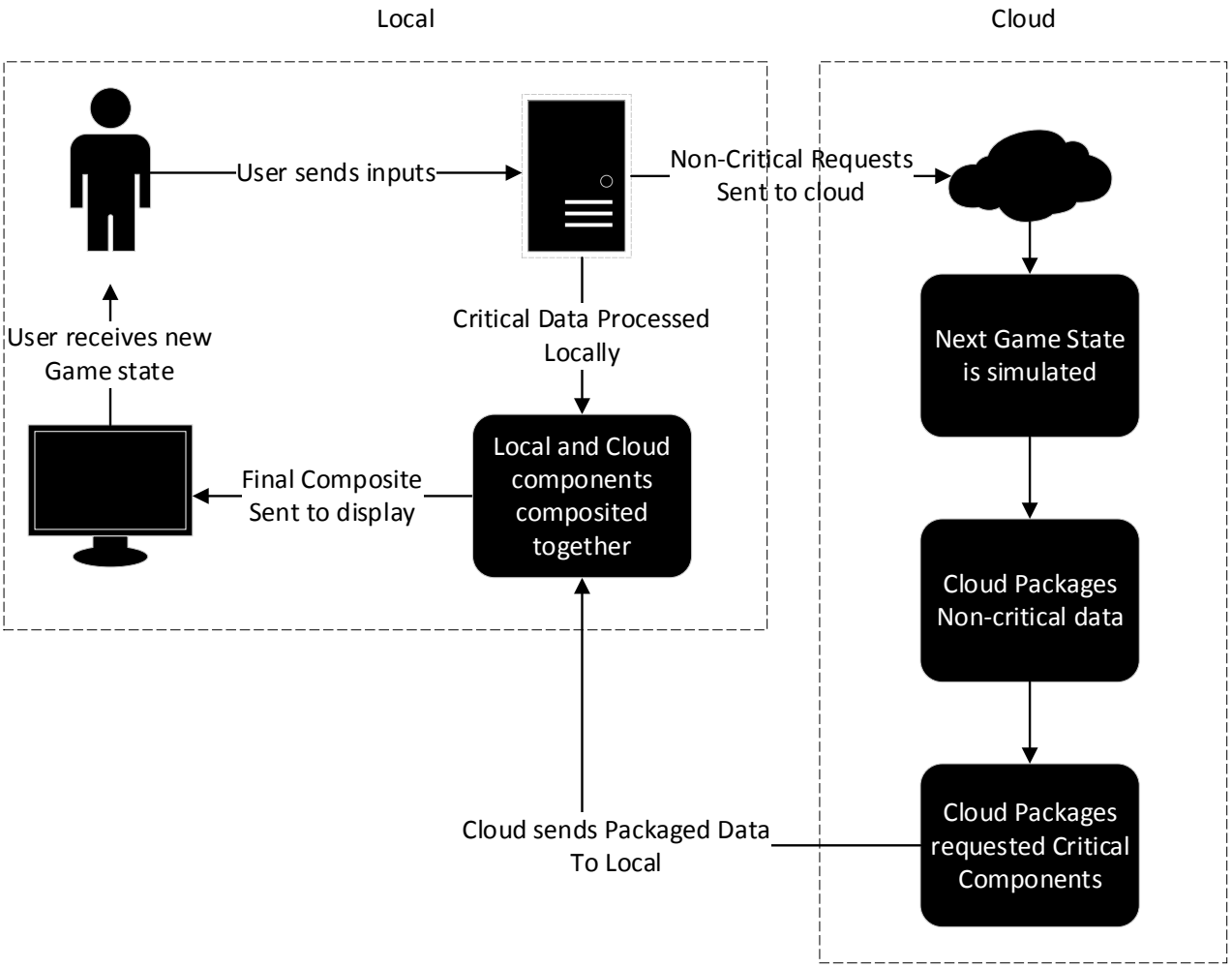

Figure 1.4 Hybrid Game System

1.5 Thesis Overview

Strategies used in video and music streaming fail to address latency issues when applied to Cloud gaming systems. Current techniques applied to video streaming do provide improvements in Cloud gaming. The primary performance metric, total system latency, is still controlled by network latency. This thesis proposes a method of sharing the processing requirements of a video game between the Cloud and Local client systems to leverage the increasing average power of client devices from personal computers to smartphones.

The Hybrid simulation proposed by this thesis suggests that we perform time-sensitive processing on the Local client leaving non-time-sensitive processes to the Cloud. With this approach, a degree of latency, the time to send and receive a message from the Cloud, can be reduced from the overall system latency. Critical files and information can be downloaded to the client device, with the result being that all of the core pieces of a game experience can be stored and treated as if the game is run locally. This leaves the less sensitive game assets and features being calculated and processed on the Cloud. Our approach aims to provide as close an experience to a Local gaming experience as possible while also preserving some of the reduced hardware requirements achieved by a Cloud system.

In this thesis, I will be comparing the results of three simulations. The first is a Local gaming simulation, the second a Cloud gaming simulation, and the third, and the final, simulation being the proposed Hybrid gaming simulation. By comparing the

results of these three simulations I aim to test if Hybrid gaming is a viable video gaming solution.

Chapter 2 will discuss the specific details of a general media stream, and the technologies that make them possible. This will cover the common issues, and differences, between current online video games and Cloud gaming. Finally, Chapter 2 will examine the state of the industry to develop solutions for current models of Cloud gaming.

Chapter 2

Background

2.1 Introduction

In this chapter, we will cover the importance and effect of streaming on general media consumers and the techniques used in both multimedia streaming and online gaming to provide high-quality user experiences. Examining the effects streaming has had on general media consumption allows us to see the effect high-quality Cloud Gaming could have on the medium, and the benefits possible through leveraging Cloud technologies. To properly understand how to improve Cloud Gaming we have to understand the techniques being used in multimedia streaming to provide a fluid experience, as well as the fundamental differences between multimedia streaming and Cloud Gaming. We will also analyze common techniques in online video game experiences for potential solutions to the latency in Cloud Gaming services. Finally, we also look at some techniques proposed specifically for increasing Cloud Gaming performance.

2.2 What is Streaming

Streaming has dramatically changed the way users consume content on the Internet [23]. A user is no longer required to obtain a physical copy or install a file on their device. Instead, streaming allows users access to almost all media content, rapidly and reliably. This discussion on streaming is necessary to our understanding of how Cloud Gaming builds on this technology. One of the effects that streaming has had is an increase in sales of games being streamed by players [24]. Boosted visibility allows for viewers to watch gameplay in an uncontrolled Livestream environment, permitting consumers to access hands-on information on gameplay, without direct influence from the marketing efforts of either the game's studio or publisher. This gives a more accurate representation of the game and allows for a more informed purchasing decision from the perspective of stream viewers. Also, when video games, with smaller player communities, are acknowledged and played up by a popular streamer, the game's sales have a chance of increasing due to its expanded visibility [24]. One example of this is the social deduction game *Among Us* (InnerSloth LLC, 2018), which saw a spike in popularity in the year 2020 due to multiple streamers discovering the game during that year [25]. Some games, such as *The Jackbox Party Pack* (Jackbox Games, 2014), have specifically designed features to create a better streaming experience such as viewer participation.

Streaming is the process of packaging, sending, receiving and presenting media that is stored on a remote service. Cloud streaming services refer to steaming services

made available to users over the Internet. These services allow for a constant, on-demand flow of data from Cloud storage to the user, allowing them to consume the content when they desire instead of only at a predetermined time. What makes Cloud storage fundamentally different from traditional media acquisition over a network is that Cloud streaming, such as video streaming, allows for content to be viewed and consumed without the entire file being present. "Video streaming addresses the problem of transferring video data as a continuous stream. With streaming, the end-user can start displaying the video data or multimedia data before the entire file has been transmitted" [26]. This allows users to access content without having to install or download complete files.

Streamed content is stored in a Cloud service where a user can nearly instantly request their desired content. This is then packaged into a useable format determined by the style of request, and the content is sent along a network to the requester's client. Most streaming services are designed to work with either thin or thick clients so the Cloud service can serve as many users as possible. The content is split into packets of data by the Cloud server and these packets are reassembled into the whole media file on the user's client [21] [27]. Packets of information are typically buffered ahead of time, allowing for breaks in the stream of packets to be hidden from the user. Figure 2.1 shows an example of how a buffer can be used to improve the playback of streamed media files.

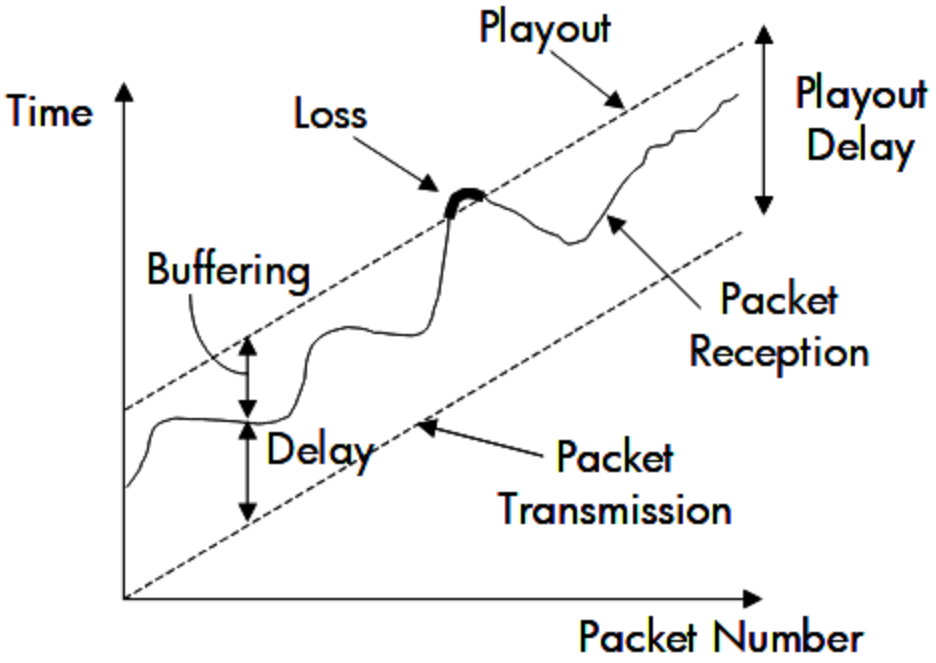

Figure 2.1 Effect of playout buffer on reducing the number of late packets [21]

This allows the client to continuously present the streaming media to the user, as long as a buffer of information is present [21] [27]. These systems do not require the full media file to be present, but instead aim to have just enough of a media file processing the data which has already arrived, giving the client enough time to receive the next section of the file. This is why many streaming services have a buffer period before the media file begins to play on the Local client. The system buffers enough data to be able to package, send, reassemble, and present the remainder of the file without pauses or interruptions in the user experience.

The delay at the beginning of a media stream creates a buffer that allows for hiding the download of the next part of a media file. The size of the buffer required

depends on multiple factors, such as file size download speed, bandwidth, and the variability of these factors. Since download speeds are not always consistent an extra amount of room is factored into the buffer. If the buffer runs out before the next piece of data can be provided then the media playback either needs to pause and wait for the missing files to arrive, or skip to the next currently available section of media.

As an example, we can look at the time interval between frames to be Δ. "Δ is 33 ms for 30 frames / s video... Each frame must be delivered and decoded by its playback time, therefore the sequence of frames has an associated sequence of deliver/ decode/ display deadlines:

- Frame N must be delivered and decoded by time T_N
- Frame N + 1 Must be delivered and decoded by time $T_N + \Delta$
- Frame N + 2 Must be delivered and decoded by time $T_N + 2\Delta$
- Etc."

[21]

By adding delay to the beginning of a video or any media stream we can increase the time Δ to allow for more space for frames to arrive on time to be used by the client. An increased delay also allows for frames to arrive out of order, meaning frames that arrive ahead of time are still displayed on time giving the stream more overall time to work with. Many current streaming services also store an already encoded version of

the media files to avoid having to encode on the fly and decrease the time to transmit a frame to the client device.

To improve efficiency video files on most video streaming services are pre-encoded and stored in multiple formats [27] [26]. This allows for videos to be stored on forward cache server locations which can then be sent to many users. Figure 2.2 shows an example of a forward cached media server.

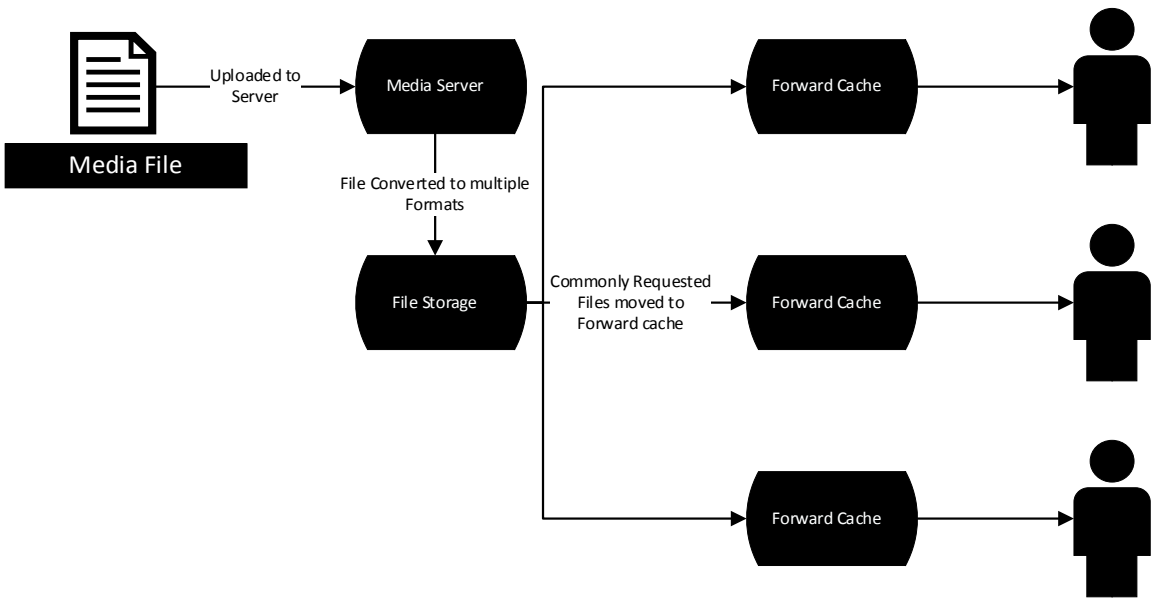

Figure 2.2 Example of forward caching

This allows for all the required files at multiple levels of quality to be sent when requested rather than needing to encode on the fly. The level of detail in a file sent

along to the user has been selected automatically to best suit the connection speed and quality [28].

A popular streaming service is *YouTube*, (Alphabet Inc., 2005) which provides a streaming video service, but is also commonly used for podcasts and music streaming. The primary advantage of streaming is convenience since it allows a user to access a wide variety of content on nearly any client without the need for the media to be stored locally. As an example, a user who accesses music through a streaming service can get access to their full music library on an Internet connection, without having to store large amounts of music files on a client device.

A disadvantage to streaming services is the quality of the content because increasing the quality causes the file size to increase, which makes it more difficult to send the file through a Cloud service. One example of this can be seen on *YouTube*, where video files are served in various resolutions from 240p, which contains 426x240, 102240 total pixels, to 2160p which contains 3840x2160 pixels, 8294400 total pixels or approximately 81 times more pixels. This results in 81 times more data needing to be moved through the network, assuming there is no compression to properly serve the video to a client. *YouTube* provides a range of video resolutions to allow a user to adjust the video they are accessing to either improve the speed at which the content is served or allow for the user to trade speed and convenience for improved image quality.

In this section we discussed the common practical techniques for improving the performance of multimedia streams, particularly through ahead of time buffering,

encoding and forward caching. These techniques work well in typical non-interactive media streams [21]. In the next section, we will discuss Cloud Gaming, how it is different from other forms of multimedia streaming, and how effective techniques for latency reduction change.

2.3 Cloud Gaming

2.3.1 What is Cloud Gaming

In this discussion of Cloud Gaming, we are referring to streaming services involving users playing video games as Cloud Gaming in place of the term video game streaming. As of this writing, the popular lexicon uses the phrase video game streaming to imply the act of watching a presenter play video games on streaming services such as *Twitch*.

Cloud Gaming has been an interest of the gaming industry for many reasons, particularly as a means of lowering the barrier to entry for new users and expanding the user base of video games. This is achieved by removing the requirement to download and/or install a full game before playing [18].

Currently, on modern video game consoles, such as the PlayStation 4 (PS4) (Sony Group Corp., 2013) or the Xbox One (Microsoft Corp., 2013), many individual video games cost between sixty to eighty dollars and require lengthy download and installation times [29] [30]. This means that simply trying a new game is costly in terms of both time and money. Cloud Gaming however allows for a game to be played without

downloading or installing the content, saving the user both time and storage space on their device of choice. Additionally, the cost to use these services commonly occurs through a monthly subscription allowing access to multiple games, which can be more cost-effective than purchasing a single title. While the exact pricing changes from service to service, Cloud Gaming services provide a way to potentially reduce cost and lower barriers for both new and enfranchised gamers.

While Cloud Gaming has great potential, previous attempts have failed to address specific game-related concerns in streaming technology. Gaming is one of the most computationally expensive uses of modern consumer-grade computational hardware and places many constraints on the performance of that hardware to create a good user experience. Most importantly, video games are greatly affected by latency and responsiveness, two issues current Cloud Gaming services fail to address adequately. The latency that Cloud Gaming services introduce can be 85% to 800% higher than Local executions [31]. Video game experiences are such that there is constant user input which creates a higher focus on responsiveness. While listening to a song through a streaming service only requires the selection or play, pause and skip for the vast majority of user engagement, video games typically require large amounts of complex inputs from the user at all times. Even in the best-case scenario network latency can introduce noticeable dips in in-game responsiveness which negatively impacts user experience.

To illustrate this disadvantage, when a video game proposes a time-reliant challenge to the user, such as dodging an enemy's attack or performing an input at the

correct time, there is a very small window of opportunity for the player to act. The additional latency imposed by streaming technologies can either greatly reduce the window in which a user has to perform the correct input, or make performing the input within the required timing window impossible. If Cloud Gaming cannot provide a good user experience, then it loses the ability to bring in new users even with the service's lower barriers to entry.

2.3.2 Cloud Gaming Advantages

Current Cloud Gaming services work as a Cloud service, where input is collected at the user's client and then sent along a connection to the Cloud where the game is running. The input is processed and a video stream, similar to *Netflix*, (Netflix Inc., 1997) is sent back to the user's client for display. This method has both advantages and disadvantages which we will analyze and discuss.

The largest advantage to Cloud Gaming is the low computational cost which allows for low power and thin clients to be utilized. The collection of user input and display of the video stream are comparatively light parts of the processing and are easily within the scope of common thin client computational power. Another advantage is that this model does not induce any additional development time onto the game's creation, as the Cloud runs the game on its system and creates and encodes a video stream that is sent along to the user. This means no additional development time is required for the creation of the game because the environment that the game will run on the Cloud is similar enough to the environment that the game would be processed on a user's

hardware. Therefore, no changes are needed to get the game running on Cloud hardware.

2.3.3 Cloud Gaming Disadvantages

The largest disadvantage of Cloud gaming is that all inputs have to be processed in the Cloud adding a not-insignificant amount of delay between an input being pressed and that input being processed by the system. This adds a large amount of operating time to the Cloud gaming service and is the main source of the increased latency over a locally run process.

