National College of Ireland

A Comparative Study on Cloud Gaming performance using Traditional, Containers in Fog Nodes, and Edge-enabled Shared GPU architectures

MSc Research Project
Cloud Computing

Sabesan Muralikrishnan
Student ID: x20149581

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sabesan Muralikrishnan
Student ID:	x20149581
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	16/12/2021
Project Title:	A Comparative Study on Cloud Gaming performance using Traditional, Containers in Fog Nodes, and Edge-enabled Shared GPU architectures
Word Count:	6769
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sabesan Muralikrishnan
Date:	16th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Comparative Study on Cloud Gaming performance using Traditional, Containers in Fog Nodes, and Edge-enabled Shared GPU architectures

Sabesan Muralikrishnan
x20149581

Abstract

The demand for cloud gaming and the user base for the same has increased drastically following various advancements in the field. After the introduction of cloud computing and its continuous evolution, various new technologies like Fog Computing, Edge computing were developed. These advancements are leveraged to overcome all fallacies and pitfalls in the traditional cloud gaming solution. Since the introduction of Edge Computing, major changes happened in the mobile network region. The new 5G technology allows users to experience a high-speed internet connection in their mobile devices. While the Edge Nodes are primarily used to collect raw data and share them with the cloud server for processing, in this research project, one more implementation for Edge Computing is inspired and proposed as a cloud gaming model. We have leveraged Edge Computing nodes, powered by virtual GPU, coupled with WebRTC streaming, in an attempt to improve the Quality of Experience and Quality of Service. The game commands will be sent to the cloud server through Edge Node. Once the game video is rendered at the cloud server, it will be encoded on an Edge Node closer to the user. This encoded video will be then streamed as real-time video onto the player's browser. The communication will pass only through the Edge Node without any additional hops. The evaluation result proves that the proposed solution maintains a constant frame rate, frame delay time below 10 ms, jitter rate below 10, and 30% reduced bandwidth consumption, thus improving the overall QoE and QoS.

1 Introduction

Cloud computing and related services have improved and optimized over various researches done by authors, students, and organizations. All these updates enable cloud computing providers to improve their service offerings. The world is shifting towards a digitization paradigm where every second of data is being gathered, analyzed, transformed, and stored. Cloud computing plays a major role in this digitization paradigm shift. Based on a survey taken during 2020, 541 hyper-scale data centers were registered across the globe. Many cloud providers leverage this increase in data centers and available resources to improve their service quality to attract more customers. One of the most profitable cloud offerings is providing Gaming as a Service. At the beginning of cloud gaming, cloud resources and infrastructure is provided to deploy game builds. This is later updated with a stable client-server architecture called the 'Traditional Architecture'.

While this approach is simple and straightforward, there are various factors that affect the quality of game-play experience and service offered by the cloud server (Cai et al.; 2014). Addressing and solving these factors created more research opportunities for students, authors, and cloud service providers. Every improvement in cloud computing and related sub-domains are leveraged and experimented to optimize and improvise the QoE (Quality of Experience) and QoS (Quality of Service) of a cloud game service.

In this research project, three different cloud gaming architectures are implemented, compared, and evaluated. The first architecture is Traditional Architecture. In this method, the game binary is deployed on the cloud server. The user will send game commands to the cloud server. Upon receiving the game commands, the cloud server will execute the game logic and render a video output. This video output is later encoded and streamed back to the player's console. The second architecture is Fog-enabled cloud game architecture. This approach leverages the Fog Computing advancements with help of container images. In this approach, non-gaming tasks like video encoding and streaming are transferred offloaded to the Fog Node that is closer to the user. This approach reduces latency and improves QoE.

The third architecture is an experimental architecture proposed in this research paper to improve both Qualities of Experience by reducing the latency, and Quality of Service by reducing the bandwidth consumption. In this architecture, game commands will be sent to the internet/ network service provider tower station from the user browser (Aguilar-Armijo; 2021). Edge computing resources deployed in these tower stations will be leveraged for video encoding and streaming. The game commands are sent directly to the cloud server from the Edge node without any Fog Node layer in between. Once the game scene is rendered by the cloud server, these packets are sent to the Edge node. The Edge node will use WebRTC protocol to stream the game video back to the user (Xu et al.; 2018). In addition to that, the Edge Node deployed will be powered by Nvidia GRID for hardware encoding support. Elimination of Fog Node layer between Edge and the Cloud server, Enabling virtual GPU for Edge Node video encoding, Reducing the communication distance from the remote cloud server to the local Edge node, introducing a peer-to-peer network for real-time video streaming, the proposed architecture is expected to produce improved QoE and QoS.

The following content of this project report is structured into multiple sections as follows. Section 2 details some related work and comparison between them and proposed system. Section 3 explains the project methodology involved in implementing all three architectures and evaluating them. Section 4 explains the required design specification to implement this project. Section 5 explains the steps involved in implementing each architecture along with the workflow overview. Section 6 details on how the assumed parameters were evaluated with proof and a summary discussion. Section 7 summarises the project findings and possible future research works.

Research Questions:

What is the difference in QoS, and QoE of Cloud Gaming, hosted using a Traditional, an Experimental Container in Fog Nodes, and a proposed Experimental architecture?

Can the QoE and QoS of the cloud gaming service be improved by leveraging Edge Computing service, allocating vGPU resources, and streaming

game video with a peer-to-peer communication?

2 Related Work

2.1 Traditional Cloud Gaming

Authors (Cai et al.; 2014) conducted a series of experiments with different cloud gaming services and cloud gaming platforms. In this experiment, they mainly focus on evaluating the Gaming as a Service paradigm and produce their findings. Based on their experiments involving games of varied genres and quality, each GaaS service is benchmarked based on its performance. Also, towards the end, the authors suggested new research ideas and comparative studies for cloud gaming and related technologies.

An experiment cloud gaming testbed is developed by authors (Huang et al.; 2013). This solution was developed to primarily test the traditional cloud gaming architecture. The performance of the cloud gaming infrastructure can be evaluated by deploying this testbed with sample 2D and 3D games. The project was published open-source for researchers, students, cloud gaming service providers to implement and evaluate their novelty idea.

2.2 Fog-enabled Cloud Gaming

In the distributed computing world, maintaining one global server to host the game is geographically not considered as an optimum solution. In general, the cloud computing infrastructure is divided into Mobile Cloud, Edge Computers, Fog Nodes, and Cloud Servers/ Data Centers. With the advancement in this new distributed computing infrastructure, authors (Lin and Shen; 2015) proposed a fog-enabled cloud gaming system. This solution will offload certain tasks to the fog node closer to the user location. This experiment proves to reduce the overall latency of the cloud gaming session. As a result, the QoE of the game-play was improved with this proposed method.

