Cloud Gaming and Simulation in Distributed Systems

Khen Cruzat

Submitted in accordance with the requirements for the degree of
Computer Science

2015/16

The candidate confirms that the following have been submitted.

<As an example>

Items	Format	Recipient(s) and Date
Deliverable 1, 2, 3	Report	SSO (DD/MM/YY)
Participant consent forms	Signed forms in envelop	SSO (DD/MM/YY)
Deliverable 4	Software codes or URL	Supervisor, Assessor (DD/MM/YY)
Deliverable 5	User manuals	Client, Supervisor (DD/MM/YY)

Type of project: _____

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) _____

Summary

Cloud computing attempts to enable access to high-end graphics intensive games to a wider audience by using powerful data centers. The process of delivering cloud gaming includes processing the user input for the game engine then encoding this as video and streamed to the end user. Due to another added layer of complexity, other problems arise and the aim of this project is to tackle networking issues in the cloud to reduce the latency experienced by the user.

The Software-Defined Networking paradigm allows the management of network technology without the need of touching individual switches. This project will discuss the effects of using SDN for cloud gaming in a distributed system.

Acknowledgements

<The page should contain any acknowledgements to those who have assisted with your work. Where you have worked as part of a team, you should, where appropriate, reference to any contribution made by other to the project.>

Note that it is not acceptable to solicit assistance on 'proof reading' which is defined as the "the systematic checking and identification of errors in spelling, punctuation, grammar and sentence construction, formatting and layout in the test";

see http://www.leeds.ac.uk/gat/documents/policy/Proof-reading-policy.pdf.

Contents

Chapter 1

Introduction

1.1 Context

Cloud computing has always been seen as a way of improving compute performance by the use of multiple computers connected to each other. The games industry has rapidly evolved along the years and a great deal of demand is apparent. Developers of games have pushed computer hardware to meet the needs of consumers for more complicated games and realism. Even with computer hardware becoming cheaper and more of a commodity, the costs of driving graphics rendering for high-end games to run at the optimal settings of 1080p at 60fps are still relatively high.

1.2 Project Aim

The aim of the project is to produce a solution which uses Software-Defined Networking to reduce the network latency in a network.

1.3 Project Objectives

- A simple game program, that is computationally expensive enough to not perform optimally on a single machine (simple flight simulator with real time procedurally generated trees).

- A simplified cloud gaming system where the game created is launched on the cloud and input on the client side in the form of button presses on the keyboard is sent to the game on the server. The game frames produced are then sent to the client's screen.

- Produce a virtual network with simulated cloud game traffic and delay. With the use of SDN, reduce latency in the network.

1.4 Deliverables

The deliverables of the project include:

- Code that demonstrates a simple game/simulation rendering graphics on a server and controlled by a client remotely.

- A manual on how to setup the client and server will be produced so the cloud gaming system can be easily setup and launched.

- Code that will create a virtual network as well as a manual on how to set it up and run the code.

- Project report that explains the problem the project is trying to solve and the schematics of the solution produced as well as an evaluation of the solution.

1.5   Project Schedule

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Chapter 2

Background Research

2.1 Problem Overview

Compute power of games can be offloaded to a server of much powerful computers and then streamed to a client with lower specification hardware such as a laptop or a mobile device. With this comes risks and shortcomings that need to be factored. One of these problems are the latency of the game. Latency can be huge factor in the gameplay such as high-paced games like first-person shooters or fighting games. The delay in pressing a button on the gamepad to seeing the action performed on the screen needs to be kept to a minimum. This idea of interaction delay tolerance being different from genre to genre of games is discussed by Shea et al. [23]. As stated above, a player of FPS games can only tolerate the least which is around 100ms whereas Role playing game (RPG) gamers can tolerate around 500 ms.

Another problem that is directly linked to delay in the system is the effect of packet loss. As stated in the *Eight Fallacies of Distributed Computing* [5], it should be assumed that latency is never zero as mentioned above as well as network is not always reliable. This means that packet loss can occur which in terms of cloud computing can mean the degradation of image quality. In the investigation conducted by Jarschel et al. [10] in which they surveyed average consumers about the importance of packet loss and delay. Generally the quality of the video streamed to the clients plays an important role as the participants were open to using such as a service if provided in good quality.

2.2 Cloud Computing

According to the National Institute of Standards and Technology [16], cloud computing is a means of providing on demand access to computing resources over the network. This should be executed with minimal management effort or service provider interaction. With the shared computing resources, cloud computing aims to process larger data and solve large scale computation.

This cloud computing model can be separated in to three different service models: Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) [9]. 'SaaS' model attempts to eliminate the need to install and run the application on the user's system. An example of this is the Microsoft Office 365 package which provides productivity software through the web browser. 'PaaS' model provides

the consumer a computing platform using the cloud infrastructure to allow running and building their own applications. The consumer does not need to manage the "underlying cloud infrastructure including networks, servers, operating systems or storage [16]". 'IaaS' model provides the capability to control the processing, storage and networks to an extent. The consumers will have to install the OS images and related application software on the cloud infrastructure [13].

A disadvantage stated by Grossman [7] is that since cloud services are often remote, it can suffer from latency and bandwidth related issues. Data centres can be physically located anywhere in the world, so the number of router hops from the client to the server may attribute to the latency. As well as distance, delays are also introduced by network hardware and error correction on data packets.

2.3   Cloud Gaming

Cloud gaming is new technology that can be seen as an alternative by having the games run remotely on a server and then streamed to the user. Performing computations remotely as with streaming games remotely is believed to gain traction in the future in the same way how streaming videos and audio have become ubiquitous through services such as Netflix and Spotify. NVIDIA's GRID Cloud Gaming advancements have shown that this is becoming the case. As stated by Mariano in *Is cloud gaming the future of the gaming industry* [15], cloud gaming is increasingly becoming an attractive option for consumers as higher end games can then work on simpler, cheaper clients as well as with devices that they may already own also known as thin clients through the use of powerful server GPUs.

These thin clients are responsible for displaying the game frames rendered on the cloud server side in the form of video frames. Also, it has to collect and process the game control inputs from the user and send these to cloud to be registered as inputs on the game engine. According to Shea et al [23], cloud gaming would be of great benefit to the game industry as it would open the user base to the thin clients. For example the recommended specifications to run the 2015 Game of the Year title Witcher 3[6] would require a system that has [18]:

- CPU: Intel Core i7 3770 3.4 GHz / AMD FX-8350 4 GHz

- GPU: GeForce GTX 770 / AMD Radeon R9 290

- RAM: 8GB

A system that has these components would cost around £400 and this does not include peripherals such as keyboard, mouse and monitor. The latest tablets and laptops can

barely if at all meet the recommended specifications to run the game natively. Furthermore, games will have to deal with running on different hardware architectures and operating systems. Cloud gaming will make it easier for developer as they will not have to deal with platform compatibility and per platform tuning.