While a focus on methods that allow for the use of thin client machines does decrease the barriers to entry further, as potential users do not need to purchase additional hardware outside of currently owned devices, it does limit the potential techniques that can be used to increase the efficiency of Cloud Gaming services. Some of the possible ways to improve a thin client-based system would include routing connections to other Cloud devices with a better connection, compression of the video stream to move the full stream faster and more efficiently. While these optimization techniques do reduce the overhead latency they still run into the issue of having to transmit inputs from the thin client to the Cloud for processing.

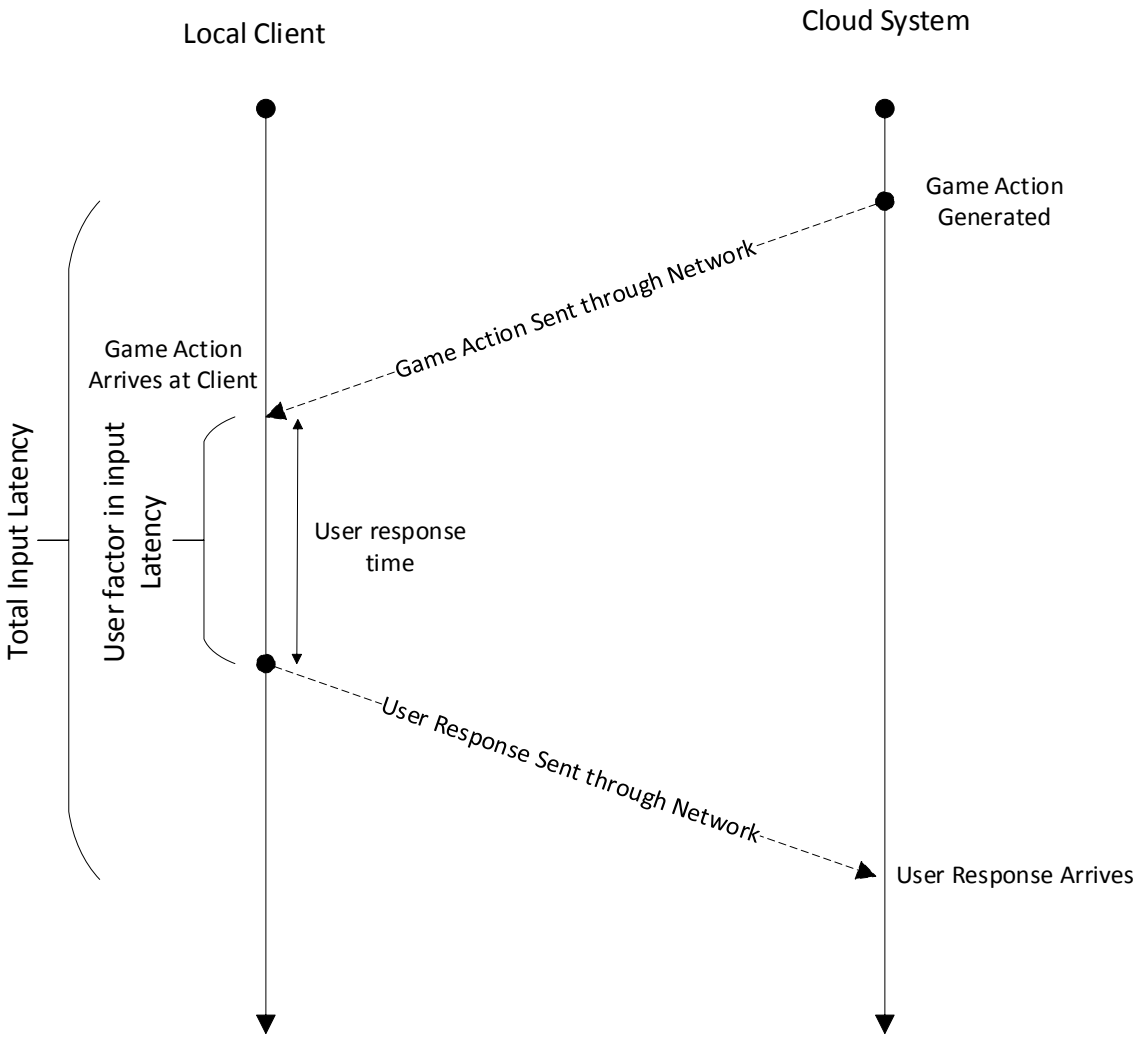

Figure 2.3 Example of typical message transfer through a Cloud gaming system

This creates a situation where all actions performed on a Cloud system is equal to the time to process input(P) + latency from client to server (L) + latency from server to client (S) while a Local system responds in only time P. Meaning the expected increase in input latency is L + S which is the largest unavoidable piece of latency. As an example of the effects of latency, we can look at the game *Street Fighter V* (SF5) (CAPCOM Co., Ltd., 2016). SF5 is a fighting game where combination attacks are performed by inputting specific sequences of commands with precise timing. One of

the smaller timing windows to perform a combination attack in three frames, and SF5 runs at sixty frames a second 3/60*1000 = 50ms. Therefore the timing window for a three-frame combination attack in SF5 is 50ms. Assuming a user has a very good connection of L + S = 20ms a player would have an additional 20ms delay on inputs. If the player uses a visual or sound cue to time their input for the combination attack that cue is sent to the player on a 10ms delay from the server then the input is sent to the server on a 10ms delay shrinking the timing window from 50ms to 30ms or changing the timing from a 3 frame window to a 2 frame window. This means that each timing window is more difficult to perform.

This problem is also seen when reacting to non-player character (NPC) actions. If a user is playing SF5 against an NPC opponent, reacting to visual and sound cues from the NPCs actions is required to avoid taking damage or failing an in-game challenge. Moves in SF5 can have very low start-up frames, which are the “wind up” animation to an action essentially the warning to a player that an action is about to occur. The fastest actions in SF5 have four frames (66ms) of start-up. In fighting games, start-up frames refer to the frames where a piece of animation is starting but the move has not had an active effect. This gives players time to react to their opponent's choices. Again using our L = 10ms and S =10ms a four-frame start-up animation is created on frame one, the visuals arrive 10ms later still during frame one of the animation. The player reacts to the action of the client which is sent with a 10ms delay. This causes the reaction of the player to be delayed by one full frame, making reacting to a move with a four-frame start-up require the same reaction time as if the move had a three-frame start-up time.

The 10ms delay on server communications is also an unrealistically fast connection in most cases and even slowing L+S from 10ms each to 16ms each which removes 2 full frames of reaction times, at 64ms total delay a four-frame start-up becomes impossible to properly react to for the player.

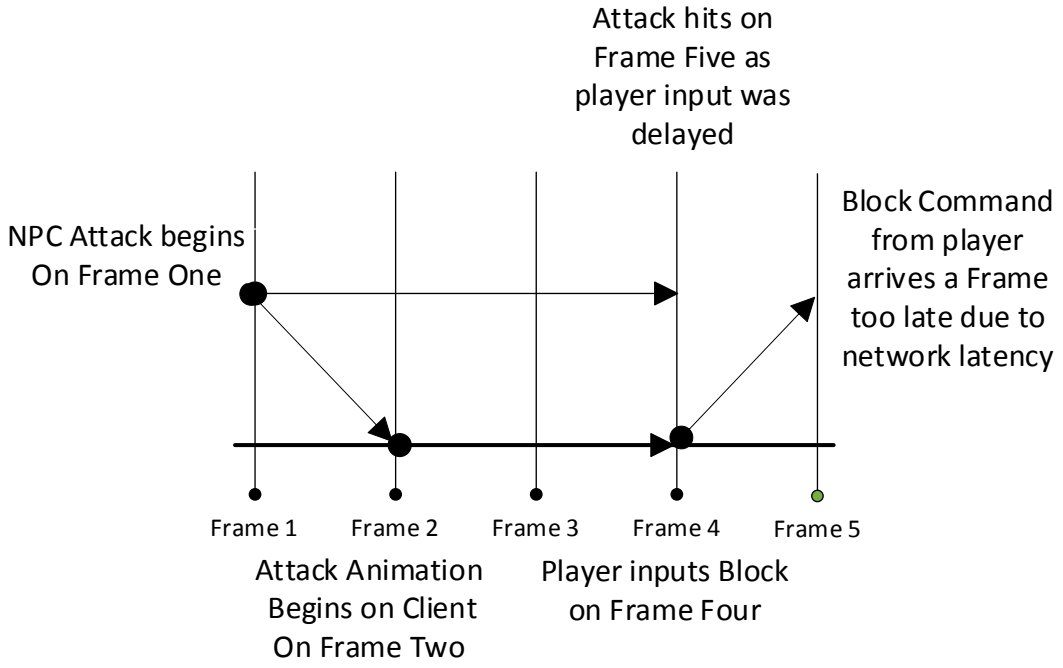

Figure 2.4 Example of effects of latency on user reactions

2.3.4 Effects of Latency in Online Gaming

Similar situations are commonly seen in online games such as *Counter-Strike: Global Offensive* (Valve Corporation, 201), *Valorant* (Riot Games Inc., 2020) or *Tom Clancy's Rainbow Six Siege* (Ubisoft Entertainment SA, 2015). Online games are when multiple users connect over a network, either to each other or to a central server to participate in a multi-player experience. This is in contrast to Cloud Gaming, which is a

singular user connecting to a Cloud service to play a single-player activity. Both services have to contend with the issues created by network latency, and some solutions from online gaming can be applied to Cloud Gaming. For this reason, a comparison of the issues caused by network latency in online gaming applies to Cloud Gaming and will be illustrated through the example of Peeker's Advantage.

Online First Person Shooters commonly run into issues with network connections, one of the most common is known as Peeker's Advantage. Peeker's Advantage refers to a reaction time advantage gained from network latency [32]. This occurs when the server managing the game knows all locations of all players and has sent that information along to all players, meaning a player's client knows the location of all players so it can properly render any information without needing to obtain additional data from the server. This can result in a situation where, for example, one player is watching a doorway while another player is approaching the doorway to pass through it. Since the locations of both players have been sent along to the client when the player approaching the doorway moves through it, their client already knows the location of the player watching the doorway and can render them without additional data needed from the server. The player that was watching the doorway instead needs to get an update from the server that a player has moved and then the client can render the new player location. This means the player walking through the doorway will be able to spot the player watching the door before the player watching the door can spot the player moving through it [33].

While there are several different ways to try and prevent or mitigate Peeker's Advantage, such as buffering incoming data, the best way to combat Peeker's Advantage in online games is faster connections [34]. It is an inherent part of client server-based online games and adding a client/server relation to a single-player game can add similar situations and delays to gameplay that is normally run completely locally and has no network latency-related delays or inconsistencies.

Another Issue faced particularly in peer-to-peer games that require the inputs of all players to generate the next game state are the delays caused by waiting until all user inputs have arrived before providing the next state. This issue is particularly common in Fighting games. Previously games would determine the amount of time required to allow for all player inputs to arrive and add a delay to synchronize these inputs to arrive at the same time for each player. However this can cause play inputs to feel unresponsive or cause the game to pause if inputs are dropped. To address this challenge Fighting games have recently embraced the use of the Good Game Peace Out (GGPO) library also known as "Rollback netcode" to use predictive techniques to allow for a more responsive game feel and provide a more smooth experience for players [35].

Rollback works by predicting the input of a user when it has not arrived in time to process the next game state, most commonly through dead reckoning techniques, discussed in section 2.5.3. In the case where a predicted input is different than the input that arrives the game will in the background rewind to the time where inputs became desynced, then replay the game with the corrected inputs to determine if those inputs

end up with a new game state or if the predicted game state is correct. If the predicted game state and reprocessed game state do not match the reprocessed game state is sent to the players as the enforced correct game state. While Rollback does allow for predictive processing to be implemented in online games and could be extended to Cloud Gaming solutions Rollback and GGPO requires significant development effort to implement requiring specific design decisions to allow for it to be integrated into a game or retrofitted into games. One example of specific requirements for implementing rollback is disconnecting rendering and game state processing, as the game state needs to able to be processed quickly in the background without effecting the rendered scene, as well as being able to remove potentially lingering graphical effects that are not part of a reprocessed game state.

2.4 Existing Cloud Gaming Services

Cloud Gaming requires a thin client, Internet connection and a Cloud system. In current Cloud Gaming systems, this thin client can be anything from a phone to a high-powered personal computer. Powerful Internet connections have become the main bottleneck for Cloud Gaming, with the newest systems requiring not only high download speeds but also high bit rates for a Cloud Gaming experience to live up to consumer expectations. Systems such as *Google Stadia* (Alphabet Inc., 2019) have a recommended 10Mbs connection speed [36] as well as a high amount of data usage per hour, with the minimum recommended settings using 4.5GB per hour. Many large games now have

playtimes in the fifty hours or above range meaning to play a full game on *Google Stadia*'s lowest settings could use 225GB or more.

2.4.1 OnLive

OnLive (Lauder Partners, 2009) was one of the earliest Cloud Gaming services beginning its service in 2009 [37]. While OnLive was an interesting first look at what Cloud Gaming could do, the service was met with middling reviews [38]. OnLive was impressive and innovative for the time, but its latency issues caused it to be seen as a less viable way to play video games than the competing options on the market, specifically home consoles and personal computers (PC). OnLive did allow for gaming on completely thin clients, such as smartphones, but unless the user had access to a fast network connection, the performance would flounder making it near impossible to consider using the system on slow connections. OnLive service ended in 2015 when Sony, owner of the current Cloud Gaming service PlayStation Now (Sony Group Corp., 2014) purchased OnLive's patents [1].

2.4.2 PlayStation Now (PS Now)

PS Now is Sony's subscription-based Cloud Gaming service, aimed specifically at streaming games that are normally exclusive to PlayStation consoles, allowing these games to be streamed on either a PC or PlayStation without downloads or installation [39]. While PS Now has had some success as a Cloud Gaming solution, it typically does not offer newer games. Instead, the service focuses on hosting older titles that users either want to revisit, but no longer have a copy on the self, or experience for the first

time at a lower cost. PS Now also has issues with internet speeds of less than 10 Mbs, especially for more demanding games that require an even faster connection to get a good quality play session. This means that even a slight connection drop will end the game session which can cause problems [40]. PS Now does provide the benefit of a large selection of games to play but does continue to suffer from imprecise controls and slight interruptions dropping the entire connection ending a session.

2.4.3 Google Stadia

Google Stadia was a recent addition to the Cloud Gaming Space and Google's entrance into the Cloud Gaming world. Google Stadia approaches the Cloud Gaming pricing model uniquely from PS Now or OnLive. Instead of providing a subscription service that allows the user to play all the games on the service, games on Stadia are purchased individually, typically for the same price as purchasing a copy of the game you can download and play locally. This places Google Stadia in an odd position for Cloud Gaming as it features the same issues that PS Now and OnLive experienced, such as reduced graphical quality and the occasional input lag or disconnect, without the benefit of a reduced price that their subscription services offered. This means that there are very few reasons for a player to purchase video games through Google Stadia, when they can buy the same title, at a consistently higher quality, for the same price elsewhere. Google closed their internal game development studios for Google Stadia, which put doubt on the continued support of the service [41]. Those doubts were confirmed when Google Stadia was closed down in January 2023 [42].

2.4.4 Xbox Cloud Gaming

Microsoft has also recently moved into the Cloud ace with their Xbox Cloud Gaming beta [43]. Which is an extension of Microsoft's Project xCloud and an addition to Microsoft's existing Xbox Game Pass product. While Microsoft's Cloud is remains similar to previous Cloud gaming solutions, the most interesting aspect of Microsoft's venture is the use of a subscription model that is more similar to services such as Spotify. The use of a monthly subscription, cost instead of a monetization plan like what was seen with Stadia where users had to pay full price for each game, could result in Xbox Cloud Gaming being better received by users. This could also lead to users being more receptive of performance issues because of the lower cost. Currently in a beta release, the list of games available for Xbox Cloud Gaming is limited but Microsoft does find themselves well positioned because of an already existing data center infrastructure that can be additionally used for the Cloud service.

2.4.5 NVIDIA GeForce NOW

NVIDIA has also moved into the Cloud Gaming space recently with the GeForce NOW service [44]. This is another subscription based service with a much larger selection of games then the current Xbox Cloud Pass service. While the GeForce NOW system provides users access to a large set of games, rather than the need to purchase each individually, they also permit free access to their game library for short sessions. These free sessions have a one-hour time limit and permit users to try the Cloud service. Time limits also apply to the paid services, with their Priority Plan limiting sessions to

six-hours and the Ultimate Plan limiting sessions to eight-hours. These time limits could reduce user interest in the system; a six to eight hour session sounds like a long time, but it is not uncommon for a game session to spend longer than that as a resulting in interruptions to a Cloud session. These interruptions could have large negative impacts on the user experience.

2.5 Previous Research

2.5.1 Video Compression

Cloud Gaming works by gathering inputs on a Local client and then sending them across a network to the Cloud system where the inputs are processed. The next game state is then created and sent back to the client and the Cloud system delivers the next game state through a video stream. The main bottleneck is the network connections as it is involved in both sending inputs and receiving a video stream from the Cloud system. A particular focus of research on improving Cloud gaming systems has been on the video stream portion. The video stream contributes in terms of input response latency, the data transmission cost of using the Cloud gaming system as well as being useful in other media streams outside of Cloud gaming.

The larger the video stream files per frame, the longer it takes for each frame of information to arrive, which increases the monetary cost of maintaining that video stream. One of the major costs of running a streaming service is transmitting media through a network. Better compression technologies reduce the amount of data being transmitted which reduces the cost and improves the performance of media streams.

These compression techniques can also be used in other video streaming systems allowing for research in this area to be used more broadly.

Most video and image compression comes from reducing similar colours among the pixels in a video frame, omitting parts of the new frame that do not change from the previous frame, and removing detail from less noticeable pieces such as backgrounds [45] [46]. For example, consider an image of something that is mostly one colour, such as a brick wall, we could have the pixels representing that wall be reduced. Instead of potentially hundreds or thousands of pixels per brick in an image, at high levels of compression, a brick can be represented by a single pixel. While this would remove any texture detail from the bricks making up this wall, it would also greatly reduce the amount of data that needs to be transferred per frame of video, reducing costs and transfer time. Compression can also be done across frames, such as if an element in the next frame is identical to an element in the previous frame, no updates to that area are needed and no new data needs to be transferred.

Three of the most common encoding schemes for video streams are H.265/MPEG-HEVC, H.264/MPEG-AVC and Google's VP9 [47]. Of Primary concern for these schemes has been the efficiency of encoding a frame of gameplay into a video stream. For a Cloud Gaming service to achieve a good user experience it would have to provide a minimum of thirty frames per second video stream of the gameplay to compete with the performance from home consoles. This means that those thirty frames need to be encoded, transferred and decoded fast enough for the end-user to

not have a noticeable increase in latency. Any inefficiencies in the encoding scheme could add an overhead of latency per frame encoded.

While video streaming services such as *YouTube* and *Netflix* have the advantage of having the complete video file ready at the time the stream is accessed, Cloud Gaming streams need to build this video file on the fly. Past work has also focused on video encoding because of the cross-discipline potential for these encoder improvements, as they can be applied to any web-based video rather than just to Cloud Gaming. Improvements in encoding reduce latency by improving video compression, lessening the load on bandwidth and improving throughput, meaning more frames of information can be transferred through the Cloud gaming system per time slice.