In addition to that, authors (Lin and Shen; 2016) proposed an improvement to the existing Cloud Fog solution where the Quality of Service is improved by reducing bandwidth consumption. The previous version proposed by authors (Lin and Shen; 2015) improved the Quality of Experience by reducing the latency alone. In this version, super-nodes (fog networks containing super-nodes) were assigned to each user for game video rendering and streaming. This process will offload the video render, streaming, and encoding tasks from the cloud server and help to produce better throughput from the cloud server.

2.3 GPU Virtualization in Cloud Gaming

The performance of cloud gaming depends on the effective allocation of hardware and software resources to power the computation tasks. Since the introduction of cloud images for operating systems, software virtualization reduced the duplication of hardware resources by virtualizing and distributing them as shared resources. After the introduction of Nvidia GRID, authors (Xu et al.; 2018) conducted an experiment to find the percentage improvement in QoE by increasing the frame per second rate on cloud media functions by leveraging Nvidia Grid virtual GPU services.

While the idea of adding more resources (GPU resources in particular) for better cloud gaming performance is straightforward, the underlying cost and energy consump-

tion will increase exponentially. Hence it is important to employ an effective GPU resource scheduler that assigns virtual GPUs to cloud nodes that runs the game logic. Author 10.1145/2632216 proposed an effective GPU resource allocation platform called VGRIS intended to improve cloud gaming experience and service quality.

2.4 Edge and Cloud Computing

Authors (Chen et al.; 2018) devised an efficient healthcare system by leveraging Edge Cognitive Computing in addition to the existing cloud computing resources. In the existing health care system, not all patients get their respective care and individual resources allocated to them. This proposed system leverages edge computing to improve the overall Quality of Service by assigning dedicated resources to each patient care unit. The data collected from these Edge nodes will be later transferred to the cloud server for analysis and storage.

In another experiment, authors Zhang et al. (2019) proposed a methodology to improve the cloud gaming QoE by enabling Edge Computing capabilities as a part of their proposed architecture. As a result of this experiment, they proposed the EdgeGame cloud gaming platform. This platform is proposed to improve the QoE by 20% of cloud gaming by reducing network latency by 50%

2.5 WebRTC Advancement

In every cloud game solution, user inputs will be transmitted to the cloud server and the resultant game scene will be streamed back to the user as a game video. With the advancement in media streaming technology, authors (Chen et al.; 2019) leveraged WebRTC for game video streaming. This experiment utilizes Google's STUN and TURN servers to register an external IP address. This system implements GPU virtualization by consuming after-market GPU drivers instead of using vGPU services like Nvidia GRID for cost efficiency.

The increasing need for interactive and real-time cloud gaming experience made many cloud computing service providers invest more in research regarding this. As a result of that, Google release its cloud gaming platform in 2019. Author (Carrascosa and Bellalta; 2020) conducted a detailed analysis of Google stadia and its performance. From the experiment, it is apparent that Google leverages WebRTC for real-time and interactive web game-play experience.

Table 1: Comparison with previous work done on cloud gaming comparative studies and optimizations

Approach	**QoE**	**QoS**	**Edge Enabled**	**WebRTC Support**
(Cai et al.; 2014)	No	No	No	No
(Lin and Shen; 2016)	No	Yes	No	No
(Lin and Shen; 2015)	Yes	No	No	No
(Xu et al.; 2018)	Yes	Yes	No	Yes
(Chen et al.; 2018)	No	Yes	Yes	Yes
Zhang et al. (2019)	Yes	No	Yes	Yes
(Chen et al.; 2019)	Yes	Yes	No	Yes
Proposed System	Yes	Yes	Yes	Yes

2.6 QoE and QoS Evaluation

Authors (Peñaherrera-Pulla et al.; 2021) conducted a series of comparative analyses on various cloud gaming architectures and open-source frameworks. In this experiment, they have compared cloud gaming platforms that are optimized with GPU resources, Fog-enabled infrastructures, and Edge computing infrastructures. The research focuses on the impact of Quality of Experience on any given cloud gaming architecture and platform. Evaluation metrics and parameters to be considered to compare the video render and streaming quality were discussed in detail.

Before implementing an existing cloud gaming architecture and the proposed architecture, it is vital to understand a major factor that affects the quality experience. Authors (Slivar et al.; 2015) conducted a series of experiments to understand the impact on quality of experience. From their experiments, it is observed that the quality of experience for a multi-player online game can be improved drastically if the video encoding strategy is optimized tailored to each user in the game session. Results obtained from this experiment are used to implement new QoE optimization strategies in cloud gaming infrastructure.

2.7 Comparing Related Work

In Table 1 the proposed architecture and its optimizations were compared with existing cloud gaming architectures, comparative analysis methodologies, and gaming test-beds.

3 Methodology

From the beginning of cloud gaming services, there have been various improvements in underlying architectures. With the raw data gathered while reviewing previous work done on cloud gaming, and the results obtained after the implementation, a detailed evaluation is performed. From the evaluation, the Quality of Service (QoS) and Quality of Experience (QoE) will be determined (Laghari et al.; 2019). In this research project, three different cloud gaming architectures were implemented and evaluated.

3.1 Traditional Architecture:

The first architecture implemented and evaluated in this project is the traditional cloud gaming architecture. At the beginning of cloud gaming services, the game engine and corresponding game logic were deployed onto a centralized server. This server creates a secured game session that is shared with users across the world. Users interact with this game session and provide game commands as input to the cloud server. This input will be processed and corresponding game logic will be executed. The outcome of this game logic will be encoded as a video and streamed back to the user. The performance of this game session depends on various criteria. While this traditional cloud gaming architecture served well for many 2D single-player games, with low graphics settings, new and modern multiplayer online games were not supported by this architecture. Author (Shea et al.; 2013) mentioned that 100 ms delay tolerance is acceptable for cloud gaming users (specifically for First-Person Shooter (FPS) games). Nonetheless, due to a huge increase in the user base for multiplayer online games, this latency should be further reduced.

3.2 Fog-enabled Architecture:

Author (Lin and Shen; 2016) through a series of experiments and analysis of this Multi-player Online Game (MOG) understood that the user base for these games is distributed among various configurations. As a part of this experiment, users were categorized based on the availability of computation resources on their end, distance from the cloud game server, and network speed. The game hosted on the cloud server was tested with increasing user load, distributed among these categories. The result revealed that, users who are far away from the cloud server (geographically distributed) experience more latency and bandwidth issues which result in bad QoE. The solution proposed by authors, (Lin and Shen; 2016) addressed this latency issue by offloading the video encoding process to a Fog Node. This Fog Node will be at the edge of the user's network thus eliminating the latency issue due to log-distance data transfer.

In above sections 3.1 and 3.2, two primary cloud gaming architectures were analyzed. The proposed third architecture in this comparative study is a hybrid architecture.