In the paper *Cloud Gaming: A Green Solution to Massive Multiplayer Online Games* [4] it mentions that NVIDIA has introduced SHIELD which is a mobile gaming device that can be connected to a desktop PC with a compatible NVIDIA GPU and stream gameplay to the device via 802.11n WiFi. Another feature is the ability to connect to one of NVIDIA's data centres to play games from their vast library of stream-ready games. One of the benefits of this service is the convenience of not having to wait for the download and installation of the game as you simply pick a game and instantly start playing. The games will not be stored on the client's device so storage limitations will not be a problem. The service also boasts gameplay performance of up to 1080p at 60fps which are deemed by gamers to be the target performance.
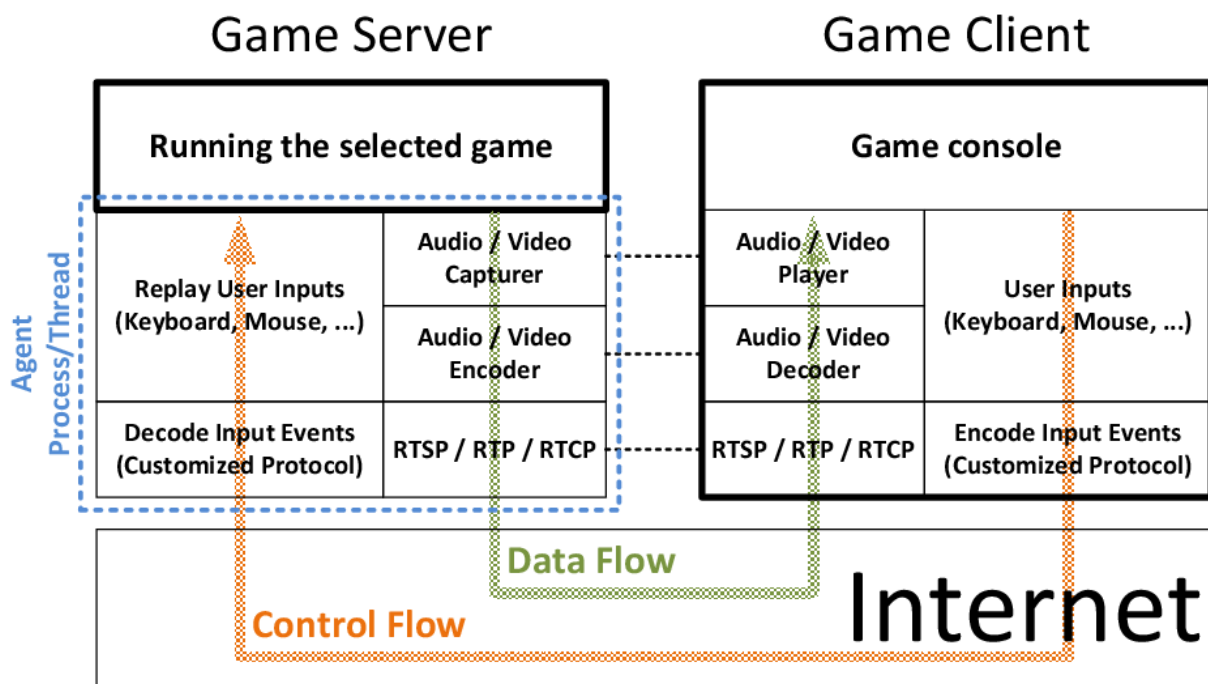

Figure 2.1: Cloud gaming system architecture

A simple cloud gaming architecture consists of several procedures which adds on top of the game engine process as shown in Figure 2.1. The client would send user inputs through some form of controller like a keyboard, mouse, or gamepad to the thin client. The game client then encodes these commands so it can easily be sent to the game server via network. When the server receives the user inputs, it simulates them so the game that is running recognises them. The game renders the frames produced as a result of the inputs and are captured and encoded to video frames. This is done so it can be easily streamed to the client through the use of Real Time Streaming Protocol

(RTSP) which is a network control protocol that manages delivery of data with real time properties [20]. The thin client uses RTSP to receive the video frames and decodes them to be displayed on a video player. The user then sees the results of the button presses sent from the game running on the server side.

Due to the many different processes involved in the architecture of a cloud gaming system, managing latency has become a problem. A traditional gaming system already experiences latency and as shown in Figure 2.2, this arises from the game pipeline and display lag. The game pipeline latency is the amount of time it takes for the game to compute and render a frame and the display lag is the time it takes to display the frames on the screen. Display lag can be caused by the display's scaler since current displays have a fixed resolution and expensive image processing such as dynamic contrast and motion interpolation [24]. Cloud gaming introduces latency from capturing/encoding game frame to video frame, network and decoding on the client side.
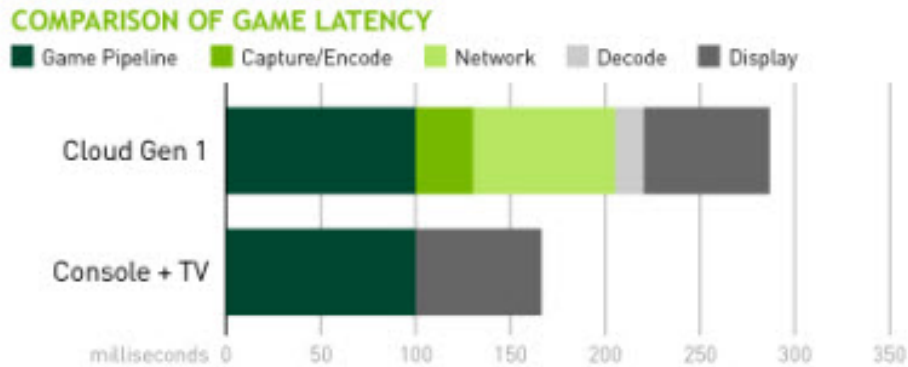

Figure 2.2: Latency in cloud gaming

2.4   Latency Mitigation

One method of latency mitigation in the network is to simply move the server closer to the clients. This means that traffic will have to travel less distance therefore latency will decrease. Unfortunately, this solution is not feasible since building and maintaining data centres are expensive. Video streaming services such as Netflix and YouTube use buffering which loads video data before playing the video in order for continuous playback. This cannot be done for live cloud gaming so other solutions to improve experiences of latency sensitive games must be explored.

2.4.1 Software-Defined Networking

As stated by Kirkpatrick [12], software-defined networking (SDN) is a new networking architecture that allows programmers to quickly reconfigure and define network usage. Whilst significant advances have been in other areas of technology, networking has not been able to evolve in the same pace.

Similar to mobile phones shifting to the world of smartphones with the help of APIs (application program interface), in an SDN environment, applications can communicate with network switches through an API. The API can be used to quickly reconfigure the resources of the network to accommodate the needs of the applications being executed. This main benefit of using SDN is also discussed in *Improving network management with software defined networking* [11]. Kim et al mentions that network operators will not need to configure all the network devices individually to make network behaviour changes, but instead make network-wide traffic forwarding decisions. The SDN controller is used for this and would have global knowledge of the state of the network.

SDN consists of two planes, the data and control plane. The data plane also known as the forwarding plane is the part of the network that carries user traffic by forwarding them to the next hop along the path to its destination in accordance to the logic in the control plane. The control plane has command where the traffic is sent by creating the routing tables and is also responsible for managing connections between switches, handling errors and exceptions.

2.4.2 Speculative Execution: Outatime

Another form of latency mitigation that is being explored is Microsoft's Outatime which uses *speculation to enable low-latency continuous interaction for mobile cloud gaming* [14]. This method acknowledges that there will be latency and attempts to reduce its effects since studies have shown that players are sensitive to latency from 75-100ms, decreasing accuracy in shooting games and decreasing their scores by almost up to 50% [2].

Outatime basically predicts multiple possible frame outputs that may appear in the future of the game's render scene on the client side. It has to predict what frames may be needed at least a full end host round trip time (RTT) ahead of time the client actually produces game input controls. So unlike standard cloud gaming where clients receive a response after more than one RTT, Outatime delivers a response immediately since the possible frames are all ready to be displayed.

Outatime was extensively tested

2.5   Related Works

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Chapter 3

Design

This chapter will outline the proposed design of the cloud gaming system as well as the game that will be used to run on the system. Also at a high level, I will discuss the virtual network design that will be used to simulate a cloud data centre as well as the proposed solution to use software-defined networking to reduce latency in the network.

3.1   Cloud Gaming System

As a result of the background research conducted, the cloud gaming system will aim to offload the game engine to the cloud data centre. This includes the game logic, physics and graphics rendering. Due to this, it will leave the client's application to just take control of receiving physical button input and send this to the game server as well as receiving and displaying the game video frames.

It can be seen at Figure 3.1 that multiple players should be able to connect to the cloud server and a new virtual machine should be generated for each player. For each virtual machine, an instance of the game will be executed with the required resources such as RAM and processing power provided by the cloud's resource manager. The resources required can be specified as a template with the parameters already set. This makes sure that each game instance has sufficient resources to run at smoothly.

A design improvement on this is to allow the client to specify the video settings to use such as 1080p resolution at 60 fames-per-second and 720p resolution at 30 frames-per-second. Enabling this option will give the client an option to manually improve their gameplay experience since sending higher resolution frames as well as more of them per second will require higher bandwidth. This also helps the cloud data centre to use different templates for virtual machine resources so they will not be wasted by providing too much and therefore can be allocated to other virtual machines.