Video encoding uses movement vectors to store movement between frames rather than redrawing an entire frame each time a new one arrives. This technique is known as interframe compression [48]. This means that if a person is walking through a frame in front of a wall of solid colour, the video is encoded by using the movement vectors of the person through the frame while the solid background does not change. Videos with high amounts of movement or changes are harder to encode and longer to transfer. For example, the appearance of confetti in a video. Confetti, and other small moving objects in a frame, can quickly overload an encoder as it is unable to keep up with all the movement between frames [49]. Within Cloud Gaming, this issue can be found in new complex particle systems and the high amount of activity that happens in modern action games. Whether separately or combined, these factors result in encoders having a difficult time keeping up with Cloud Gaming streams. So while the

improvements in encoding and compression seen in current systems do decrease latency, they can, in some cases, degrade video quality to the point of having a detrimental impact on user experience. With new improvements on the horizon in home gaming hardware, as with the PlayStation 5 (Sony Group Corp., 2020) and Xbox Series X (Microsoft Corp., 2020) consoles, there is an expectation for particle systems and effects to get more complex, thus further taxing current encoding schemes more than ever.

In Are All Games Equally Cloud-Gaming-Friendly? [50] Lee et al. explore the challenges created by Cloud Gaming and the effects of video encoding. They find that a game's playability, which they defined as “friendliness”, on Cloud Gaming setups could be modelled by comparing the screen dynamics to the input rate required by the game. Lee et al. found that many techniques used in “‘traditional’ network games, which render graphics on a players' computers, [the] impact on network games can be somewhat mitigated by using delay compensations techniques, such as dead reckoning. However, such techniques cannot be applied to Cloud games because their functions all require game state information, which is not available in Cloud gaming clients." [50]. This implies that Cloud Games cannot use all techniques normally found in online gaming, and are unable to use techniques used in video streaming, meaning improving the performance of Cloud Gaming either requires new techniques or a new architecture.

Lee et al.’s approach to determining the friendliness of a game in a Cloud Gaming system was by having subjects play nine different games at five different latency levels. Following this, the authors used their findings to create an approximate on how

friendly a game will be to a Cloud Gaming environment. The genres of games tested were First Person Shooters (FPS), role-playing games (RPG) and action games, with the emotional responses of the players, which were measured at various latencies. Lee et al. found that there were negative emotions at all levels of increased latency, with different games having different baselines of when the player began to experience these negative emotions, and that the effect of increased latency was also varied for each game.

Using the results from these experiments the authors developed a computation to measure how friendly a game will be to Cloud Gaming, which they called *command heaviness*.

$$command\ heaviness = \frac{screen\ dynamics}{input\ rate}$$

Where *screen dynamics* is the amount of movement required between each frame and *input rate* is the rate of inputs required by the game. "If a game's commands are mostly "heavy," i.e., associated with long and large amounts of screen changes, such as power attacks in SM5 [Sangoku Musou 5], the game will be less susceptible to latency because the timing of such attack commands is not so critical" [50]. The basic idea is that if an input has a large effect on the game state, and is not very input sensitive, the game will play well with Cloud systems.

Turn-based strategy games would be only lightly affected by Cloud Gaming systems as there is no time sensitivity on the inputs and the effects these inputs have on gameplay are large. Conversely, FPS games that require quick responses that are generally considered "light" commands, would be greatly affected by Cloud Gaming. The

authors used *command heaviness* as a predictor of real-time strictness (or latency sensitivity) and found a correlation between command heaviness and real-time strictness in games. While this research is based on a small set of games, and the classification of games into genres is not perfect, it does provide evidence that *command heaviness* could be a good predictor of how well a game performs on a Cloud service.

As games improve graphically and aim for even more detailed and realistic scenes, the limitations imposed by video encoding will become more pronounced. This could result in Cloud Gaming having issues going forward and requiring the use of less compressed video to keep up with the increases in graphical quality of games. The implication is that Cloud Gaming may provide a lower quality visual performance, as games trend towards more graphically complex and chaotic environments which require faster and more reliable network connections to provide an enjoyable experience to users. This is especially noticeable with particle-effect heavy games and fast-paced games, as they create visuals similar to confetti. This can refer to large amounts of small objects and lights moving quickly, which is difficult for current encoding solutions to properly preserve visual quality.

These current trends regarding in-game graphics also shift more actions towards *light* inputs, making the game more sensitive to network delays and reducing visual quality, both of which have negative effects on user experience. Improvements in video encoding would help to balance the effects caused by a shift towards more confetti-like visuals and *light action* inputs, but the main limiting factor will still be network

connections and bandwidth. With video encoding being primarily developed for non-interactive video streaming such as television and movies, improvements specific to the issues found in Cloud Gaming remain to be solved. Better encoding would improve the user experience in Cloud Gaming but at this time, no solution can remove the quality and latency losses caused by a network connection.

2.5.2 Graphics Culling

Graphics culling is a technique that removes graphical details not visible in the current scene. One of the most common techniques is occlusion culling, which aims to hide polygons that are behind other polygons and scene objects to reduce the render load of a scene [51]. By determining if a polygon or set of polygons are visible in the current scene, effects, objects, and lighting can be removed from the scene to greatly reduce the load on the hardware rendering. This is one of the core techniques that allow large worlds to be feasible in technically advanced video games. Something one can observe in older games is that they tend to be made up of hallways and rooms, taking place in smaller areas which greatly limit sightlines into the distance. This was done so a game could easily be divided into small sections that only need to be rendered in small numbers [52].

Consider the 2002 game *Metroid Prime* (Retro Studios, 2002). This game was made on a series of rooms of various sizes, with distinct doors between each one [53].

While the game world is rather large, very few rooms are rendered at any time with load triggers that a player needs to activate before the game loads the next room. When a player activates the load trigger to the next area, the game unloads the graphical information for the rooms the player is not in, then loads the graphical information for the room the player is trying to access. Once the room has been loaded, the door opens and allows the play to move into the new area. This keeps the load on the system manageable as the player explores a large game world. Though this system hides loading screens in the opening of doors, this does lead to small breaks in pacing and action on occasion and prevents scenarios where a player can be in one room and see into another room. With changes in video games to having more open areas, the player now has larger open sightlines. Different techniques have become popular to allow for games to move away from the load one room at a time method; a common technique being *occlusion culling*.

Most research into improving Cloud Gaming services has focused on improving the performance of the Cloud system processing the game, or reducing the size of the video stream through compression or visual culling [54]. Hemmati et al. propose an approach to cull unimportant or unneeded details from a computer graphic environment, specifically to improve render times. This has the added benefit of allowing common video compression schemes to further reduce the size of the video streams. Moreover, culling is a commonly used practice in video games as well, allowing for the removal of any unnecessary rendering [54]. Hemmati et al. propose a system that removes superfluous assets and models from a game to reduce the load on the

Cloud system, as well as to improve the compression on the video stream. While this does achieve some speed improvement, this approach has a greater impact on visual quality and stream bit rate. These practices reduce the costs of running a Cloud system but do not improve the service noticeably for the end-user. In some cases, the removal of scenery by the system detracts from the end-user experience. Culling assets may improve the performance of the Cloud system and the performance of video stream compression, but it does not improve the time delay between user input and a response from the server in a noticeable way [54].

Hemmati et al.'s approach found a 2-8% decrease in file size per encoded video frame. Their system does have some issues. The first is that it requires game developers to mark what elements in the game world are viable to be removed from a scene, and under what conditions that element could be removed. The authors propose removing elements that serve as obstacles to movement while the player is aiming, as these elements are irrelevant to the task at hand. If elements are being added and removed as the player's actions change, this could lead to pop-in which can be distracting to a user's experience. Pop-in is when an asset finishes loading and is added to a scene after the user has already absorbed the state of the game world. A common example is trees in the distance loading slower than closer assets, thus a bare area in the distance can suddenly be covered by a forest. Since the eye is drawn to changes and movement, the sudden addition of trees appearing on a landscape can distract the user. This type of situation further reminds the user that they are interacting with a video game, which in turn draws them out of the experience.

2.5.3 Remote Rendering

An important part of Cloud gaming is the rendering system which can deploy various solutions for reducing the latency experienced by the end user. This section will cover techniques used for Cloud gaming rendering systems. The first technique is the process of rendering the scene remotely and sending an image along the network to display on the client computer. This technique is common in Cloud technologies for its lower bandwidth requirements and ease of implementation, as well as allowing for a thin client [55]. While rendering scenes in this fashion allows thin clients to display the game using lower bandwidth than transmitting the assets required for a scene to be rendered locally, it is the least responsive of potential solutions. This is because scenes delivered to the thin client incur a network delay with responses also being delayed by the network. This solution does allow for an easy to implement, high quality image stream to be transmitted but adds a large amount of latency which could negatively impact the experience.

One technique for reducing this latency is Dead Reckoning [55]. Dead Reckoning is a technique used to predict the next game state depending on user input and the current game state. Dead Reckoning is also commonly used to reduce the appearance of latency in online games. An example of the use of Dead Reckoning would be in a Racing game; the speed and direction of the vehicle being piloted by the player is known by the game as well as the previous inputs from the player. From this data, the Dead Reckoning algorithm can predict the likely next game state. This allows a Cloud gaming system to create the next game state images ahead of time and transmit over the network in an

attempt to get ahead of the latency. This Solution can cause problems in cases where the Dead Reckoning algorithm is incorrect in its prediction; images are either displayed incorrectly or the system has to throw the images out and wait for the correct images to arrive after a delay.

Additionally, while Dead Reckoning is a powerful strategy for Cloud gaming as well as Online gaming, it works best in games with a lower amount of possible next game states. These include games where the players (and other game entities) next position is predictable, such as racing, rhythm and other on-rails type games. Games with a large number of potential next moves, such as character action or open world games are more difficult for Dead Reckoning techniques to accurately predict.

Pre-fetch strategies can also be used to reduce the latency of a Cloud gaming system. Similar to Dead Reckoning, Pre-fetch solutions predict the most likely set of next moves by a user and asks for all possible situations to be transmitted [55]. If the predictions are accurate, the pre-fetched scene can be displayed and the next scene requested. While this approach does lead to greatly reduced latency, it does have many potential draw backs. Video games are very intensive programs which render complex scenes. Having to render (and transmit) many potential scenes would be a large increase in processing time which can be costly in terms of the necessary hardware available on the Cloud to perform the required Pre-fetch rendering. This would also significantly increases the bandwidth required by the Cloud Gaming System as multiple image streams would have to be sent with only one of those streams being used. While Pre-fetch is a powerful technique for reducing latency, it also increases the cost to run a

Cloud gaming system significantly and could potentially preventing the use of lower quality network connections.

Chapter 3

Simulation Development

3.1 Introduction

This chapter provides an overview of the three file service models developed and compared for the core work of this thesis. This section begins with a discussion on how file requests and cache hit events are modelled. We designed three different models to represent the three use cases that we envisioned when running video games. The first model is the Local System Model, with the client-side running of a video game. The second model is the Cloud System Model which captures current Cloud video gaming solutions. The third, and final, model is our proposed Hybrid System Model which was created to test the performance of a system that uses local caching with the goal of gaining some performance advantages over the Local and Cloud-based Systems. We begin by discussing file request generation as it is a core component of all three models.

3.2 File Request Generation

File requests are generated from an exponential distribution to model the time between batches of file requests. We use two file request generators to better match the two common load cases for video games; the first creates frequent small requests to match minute-to-minute gameplay, and frequent loads of small assets; the second models the loading screens where gameplay is paused to complete a large file load of game assets. File requests from the load screen generator are issued in batches to

better capture the scenario of loading large sets of game information as observed with loading screens or level-streaming events [56]. For example, if a game needs to load an scene, e.g. a room, all the components of that scene must be loaded with each asset in a scene requiring one or more files with each file generating a request. The location of the file requested is generated at the same time the batch is generated. File requests are processed in order of time created with the earliest created file requests being completed first.

We assume three potential file sizes, with the file sizes being relative to one another and can be adjusted for different runs of the simulation. The sizes under consideration are small, (files measured in kilobytes), medium (files measured in megabytes) and large (files measured in gigabytes). The file sizes are relative to each other rather than being absolute values. Examples of small-sized file requests would be text files used for in-game descriptions. For a medium-sized file request, an example would be background textures. Finally, an example of a large file request would be level topography and three-dimensional models. However, it should be noted that file sizes are very game-dependent.

Each file size also has a probability that a request for a file of that size being generated: these are given by P(small), P(medium), and P(large). As there was not currently a large base of research and subsequent analysis of file sizes in video games, we used a cross-section of the games in a personal PC game library and used those as an estimate of the file size distribution in video games. See the Appendix for a full list of games used. We found that overwhelmingly, the majority of files in the collection of

video games analyzed were extremely small, about 4KB or less, these results are presented in Figure 3.1 below.

The selection of games chosen gives a representation of video games by having a mix of smaller independent games, midsized games as well as many games produced by large studios. Some files were removed from our data, specifically BLOB files. These were omitted because they are contiguous collections of binary data made up of multiple image, audio and other files [57]. BLOB files are the binary of multiple files connected together to make storing them and performing operations to move all of the files faster and more efficient, as well as reducing total size on disc for files by compressing multiple files together. Because we cannot profile file loads for the games in our data set, BLOB file loading cannot be profiled with the data we have. Since BLOB files are a collection of data they are not commonly loaded in their entirety and instead specific subsections of a BLOB are requested and loaded. Because we cannot profile how any of the games access and load these files they were all omitted from our data set as outliers. The removal of BLOB files does not negatively affect our dataset as our analysis included a wide variety of file sizes reaching 10+ GB in size as shown by the large tail in Figure 3.1 and continues to contain files in a wide variety of sizes and from a wide variety of games.

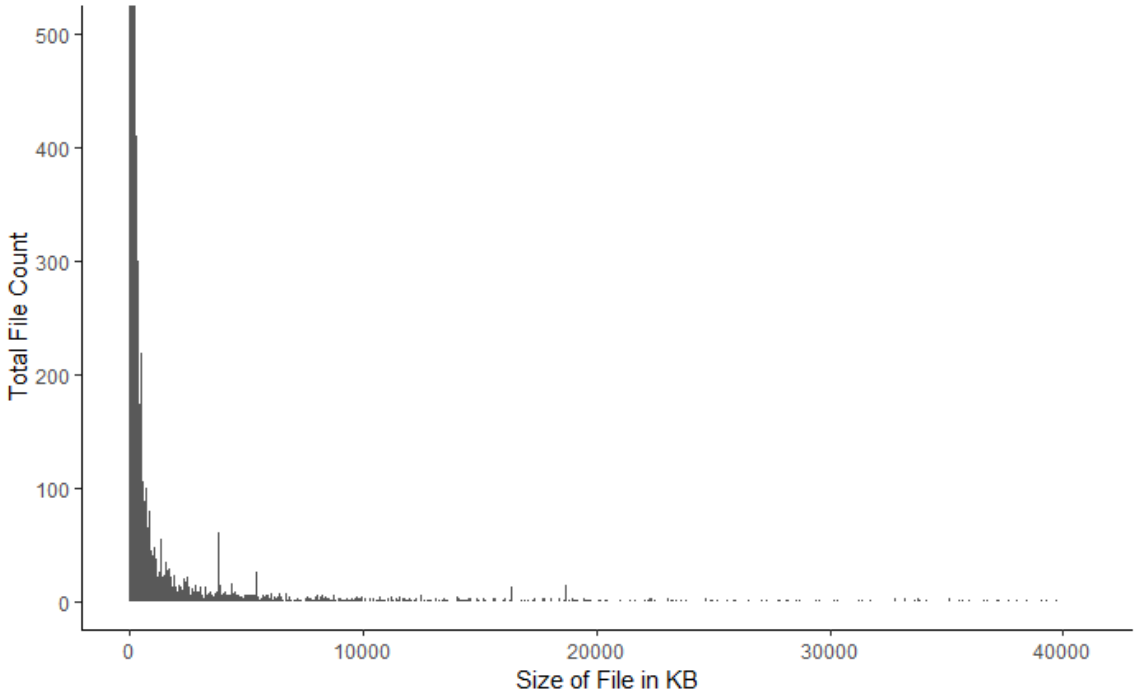

Figure 3.1 Histogram of File Size Distribution (bin width 5 KB) Y limited to 500 for ease of reading size 4KB has 25000 observations

Using this information, we determined that if a small file was being accessed by the models, it would likely be about 4KB in size because the vast majority of the files we examined were in this range. This information was used as the basis for designing the small file request event generator, which creates requests for small files with a short interarrival time. This still left out large loading events in video games, such as loading screens, where the data set included a large number of files that were in the MB and GB in size. To better capture this we decided to introduce a second separate file generator that would produce medium and large file requests (of the magnitude of MB and GB-sized files). For MB-sized files, we found the majority of files were in the 40MB to 300MB range (See Figure 3.2). For GB-sized files, we found that the majority of files were centred around 1GB ranging down to 0.5 GB and up to 3.0 GB (See Figure 3.3).

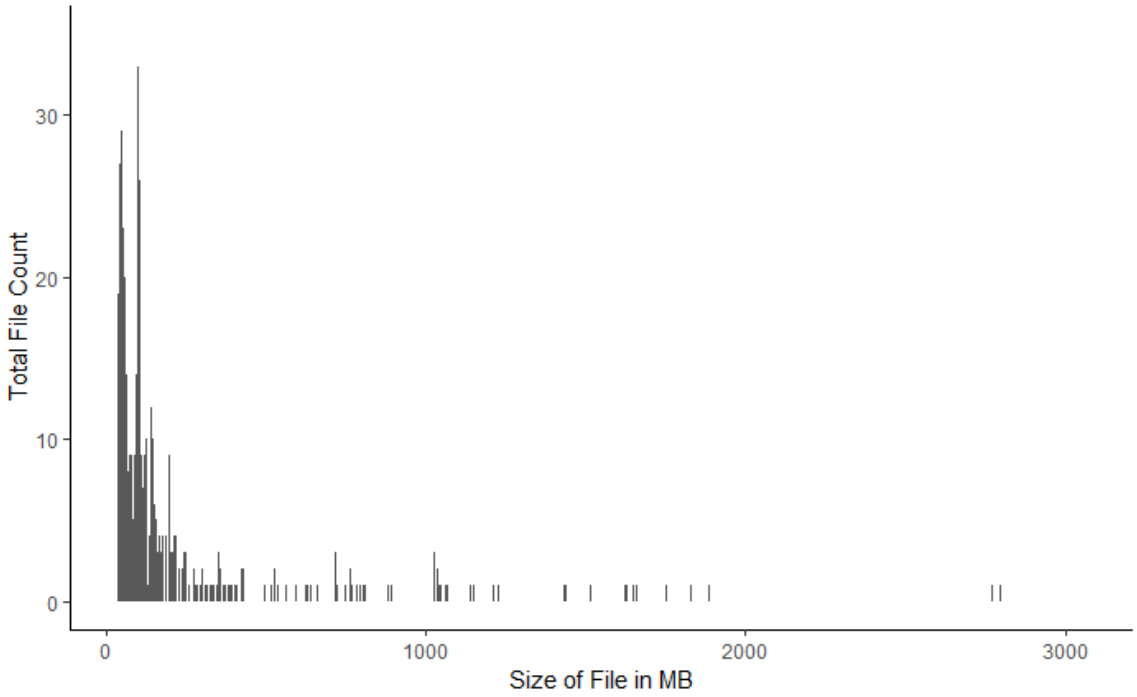

Figure 3.2 Analysis of MB Size Files (minimum size is 40MB) (bin width 1 MB)

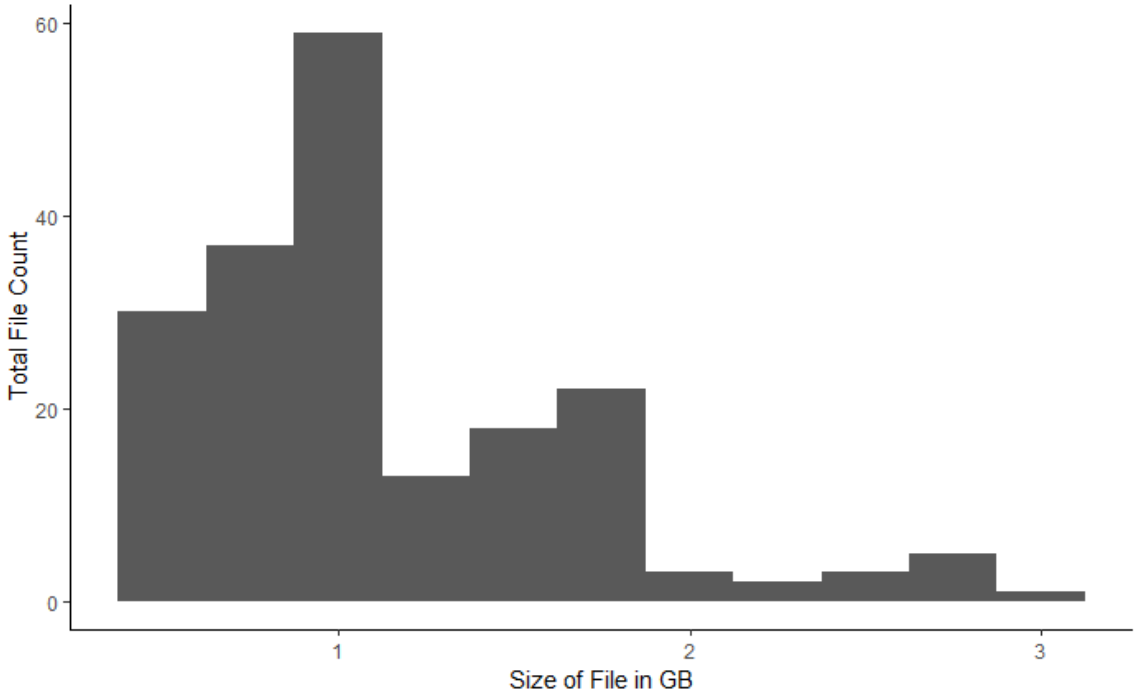

Figure 3.3 Analysis of GB Size Files (minimum size 0.5GB) (bin width 0.25 GB)

The analysis of the file size distribution proved challenging; while many distributions could account for the large number of very small files in the data set, it was difficult to find a distribution that could properly capture the long tail, along with the order of magnitude difference between the largest and smallest files. We eventually concluded that the closest match would be the Pareto distribution, particularly because of its ability to handle large tails in file sets. After performing fit testing with the data set in the programming language R, we found the Pareto distribution to be a good match for the data. When we applied the Pareto distribution to medium and large file requests, we found the majority of the results generated fell near the mean but also included the long tail found in video game file sizes.