3.3 Proposed: Hybrid Architecture:

The proposed architecture is a hybrid implementation, combining the advancements through Fog-enabled cloud gaming and the invention of vGPU (virtual Graphical Processing unit). The vGPU solution proposed and provided as a service by (Herrera; 2014) is leveraged in this hybrid architecture. While the Fog-enabled cloud gaming reduces the latency by bringing the video encoding node closer to the user, the video encoding process will consume more computing resources and time. This will include a Fog Node overhead latency in addition to the existing cloud server overhead latency. This new overhead latency on the Fog node end can be reduced by assigning a vGPU core for the Fog node. Once the game scene is rendered as encoded video at the edge node it will be streamed in real-time to the user. In this proposed architecture, WebRTC (Web Real-Time Communication) protocol is used to stream game videos to all active users in a game session. WebRTC, a peer-to-peer real-time communication protocol (Loreto and Romano; 2014) proven to be most reliable to stream multi-casting video on real-time to users (peers) in a network (Jansen et al.; 2018). For this proposed architecture, WebRTC is used for video streaming due to the below-mentioned reasons,

- Being a Peer-to-Peer network, there is no requirement for a central network administrator. This technology will be more suitable to stream real-time game videos to a large number of participants.
- Compared to other streaming services, the WebRTC network is easy to set up, maintain, and better performing.
- Data backup and security are maintained within the network.
- For any personal multi-media application over cloud, peer-to-peer network is proven to be easy to set up.

During the implementation stage of this proposed architecture, various cloud gaming open-source platforms were gathered, evaluated. Below are the various stages defined during the project methodology planning,

- Stage 1: A flexible game engine/ platform is selected. In all comparative studies conducted in the past and reviewed for cloud gaming in section 2.1, the cloud gaming infrastructure and deployment platform were critically analyzed. In this research project, the proposed cloud gaming architecture is implemented by creating a sample 3D and 2D game with custom build WebRTC components to stream game audio and video. As a result of this stage, in this research project, the Unity 3D game engine is selected to develop the sample games which will later be deployed in the proposed architecture. The primary reason for selecting this engine to experiment with is the percentage of the gaming industry covered by the engine.

- Stage 2: A custom-built WebRTC component that will be a part of the game development code. This stage will give the flexibility of streaming the audio and video of the game scene (in Unity terminology, each game scenario/ game-play is referred to as 'scene'). This custom-build WebRTC component is backed up by the open-source WebRTC package developed by Google, Ericsson, and supported by a group of open-source communities (Loreto and Romano; 2014).

- Stage 3: An interactive web-based platform to play the game on any desktop, mobile, and tablet device browsers. As explained in detail in books (Angel and Shreiner; 2014) and (Parisi; 2012), the WebGL technology is platform-independent, built for interactive and dynamic web applications as an improvement over OpenGL. WebGL commands are accessed through JavaScripts-based programming interfaces.

- Stage 4: As a part of the Unity game development code, the nearest MobiledgeX (Aguilar-Armijo; 2021) cloudlet pod will be selected. The video encoding job will be deployed as a container image in this cloudlet. This service is selected as an Edge Computing layer of this proposed architecture after a detailed comparison among other open-source and commercial Edge Computing service providers. With MobiledgeX we can offload the video encoding process to the nearest edge node. This edge node will be selected based on the nearest network service provider tower from the user's location. This node selection logic will be a part of the game development code itself. Once the game logic is executed on the cloud side, raw data and image frames are transferred to the identified close edge node. In this edge node, the game video will be encoded and streamed to the user using WebRTC.

- Stage 5: A virtual GPU processing service offered by Nvidia GRID to accelerate the video encoding process. With this service, the native GPU hardware can be divided into sizable virtual graphics cores. This vGPU core can be then assigned to the container where the video encoding is being performed. While software encoding increases the video encoding quality, the overall speed of the encoding can be improved by assigning a dedicated graphical processing unit.

After gathering adequate data on Fog-enabled computing, GPU virtualization, and WebRTC Protocol through a detailed technical content review on sub-sections under section 2.5, a detailed classification of this project methodology is explained in this section.

3.4 Evaluation Methodology:

Author (Slivar et al.; 2016) conducted a detailed evaluation of cloud gaming architectures and platforms by considering various factors. In addition to that, author (Peñaherrera-Pulla et al.; 2021) carried out a detailed experiment to compare different cloud gaming

platforms and architectures. On a high-level, the performance of the cloud gaming session is determined by the cloud server resources, distance between user and cloud server, video encoding and streaming services. To evaluate all above areas of the proposed architecture, below metrics were selected,

Table 2: Proposed Evaluation Parameters

Evaluation Parameter	**Description**
Frame Rate	Average frame rate received by the client browser
Frame Decoding Rate	Number of frames decoded after receiving by the client browser
Delay Between Frames	Average delay between frames received during each transaction
Jitter Rate	Average amount of Jitter occurred during the video rendering and streaming
Frames Drop Rate with respect to Jitter	Average number of frames dropped as a result of the Jitter
Decoding Time Taken	Average time taken to decode the frames. This is not calculated for frames that are discarded during the transmission
Bandwidth Consumption	Client's network bandwidth consumed during the game session

4 Design Specification

4.1 Common Design Requirements:

As a primary requirement, for this experiment, two games should be base-lined, implemented, and evaluated using all three architectures.

- To develop 2D and 3D sample games used throughout this experiment Unity 3D 2019.4.33f1 edition is used.
- Unity WebGL: Library plugin to build the game as WebGL supported HTML application.
- Unity Render Streaming: Package to create custom code and configuration that is used for streaming the game video with WebRTC

4.2 Traditional Architecture:

As discussed in the sub-section 3.1, the design requirements to implement are as follows,

- A Google Cloud Compute Engine is used to deploy the game build. The system specification are as follows,
  - Machine Type: c2-standard-4

  - Ubuntu Server v18.04
  - vCPU 4
  - RAM 16 GB
  - Storage 50 GM for game and container installation
  - GPU: Nvidia P100 (Available in europe-west4-a region inside EU)
  - External IP enabled: WebRTC communication requires an external IP to register with a STUN or TURN server
  - Enable UDP ports 8000 to 9000 as the WebRTC uses a range of ports for peer-to-peer communication and to contact the relay server

- Docker Container is used to deploy the WebGL application built using the Unity Game Engine.
- coTURN Server package deployed as docker container used to register the GCP compute engine's external IP with the STUN and TURN server
- Unity WebClient Package to host the game build as a JavaScript-based web application. This application access will be shared through the cloud infrastructure in this traditional architecture.

To re-create the implemented traditional cloud gaming solution, as a part of this project artifact, a final docker image will be backed and provided. All above-mentioned libraries, game build, packages, plugins, and commands will be pre-built in that docker image. This image can be executed on a container engine that is running as a service inside the GCP compute instance.

4.3 Fog-enabled Architecture:

As discussed in the sub-section 3.2, the design specifications required to implement Fog-enabled cloud gaming architecture are,

- A similar GCP Compute instance as specified in 4.2 will be required to deploy and execute game commands.
- Docker image is built including all required libraries for game video rendering, encoding, and media streaming.
- The WebRTC Audio and Video streamer components included in the game build will send audio and video data frames to the selected node to encode as video and stream to the user.