The software-defined networking controller will manage the networking inside the data centre. It will have global knowledge of the all switches and the links between them. A load balancing application will be used to make sure that traffic generated with the video streaming is routed appropriately to help keep latency in the network to a minimum. This will be discussed later on section 3.4.
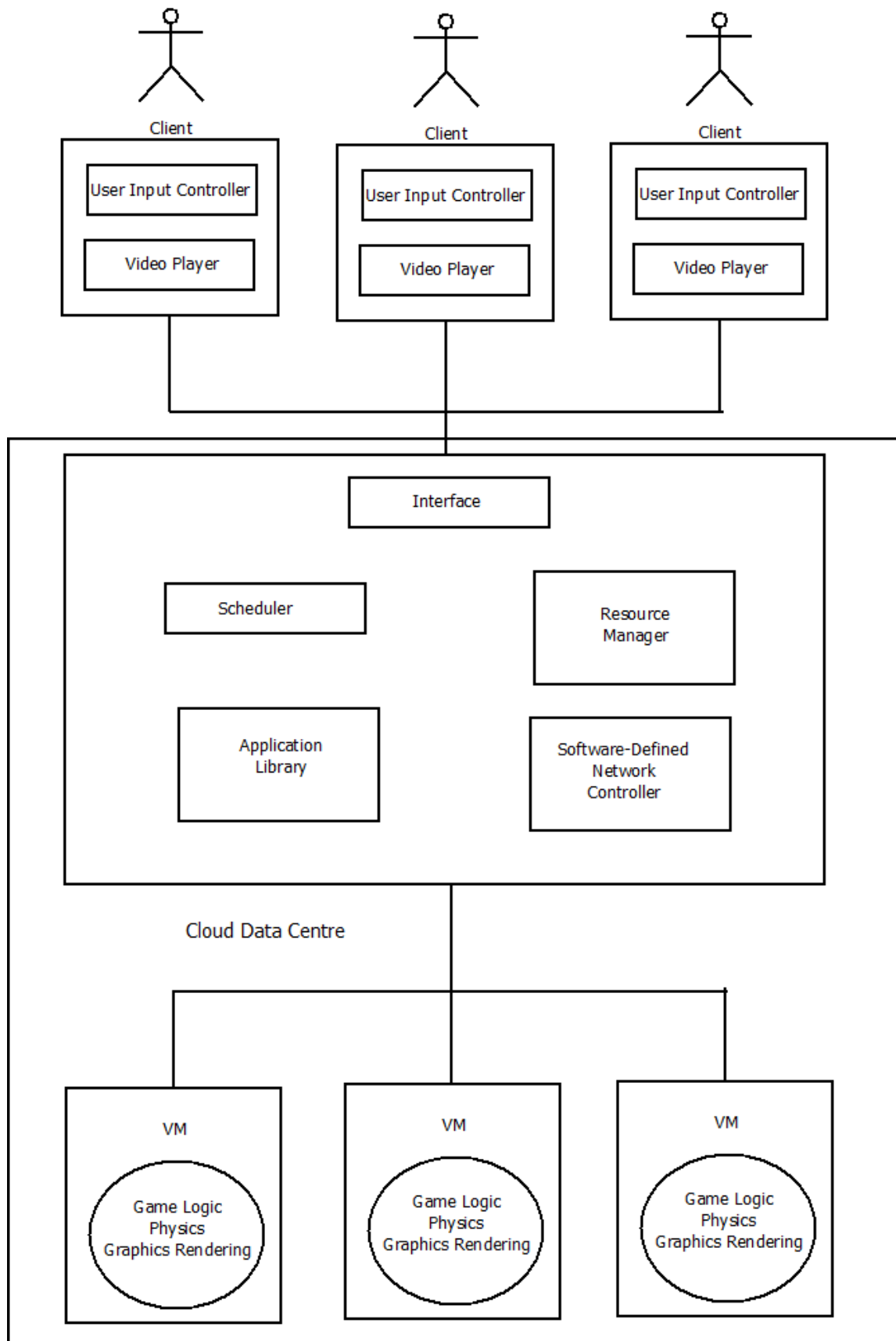
Figure 3.1: Cloud Gaming System Model

As for the framework of the cloud gaming system, it will start with a local program on the client's machine. This program will connect to a server on another machine using its IP address and the port number it listening at. Once a connection is made the server will launch the game and start capturing the rendered frames and encoded them to video to be streamed to the client. The client program will receive the video frames and display them using a video player embedded on the program. It will also be actively waiting for player interaction through the keyboard. The only commands that will register with the client program will be the ones binded for the flight simulator controls. Once the player has pressed one of these, it will send the input to the server where it is processed and simulated as an in game command. Then similarly to traditional games, the game pipeline uses the game command to process the game logic with this new information for rendering updated video frames. This process is displayed in Figure 3.2 as a cycle where the player receives video feedback of the game and responds with keyboard game commands.
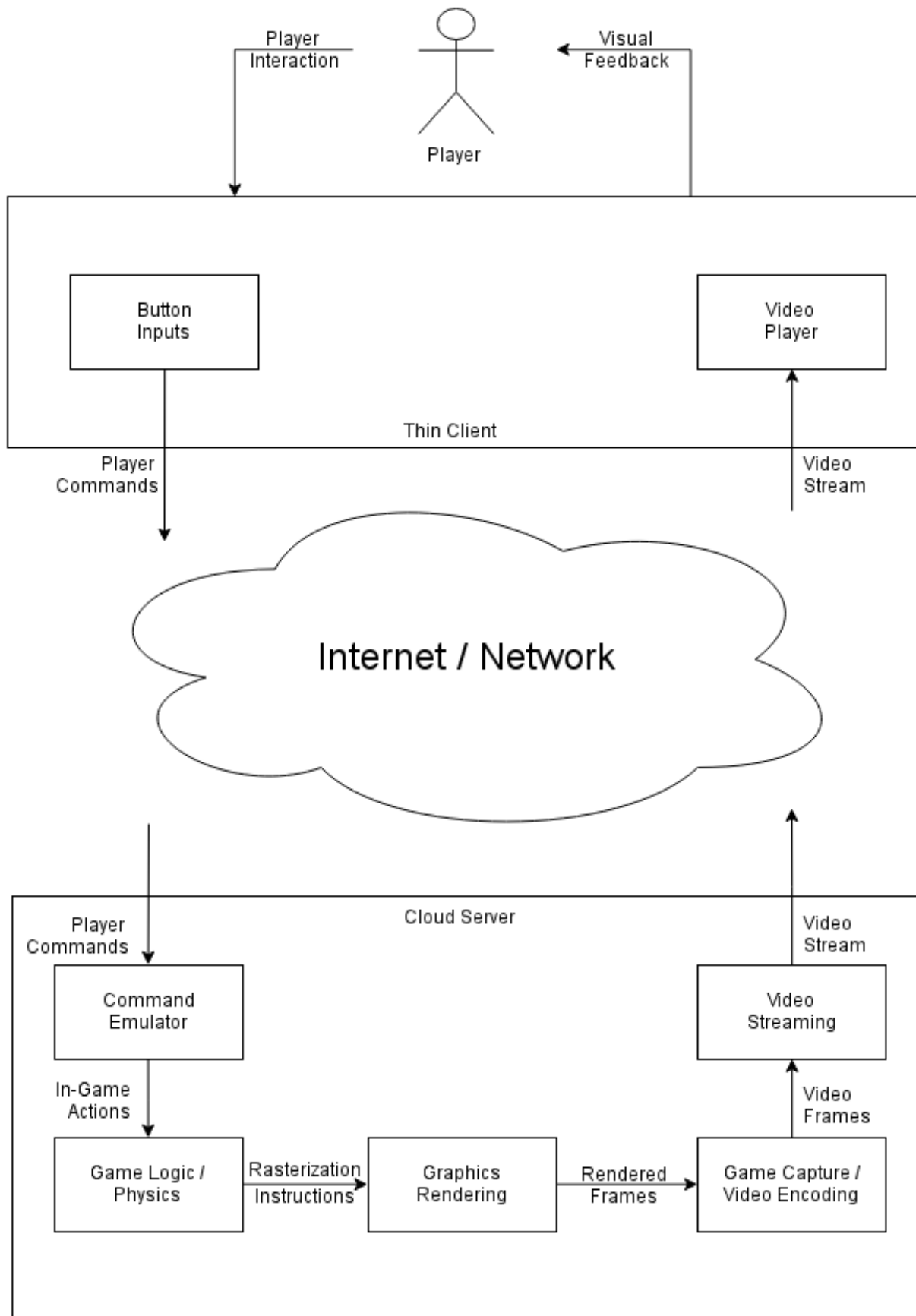
Figure 3.2: Cloud Gaming Framework

3.2 Game Design

In this section I will indicate the design of the game at a high level that will developed to be used as the program to benchmark on the cloud gaming system. The aim of this game is to be somewhat computationally expensive to warrant as a program that will improve by being on a system that is more powerful than a mainstream consumer system. The game should be configurable so it can test out the limits of the system by increasing the amount of computer power it needs. It should also implement a method to measure latency that will arise not just numerically but through user input and immediate visual feedback. The client will experience this lag when a button is pressed, but the expected visual feedback is not being displayed as soon as expected.