The Pareto distribution models used in our research were based on the same data set that was used to generate Figures 3.1-3.3. Upon deciding on the Pareto as the distribution to use for our model we performed a parameter estimation for the Pareto distribution using our data set. This resulted in a Pareto distribution for the medium files with a mean centred at 40 MB and a shape of 0.9137, and a Pareto distribution for the Large file requests with a mean of 518 MB and a shape of 1.395. These distributions were then used for all medium and large file request generations for our three models. The Pareto's long tail being able to create infrequent very large files while also matching the shape of the files we found in our analysis of real game file sizes made it a great fit for our file generators.

3.3 Local System Model

We begin with a discussion of our Local System Model which has two storage locations that are utilized when a file is loading (shown in Figure 3.4). There is both short-term storage (RAM) and long-term storage (Hard Drive) that model the available storage in a local gaming setup.

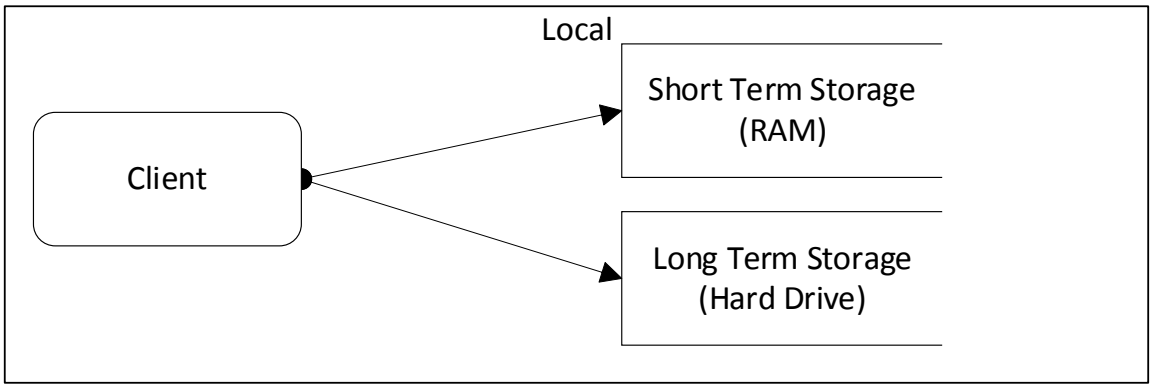

Figure 3.4 Local simulation Overview

When a file request is generated, it is either determined to be a cache hit, (the file request is in short-term storage), or a cache miss, (the file request is in long-term storage). In all Models, the storage locations are implemented as priority queues, that prioritize the earliest created file request. As the oldest request for a storage location is served first this created a First in First out(FIFO) system for each storage location. The priority queue then calculates the file request's time in the system and records that information for all file requests moving through the system. The short-term and long-term storage have a performance relative to each other. The short-term storage is the

faster of the two and can complete the service of a file request faster than the long-term storage. The time that a storage component takes to process a file request through the system is: $\frac{file\ request\ size}{transfer\ speed}$. This gives us our service time for a file in all models as a linear relation between the file transfer speed of the storage location and the file size. Both storage locations can move data independently. The storage components in all three of our models are implemented as priority queues, which we prioritize by file request arrival time. The Model calculates how long it would take for a file to be processed and then move the file out of the system, each file request's time in the system is recorded for analysis. This is done by using time slices where at the beginning of each time slice the file request at the front of the queue is checked using $\frac{file\ request\ size}{transfer\ speed}$ to see if it can be completed in the current time slice. If a file request can be completed in the current time slice it is completed marked with a completion time and moved to the client, if a file request cannot be completed within the time slice it is marked as partially complete and placed at the front of the queue to be addressed in the next time slice. When a file request is completed in less time than a full-time slice the remainder of the time slice is applied to the following file request, continuing through the queue until the entire time slice is used. Neither the short-term nor long-term storage components factor in network latency since all components are local to the client.

3.4 Cloud System Model

The Cloud System Model has two storage locations that are accessed for file loading: Cloud-based short-term storage (conceptually similar to RAM) and Cloud-based long-term storage (conceptually similar to Hard Drives) shown in Figure 3.5. As with the Local System Model, both storage locations have an access time, however, Cloud-based storage would also include network latency.

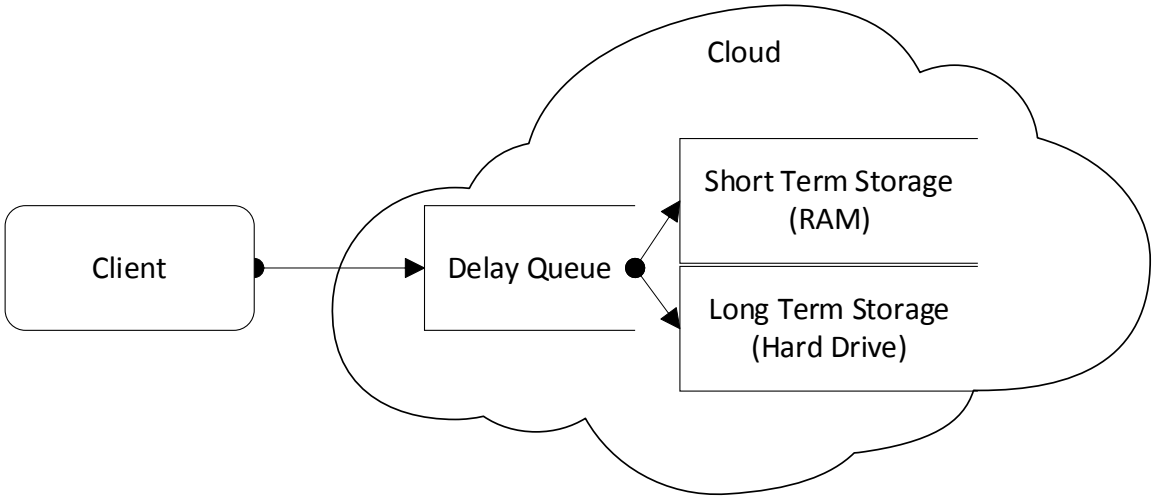

Figure 3.5 Cloud simulation Overview

Network latency is modelled using a delay queue, where by all file requests that move through the network are also passed through the delay queue which adds to a file requests time in the system. This behaviour is similar to the Local System Model because current Cloud-based gaming implementations are constructed by connecting to a Local system via a network.

3.5 Hybrid System Model

The Hybrid System Model has four storage locations that are accessed for file loading (shown in Figure 3.4); Local short-term storage (RAM); Local long-term storage (Hard Drive); Cloud-based short-term storage (conceptually similar to RAM); and Cloud-based long-term storage (conceptually similar to Hard Drive).

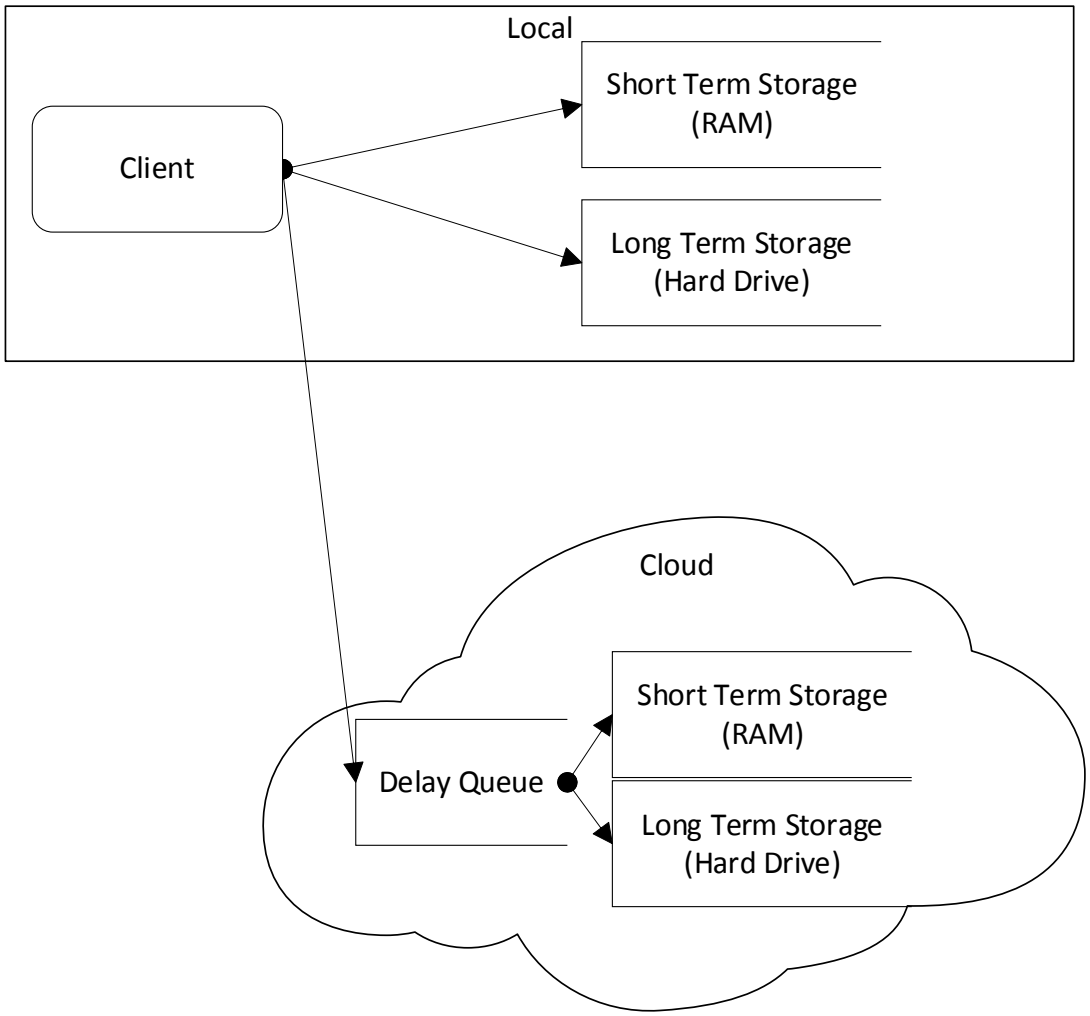

Figure 3.6 Hybrid simulation Overview

Network latency is also present in the Hybrid System Model for requests that involve files stored in Cloud locations. The Hybrid System Model assumes that files loaded into Local short-term storage or Local long-term storage are the pre-loaded and cached files, while the files loaded from the Cloud-based storage sites are the result of cache misses.

When a file request is Local, no network latency is incurred. In this situation, a file request from both short-term and long-term storage would be pre-cache hits in our system with the short-term storage serving file requests faster than the long-term storage mirroring operation of the Local System model.

When a file request requires a file from Cloud storage, a delay queue is used to model a network latency. The Cloud-based short-term stores files that are pre-cached on the Cloud system, while a Cloud-based long-term request is a cache miss for files on the Cloud system, this mirrors the Cloud Model. The goal of this model is to evaluate a system that shares the load between Local and Cloud hardware permitting time-sensitive user inputs to be handled on the Local hardware while less time-sensitive requests are handled in the Cloud. This would allow the model to perform actions that are most greatly impacted by increases in latency, completely locally and bypassing the most common impacts of latency on the users experience. The Hybrid Model aims to provide a performance improvement by leveraging resources from both Local and Cloud locations.

3.6 Model Verification

To ensure that the simulation models produced valid results multiple stages of testing and verification were used. The following section outlines the issues found in the

simulations and how they were corrected, as well as any assumptions that were made for reducing computational complexity.

Packet loss is omitted from the Cloud Model. While packet loss does affect Cloud Systems, the primary focus of our research is in reducing average input latency and file load performance. Additionally packet loss in the Cloud system does not affect the file transfer speeds since all of those transfers are handled by the Cloud server.

On the user side, packet loss is normally only noticeable in a video stream when multiple full frames are lost and the systems in place for current video streaming technology hide most packet losses from the user [57]. Packet loss can also be noticeable to the user if an input is lost. The impact of lost user input can be lessened by sending multiple copies of the same input, which would mean that multiple adjacent packets need to be lost in a row to prevent the input from arriving. If enough packet losses have happened to be noticeable to the user, then the system's quality of service will be low enough that users will likely stop using the system [58].

Packet loss is also omitted from the Hybrid Model as the effect of packet loss is a slight increase in total file transfer time based on a probabilistic amount of packet loss. Since the model already uses a distribution of total file transfer time, adding packet loss to the model adds an extra layer of complexity without having a noticeable effect on the results of the model.

The file request generation process was chosen to be the same for the Local, Cloud and Hybrid models. This was because most current Cloud gaming systems are very

similar to Local systems in implementation with the addition of a network delay for user inputs and results. The metric we were primarily interested in for this research, mean file transfer time, is the of time it takes for game files to be transferred to a location where the game system can utilize those files. This means that for models operating over a network, actions such as creating, compressing and transferring a video stream are not factored into model analysis. The Network in our Models is represented by a delay queue, adding a network delay to all file requests that are transmitted along the network. This delay queue is handled as a FIFO queue and adds a set amount of delay to all files transmitted along the Network.

Chapter 4
Results

4.1 Introduction

As discussed in Chapter 3, we have created three models to examine streaming systems for Cloud gaming: Local, Cloud and our proposed Hybrid approach. In this chapter, each of these models will be tested against twelve different scenarios factoring in; changes to file size frequency, cache hit miss rate, and file request interarrival time. We will discuss the results of each model, how they were impacted by the changes to input parameters, and how the models relate to one another.

The first parameter that we will investigate is the interarrival time of file requests. We use two scenarios: a short interarrival time ($\lambda = 10$), and a long interarrival time ($\lambda = 20$). This permits us to evaluate how each model performs as the interarrival time between file requests decreases.

The second parameter to be varied is the file size frequency, where these values are determined from our analysis in Chapter 3. We use a Pareto distribution for generating file sizes for the large and medium file requests. Let P_L be the probability of requesting a large file, let P_M be the probability of requesting a medium-sized file.

The three scenarios we used for file size were: 1) High Frequency of Medium File Size Requests where $P_M = 80\%$ and $P_L = 20\%$, 2) Balanced Frequency of Medium and Large File Size Requests where $P_M = 50\%$ and $P_L = 50\%$, and 3) High Frequency of Large File Size Requests where $P_M = 20\%$ and $P_L = 80\%$. Note that the number of

small file requests generated is not affected by changing this parameter. Small file requests are generated independently with their own separate file request generator. The rate of small file request generation is consistent across all cases for all models. All Models used the same file size distribution as we were primarily interested in file transfer speeds to move files to locations where the system can generate the next game state rather than transmitting image streams for the Cloud and Hybrid models.

The final parameter considered for the models is the cache hit rate. We let P_C be the probability of a cache hit and examine two cases: $P_C = 50\%$, and $P_C = 25\%$. These higher cache miss rates were used in our experiment as less optimal system operation was a primary performance concern of our research.

Using these various parameters, we end up with twelve Model Test Cases:

Table 4.1 Model Test Cases

Model Test Case (short hand)	λ	P_M	P_L	P_{hr}
SFBH	$\lambda = 10$	$P_M = 50\%$	$P_L = 50\%$	$P_{hr} = 50\%$
SFBM	$\lambda = 10$	$P_M = 50\%$	$P_L = 50\%$	$P_{hr} = 25\%$
SFLH	$\lambda = 10$	$P_M = 20\%$	$P_L = 80\%$	$P_{hr} = 50\%$
SFLM	$\lambda = 10$	$P_M = 20\%$	$P_L = 80\%$	$P_{hr} = 25\%$
SFMH	$\lambda = 10$	$P_M = 80\%$	$P_L = 20\%$	$P_{hr} = 50\%$
SFMM	$\lambda = 10$	$P_M = 80\%$	$P_L = 20\%$	$P_{hr} = 25\%$
LFBH	$\lambda = 20$	$P_M = 50\%$	$P_L = 50\%$	$P_{hr} = 50\%$
LFBM	$\lambda = 20$	$P_M = 50\%$	$P_L = 50\%$	$P_{hr} = 25\%$
LFLH	$\lambda = 20$	$P_M = 20\%$	$P_L = 80\%$	$P_{hr} = 50\%$
LFLM	$\lambda = 20$	$P_M = 20\%$	$P_L = 80\%$	$P_{hr} = 25\%$
LFMH	$\lambda = 20$	$P_M = 80\%$	$P_L = 20\%$	$P_{hr} = 50\%$
LFMM	$\lambda = 20$	$P_M = 80\%$	$P_L = 20\%$	$P_{hr} = 25\%$

Model results are measured in two ways, mean file transfer time and performance consistency as seen through interquantile range (IQR) values.

Mean file transfer time is the mean of time taken for a file to be transferred (i.e. the time it is requested until the time it is delivered). The lower the mean file transfer time, the faster the system can perform. A faster file loading time results in a higher quality user experience, typically demonstrated by faster loading screens and reduced pop-in. For our analysis we use a generic time unit for our mean file transfer time as we are interested in the relative performance between models rather than the absolute performance of each model.