To re-create the implemented Fog-enabled cloud gaming solution, the docker image created for traditional architecture will be divided into two-part. The WebRTC encoding and streaming services will be separated as a new docker image and deployed onto the Fog node identified. While the actual game logic developed will be backed as a separate docker image and deployed onto the same GCP compute instance.

4.4 Proposed: Hybrid Architecture:

As discussed in the sub-section 3.3, the design specifications required to implement the proposed architecture should be divided into five stages, as follows,

- Stage 1: As explained in the common design requirements section 4.1 one 2D game and one 3D game is developed using Unity 3D 2019.4.33f1 edition.
- Stage 2: WebRTC Plugin: This is a Unity Technologies plugin attached with each game object for audio and video streaming through WebRTC.
- Stage 3: WebGL and Unity Render Streaming: Each Unity game will have multiple audio sources (for the player, AI characters, and surrounding sound), multiple game camera objects that capture the game like a scene. This Unity Render Streaming component will collect data from these audio and video sources to render into a single data packet. This package is finally built as a WebGL application for interactive game-play through desktop and mobile browsers.
- Stage 4: MobiledgeX SDK: This is a C based SDK provided by MobiledgeX service providers. This SDK is used to get all list of cloudlets (Edge Computing nodes) that are assigned to a subscribed user through C# APIs. Also, the nearest cloudlet to the target user can be determined by the same SDK. By including this SDK and logic to select the closed Edge Node to the end-user, the cloud gaming server (in this case the GCP Instance) will send the game scene to that cloudlet for encoding and streaming.
- Stage 5: Nvidia GRID drivers: As an optimization to the existing cloud gaming architecture and to support the proposed hybrid architecture, Nvidia vGPU solution is leveraged.

5 Implementation

With all data gathered from sections 2 and 3, the required design specification for the project is planned in section 4. For each architecture implemented in this project, different design specifications and implementation workflow are involved. This section will explain in detail how all three architectures were implemented based on the methodology and design specification established. For better understanding, implementation of each architecture is explained in three different sub-sections 5.2, 4.3, and 5.4.

5.1 Sample Game Implementation:

Before implementing the architectures for our comparative experiment, the game samples are to be created. In this research project, we evaluate the QoE and QoS in depth by creating our own game samples. Compared to other comparative studies and evaluation projects as explained in section 2, we have created our own game samples to have better flexibility and rigorously analyze the cloud game infrastructure. Below are the steps involved in implementing the sample games along with a few custom-built components to support WebGL game-play, and WebRTC game streaming.

- Step 1: Before starting to create the sample games to be deployed onto the server, supporting libraries and packages should be imported to the Unity project. To support the WebGL build of the game, from the Unity HUB, WebGL Build Support package should be included in the package. In addition to that, to support WebRTC video and audio streaming, Unity Render Streaming and Unity WebRTC packages should be imported to the project.

- Step 2: Once all the game objects were created with corresponding game logic, custom-built Rendering and Streaming components should be coupled with those game objects. The WebRTC package will have three streaming custom build libraries. The first one is to stream audio sources. Hence, this library will be included in the game object that creates sound effects. The second one is to combine multiple video sources (in Unity terms, cameras) into one video stream. The final component is a video streamer that will stream the resulting video.

- Step 3: Once the Audio and Video sources are combined using the custom build library, they will be rendered using the Unity Render Streaming library. As an output, the rendered video will be streamed using the Unity WebRTC library.

- Step 4: Once all the logic is in place with required libraries, using the WebGL build support from Unity, the game package will be built into a WebGL project.

- Step 5: As the last step, in this project, we have created a docker image for each sample game. This docker image will be deployed in the cloud server in each architecture implementation. This is done to reduce the time taken by the server to build the game package each time when a new game session is created.

5.2 Traditional Architecture Implementation:

In the traditional cloud gaming architecture, the game logic is deployed on a remote cloud game server. This server will be placed in a fixed location to which users will connect through a secured connection. As depicted in figure 1, the implementation of the traditional cloud gaming architecture will happen as below,

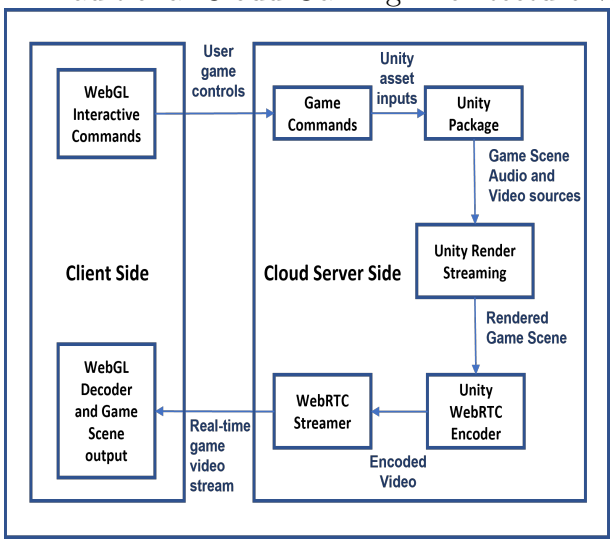
- At the beginning, each user will log into the game session created by the cloud game server. The game will be provided as a service by the cloud service provider (Cai et al.; 2014). This game session will take commands from the user through various protocols. Predominantly, UDP and WebSocket communication are being used due to their transfer rate. These game controls will be encrypted and sent to the cloud server. We have implemented a WebGL Interactive session to send game commands from the user end.

- Upon receiving the user commands, these commands are decrypted and sent as input to the game package. As explained in the design specification section 4.2, the Unity game package is used in this implementation.

- After receiving game inputs, Unity binary build will execute corresponding game logic and generate Audio and Video sources as output. We have implemented a custom C utility that will collect all audio and video sources and send it to the rendering module.

- These Audio and Video data from each source will be rendered using the Unity Render Streaming package included in the game binary itself. We have implemented another custom C utility using the Unity Render Streaming program level interface. This Utility will render all audio and video sources into a single game scene.
- Once the combined packets are received by the Unity WebRTC Encoder module, the game video will be encoded into packets of the video stream. This encoding is accelerated by Software Encoding and Hardware Encoding (using GPU) libraries. As a result, packets of encoded video are sent to the WebRTC Streamer Module.
- In general, Unity supports three video streaming options. In this experiment, we have selected WebSocket communication coupled with WebRTC public server. As WebRTC is a peer-to-peer communication, it is easy to set up and maintain the entire network. This WebRTC server will register the External IP of the cloud server to a STUN/ TURN server.
- Finally, the video stream will be received by the WebGL build generated for the Unity game. This application build will decode the video and display it to the user. In this WebGL module, we have implemented JS functions to send user commands to the cloud server and display the WebRTC video stream.