To make the game computationally challenging, trees will be modelled and generated in real time. Plants and trees are part of the natural environment and are often used in games to create a realistic scenery. They are geometrically complex and difficult to generate realistic models in real time. Such trees are usually pre-generated and saved locally so it can be loaded easily when needed, but this means the game is limited to using those models and tree models will have to be repeated. This degrades the player's experience as they see duplicate trees in the scenery reminding them that they are in a game which breaks the immersion.

Lindenmayer systems or 'L-systems' is a part of formal language theory to write parallel grammars describing growth similar to the way DNA is a programming language of the human body [19]. Plants tend to have patterns in their growth, but fundamentally they grow forward, rotate then branch out in a hierarchy starting from the root. L-systems are stated as production rules and correspond to each stage of growth for each part of the plant according to a fixed pattern. The notation in the pattern can be simplified to:

- F : move forward and draw

- $+,-$ θ : rotate θ around x-axis

- $\&,\wedge$ θ : rotate θ around y-axis

- $/,\backslash$ θ : rotate θ around z-axis

- [,] : push / pop

Using the symbols above, grammars for tree generation can be produced that will specify the pattern to be followed. The symbols '[' and ']' which represent push and pop respectively refer to a Last In First Out stack. When the symbol '[' is reached, the current position and angle is saved and are restored when the symbol ']' is encountered. Variations can be implemented through randomized parameters such as random angles

for rotation and length of branches. For more complex games, environmental factors can be used to determine the parameters such as competition for light, food, diseases and animals [3]. For the game produced from this project, random rotation angles will be used to generate variation. The production rules below are recursive and will produce two branches for each branch on each recursion.

$$Trunk \rightarrow F[+\theta/\theta Branch][-\theta \backslash \theta Branch]$$

$$Branch \rightarrow F[+\theta/\theta Branch][-\theta \backslash \theta Branch]$$

To better illustrate how a tree is produced at each recursion, Figure 3.3 shows a two dimensional version. The production rules are based on the Pythagoras tree which is a tree constructed from squares and is named due to each triple of squares enclose a right angle triangle. The tree starts with one square and with each recursion two squares are used for each square from the previous recursion. As shown in Figure 3.3, the amount of branches grows in size and complexity pretty quickly, in fact it grows exponentially where the total branches equal to $2^n - 1$ with n being the number of recursions.
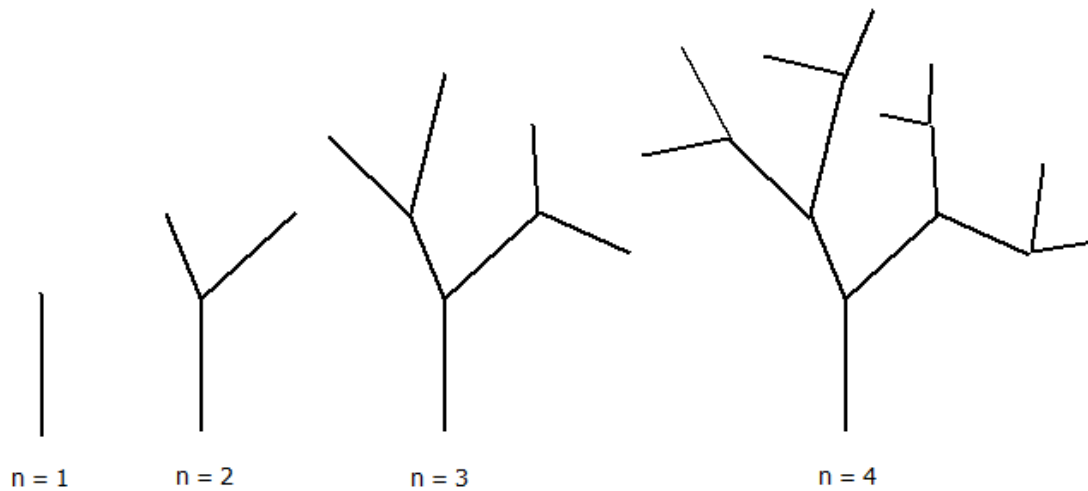

Figure 3.3: 2D Pythagorean L-system tree

Computing the vertices of these many branches in real time will be expensive and adding lighting and shadows will add even more to the resource requirement. Projective shadows is a shadow technique that casts shadows by point light sources onto planes. It takes advantage of a shadow matrix that uses perspective transformation to render an object on to the plane depending on the position of the light source [8]. The problem with this is that it doubles the computation since the object is being rendered twice; the actual object and its shadow. To measure the performance, a frames-per-second counter should be used. The higher the number the FPS counter shows, it means the higher performance the system is capable of since it can push more frames at a given time which means the player will have a smoother gameplay experience. An important factor that should also be considered is the variance in the framerate. A steady framerate

should be the aim as frame drops would cause stuttering.

In order for latency to be evident when playing the game, a form of interaction needs to implemented. A simple flight simulator was chosen so the user can traverse the game and be a way to view the tree model from different angles. The flight simulator will support basic controls such as acceleration and deceleration and roll, yaw and pitch movement. Latency will be easily detected when the user inputs a command then the game's camera doesn't move immediately.

3.3   Virtual Network

Due to limitations with the School of Computing cloud testbed, the proposed solution of software-defined networking to mitigate latency will not be able to be conducted. An alternative solution is to simulate the video traffic that would have been using the data centre's network. This can be done through virtual networks where it can run real kernel, switch and application code on a single machine. Simulating a network also means that switched and links can be easily configurable for software-defined networking controller to work and to simulate real world factors such as latency and bandwidth limitations.

Figure 3.4 shows a basic topology of a cloud network that is currently in use by three different clients. Each client is connected to a single switch which is the core switch acts as an entry point to the data centre. The core switch is linked to other switches called aggregation switches and is eventually linked to a Top of Rack (ToR) switch which is connected to the cloud server host machine running multiple virtual machines. This network design is called fat tree topology. A tree topology is used since it is commonly used in data centres for its good scalability, accessibility, cost effectiveness and low latency [25]. The advantages of using fat tree instead of a binary tree is that multiple paths will be available between nodes which improves fault tolerance and increases inter-rack bandwidth [1]. The main disadvantage that has been considered with using this topology is its single point of failure at the root switch. If this core switch has a fault the entire tree network collapses. Solving this problem is beyond the scope of this project and this topology is used mainly for the fact that multiple paths will be available between two nodes.