The other important result of interest is performance consistency, which is based on IQR. In gaming, a consistently performing system is a much better user experience than an inconsistently performing system. In particular, consistency is experienced and perceived through input latency. In fact, game performance that has a consistently longer input latency would provide a better user experience compared to games that only intermittently experienced periods of longer input latency. Inconsistent performance changes are much more noticeable to a user than consistent game performance. This is analogous to driving on a highway through a large city. If you can drive completely through at a consistent speed, the driving experience is better than if you have to speed up and slow down multiple times, even if the consistent speed is lower than the maximum speed limit. Smaller IQR values demonstrate greater consistency since there is less difference in the majority (50% of all results) of mean file transfer times from the median mean file transfer time of a model. This means that a

model with a larger IQR has less consistent performance than a model with a smaller IQR value.

4.2 Local Model Results

The first of the three models we will be discussing is the Local Model. The Local Model will provide the baseline performance for both latency and file loading. The Local Model represents local game systems that are considered specialized hardware. Notable examples of these include the PlayStation or Xbox gaming consoles, as well as consumer-grade personal computers. In this section, we investigate the performance of the Local Model by examining the various test cases mentioned in Section 4.1.

The performance of Local Model by test case is presented in Figure 4.1. This figure illustrates the Local Model's performance by case.

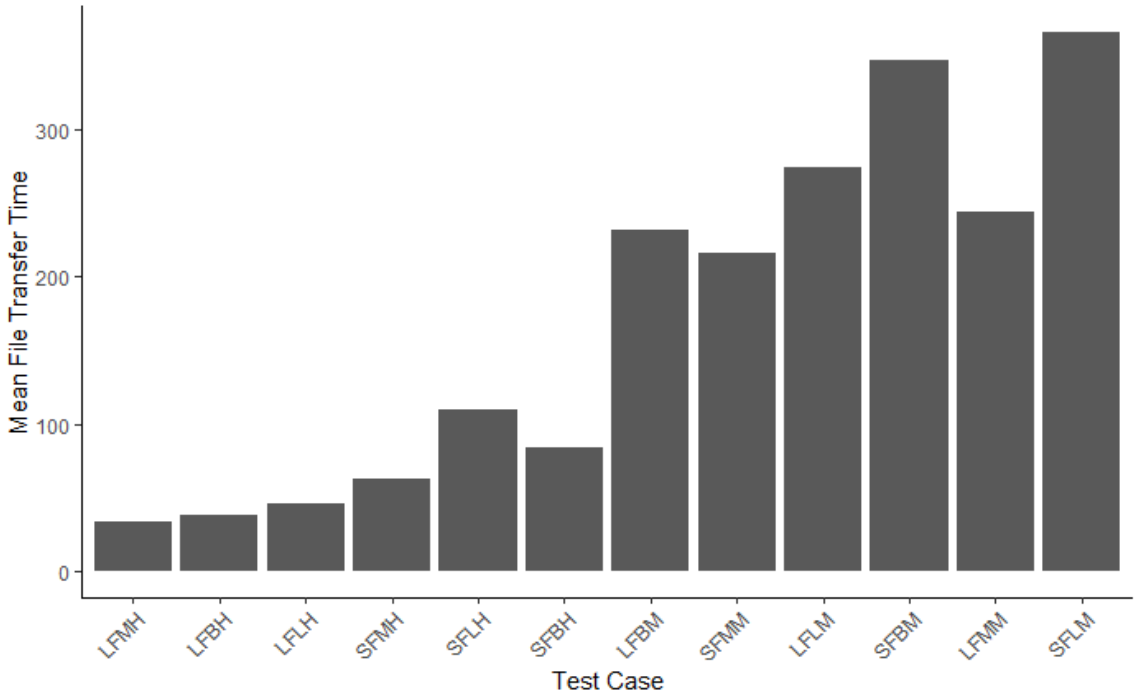

Figure 4.1 Performance by Case in Local Model

Interestingly, the Local Model performed most inefficiently in terms of mean file transfer time with the LFMM case. This could be a result of higher variability in performance with a high cache miss rate as other test cases performed as expected. Observing the overall performance of all test cases for the Local Model, we observe that having a high cache miss rate is the largest predictor of a higher mean file transfer time and as such, the most accurate predictor of poor performance in the Local Model.

We now move on to examine the three input parameters introduced in Section 4.1. We start with file interarrival time where we will compare the scenarios of $\lambda = 10$ and $\lambda = 20$. The results are shown in Figure 4.2 where we compare the mean file transfer times for the Local Model for $\lambda = 10$ and $\lambda = 20$.

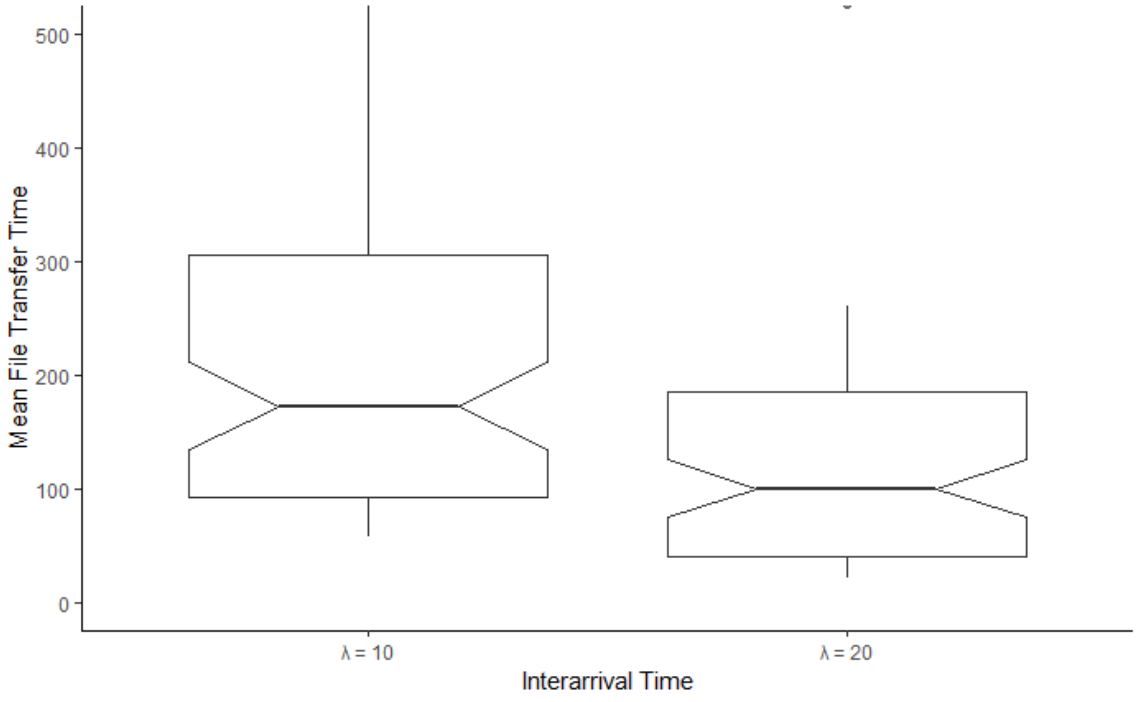

Figure 4.2 Comparison of Mean File Transfer Time in Local Model for Short and Long File Request Interarrival Times

Our results show that on average, cases with a shorter interarrival time achieve inferior performance to cases with a longer interarrival time. This follows directly from the fact that a lower file request interarrival time will lead to an increase in the number of file requests which results in an increase in the amount of load placed on the system. The λ = 10 scenario results in a 15% higher mean and a 72% higher median file transfer time over the λ = 20 scenario. We can conclude from these results that file interarrival time has a significant effect on the performance of the Local Model. There is also a significant effect on the variability in the file transfer time: the IQR for $\lambda = 10$ is 214.37 while for $\lambda = 20$ it is 145.05 (a 32.34% lower value). The IQRs are of particular interest for our scenarios because a smaller IQR represents a more consistent performance. We see an increase in both the file transfer times and the IQR values in the λ = 10 scenario because of an increased time spent waiting for all file requests regardless of file size. The reduced time between file request arrivals gives the model less time to move through all current requests before more file requests arrive and creates more situations where a large file request prevents many small file requests from being completed. The IQR increase also tells us that this creates more inconsistency in file transfer times as a string of small file requests can be served quickly while the arrival of a large file request negatively impacts the mean file transfer time of multiple requests.

The next parameter we evaluate for our Local Model is the proportion of the file sizes being requested. We will consider three scenarios: 1) there is a small number of large file requests, 2) there is a balanced number of medium and large size file requests,

and 3) there is a large number of large file requests. The primary effect of these scenarios is to change the time to service a file request rather than a change in the total number of file requests as seen with the previous parameter. This implies that while the average number of file requests in the model is similar for all three scenarios, the large file requests require more work and as such, should decrease the performance of the system. We can observe the results in Figure 4.3 for the Local Model.

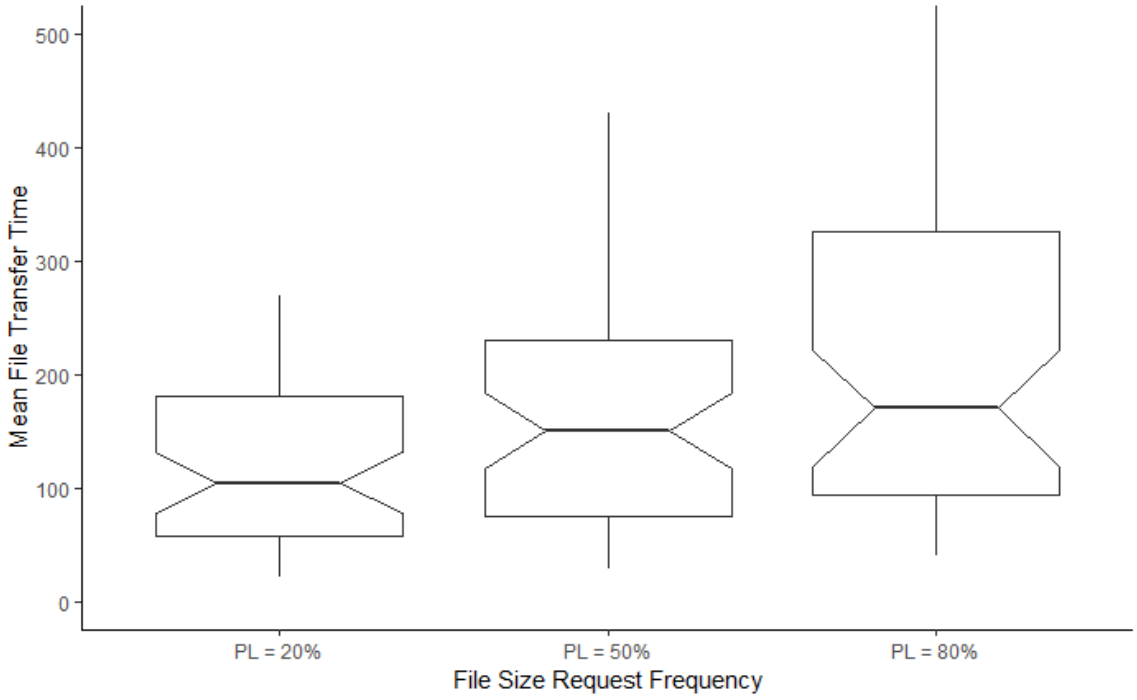

Figure 4.3 Comparison of Mean File Transfer Time in Local Model for all File Size Frequency Scenarios

From these results we can observe that as the size of the files requested increases, so does the mean file transfer time. This demonstrates that when the majority of file requests are not for large files, the mean file transfer time is smaller.

When a large percentage of file requests are for large files, the mean file transfer time increases and this increase is consistent across all of our scenarios: P_L = 20% where most files requested are medium sized, P_L = 50% where a balanced number of medium and large files are requested and P_L = 80% where the majority of files requested are large. Numerically, this is captured in the 43.5% percent increase between $P_L = 20\%$ and $P_L = 50\%$ scenarios, as well as a 13% percent increase between the $P_L = 50\%$ and $P_L = 80\%$ scenarios. We can see from these results that when the majority of file requests are medium the model performs the best and moving to a 50% of file requests being large gives us our most significant decrease in performance, continuing to increase the number of large file requests impacts the mean file transfer times less as seen in the only 13% increase in mean file transfer time from the 50% to 80% scenario. The decrease in mean file transfer time performance from the 20% to 50% scenario is likely from two aspects; a) more time is being spent waiting for small and medium files to be transferred while waiting for large files to be processed (grocery analogy) and b) an increase in mean file size which leads to an increase in time to transfer each file. The smaller increase in mean file transfer time is likely from a small increase in the minimum time that files take to transfer.

Now concerning the IQRs, the $P_L = 20\%$ scenario has a value of 122.38, the $P_L = 50\%$ scenario has a value of 154.86, and the $P_L = 80\%$ scenario has a value of 232.18. This gives us a 26.5% increase in the IQR between the $P_L = 20\%$ and $P_L = 50\%$ scenarios and a 49.9% increase in the IQR between $P_L = 50\%$ and $P_L = 80\%$. We suspect that the larger increase in IQR between the $P_L = 50\%$ and $P_L = 80\%$ test

scenarios come from larger file requests not only taking longer to process individually but delaying the system from moving onto new file requests, both increasing the time file requests spent waiting and the time file requests spent processing. The increase in IQR from the 20% and 50% scenario follows the increase seen in the mean file transfer times but interestingly the increase in IQR from the 50% to 80% is larger than the increase in mean file transfer time seen from the 50% to 80% scenario. This is likely because of an increase in the inconsistency of performance for small files that end up waiting more often for large file requests to be processed.

The final subset of experiments we conduct for the Local Model is with varying the cache hit rates. A high cache hit rate value indicates a scenario where files are pre-cached into fast access storage which should significantly cut down on the time to transfer a file to the user. A cache miss is where a file is stored in a distant or slow access memory which impacts the speed at which a file can be sent to the user. In Figure 4.4, we compare the mean file transfer times for the Local Model using a high cache hit rate ($P_c = 50\%$) and a low cache hit rate ($P_c = 25\%$).

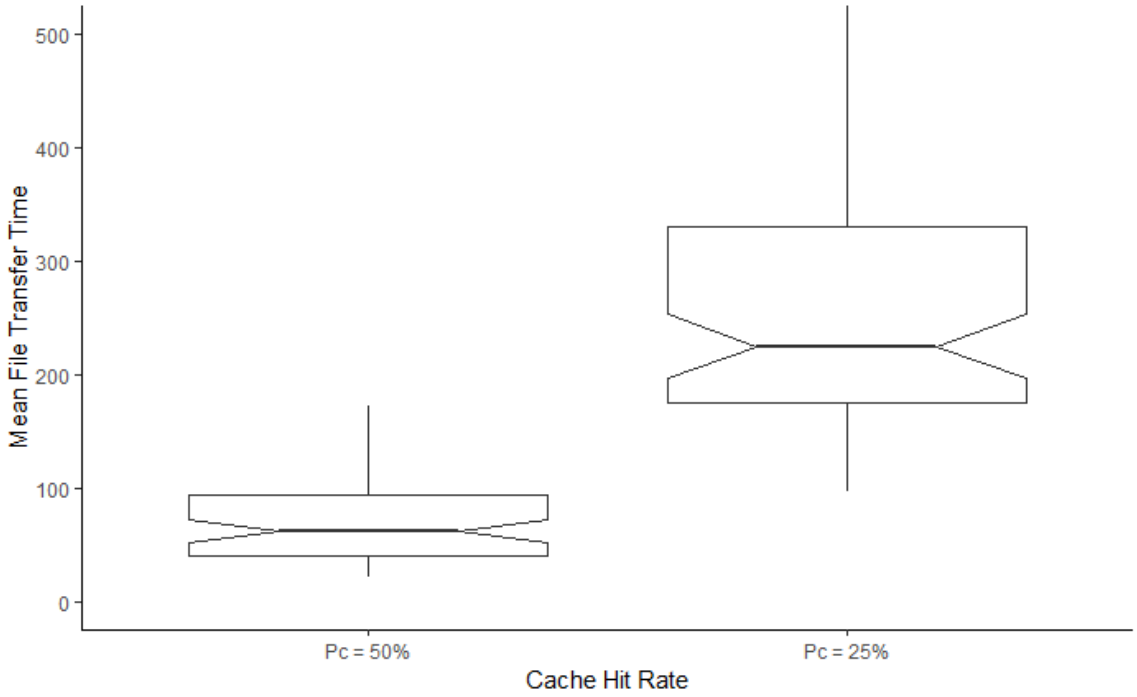

Figure 4.4 Comparison of Mean File Transfer Time in Local Model by Cache Hit Rate

We observe that based on medians, there is a 262% difference in the file transfer time between the high and low cache rate scenarios. The cache hit rate seems to have the greatest impact on our mean file transfer time performance metric. This makes sense as a much larger percentage of file requests are using the much slower long term storage, which greatly decreases the throughput of files through the system. As for the IQRs, we find a value of 53.90 when $P_c = 50\%$, and a value of 155.27 when $P_c = 25\%$. This represents a 188% increase in IQR. The cache hit and miss rate is the largest contributor to the performance of our Local Model in terms of mean file transfer time. Additionally, the IQR results show a high cache hit rate leads to a significantly more consistent model performance.

4.3 Cloud Model Results

The second of the three models we will be examining is the Cloud Model. This model represents the current video streaming methods used in Cloud gaming and is expected to have a similar, but slightly decreased, overall performance to the Local Model. This is because the current Cloud solutions for Cloud gaming have very similar hardware to that used in Local Model solutions.

Of the results for the performance of the Cloud Model are presented in Figure 4.5 for the twelve test cases. The Cloud Model performance is in line with our expectations; they follow closely the trends seen in the Local Model. This is expected because the underlying design is very similar between the Local and Cloud Models.

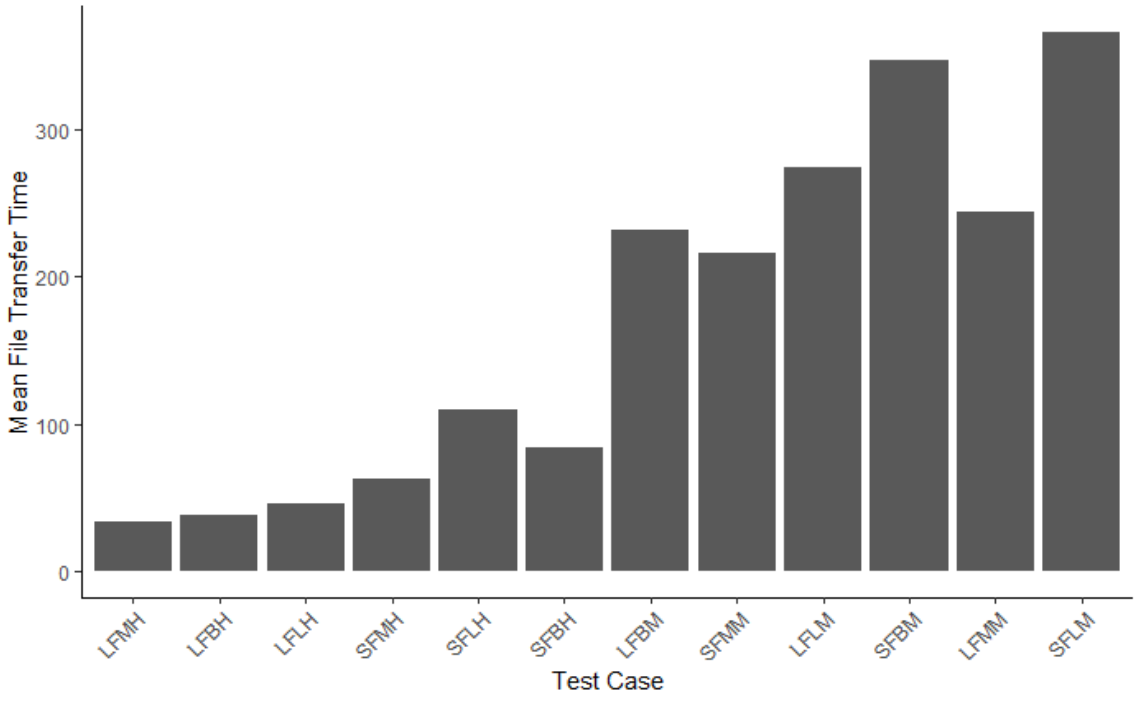

Figure 4.5 Performance by Case in Cloud Model

The Cloud Model performed worst in the SFLM case which follows the trends shown in the Local Model that cases with a higher amount of inconsistency in file requests, in this case with more file requests arriving closer together and with a higher chance of a cache miss, perform worse overall compared to the more balanced file arrival cases seen in LFMH, LFBH and LFLH cases.