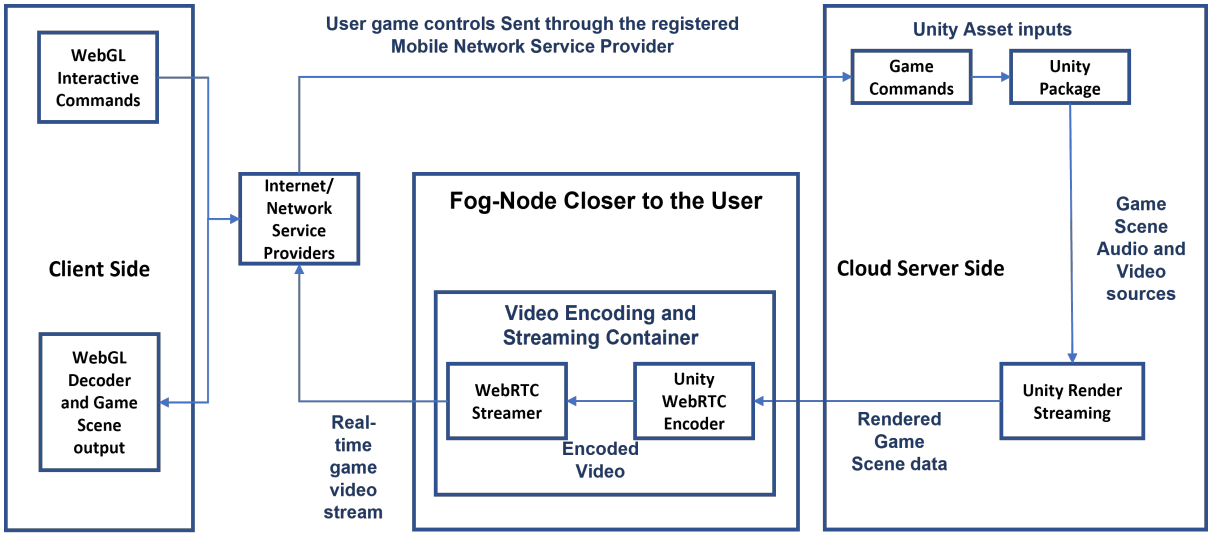
Figure 1: Traditional Cloud Gaming Architecture Workflow



5.3 Fog-enabled Architecture Implementation:

In traditional cloud gaming architecture implementation explained in sub-section 5.2, there will be two ties. One is the client-side where the game inputs are transferred from and the game video is streamed to. The other one is the cloud server end where the actual game logic is executed and the game scene is encoded as a video stream. In this Fog-enabled architecture, there will be an additional layer in between the client and cloud game server. The Fog-node layer will take part in encoding the game audio and video packets into a single game scene. Later, the WebRTC module in the same fog node will

Figure 2: Fog-enabled Cloud Gaming Architecture Workflow



stream the game scene back to the user. This implementation is based on the proposal Lin and Shen (2016) where fog node is selected based on the distance between the user and the fog node.

To create a Fog cluster of cloud instances, in this experiment, the Google Cloud Platform is used. As mentioned in the design specification section, 4.3, the docker image is built with all required libraries for video rendering, encoding, and streaming.

5.4 Proposed: Hybrid Architecture Implementation:

Figure 3: Signalling in proposed architecture

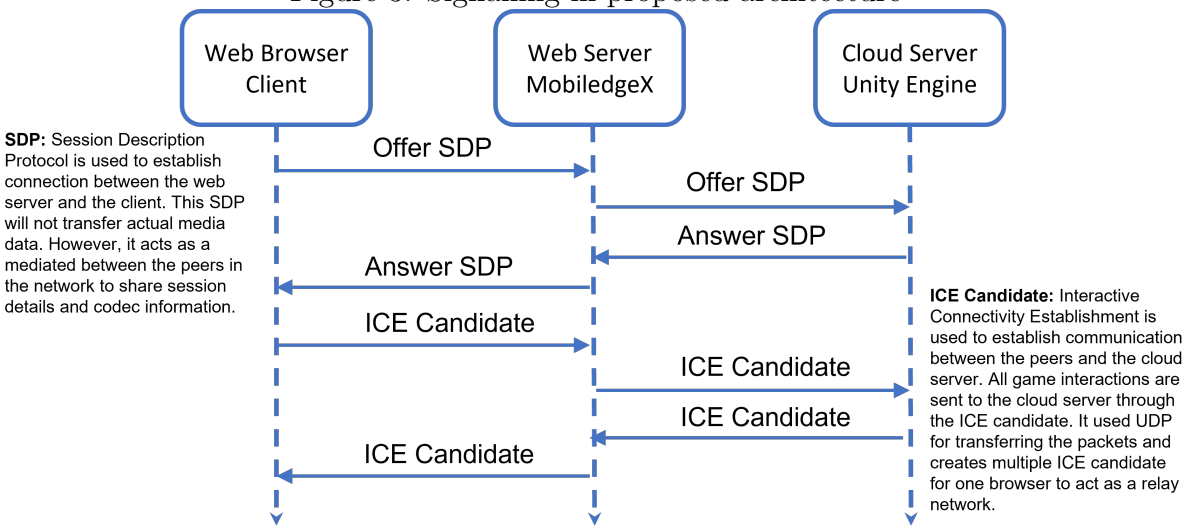
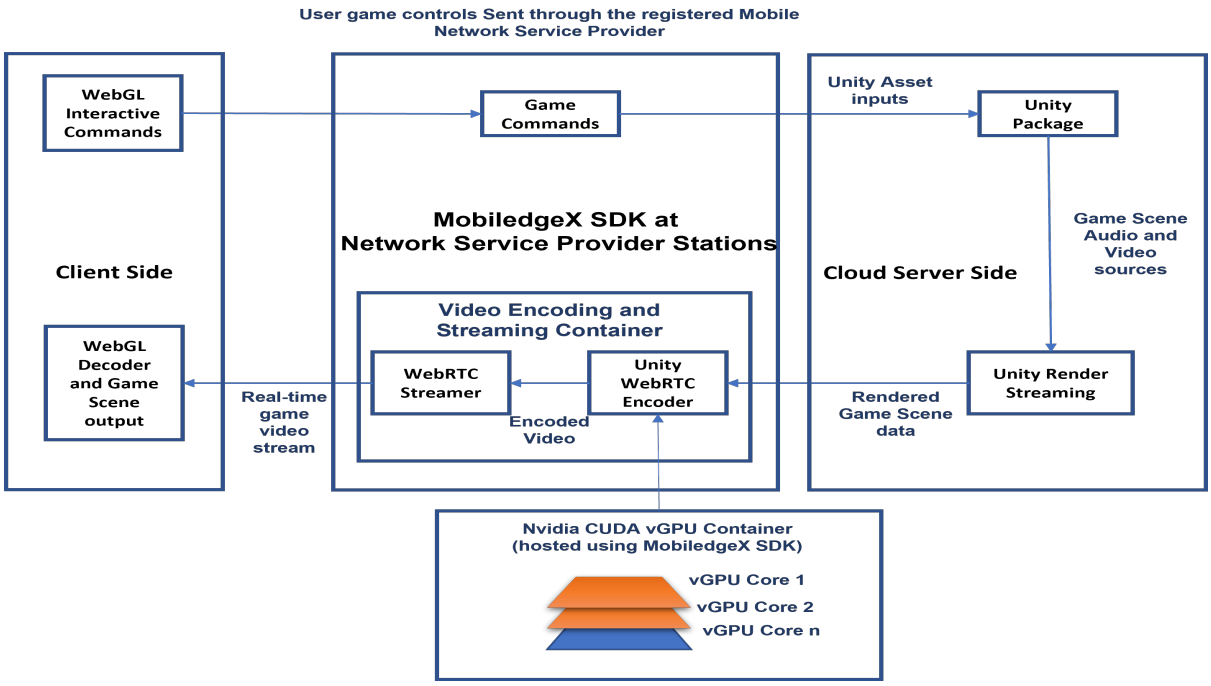