Simulated video traffic will be generated at a cloud game server source and sent to the client machines. Large amounts of video data will have be to be transmitted which makes the network links prone to congestion. Latency will be simply measured by pinging between host machines while network is under load.
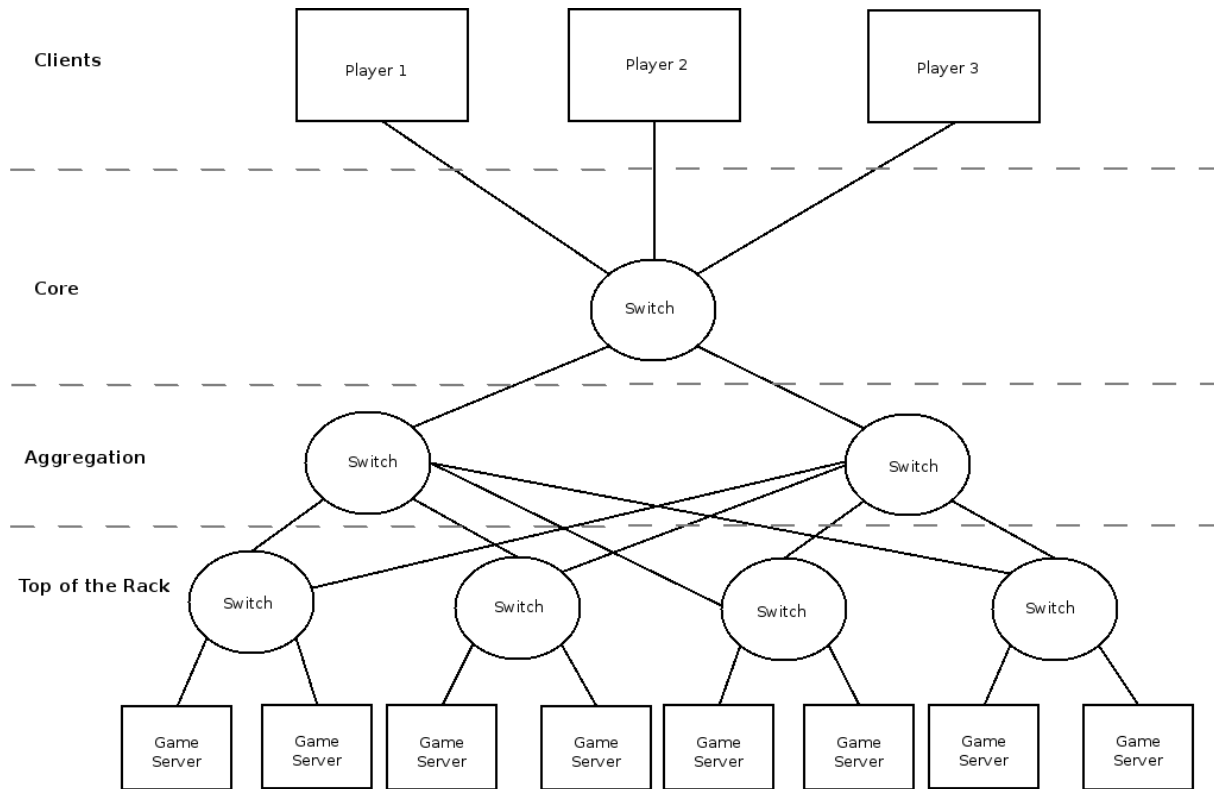
Figure 3.4: Virtual Network Topology

3.4   SDN Application Design

In a traditional cloud gaming system, several methods allow optimal paths in the network to be determined but use link status and parameters provided when the network was set up. Current states of network links when in use such as the amount of bandwidth currently available are not considered. The software-defined network application that will be used to control network traffic through a centralized network controller will aim to lower the latency. This controller will have global knowledge of the network by listening on a certain port where switches can set up a connection. It will use the information collected from switches to compute the best route for packets.

A load balancing concept aims to evenly spread traffic occurring in the network by making use of available paths between a source and destination. This in turn should lower the congestion in network links and reduce variance in network load so the user will experience less stuttering and delay. In order to find optimal paths, the shortest paths between two nodes needs to be computed. The algorithm that will be used to do this is Dijkstra's algorithm which will result in finding all the paths of the shortest length to reduce the inspection for load statistics. The path with the least load is determined from the set of all shortest path. The load will be computed by querying the switches for the amount of packets received and transmitted. The flows for the path with the lowest cost is pushed to the flow tables of all the switches in that path so new

packets will know to be routed using that path.

The main difference with this method is unlike traditional networking this process will be executed repeatedly so links costs are dynamically updated which leads to the path between the two nodes to always be the optimal path for every communication. The load balancer should only update about every minute so it will not use a lot of computation resources. This load balancing solution will make sure that all possible paths are being utilised evenly so no path is heavily congested. Using the virtual network topology above, improvement in latency should still show while performing transmission of simulated video traffic. Better results may show when using this solution with large data centre networks since more paths will be available, but it means the SDN application will have more computation to do as there will be more links and nodes that will have to be factored.

The load balancer application will communicate with the SDN controller by utilising its Northbound APIs. This will be done using REST calls that are already predefine with the SDN controller [21]. Parameters and information can be sent using XML format and responses can be received using JSON format due to lower overhead that comes with these serialized formats. As shown in Figure 3.5, the SDN network controller takes the information from the load balancer application and processes this in to instructions that can be sent to the network elements through the Southbound APIs communication protocols. OpenFlow is an example of a software adapter that the SDN controller can use to communicate with the switches [22].
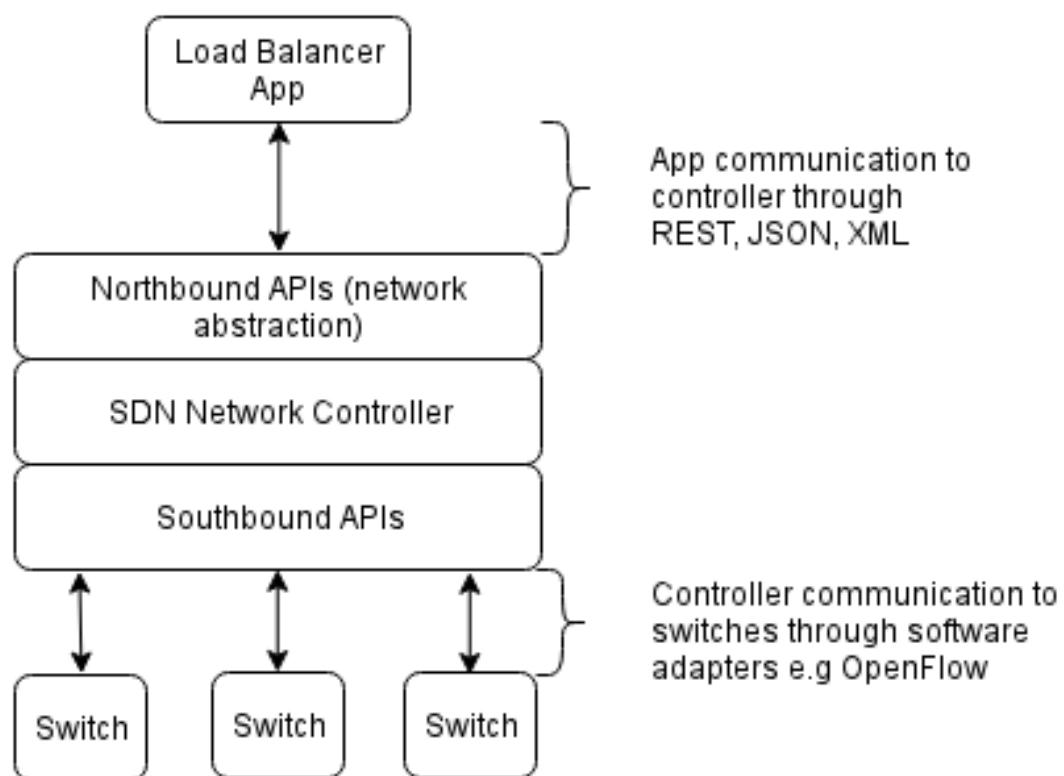
Figure 3.5: SDN Basic Architecture