We will begin our more detailed examination of the Cloud Model performance with file request interarrival times. In Figure 4.6, we plot the Mean File Transfer Time against Interarrival time for $\lambda = 10$ and $\lambda = 20$.

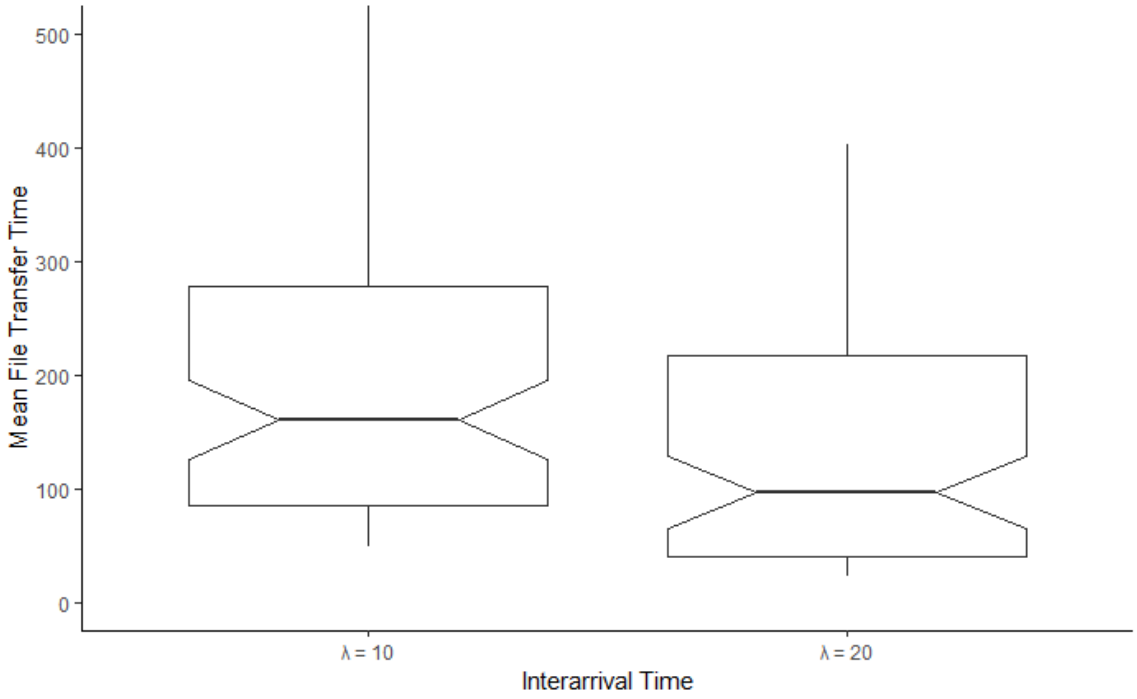

Figure 4.6 Comparison of Mean File Transfer Time in Cloud Model for Short and Long File Request Interarrival Times

We see from Figure 4.6 that as file interarrival time decreases the mean file transfer time increases. This is caused by an increased amount of files waiting in a queue to be processed. The results from the lower file interarrival time ($\lambda = 10$) were 36.62% higher in mean file transfer time than the results with the larger interarrival time ($\lambda = 20$). The IQR for the two test scenarios are not substantially different for the $\lambda = 10$ scenario the value is 192.22, while for $\lambda = 20$, the result is 177.37. This illustrates that while the long file interarrival time scenarios have shorter mean file transfer times, the consistency of performance shown by IQR values does not differ substantially between the test scenarios. Interestingly the difference in the IQRs is smaller than in the Local Model which could be a result of the additional overhead added to file transfers by having to pass through a network. This would raise the lower file transfer times but the large file transfer time would not increase significantly, causing a smaller difference in IQRs.

As with the Local Model, the next experiment (results shown in Figure 4.7) examines the three scenarios based on varying proportions of file sizes: 1) a small number of large file requests ($P_L = 20\%$), 2) a balanced number of medium and large file requests ($P_L = 50\%$), and 3) a large number of large file requests ($P_L = 80\%$).

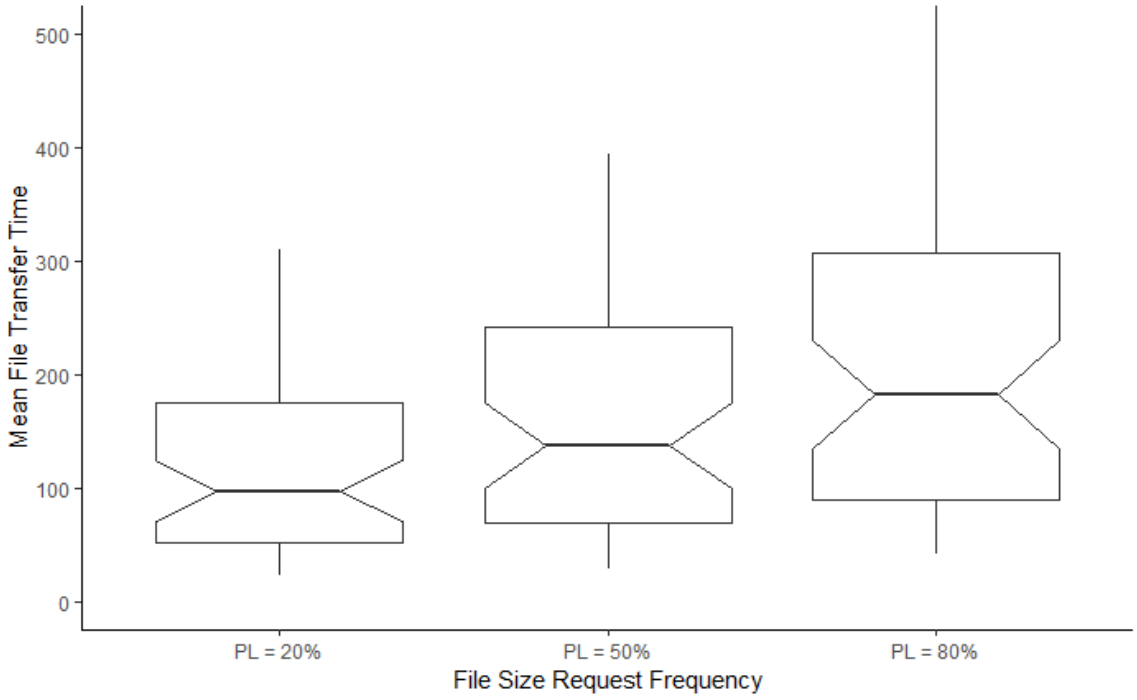

Figure 4.7 Comparison of Mean File Transfer Time in Cloud Model for all File Size Frequency Scenarios

As shown in Figure 4.7, as the proportion of the larger files being requested increases, the mean file transfer time also increases. This is demonstrated by the 41.4% increase in median file transfer time between the results for $P_L = 20\%$ and $P_L = 50\%$. There is a further 32.6% increase in median file transfer times between $P_L = 50\%$ and $P_L = 80\%$. The Cloud Model follows the Local Model in seeing the largest increase in mean file transfer time between the 20% and 50% scenarios, interestingly the Cloud Model is more impacted by the change from the 50% to 80% scenarios. This is most likely due to an additive combination of the increase in delays in file requests moving through a network and the increased average size of a file requested.

In the final subset of experiments we conduct on the Cloud Model we vary the cache hit rate. Displayed in Figure 4.8 is the mean file transfer time performance between a high cache hit rate ($P_c = 50\%$) and a low cache hit rate ($P_c = 25\%$).

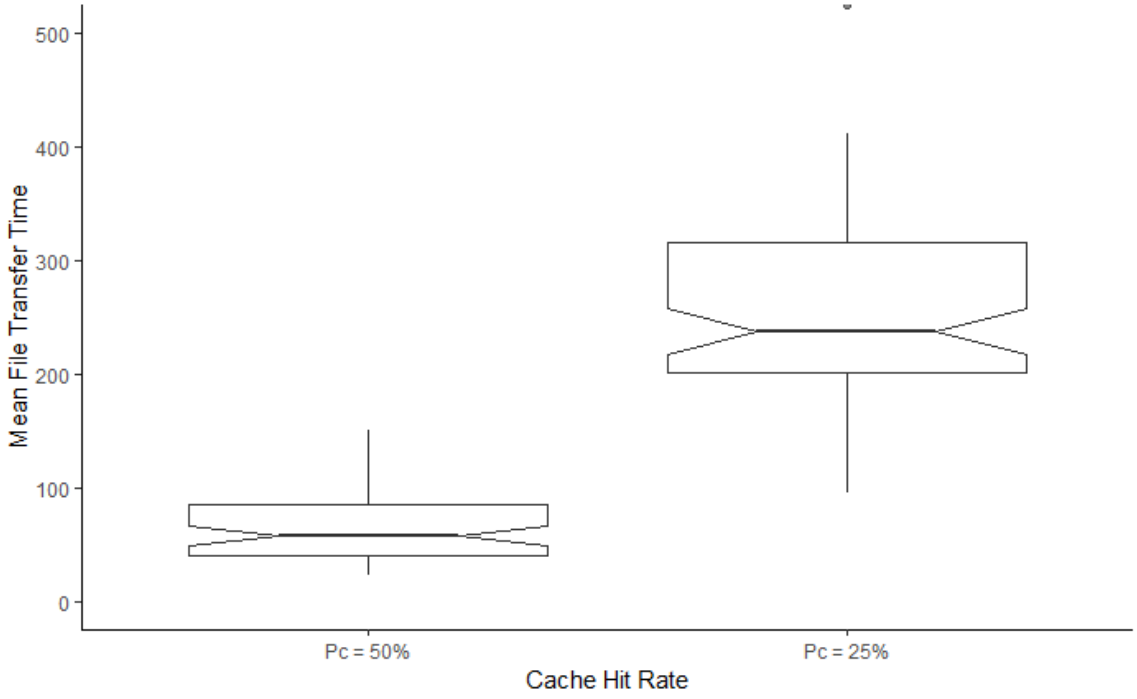

Figure 4.8 Comparison of Mean File Transfer Time in Cloud Model By Cache Hit Rate

In the scenario with the higher cache hit rate ($P_c = 50\%$), the IQR is 46.14 and tends to perform substantially better than the scenario with a lower cache hit rate ($P_c = 25\%$). For the scenario where $P_c = 25\%$, the IQR is 114.20, which is a 147.5% increase over $P_c = 50\%$. The low cache hit rate scenario also has a 312.4% higher median file transfer time. The large increase in mean file transfer time for the $P_c = 25\%$ scenario likely comes from the Cloud Model having to not only wait for a file transfer on a cache miss but also the additional network delay resulting from having to access the

correct locations to retrieve the requested files. Overall, the cache hit rate seems to be a good predictor of performance in the Cloud Model.

4.4 Hybrid Model Results

The final model we will analyze is our proposed Hybrid Model for a Cloud gaming system. This model is different from the Local and Cloud Models as it contains multiple locations from where files can load. We are particularly interested in how this model responds to the various test cases as good performance could indicate a possible solution for enhancing the user experience of Cloud Gaming.

In Figure 4.9 we examine the performance of the Hybrid Model for the twelve specific test cases.

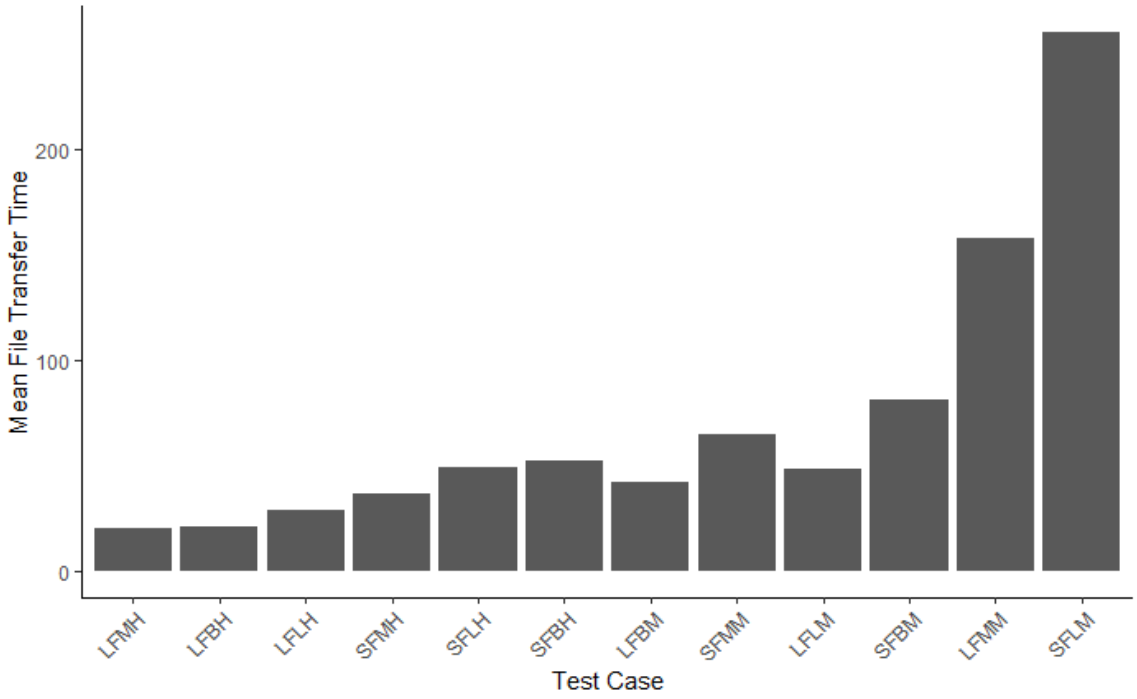

Figure 4.9 Performance by Case in Hybrid Model

Our initial observation is the performance of the Hybrid Model seems to follow the trends found in both the Local and Cloud model. Scenarios that performed well in both Local and Cloud models continue to perform well in the Hybrid Model showing that the contributing factors to model performance are similar between all models. Interestingly the Hybrid Model did not see large performance decreases until reaching the worst performing scenarios from the Local and Cloud models (e.g. scenarios with larger files with lower cache hit rates).

We now move on to the more detailed analysis of the Hybrid Model with an examination of the impact of varying the file interarrival times ($\lambda = 10$, $\lambda = 20$). The results are shown in Figure 4.10.

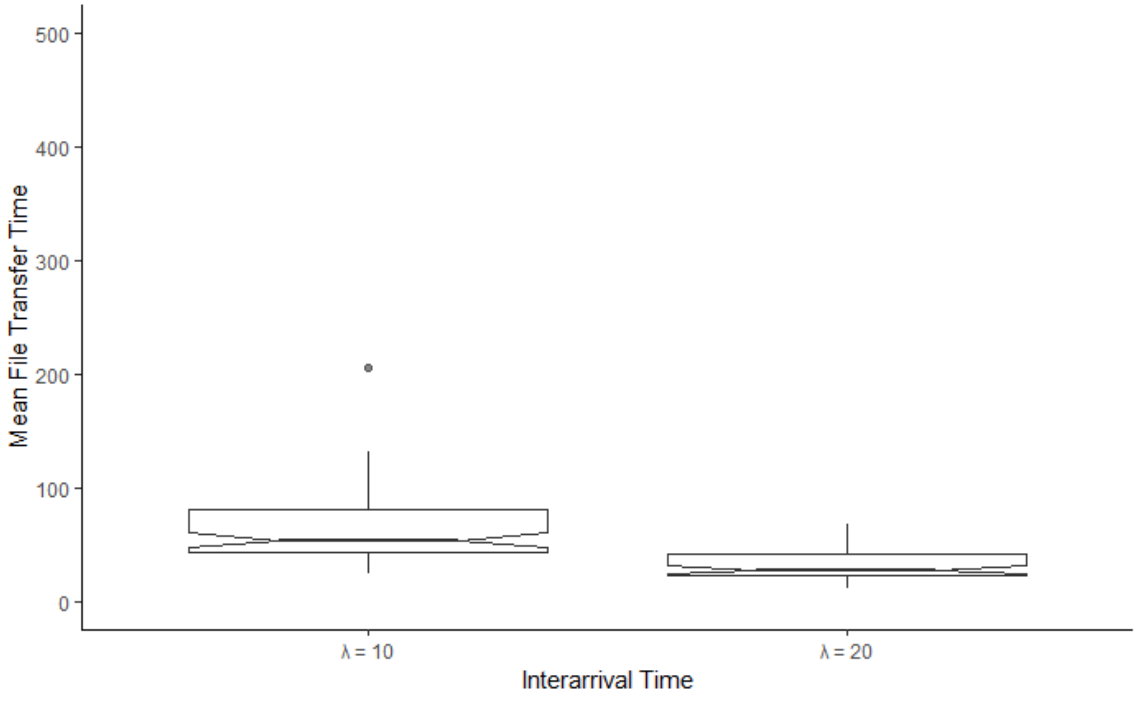

Figure 4.10 Comparison of Mean File Transfer Time in Hybrid Model for Short and Long File Request Interarrival Times

Observing the results, we can see that there is a large difference in performance between $\lambda = 10$ and $\lambda = 20$. The higher file request interarrival time ($\lambda = 20$) has a mean file transfer time which was 93.2% faster than the system with the lower file request interarrival time ($\lambda = 10$). The IQR for $\lambda = 10$ is 38.29 and for $\lambda = 20$, the IQR is 19.37. While both scenarios have small IQR values, the IQR for $\lambda = 10$ is 97.6% higher than that of the $\lambda = 20$. We conclude that a decrease in time between file requests does significantly increase the time waiting to be processed causing large increases in both the mean file transfer time and IQR.

From Figures 4.2 and 4.6, we can observe that the mean file transfer time for the Hybrid Model is significantly lower than the mean file transfer time of the Local and Cloud Models. This is likely because the Hybrid Model has two file transfer locations, avoiding the bottleneck of a single file transfer location as is present in the Local and Cloud Models. When one file transfer queue is delayed in the Hybrid Model because of a backlog of rapidly arriving file requests, the second queue may be unaffected and can deliver file requests without a significantly increased waiting time. When a backlog forms in the Local or Cloud Model, this affects all files waiting to be transferred and has a much greater impact on both the mean file transfer time and the IQR. The Hybrid Model has significantly lower mean file transfer times than the other two models, however, increasing the frequency of arrivals still has a notable impact on performance. While the Hybrid model has an increased ability to process file requests, overall performance is still negatively impacted when a backlog of file requests forms.

We now examine the effect of the proportion of file size has on the performance of the Hybrid Model. As before, we examine three choices: (1) a small number of large file requests, $P_L = 20\%$, (2) a balanced number of medium and large file requests, $P_L = 50\%$, and finally (3) a high number of large file requests, $P_L = 80\%$. The results are presented in Figure 4.11.