Figure 4: Proposed: Hybrid Cloud Gaming Architecture Workflow:



For the proposed hybrid architecture, represented as a workflow in figure 4, two major changes/ optimizations were implemented. The primary optimization is, in addition to the second tier implemented in the Fog-enabled cloud gaming architecture, a virtualized GPU server will be coupled with the second tier. However, implementing the Fog Nodes for this proposed architecture also has some latency issues. This issue occurs when the Fog Node is not at the edge of the user's network. Even though other existing optimization proposals like Lin and Shen (2016) to select a fog node that is closer to the user game-play device, the latency issue occurs due to multiple hops. Initially, the game commands from the user end will go to the Internet/ Network service provider, registering transaction number one. From there the command will be sent to the Cloud Game server, registering transaction number two. Once the game scene is rendered on the cloud server, the video and audio packets will be sent to the identified Fog node to encode the video, registering transaction number three. Once the video is encoded, the video stream is sent back to the network/ internet service providers registering transaction number four. Finally, the video stream reaches the user's desktop or mobile browser.

To overcome this latency issue created due to multiple network hops, in the proposed architecture, a mobile edge network will be used as the second tier, replacing the Fog node. This reduces the number of transactions from five to four. The general idea linking Edge and Fog computing is, the Fog Node acts as a layer in between the Edge Computing nodes. This layer unites the Edge Node into one cluster and transfers the raw data collected by the Edge Nodes (Yousefpour et al.; 2019). The proposed architecture eliminates this additional layer between the Edge nodes and the cloud computing services. This proposed idea is inspired by the research experiment conducted for healthcare solutions leveraging Edge Cognitive Computing Chen et al. (2018). In this approach, the data (game controls) gathered by the Edge nodes that are set up in the internet and network service providers'

end are sent directly to the Cloud Computing servers. In return, the game scene packets are sent to the Edge Nodes where the game video is encoded and transferred back to the client end.

To implement both optimizations discussed above, two services were selected after eliminating a set of similar services. Based on the recent comparative study published by authors (Rohith et al.; 2021), and based on the evaluation performed on other Edge Computing service providers, for this research experiment, MobiledgeX service and provided SDK packages are implemented in the game logic Aguilar-Armijo (2021). In addition to eliminating the Fog Node layer, we have implemented a virtual GPU layer on the same Edge Computing layer (Herrera; 2014). This will increase the video encoding performance and reduce the overall latency.

Once the implementation is complete, the communication between the client and the cloud server through MobiledgeX server will be as represented in the figure 3

6 Evaluation

After implementing three architectures 5 with sample games developed in 5.1, each architecture were evaluated based on the evaluation methodology described in the sction 3.4. Below subsections from 6.1 till 6.7, display graphs plotted based on the data received during the evaluation of each architectures. For each evaluation parameter defined in section 3.4, three experiments were conducted. The sample game designed using the Unity engine and WebRTC library is used in each experiment. All these three experiments were performed for traditional, Fog-enabled, and Proposed architectures respectively. Also, the gameplay session was kept active for 20 minutes. From that, a sample of 1 minute and 20 seconds of raw data is considered to plot all graphs and derive our findings. In the following sections, findings from each experiment graph will be explained.

6.1 Frame Rate

Frame Rate is defined by the number of frames received per second during the game session was active. Before the beginning of the game session, the client was configured to receive 60 FPS (Frames Per Second) as a baseline for all three experiments. In the proposed architecture, frame rate data plotted in figure 5, is showing a constant average of 60 FPS reception. In the Fog-enabled architecture, frame rate data plotted in figure 6, is showing some frame loss at time stamp 00:25. This frame loss is negated by the next transaction in 00:25, causing a reduction in overall QoE compared to the proposed architecture. In the Traditional architecture, frame rate data plotted in figure 7, is showing 2 occurrences of total frame loss after 00:47 and close to the 01:00 time stamp. During the game-play session, the game scene lag of 2 seconds was observed during the same timestamp. This is because, the cloud server was configured in a time zone far away from the client zone, resulting in packets loss, increased frame decoding time due to frame rate loss. These factors have impacted the QoE and QoS considerably as compared to the other two architectures.

6.2 Frame Decoding Rate

The game session, especially, a cloud game session is claimed to have better QoE if there is no/ not-noticeable amount of streaming lag. Frame Decode Rate plays a vital role

Figure 5: Proposed Architecture

Figure 6: Fog-enabled Architecture

Figure 7: Traditional Architecture

in producing a better QoE in cloud gaming. When there is a delay in Frame Rate (the rate at which frames are received), the Frame Decoding Rate will be affected as well. In figure 7, at 00:50 time stamp, no frames were received. This registered a similar effect in the Frame Decoding Rate graph of the traditional architecture experiment in figure 10. In the proposed architecture, in figure 8, fluctuation in the Frame Decode rate is not noticeable. It maintains a 60 FPS decoding rate as expected. This is less common in graph plotted for Fog-enabled architecture, in figure 9. The steady frame decode rate in the proposed architecture proves that all 60 Frames were received and decoded at the given time, in turn proving the better QoE and QoS compared to the other two experiments.

Figure 8: Proposed Architecture

Figure 9: Fog-enabled Architecture

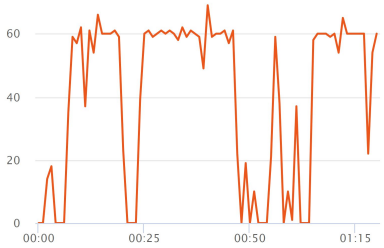


Figure 10: Traditional Architecture

6.3 Delay Between Frames (in ms)

The delay between the rate at which the frames are received will act on the smoothness of the game-play. Any delay in packets reception will prolong the next command input duration. Cumulatively, a big delay will have a huge effect on the overall QoE. The graphs plotted in figures 11, 12, and 13 have the time stamp at which the delay occurred in x-axis and the overall delay in milliseconds in y-axis. This graph is plotted by calculating the standard deviation of the overall delay population. The deviation plot reveals that the proposed architecture has a delay ranging from 5 to 10 milliseconds, the Fog-enabled architecture has a delay ranging from 10 to 15 milliseconds, and the traditional architecture has a peak delay of more than 800 milliseconds. The proposed approach is proven to have the least, negligible amount of delay with better QoE and QoS.