Chapter 4

Implementation
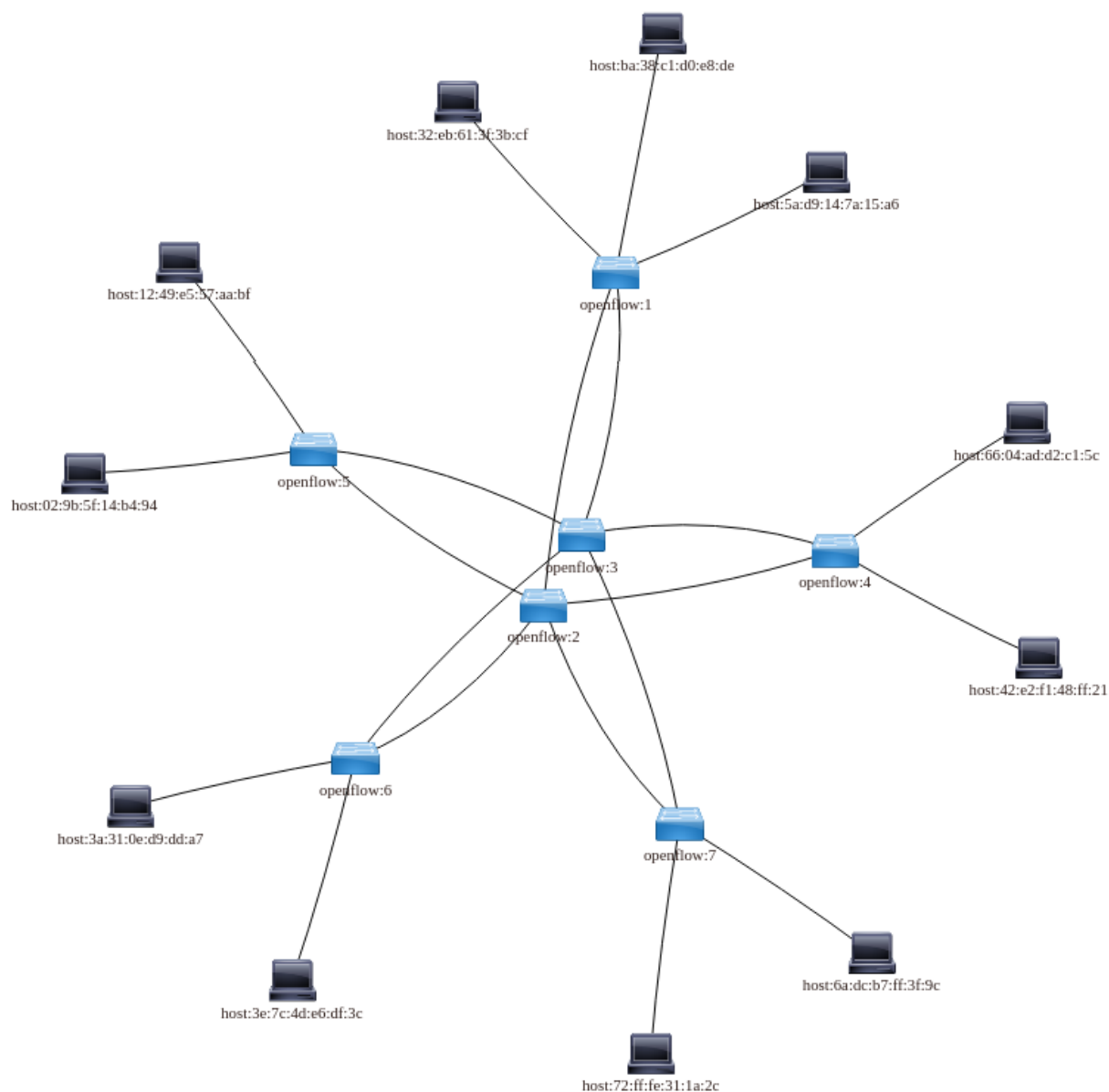
4.1 Implementation

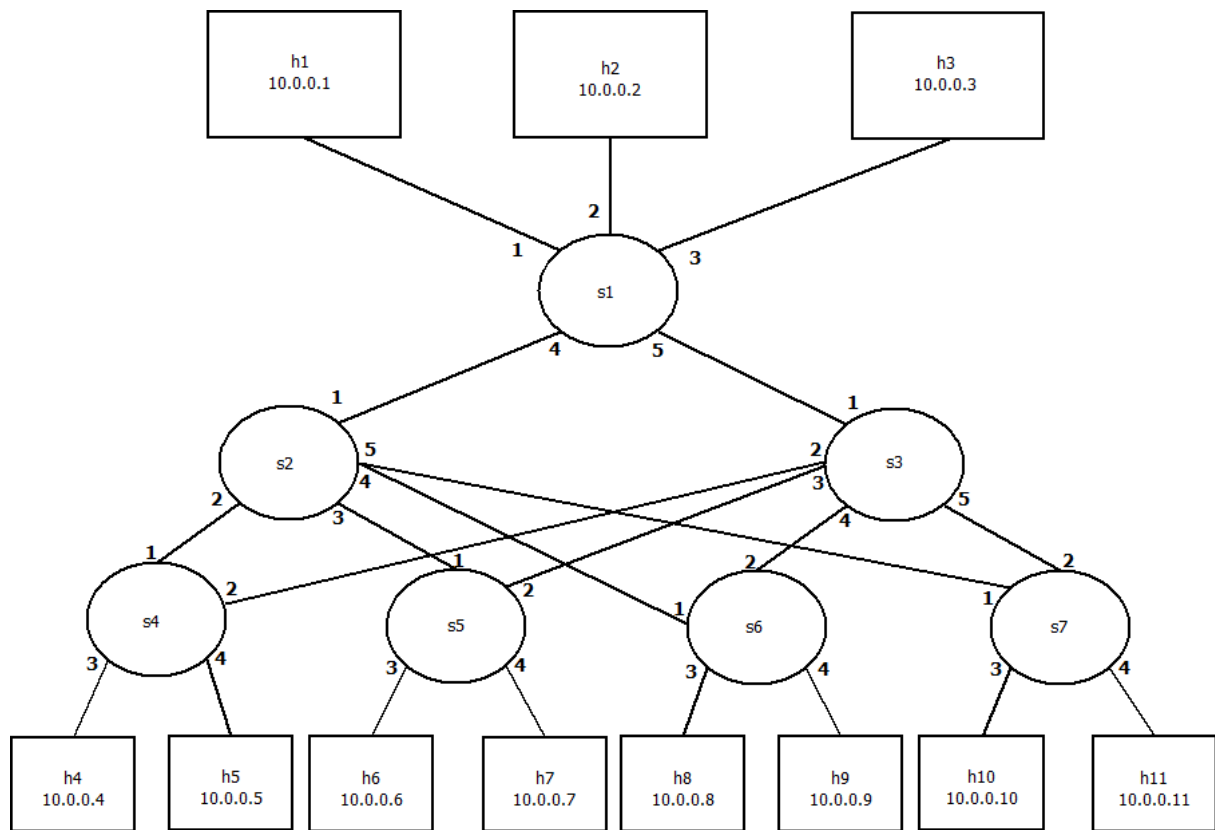Figure 4.1: OpenDaylight network topology

Figure 4.2: Mininet network topology with port numbers

Chapter 5

Testing and Evaluation

5.1 Test Cases

In order to test the load balancing application, test scenarios were designed to see how it would effect real world cases. Using the virtual network created with the mininet program and the iperf network bandwidth measurement tool, traffic is generated between the player hosts and game server hosts. The bandwidth of this traffic will be based on network requirements suggestions for 1080p at 60FPS, 720p at 60FPS and 720p at 30FPS given by nVidia's GeForce Now cloud gaming system requirements [17] as shown in Table 5.1. The iperf tool used to generate the traffic will set the player hosts in server mode and the game servers in client mode. This is so the game server can send UDP traffic to it's connected player host machine. UDP is used since its traffic bandwidth can be set and is the protocol that is used for streaming video data over TCP.

Latency will be simply tested by sending 10 ping requests from player host to game server whilst the iperf traffic is being sent. Each test case will contain multiple tests using each of the settings in Table 5.1 with tests before load balancing and then with load balancing in effect. To make sure th tests without load balancing doesn't depend on routes specified by previous load balanced runs, flows created by previous load balanced runs are deleted using a python script that makes REST API calls to the OpenDaylight SDN controller.

Video Settings	Network Bandwidth
1080p + 60FPS	50 Mbps
720p + 60FPS	30 Mbps
720p + 30FPS	10 Mbps

Table 5.1: Traffic Simulation Settings

5.1.1 Case 1

The first case is two players using the cloud gaming system with their game instance being in two separate servers but under the same switch. Player host 1 'h1' (10.0.0.1) and player host 2 'h2' (10.0.0.2) are used and are connected to 'h4' (10.0.0.4) and 'h5' (10.0.0.5) respectively. This shows a basic example of multiple players connected to different game servers but using the same Top of the Rack switch. As shown in Figure