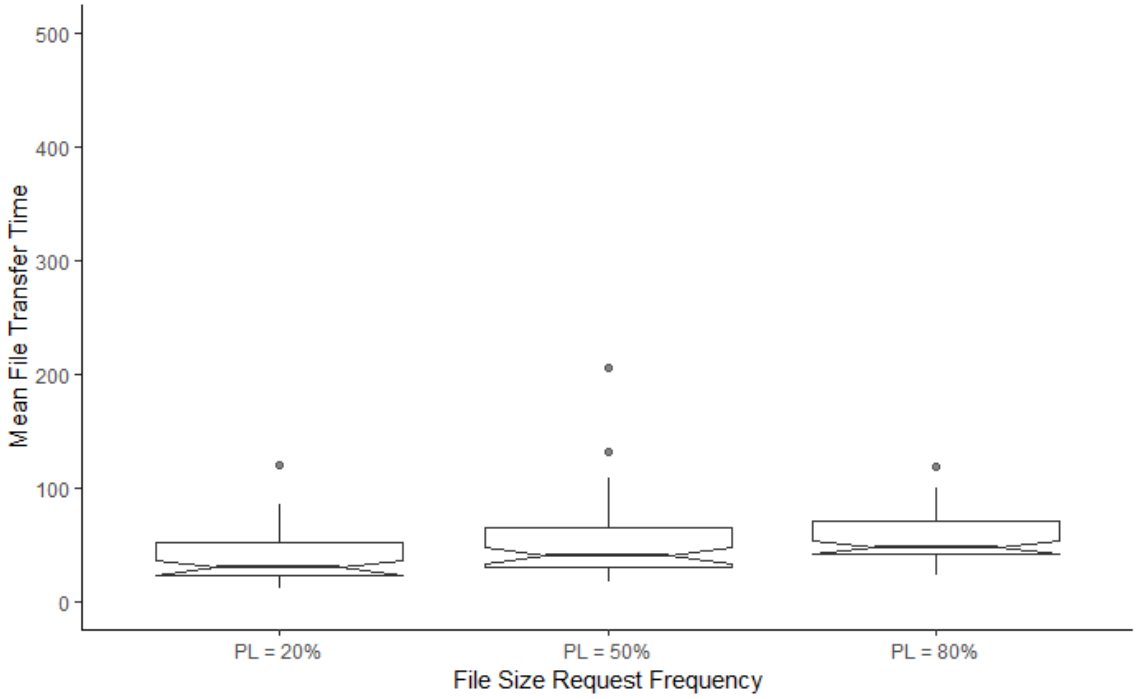

Figure 4.11 Comparison of Mean File Transfer Time in Hybrid Model for all File Size Frequency Scenarios

We found that the impact on the performance of the proportion of files sizes (P_L) is less than with the file request interarrival time (λ). For mean file transfer time, the difference between $P_L = 20\%$ and $P_L = 50\%$ is 35.1% while the difference between $P_L = 50\%$ and $P_L = 80\%$ is only 18.2%. The $P_L = 20\%$ scenario has an IQR of 27.98, the $P_L = 50\%$ scenario has an IQR of 35.00, and the $P_L = 80\%$ scenario has an

IQR of 28.81. The IQR values increased by 25% between the $P_L = 20\%$ and $P_L = 50\%$, but decreased by 17.6% between $P_L = 50\%$ and $P_L = 80\%$. The similar ranges, as well as a decrease in range size from balanced to large scenarios, lead us to believe that file size proportions do not have a significant effect on the Hybrid Model's performance. We observe that the greatest reduction in performance occurs between the $P_L = 20\%$ and $P_L = 50\%$ scenarios. This most likely occurs because the waiting time for the results are not significantly increased from the $P_L = 50\%$ to $P_L = 80\%$ scenario. The reduction in IQR when moving to the $P_L = 80\%$ is interesting and likely caused by an increase in the shortest length of time for a file transfer to occur without seeing a significant change in the longest lengths of time files take to transfer. This likely occurs because smaller files interspersed amongst larger files are taking longer to complete due to increased waiting times while larger files are being processed. Essentially this causes the range of file transfer times to narrow decreasing IQR while increasing mean file transfer time.

The final results we present for the Hybrid Model involve the effect that the cache hit rate (P_c) has on mean file transfer time. In Figure 4.12 we examine two scenarios, a high cache hit rate ($P_c = 50\%$) and a low cache hit rate ($P_c = 25\%$).

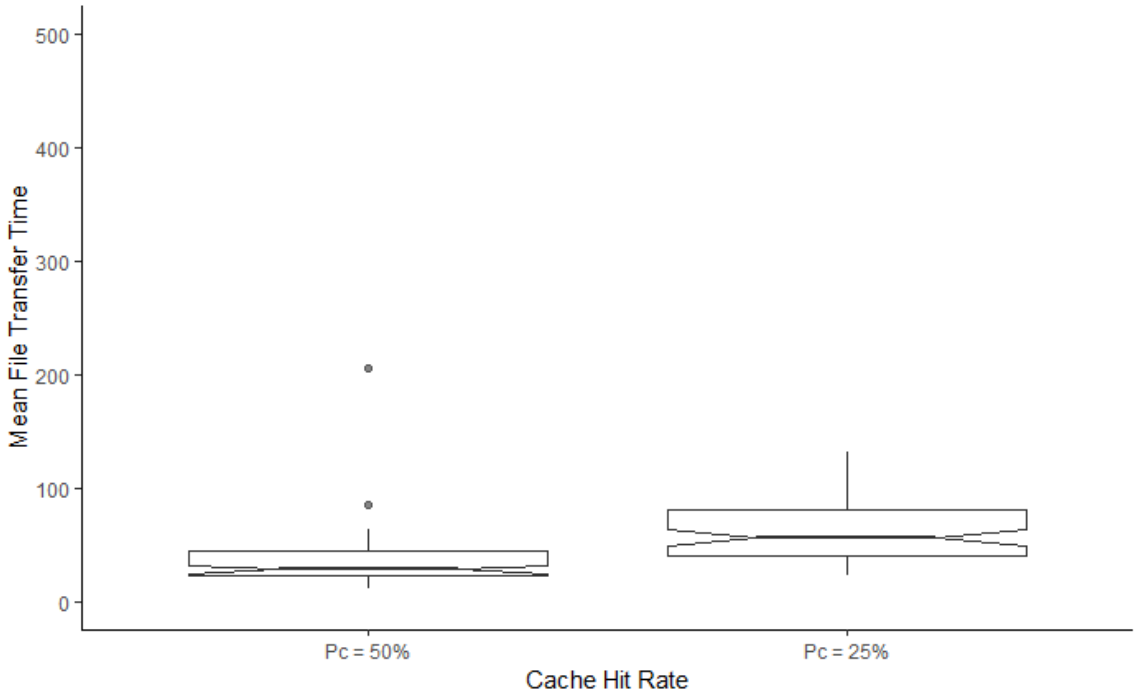

Figure 4.12 Comparison of Mean File Transfer Time in Hybrid Model for High and Low Cache Hit Rates

We found that a high cache hit rate seems to have a moderate impact on mean file transfer time. There is a 46.52% increase in median times between the high hit rate and the low hit (high miss) rate scenario. The IQRs were also greatly affected by cache hit rate, with the $P_c = 50\%$ scenario having an IQR of 21.93 while the $P_c = 25\%$ scenario has an IQR of 40.03. This is an 82.5% increase in IQR between the two scenarios. The increase in mean file transfer time is as expected and it follows the results of the Local and Cloud Models. We see that cache hit rate is the largest predictor of IQR in the Hybrid model, a lower cache hit rate greatly widens the times taken to serve files. The higher number of file transfer times at each end of the speed spectrum (shorter and longer) gives a greater variation in performance that is seen in the large difference in IQRs for each cache hit scenario.

4.5 Comparison of Model Results

The final results were interesting with the Local and Cloud Models performing as expected relative to each other. Both the Local and Cloud Models responded similarly to changes in cache hit rates, file interarrival times, and file size variations. The Cloud Model performed slightly worse than the Local Model overall, largely due to the time to send data through a network. From the results across all 12 test cases, the median file transfer time for the Local Model is 115.93, while the median time for Cloud Model was 124.4, resulting in a 7% degradation in performance from the Local to the Cloud Model. The overall difference between the performance of the two models is small with the main difference being attributed to the additional time required for transmitting files through a network.

The performance of the Hybrid Model was especially noteworthy. The Hybrid Model responded well to changes in cache hit rates, file interarrivals and file sizes with less variation in mean file transfer time compared to both the Local and Cloud Models.

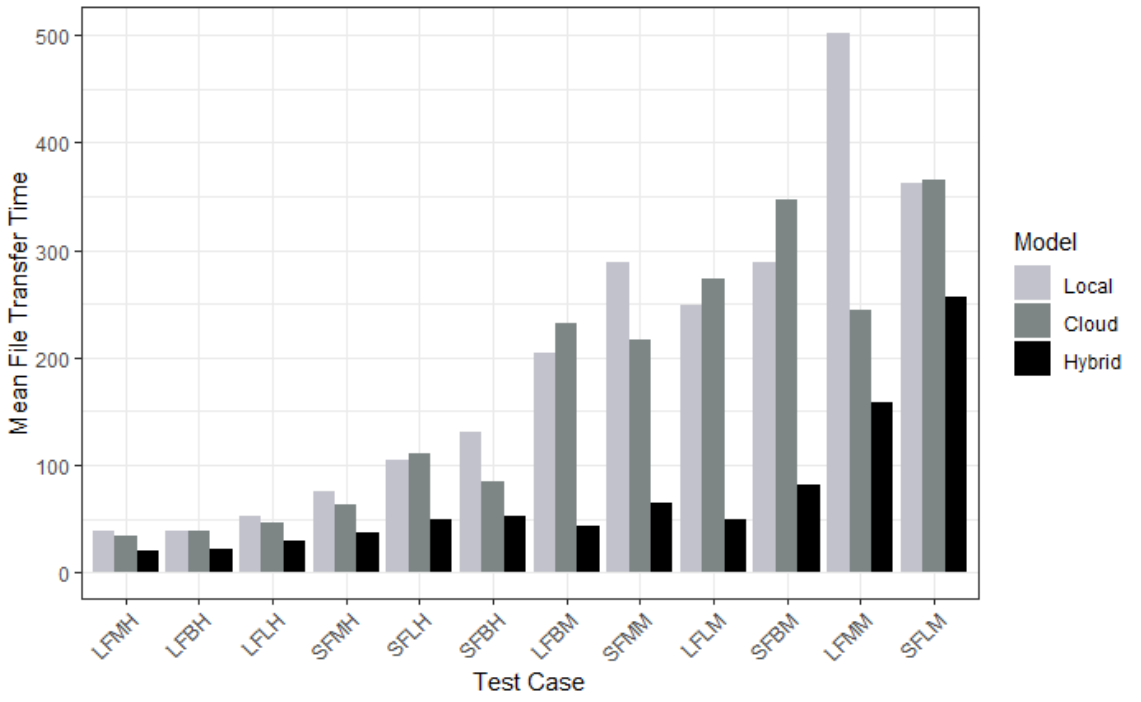

Figure 4.13 Comparison of Mean File Transfer Time for the Test Cases

Figure 4.13 presents the results for all three models for the 12 test cases. In all test cases, the Hybrid Model outperformed the Local and Cloud Models when it comes to mean file transfer time. The Hybrid Model had a median file transfer time of 42.45 across the 12 test cases which is a 63.00% decrease in the mean file transfer time compared to the Local Model, and a 65.87% decrease compared to the Cloud Model. We feel that the reason for the Hybrid Model's improved performance can be correlated to the bottlenecks imposed in the Local and Cloud Models. The Local and Cloud Models only have one path for files to move through the system. While the Hybrid Model has two paths that share the load of files. Thus, the Hybrid Model results in a higher throughput of files in the system and a large portion of the performance increases can be attributed to this increase in resources.

Another interesting observation from our performance analysis is the Hybrid Model performed better overall twelve scenarios with less variation. This is best illustrated in Figure 4.14 where we use a Violin Plot to show the mean file transfer times for all 12 test cases for each of the three models. The Hybrid Model shows a concentration of values falling within the lower range of mean file transfer time which is represented in the graph by the wide base in the violin shape of that model.

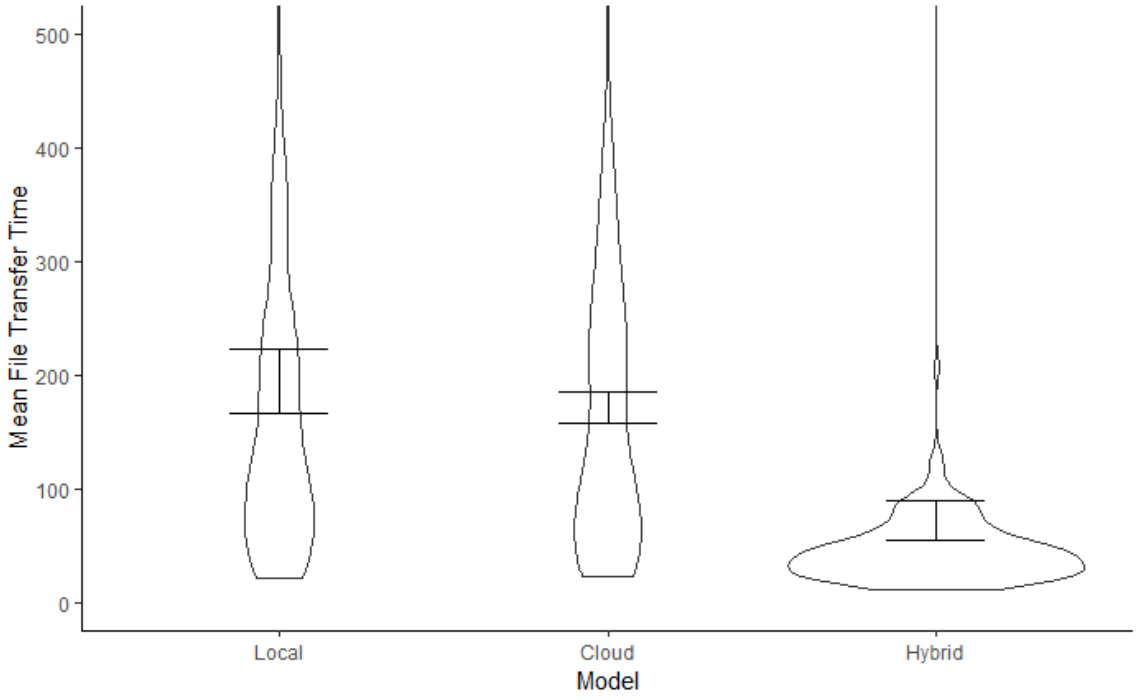

Figure 4.14 Comparison of Mean File Transfer Time by Violin Plot for all Models

The Local and Cloud Model plots are similar with a slight increase in the results in the higher mean file transfer time for the Cloud Model compared to the Local Model (which can be attributed to network delay). Figure 4.14 does show a stark difference between the Local and Cloud Models compared with the Hybrid Model. The Hybrid Model has the

vast majority of its mean file transfer times in the lower fifth of all performance and overall has much higher consistency in mean file transfer times for all cases. The Hybrid Model shows a significant decrease in the mean file transfer time across all test cases.

This increase in consistency is particularly noteworthy under cases with an expected higher amount of variations in file transfer time, as it shows that the Hybrid Model is more consistent in cases that tended to have high IQRs in the Local and Cloud Models. This is demonstrated by the concentration of results in the bottom section of the Violin Plot, compared to the evenly distributed results seen in the Local and Cloud Model plots. These performance characteristics could potentially provide not only a better user experience than current Cloud technologies, but also potentially have a more consistent performance under less-than-ideal networking capacity. This is because of the Hybrid Model's ability to perform faster file transfers throughout variable conditions and would allow the Hybrid Model to better address current Cloud gaming issues and potentially expand the availability of Cloud gaming to users with lower-end network capabilities.

4.6 Summary

The Local Model is our baseline performance for gaming solutions and performed within our expectations. Overall, for all test cases the Local Model has an IQR of 164.88 and an average of 115.93 for the median file transfer time. As illustrated in Table 4.1 the Local Model performed best in cases LFBH and LFMH with very small

differences between the performance. The worst performances for the Local Model were observed in test cases SFLM and LFMM.

Model Test Cases	Mean File Transfer Time	Total Observations
SFBH	130.18	1883738
SFBM	289.07	1771816
SFLH	105.22	1887910
SFLM	362.48	1769790
SFMH	74.94	1881607
SFMM	287.82	1772925
LFBH	38.57	1594304
LFBM	204.99	1592166
LFLH	51.98	1592830
LFLM	249.44	1596238
LFMH	38.93	1592404
LFMM	504.1	1592676

Table 4.2 Mean File Transfer Time for the Local Model by Test Case

The Cloud Model performed very similarly to the Local Model and again was within our expectations as the technology that powers a Cloud gaming solution is currently very similar to a Local gaming solution. The Cloud Model performed slightly worse in most cases than the Local Model, which can be attributed to the delays found from having to utilize a network connection. The Cloud Model has an IQR of 178.84 and a median file transfer time of 124.4, which is a decrease in overall performance when comparing it to the Local Model. As illustrated in Table 4.2 the Cloud Model performed best in the LFBH case and the LFMH case with very small differences between the performance.

Model Test Cases	Mean File Transfer Time	Total Observations
SFBH	84.15	1885351
SFBM	346.91	1771460
SFLH	110.23	1879261
SFLM	365.97	1771316
SFMH	62.96	1883680
SFMM	216.17	1768772
LFBH	38.47	1590926
LFBM	232.4	1591254
LFLH	45.78	1593792
LFLM	273.92	1594000
LFMH	33.91	1592533
LFMM	243.64	1590133

Table 4.3 Mean File Transfer Time for the Cloud Model by Test Cases

This coincides with our understanding of current Cloud infrastructures being slightly modified versions of the Local solutions. Cache hit rate continues to be the best predictor and largest influence on case performance, as all cases with a high cache hit rate performed substantially better than their low cache hit counterparts.

The Hybrid Model gave some surprising results with an IQR of 35.07 and a median file transfer time of 42.45. This points toward the large potential performance gains to be found in Cloud technology by implementing a Hybrid Model solution. As illustrated in Table 4.3, the Hybrid Model has a similar performance characteristic to the Local and Cloud Models in the areas of its best-performing cases. The Hybrid Model's best-performing cases are the LFMH, LFBH and LFLH.

Model Test Cases	Mean File Transfer Time	Total Observations
SFBH	52.44	1886547
SFBM	81.33	1884610
SFLH	49.52	1884186
SFLM	256.47	1884216
SFMH	36.55	1884216
SFMM	65.34	1884938
LFBH	21.08	1590460
LFBM	42.37	1592912
LFLH	28.85	1592195
LFLM	48.76	1590034
LFMH	20.05	1591769
LFMM	159.02	1591249

Table 4.4 Mean File Transfer Time for the Hybrid Model by Test Case

Interestingly, unlike the Local and Cloud Models, a balanced case, rather than a small file size case, was within its two best-performing cases. This shows that the Hybrid Model is more resilient to a wide variety of file requests. We can also note that while cache hit rate was still the largest predictor of performance and the largest difference in performance between cases, in general, most cases for the Hybrid Model did not see as large an increase in mean file transfer time in a low cache hit scenario compared to a high cache hit rate scenario when compared to the Local and Cloud Models. The increase of resources in the Hybrid model from leveraging Cloud components along with Local components allows for a more stable and better performing experience for the user.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

Observations of the performance of the Local and Cloud Models were consistent with our expectations at the outset of the analysis. However, the performance of the Hybrid Model under our scenarios has resulted in some interesting observations. The improved consistency and mean file transfer speeds provided by the Hybrid Model imply that it is a viable potential solution for improving performance in Cloud gaming technologies. The particularly important observation is the increase in consistency of performance when utilizing the Hybrid Model, as consistency is a crucial factor in a gaming experience. A user tends to remember the instances where a connection or responsiveness is unstable rather than the majority of the time where performance is reaching the user's expectations. A blip or a drop in performance at an inopportune time can result in a large loss in the quality of a user’s experience, and repeated or continuous drops would greatly degrade the user experience. The potential improvement in consistency provided by the Hybrid Model would be a large boon to improving the user experience for a Cloud gaming solution.