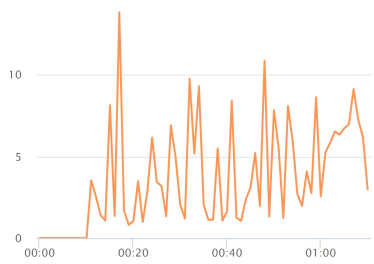


Figure 11: Proposed Architecture

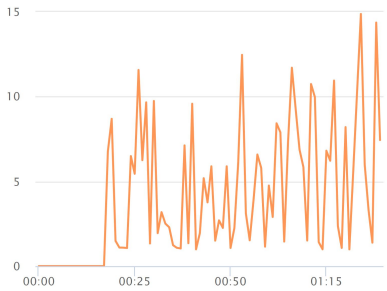


Figure 12: Fog-enabled Architecture

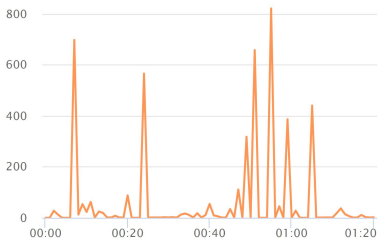


Figure 13: Traditional Architecture

6.4 Jitter Rate

Jitters are unexpected fluctuations in the transmission of media files over a communication protocol. In this evaluation, the jitter rate is calculated in milliseconds. For any video streaming application, the jitter rate should be less than 30 milliseconds. In a multi-playing cloud gaming scenario, the jitter rate is expected to be below 15 milliseconds. As the user commands are transferred to the cloud server to a series of network transactions, packets are lost/ carried to the next transaction as overhead. In the conducted experiment, the proposed architecture is proven to have an average jitter rate of fewer than 15 milliseconds 14. Whereas, the Fog-enabled architecture and Traditional architecture resulted in an average jitter rate of 10 and 15 milliseconds. This directly shows that the proposed architecture results in better QoE with a smoother and negligible amount of jitters during the video streaming.

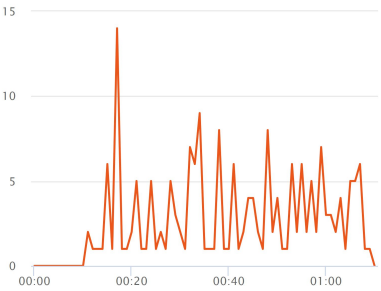


Figure 14: Proposed Architecture

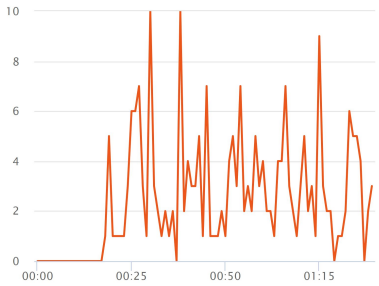


Figure 15: Fog-enabled Architecture

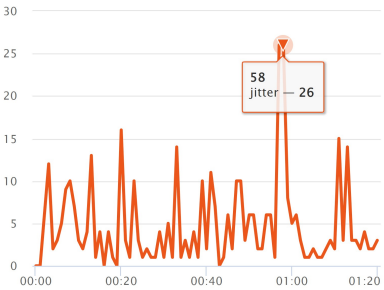


Figure 16: Traditional Architecture

6.5 Frames Drop Rate With Respect To Jitter

While comparing the rate in which frames are dropped and the rate in which the jitter occurs, it is observed that, in the Traditional architecture, figure 19, three timestamps before 00:25, close before 00:50 and close after 00:50, the heavy rate in frame loss is observed to be the primary reason for the jitter occurred during the video streaming. The rate of fluctuation in the frame loss and jitter, in the proposed architecture, remains below the threshold value, where the frames are maintained close to the expected 60 FPS and the jitter is less than 15 milliseconds. The constant frame rate as configured by the client ensures better QoS and the lesser jitter rate ensures better QoE.

Figure 17: Proposed Architecture

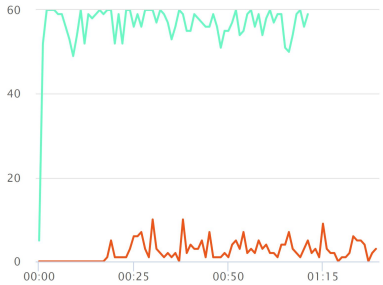


Figure 18: Fog-enabled Architecture

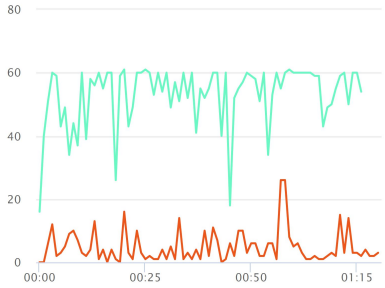


Figure 19: Traditional Architecture

6.6 Decoding Time Taken

In any cloud gaming architecture, the game scene will be streamed to the end-user as a video. This video will be encoded in the cloud sever or fog server or edge server (proposed system). Based on the time taken for packet encoding and the time is taken for transmission, the proposed architecture is proven to have the least time to decode the video packets received 20. This is because of the proposed optimizations over the existing architectures. Replacing the fog server and transmitting data to and from the edge server reduces the number of transactions and overall time to receive decoded frames. In figure 21 the time taken to get the decoded frame is maintained close to the proposed architecture. This is because the encoding is not done from the cloud server end as in traditional architecture where the average time taken to receive the decoded video frame is more than the other two architectures 22.

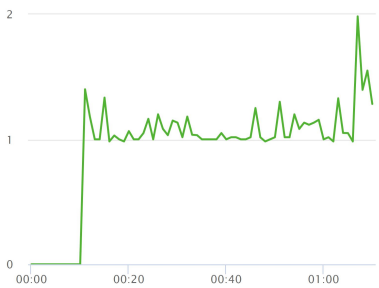


Figure 20: Proposed Architecture

Figure 21: Fog-enabled Architecture

Figure 22: Traditional Architecture

6.7 Bandwidth Consumption

Finally, one of the factors considered to evaluate the QoS is the total bandwidth consumed during the given time. By comparing the graphs in figures 23, 24, and 25, the proposed and fog-enabled architecture maintains the overall bandwidth consumption below 100 kilobytes. In the traditional architecture, more peak values are observed above 100-kilo bytes indicating more bandwidth consumption from the client network. This will affect the overall QoS and increase the cost from the client network end.