5.1, there are two possible routes for traffic flowing from the core switch 's1' to switch 's4' (1::2::4 and 1::3::4). When the load balancer is running for each flow, it will assign a different route for each of them. Figure 5.1 shows one out of the two possible load balanced routes with the other being just a swap with the red and blue path between 's1' and 's4' switches.
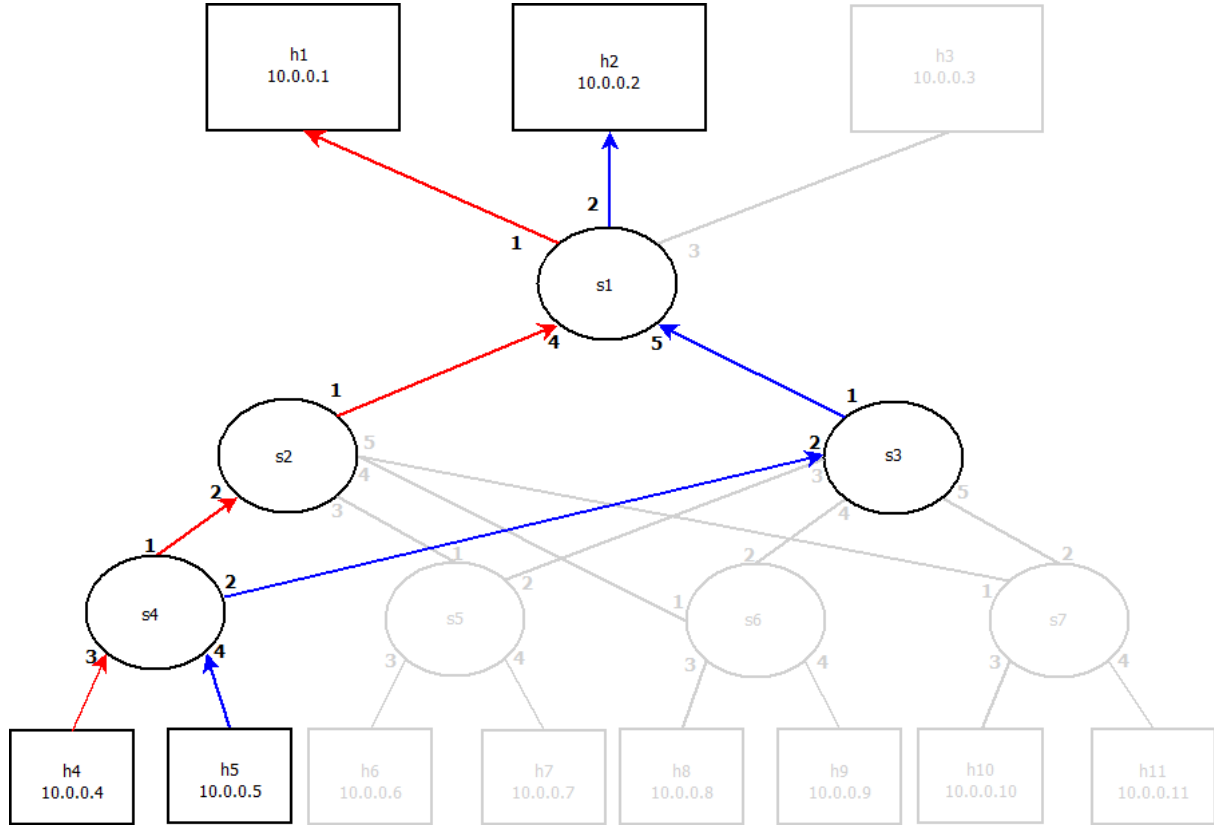

Figure 5.1: Test Case 1: Possible load balanced routes

5.1.2   Case 2

The second case is when two players are connected to the same physical game server host since it can contain multiple virtual machines with multiple game instances. Also it can represent two players playing the same game instance such as a multiplayer game so two video traffic will be needed to be generated and sent to the two separate clients. For this test case, hosts 'h1' and 'h2' will be used to be connected to host 'h4'. The possible load balanced routes between the switches will be the same as in test case 1 but with two separate traffic being generated from 'h4' as shown in Figure 5.2.

5.1.3   Case 3

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque.
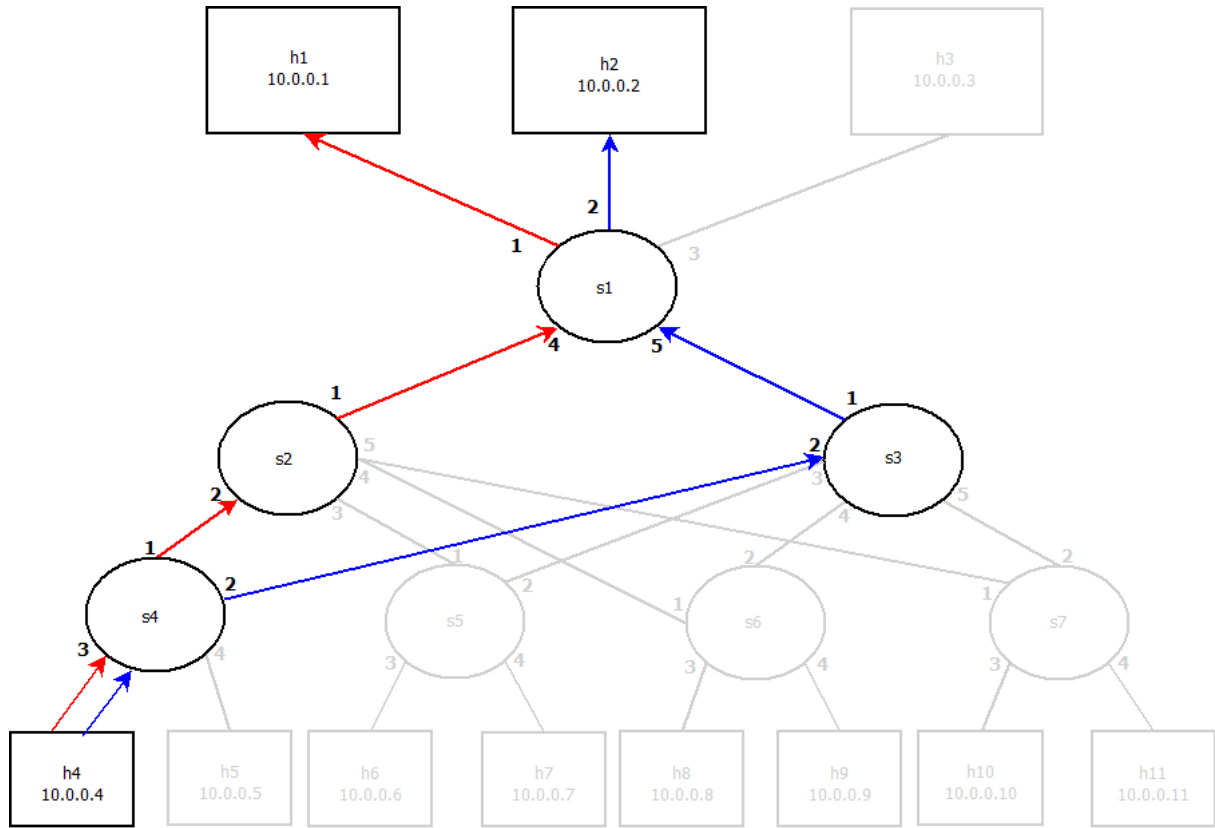
Figure 5.2: Test Case 2: Possible load balanced routes

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

5.2 Results

5.3 Evaluation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla.
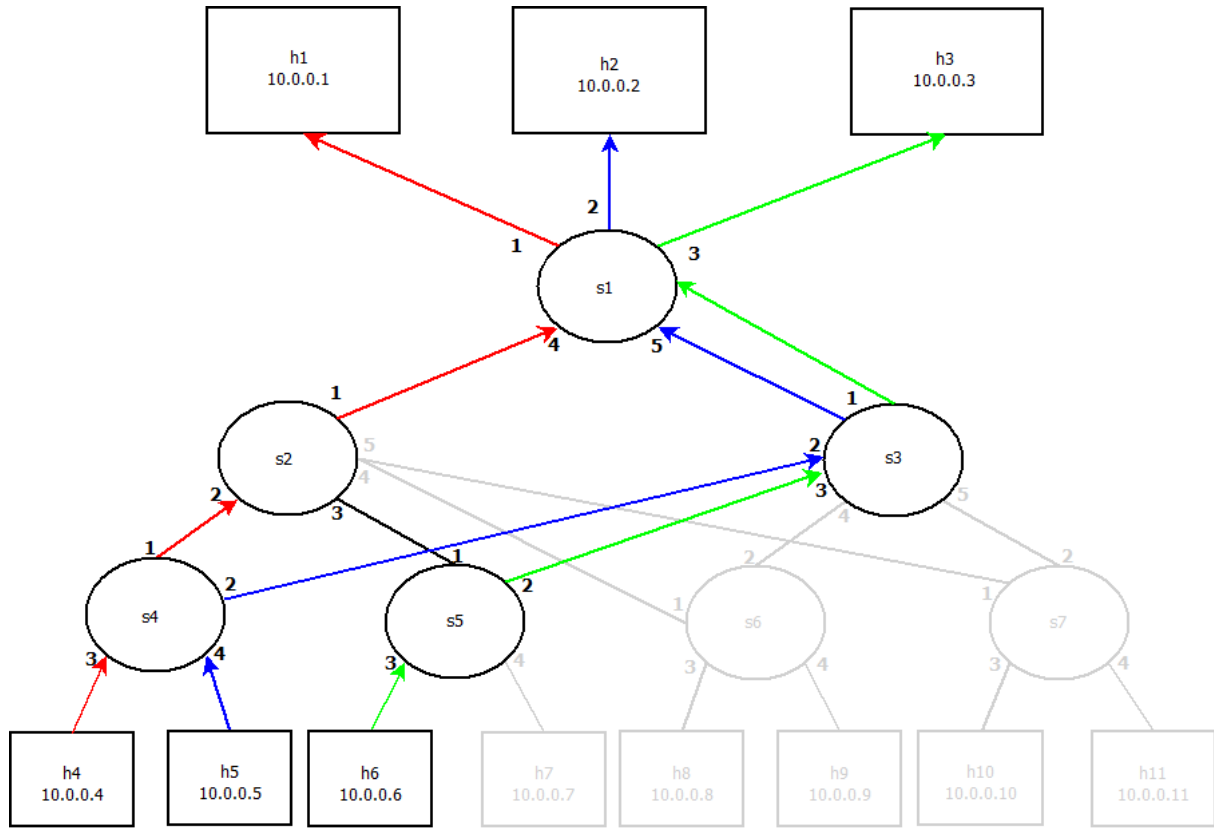
Figure 5.3: Test Case 3: Possible load balanced routes