5.1.1 Cost Benefit Analysis

While our exploratory analysis of a Hybrid Model for Cloud gaming has shown potential for a substantial increase in performance in file loading and reducing user input latency, the Hybrid Model solution also has several detriments that could potentially impact the decision to implement this type of system. Further work into the performance increases and costs is necessary to determine if a Hybrid Model is feasible. Table 5.1 presents a cost-benefit analysis of implementing a Hybrid system for Cloud gaming.

Benefits	Costs
Noticeable reduction in input latency because time sensitive processing can be performed locally	Higher costs for users to supply processing capable hardware locally
Noticeable reduction in loading times (both initial game load and ongoing gameplay loading)	Higher costs for developers due to the added complexity of the file request management, plus likely reengineering costs for existing games to migrate to the Hybrid model
More resilient to unstable connections than traditional streaming	Effort required to synchronize the game state between Local and Cloud

Table 5.1 Cost Benefit Analysis of the Proposed Hybrid Model

As noted in Table 5.1, the largest benefit gained by the Hybrid Model is the reduction of input latency to be equal to, or near Local system levels. This leads to a direct increase in performance on the most influential part of a Cloud gaming system's user experience. Another important benefit is found in the file transfer speeds through

the Hybrid system which could potentially lead to improved loading speeds when compared to a completely Local system. The Hybrid system also demonstrated itself as being more resilient to unstable connections when compared to traditional Cloud gaming solutions. This is very important when dealing with lossy connections, such as Wi-Fi, or on connections that are not as powerful within areas with less-developed Internet infrastructure, as this would allow a Hybrid system to better reach these areas and users. These three factors taken together could allow for a greatly improved user experience compared to current traditional Cloud gaming systems, and potentially provide a better user experience than a Local system in some cases.

While the Hybrid system could offer some very powerful improvements in Cloud gaming technology, it does also have some detriments that are seen by the user. The first of these is that a Hybrid system would come at a higher cost to users than current Cloud gaming experiences resulting from the necessity to purchase the capable hardware. Current Cloud gaming experiences can work on an entirely thin client such as a laptop or older mobile phone that have a lower processing power. A Hybrid system would require additional processing power to handle the user interaction locally, which in turn would require a more powerful device which would be a cost borne by the user. The ability to use a thin client is one of the main draws to current Cloud gaming as it greatly reduces the barrier to entry. As a result, a higher barrier required by the Hybrid system may reduce the number of users willing to enter Cloud gaming.

The second major detriment to implementing a Hybrid system would be the increased cost on the development side. Currently, a Cloud gaming solution is a very

easy migration from a Local version to a Cloud version, as Local and Cloud solutions currently run very similar architecture. This allows for the migration costs from Local to Cloud to be kept down, as a Local version will run on most Cloud systems without a large number of edits required. For a Hybrid system however additional work would be required to determine which components of a game need to be run locally and which components of a game will need to be run in the Cloud location. Also, more work would be required into a file loading scheme specific to each game developed for the Hybrid system. As seen in our performance metrics in Chapter 4, the Hybrid system's primary performance increase comes from an anticipatory file loading scheme and a high cache hit rate of file loads.

The final barrier to a Hybrid system that we mentioned in Table 5.1 is the requirement to develop a solution for keeping the Local and Cloud components of the Hybrid system in sync. If a part of the system ends up ahead of the other, the lack of synchronization in the data being processed at both locations could greatly reduce user experience by causing a visual tearing in the game world, making the game unplayable. Some work has already been done in synchronizing two games across multiple clients, particularly in one-on-one multi-player competitive games, such as fighting games with GGPO netcode [59], synchronization is instrumental to an enjoyable and balanced user experience. Existing examples such as GGPO might allow a starting point into synchronizing the Local and Cloud components of the Hybrid system, but would also introduce further costs into the development of a Hybrid system as one has to consider

the migration of any existing games as well as in the development of any new games specifically targeting to a Hybrid system.

5.2 Future Work

Taking into account these benefits and detriments of a Hybrid system solution to Cloud gaming, it is not clear at this time if a Hybrid system is the best choice for moving forward with Cloud gaming technology. Additional work needs to be done to determine if the benefits of the system outweigh the various costs and if the synchronization of Local and Cloud components is viable for a system being used for Cloud gaming.

Future work on this topic could focus on the implementation of a synchronization system for Local and Cloud components, as this would be one of the major hurdles for creating a Hybrid system, and/or building and designing a Hybrid system using a current video game as a proof of concept. Building a Hybrid system would be a costly endeavour, however, if one was attempted, it would be best to not use an older game as the performance profiles are not indicative of modern game performance profiles and would not well represent modern Cloud gaming. Creating and testing a Hybrid system would allow for better comparisons of end-to-end input delay and give more accurate metrics to test the Hybrid systems performance. An area that could be expanded on the current model would be a more complex network layer, as the current network layer is greatly abstracted, adding variable network delay or testing

the model under variable bandwidths. Additionally testing multiple clients accessing a Cloud and Hybrid system would allow insights into how scalable these methods are.

References

[1] OnLive, "farewell," 7 April 2015. [Online]. [Accessed 28 March 2021].

[2] Yahoo, "NFLX profile," Yahoo, 14 April 2021. [Online]. Available: https://finance.yahoo.com/quote/NFLX/profile. [Accessed 14 April 2021].

[3] Yahoo, "SPOT profile," Yahoo, 14 April 2021. [Online]. Available: https://finance.yahoo.com/quote/SPOT/profile. [Accessed 14 April 2021].

[4] Yahoo, "YouTube Overview," Yahoo, 14 April 2021. [Online]. Available: https://finance.yahoo.com/company/youtube?h=eyJlIjoieW91dHViZSIsIm4iOiJZb3VUdWJlIn0. [Accessed 14 April 2021].

[5] Yahoo, "Twitch Overview," Yahoo, 14 April 2021. [Online]. Available: https://finance.yahoo.com/company/twitch?h=eyJlIjoidHdpdGNoIiwibiI6IlR3aXRjaCJ9. [Accessed 14 April 2021].

[6] Alexa, "The top 500 sites on the web," 2020. [Online]. [Accessed 23 June 2020].

[7] SimilarWeb, "Top Website Ranking," 1 May 2020. [Online]. [Accessed 23 June 2020].

[8] M. R. Johnson and J. Woodcock, "The impacts of live streaming and Twitch.tv on the video game industry," *Media Culture & Society,* pp. 670-688, 2018.

[9] O. L. Gallaga, "What if cloud gaming doesn't catch on?," Polygon, 15 October 2020. [Online]. Available: https://www.polygon.com/2020/10/15/21499250/cloud-gaming-fad-trend-not-catching-on. [Accessed 17 April 2021].

[10] Chikhani and Riad, "The History Of Gaming: An Evolving Community," 31 October 2015. [Online]. Available: https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/?guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAKe4Q0fWeVSicd1tZLBrngclW3KUh-3FHUKp5CQ_mpJ_YreC52xkMoBXo04plxqMBYc8v0QnRarhQqmWDvYDQbQx-bdRVm909QOoH9zxLOgK. [Accessed 14 March 2021].

[11] National Museum of Play, "Video Game History Timeline," [Online]. Available: https://www.museumofplay.org/about/icheg/video-game-history/timeline. [Accessed 14 March 2021].

[12] SNK Corporation, Neo-Geo Hardware Specification, 1990.

[13] B. Nicoll, "Bridging the Gap: The Neo Geo, the media Imaginary, and the Domestication of Arcade Games," *Games And Culture,* pp. 200-221, 2017.

[14] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang and C.-H. Hsu, "On the Quality of Service of Cloud Gaming Systems," vol. IEEE Transactions on multimedia Vol 16, no. 2, 2014.

[15] S. Schneider, "Introducing NVIDIA Reflex: Optimize and Measure Latency in Competitive Games," 1 September 2020. [Online]. Available: https://www.nvidia.com/en-us/geforce/news/reflex-low-latency-platform/. [Accessed 19 February 2021].

[16] L. *. Petit, "Controller Input Lag - How to measure it?," 19 January 2019. [Online]. [Accessed 23 June 2020].

[17] Teyah, "ARCADE STICK INPUT LAG TESTING - OVERVIEW," 7 July 2014. [Online]. [Accessed 23 June 2020].

[18] M. Carrascosa and B. Bellalta, "Cloud-gaming: Analysis of Google Stadia traffic," vol. arXiv preprint arXiv:2009.09786, 21 September 2020.

[19] A. M. D'Argenio, "The Past and Future of Cloud Gaming: Will it ever work," 10 October 2018. [Online]. [Accessed 23 June 2020].

[20] OnLive, "OnLive," 30 April 2015. [Online]. [Accessed 23 June 2020].

[21] J. Apostolopoulos, W.-t. Tan and S. J. Wee, "Video streaming: Concepts, Algorithms, and Systems," *HP Laboratories, Report HPL-2002-260,* 2002.

[22] M. Claypool and C. Kajal, "Latency and Player Actions in Online Games," *Communications of the ACM,* pp. 40-45, 2006.

[23] P. Falkowski-Gilski and R. Uhl, "Current Trends in consumption of multimedia content using online streaming platforms: A user-centric survey," vol. 37, 2020.

[24] D. Hernandez, "Game Creator Success on Twitch: Hard Numbers," 13 July 2016. [Online]. [Accessed 9 March 2021].

[25] N. Grayson, "Among Us' Improbable Rise To The Top Of Twitch," Kotaku, 9 September 2020. [Online]. Available: https://www.kotaku.com.au/2020/09/among-us-improbable-rise-to-the-top-of-twitch/. [Accessed 14 December 2021].

[26] H. Sun, A. Veto and X. Jun, "An overview of scalable video streaming," *Wireless Communications and Mobile Computing 7.2,* pp. 159-172, 2007.

[27] S. Riley and B. Haran, "How YouTube Works - Computerphile[Video]," 15 November 2013.

[Online]. [Accessed 22 February 2021].

[28] S. Riley and B. Haran, "How YouTube Works - Computerphile," 2013.

[29] B. Gilbert, "The standard price for video games is increasing to $70 for the PlayStation 5 and next-gen Xbox consoles, ending 15 years of $60 games," BusinessInsider, 16 September 2020. [Online]. Available: https://www.businessinsider.com/video-game-price-increase-60-70-nba-2k-xbox-ps5-2020-7. [Accessed 17 April 2021].

[30] T. Wilde, "How game sizes got so huge, and why they'll get even bigger," PCGamer, 9 February 2018. [Online]. Available: https://www.pcgamer.com/how-game-sizes-got-so-huge-and-why-theyll-get-even-bigger/. [Accessed 17 April 2021].

[31] W. Cai, R. Shea, C.-Y. Huang, J. Liu, v. C. M. Leung and C.-H. Hsu, "A Survey on Cloud Gaming: Future of Computer Games," *IEEE Access,* pp. 7605-7620, 2015.

[32] A. Donlon and J. Ziegler, "04: On Peeker's Advantage & Ranked," 13 May 2020. [Online]. Available: https://playvalorant.com/en-us/news/game-updates/04-on-peeker-s-advantage-ranked/. [Accessed 3 February 2021].

[33] Ubisoft, "Dev Blog: Ping Abuse, Peeker's Advantage, and Next Steps," 18 October 2017. [Online]. Available: https://www.ubisoft.com/en-us/game/rainbow-six/siege/news-updates/1su64agGqZZQWb0mBeJUem/dev-blog-ping-abuse-peekers-advantage-and-next-steps. [Accessed 3 February 2021].

[34] M. deWet and D. Straily, "Peeking into valorant's netcode," Riot Games, 28 July 2020. [Online]. Available: https://technology.riotgames.com/news/peeking-valorants-netcode. [Accessed 1 May 2021].

[35] GGPO, "GGPO Rollback Networking SDK," GGPO, [Online]. Available: https://www.ggpo.net/. [Accessed 28 May 2023].

[36] K. Webb, "Google's Stadia can stream Xbox One and PlayStation 4 games to your phone or computer, but it might kill your data plan," 7 June 2019. [Online]. [Accessed 23 June 2020].

[37] D. Terdiman, "GDC 2009: OnLive unveils on-demand game-streaming," GameSpot, 25 March 2009. [Online]. Available: https://www.gamespot.com/articles/gdc-2009-onlive-unveils-on-demand-game-streaming/1100-6206620/. [Accessed 13 April 2021].

[38] A. Oxford, "OnLive review," PCGAMER, 3 December 2011. [Online]. Available: https://www.pcgamer.com/onlive-review/. [Accessed 3 April 2021].

[39] Sony Interactive Entertainment, "PlayStation Now," Sony Interactive Entertainment, 2021. [Online]. Available: https://www.playstation.com/en-ca/ps-now/. [Accessed 3 April 2021].

[40] N. Pino and H. St Leger, "PlayStation Now review," techradar, 16 September 2020. [Online]. Available: https://www.techradar.com/reviews/gaming/playstation-now-

1213666/review. [Accessed 3 April 2021].

[41] S. Totilo, "Google Stadia Shuts Down Internal Studios, Changing Business Focus," Kotaku, 1 February 2021. [Online]. Available: https://kotaku.com/google-stadia-shuts-down-internal-studios-changing-bus-1846146761. [Accessed 3 April 2021].

[42] P. Harrison, "A message about Stadia and our long term streaming strategy," Google, 29 September 2022. [Online]. Available: https://blog.google/products/stadia/message-on-stadia-streaming-strategy/. [Accessed 27 May 2023].

[43] Microsoft, "Xbox Cloud gaming (Beta)," Microsoft, [Online]. Available: https://www.xbox.com/en-CA/cloud-gaming. [Accessed 28 May 2023].

[44] NVIDIA, "GeForce NOW," NVIDIA, 2023. [Online]. Available: https://www.nvidia.com/en-us/geforce-now/. [Accessed 28 May 2023].

[45] T. Scott, "Why Dark Video Is A Terrible Blocky Mess," 2020.

[46] T. Wiegand, G. J. Sullivan, G. Bjontegaad and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology,* p. 560576, 2003.

[47] D. Grois, D. Marpe, A. Mulanyoff, B. Itzhaky and O. Hadar, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders," *2013 Picture Coding Symposium,* pp. 394-397, 2013.

[48] H. K. Joy and M. R. Kounte, "An Overview of Traditional and Recent Trends in Video Processing," *2019 International Conference on Smart Systems and Inventive Technology,* pp. 848-851, 2019.

[49] T. Scott, "Why Snow and Confetti ruin YouTube Video Quality," 2016.

[50] Y.-T. Lee, K.-T. Chen, H.-I. Su and C.-l. Lei, "Are All Games Equally Cloud-Gaming-Friendly?," *2012 11th Annual Workshop on Network and Systems Support for Games,* pp. 1-6, 2012.

[51] N. D. Scott, D. M. Olsen and E. W. Gannett, "An Overview of the visualize fx Graphics Accelerator Hardware," vol. 49, 1998.

[52] P. R. Boulian, "Mandala: A Portal Engine," 2006.

[53] esi, "Metroid Prime OOB Movement Tutorial," 6 August 2016. [Online]. Available: https://www.speedrun.com/mp/guide/2vxrj. [Accessed 4 April 2021].

[54] M. Hemmati, A. Javadtalab, A. A. Nazari Shirehjin, S. Shirmohammadi and T. Arici, "Game as Video: Bit Rate Reduction Through Adaptive Object Encoding," *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video,* pp. 7-12, 2013.

[55] S. Shi and C.-H. Hsu, "A Survery of Interactive remote Rendering Systems," *ACM Computing Surverys,* vol. 47, pp. 1-29, 2015.

[56] EPIC Games, "Level Streaming Overview," EPIC Games, [Online]. Available: https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LevelStreaming/Overview/. [Accessed 14 December 2021].

[57] A. Huzak and S. Imre, "Analysing GOP Structure and Packet Loss Effects on Error Propagation in MPEG-4 Video Streams," *Proceedings of the 4th International Symposium on Communications,* pp. 1-5, 2010.

[58] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," *IEEE/ACM Transactions on Networking,* vol. 21, pp. 2001-2004, 2013.

[59] "GGPO Rollback Network SDK," 7 November 2019. [Online]. Available: https://github.com/pond3r/ggpo. [Accessed 11 December 2022].

[60] S. Chakrabory, S. Paul and K. M. A. Hasan, "A Transfer Learning-Based Approach with Deep CNN for COVID-19- and Pneumonia-Affected Chest X-ray Image Classification," *SN Computer Science,* vol. 3, 2022.

Appendix

Sean Baxter Personal Steam library of installed games as of July 6th 2020
Large Blob files removed from dataset to avoid skewing the numbers to larger file request results. This is because these files are very large in size but are not normally completely loaded into game memory, instead parts of the file are loaded. Keeping the full sized files in the data set would have skewed the file requests to being much larger on average.

Game Title	**Total Size in MB**
Arcade Spirits	498
Armello	1583
Atelier Sophie The Alchemist of the Mysterious Book	6563
Audiosurf 2	1022
Autonauts	775
Baba Is You	67
Bastion	1264
BATTLETECH	35416
Cities_Skylines	11167
Cosmic Star Heroine	1962

Counter-Strike Global Offensive	22594
Crusader Kings II	2947
DarkestDungeon	4403
Dead Cells	685
DemonCrawl	399
Disco Elysium	14259
Divinity Original Sin 2	59122
DNFTM	951
Donut County	310
dota 2 beta	25665
Dungeons 3	6459
Elite Dangerous	131
Eternal Card Game	1928
Evil Genius	1365
Factorio	1512
Fallen ~Makina and the City of Ruins~	973
Fell Seal	1059
FINAL FANTASY XIII	58986
FINAL FANTASY XIV Online	47359
Gang Beasts	1886
GarrysMod	4349
Grim Dawn	9639
GRIS	3868
HatinTime	7525
Hearts of Iron IV	2583
Heaven's Vault	4120
Helltaker	366
HITMAN2	14037
Hob	2477
Hollow Knight	7566
Jotun	3911
Katana ZERO	216
Kindred Spirits on the Roof	794
Left 4 Dead 2	12721
Momodora RUtM	242
Monster Train	1158
Noita	929
One Step From Eden	493
OxygenNotIncluded	1664
Path of Exile	129
Pillars of Eternity II	52652

Planet Zoo	10766
Pyre	8606
Risk of Rain 2	2072
Salt and Sanctuary	6
Sid Meier's Civilization VI	17489
Skyrim Special Edition	12997
SlayTheSpire	537
Slime Rancher	1418
Sniper Elite 4	27699
SoulcaliburVI	3675
Spin Rhythm	640
Stardew Valley	497
Sundered	2347
Temtem	4590
Terraria	280
they bleed pixels	572
Thumper	975
tModLoader	36
Torchlight II	1764
Transistor	3764
TreeOfSavior	418
Underlords	2918
wallpaper_engine	742
Warframe	7124
West of Loathing	233
XCom-Enemy-Unknown	18967
Grand Total	**575882**