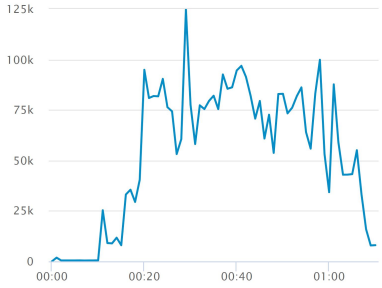


Figure 23: Proposed Architecture

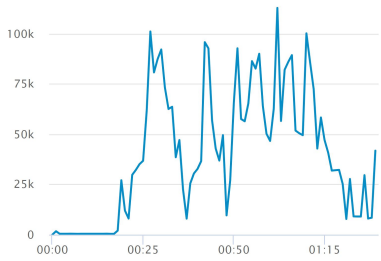


Figure 24: Fog-enabled Architecture

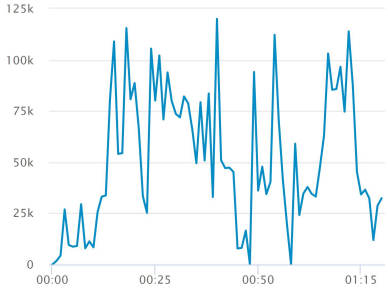


Figure 25: Traditional Architecture

6.8 Discussion

From the detailed experiment and comparison of considered metrics, it is proven that the proposed architecture resulted in better QoE and QoS compared to traditional and fog-enabled architectures. Also, the graphs are sampled from a huge timeline of more than 20 minutes of game-play. If the entire game-play sample is to be considered, the fog-enabled and traditional architecture will result in a frame loss of more than 40 frames and bandwidth consumption of more than 60 MB of network data. This amount of frames loss and bandwidth consumption will reduce the overall QoS and QoE. On the other hand, the proposed architecture is not having any cumulative effect on the frame rate, jitter, decode delay, and bandwidth consumption. From the sample considered for evaluation and overall game-play sample, the proposed architecture is proven to have better consistency in all the above-mentioned metrics.

7 Conclusion and Future Work

The proposed experiment is to compare three different cloud gaming architectures and evaluate the resultant QoE and QoS. Also, as a part of this research project, a new system that leverages Edge Computing, vGPU, and WebRTC is evaluated along with Traditional and Fog-enabled architectures. After the implementation of all three architectures, they were tested using sample games developed as a part of this experiment. The result revealed that the proposed hybrid architecture maintains constant input and decoded frame rate (at 60 FPS as configured), maintains delay between frames and jitter rate below 10 ms on average, frame decoding time below 2 ms on an average, and 100 Kilobyte of bandwidth consumption for a single gameplay session. All above-produced result improves the QoE and QoS as expected.

Future research work can focus on improving the video encoding library. At present WebRTC uses H.264 for video encoding and opus for audio encoding. This encoding format produces low to medium quality video streaming based on the player's network bandwidth. To experience better video quality with the same reduced latency, H.265 video encoding can be integrated with WebRTC protocol. Also, device statistical methods to calculate QoE and QoS based on the data gathered from the game session.

References

Aguilar-Armijo, J. (2021). Multi-access edge computing for adaptive bitrate video streaming, *Proceedings of the 12th ACM Multimedia Systems Conference*, MMSys '21, Association for Computing Machinery, New York, NY, USA, p. 378–382.

Angel, E. and Shreiner, D. (2014). *Interactive computer graphics with WebGL*, Addison-Wesley Professional.

Cai, W., Chen, M. and Leung, V. C. (2014). Toward gaming as a service, *IEEE Internet Computing* **18**(3): 12–18.

Carrascosa, M. and Bellalta, B. (2020). Cloud-gaming: Analysis of google stadia traffic, *arXiv preprint arXiv:2009.09786* .

Chen, H., Zhang, X., Xu, Y., Ren, J., Fan, J., Ma, Z. and Zhang, W. (2019). T-gaming: A cost-efficient cloud gaming system at scale, *IEEE Transactions on Parallel and Distributed Systems* **30**(12): 2849–2865.

Chen, M., Li, W., Hao, Y., Qian, Y. and Humar, I. (2018). Edge cognitive computing based smart healthcare system, *Future Generation Computer Systems* **86**: 403–411.

Herrera, A. (2014). Nvidia grid: Graphics accelerated vdi with the visual performance of a workstation, *Nvidia Corp* pp. 1–18.

Huang, C.-Y., Chen, D.-Y., Hsu, C.-H. and Chen, K.-T. (2013). Gaminganywhere: an open-source cloud gaming testbed, *Proceedings of the 21st ACM international conference on Multimedia*, pp. 827–830.

Jansen, B., Goodwin, T., Gupta, V., Kuipers, F. and Zussman, G. (2018). Performance evaluation of webrtc-based video conferencing, *ACM SIGMETRICS Performance Evaluation Review* **45**(3): 56–68.

Laghari, A., He, H., Ali, K., Laghari, R., Halepoto, I. and Khan, A. (2019). Quality of experience (qoe) in cloud gaming models: A review, *Multiagent and Grid Systems* **15**: 289–304.

Lin, Y. and Shen, H. (2015). Leveraging fog to extend cloud gaming for thin-client mmog with high quality of experience, *2015 IEEE 35th International Conference on Distributed Computing Systems*, pp. 734–735.

Lin, Y. and Shen, H. (2016). Cloudfog: Leveraging fog to extend cloud gaming for thin-client mmog with high quality of service, *IEEE Transactions on Parallel and Distributed Systems* **28**(2): 431–445.

Loreto, S. and Romano, S. P. (2014). *Real-time communication with WebRTC: peer-to-peer in the browser*, " O'Reilly Media, Inc.".

Parisi, T. (2012). *WebGL: up and running*, " O'Reilly Media, Inc.".

Peñaherrera-Pulla, O. S., Baena, C., Fortes, S., Baena, E. and Barco, R. (2021). Measuring key quality indicators in cloud gaming: Framework and assessment over wireless networks, *Sensors* **21**(4): 1387.

Rohith, M., Sunil, A. and Mohana (2021). Comparative analysis of edge computing and edge devices: Key technology in iot and computer vision applications, *2021 International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT)*, pp. 722–727.

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

Slivar, I., Skorin-Kapov, L. and Suznjevic, M. (2016). Cloud gaming qoe models for deriving video encoding adaptation strategies, *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, Association for Computing Machinery, New York, NY, USA.

Slivar, I., Suznjevic, M. and Skorin-Kapov, L. (2015). The impact of video encoding parameters and game type on qoe for cloud gaming: A case study using the steam platform, *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, pp. 1–6.

Xu, Y., Shen, Q., Li, X. and Ma, Z. (2018). A cost-efficient cloud gaming system at scale, *IEEE Network* **32**(1): 42–47.

Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J. and Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey, *Journal of Systems Architecture* **98**: 289–330.

Zhang, X., Chen, H., Zhao, Y., Ma, Z., Xu, Y., Huang, H., Yin, H. and Wu, D. O. (2019). Improving cloud gaming experience through mobile edge computing, *IEEE Wireless Communications* **26**(4): 178–183.