Connection	Before Load Balancing				With Load Balancing			
	min	avg	max	mdev	min	avg	max	mdev
50 Mbps								
h1 ->h4	210.317	223.223	244.866	9.440	37.409	40.268	47.533	2.685
h2 ->h5	278.072	280.276	281.873	1.522	265.200	269.559	274.839	2.182
30 Mbps								
h1 ->h4	57.012	75.600	108.208	15.111	36.755	37.891	41.927	1.428
h2 ->h5	274.877	280.685	286.498	4.096	36.628	38.28	45.249	2.373
10 Mbps								
h1 ->h4	41.814	43.108	46.786	1.469	37.211	38.107	39.057	0.596
h2 ->h5	42.486	43.516	44.502	0.675	37.550	38.200	39.239	0.653

Table 5.2: Test Case 1 Results

Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Connection	Before Load Balancing				With Load Balancing			
	min	avg	max	mdev	min	avg	max	mdev
50 Mbps								
h1 ->h4	268.850	296.824	320.158	18.250	209.526	213.216	216.632	3.011
h2 ->h4	357.026	377.026	404.797	20.273	272.451	274.735	275.937	1.167
30 Mbps								
h1 ->h4	213.000	221.689	229.219	6.238	38.468	39.232	39.960	0.448
h2 ->h4	296.504	302.919	321.211	7.733	38.525	39.243	39.995	0.469
10 Mbps								
h1 ->h4	42.832	43.443	44.234	0.484	39.985	38.016	38.572	0.432
h2 ->h4	41.622	42.396	44.543	0.936	36.992	38.007	39.062	0.739

Table 5.3: Test Case 2 Results

Connection	Before Load Balancing				With Load Balancing			
	min	avg	max	mdev	min	avg	max	mdev
50 Mbps								
h1 ->h4	208.303	216.48	220.662	4.465	37.631	40.998	49.488	3.256
h2 ->h5	269.297	273.377	277.458	4.113	269.133	272.394	274.474	2.096
h3 ->h6	426.205	435.244	443.197	6.979	421.068	424.151	425.826	1.589
30 Mbps								
h1 ->h4	205.742	210.754	216.244	3.768	36.542	38.055	40.507	1.168
h2 ->h5	271.333	273.377	278.936	2.629	36.397	37.288	38.337	0.657
h3 ->h6	425.217	427.494	431.469	2.499	425.185	426.292	430.231	1.489
10 Mbps								
h1 ->h4	41.517	42.602	44.918	0.966	37.082	38.156	40.023	0.975
h2 ->h5	41.602	43.132	44.731	0.943	37.598	38.386	38.999	0.387
h3 ->h6	426.631	428.373	430.332	1.518	36.981	37.702	38.944	0.523

Table 5.4: Test Case 3 Results

Chapter 6

Conclusion

6.1 Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.2 FutureWork

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.3 Personal Reflection

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est,

iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review*, 38(4):63–74, 2008.

[2] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The effects of loss and latency on user performance in unreal tournament 2003®. In *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 144–151. ACM, 2004.

[3] H. Carr. 15: Natural phenomena: Terrain & plants. University Lecture, 2015.

[4] S.-P. Chuah, C. Yuen, and N.-M. Cheung. Cloud gaming: a green solution to massive multiplayer online games. *Wireless Communications, IEEE*, 21(4):78–87, 2014.

[5] P. Deutsch. The eight fallacies of distributed computing. *URL: http://today. java. net/jag/Fallacies. html*, 1994.

[6] GameCentral for Metro.co.uk. The witcher 3 wins game of the year at golden joysticks 2015, 2015. [Online]. [Accessed 15 April 2016]. Available from: http://metro.co.uk/2015/10/30/the-witcher-3-wins-ultimate-game-of-the-year-at-golden-joysticks-2015-5471812/.

[7] R. L. Grossman. The case for cloud computing. *IT professional*, 11(2):23–27, 2009.

[8] K. Hawkins and D. Astle. *OpenGL game programming*. Cengage Learning, 2001.

[9] Y. Jadeja and K. Modi. Cloud computing-concepts, architecture and challenges. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, pages 877–880. IEEE, 2012.

[10] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld. An evaluation of qoe in cloud gaming based on subjective tests. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pages 330–335. IEEE, 2011.

[11] H. Kim and N. Feamster. Improving network management with software defined networking. *Communications Magazine, IEEE*, 51(2):114–119, 2013.

[12] K. Kirkpatrick. Software-defined networking. *Communications of the ACM*, 56(9):16–19, 2013.

[13] A. Kumawat. Cloud service models (iaas, saas, paas) + how microsoft office 365, azure fit in, 2013. [Online]. [Accessed on 22 April 2016]. Available from: http://www.cmswire.com/cms/information-management/cloud-service-models-iaas-saas-paas-how-microsoft-office-365-azure-fit-in-021672.php.

[14] K. Lee, D. Chu, E. Cuervo, J. Kopf, A. Wolman, Y. Degtyarev, S. Grizan, and J. Flinn. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. *GetMobile: Mobile Computing and Communications*, 19(3):14–17, 2015.

[15] B. Mariano and S. G. Koo. Is cloud gaming the future of the gaming industry? In *Ubiquitous and Future Networks (ICUFN), 2015 Seventh International Conference on*, pages 969–972. IEEE, 2015.

[16] P. Mell and T. Grance. The nist definition of cloud computing. *NIST*, 2011.

[17] nVidia. What are the system requirements for geforce now? | shield, 2016. [Online]. [Accessed on 4 May 2016]. Available from: https://shield.nvidia.co.uk/support/geforce-now/system-requirements/1.

[18] D. Piner. Witcher III system minimum and recommended system requirements and upgrade guide, 2016. [Online]. [Accessed 15 April 2016]. Available from: http://www.tentonhammer.com/guides/witcher-iii-system-minimum-and-recommended-system-requirements-and-upgrade-guide.

[19] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[20] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol, 1998. [Online]. [Accessed 20 April 2016]. Available from: https://www.ietf.org/rfc/rfc2326.txt.

[21] SDxCentral. What are sdn northbound apis?, 2015. [Online]. [Accessed on 1 May 2016]. Available from: https://www.sdxcentral.com/resources/sdn/north-bound-interfaces-api/.

[22] SDxCentral. What are sdn southbound apis? - where they are used., 2015. [Online]. [Accessed on 1 May 2016]. Available from: https://www.sdxcentral.com/resources/sdn/southbound-interface-api/.

[23] R. Shea, J. Liu, E. Ngai, and Y. Cui. Cloud gaming: architecture and performance. *Network, IEEE*, 27(4):16–21, 2013.

[24] A. Soomoro. What is input lag: The breakdown, 2013. [Online]. [Accessed 21 April 2016]. Available from: http://www.displaylag.com/what-is-input-lag-the-breakdown/.

[25] C. Wu and R. Buyya. *Cloud Data Centers and Cost Modeling: A Complete Guide To Planning, Designing and Building a Cloud Data Center.* Elsevier Science, 2015.

Appendices

Appendix A

External Material

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Appendix B

Ethical Issues Addressed