B.Sc. (Hons) in Software Development

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Multiplayer Networks in Gaming

By
Uan Murphy

for
Mr. Joseph Corr

April 24, 2023

Minor Dissertation

Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.

Contents

Chapter 1

Introduction

Multiplayer networks in gaming have been around since the early days of gaming, but they have come a long way since then. In the past, multiplayer games were typically limited to local area networks or split-screen play, which required players to be physically present in the same location. However, with the advent of the internet, multiplayer networks have become more accessible and sophisticated. The popularity of online gaming has fueled the development of multiplayer networks capable of supporting enormous numbers of players from all over the world. As a result, various types of multiplayer networks have emerged, each with its own set of pros and limitations.

The purpose of this minor dissertation is to investigate the use of multiplayer networking systems in multiplayer games, with a higher focus on hybrid server connections. The applied project portion of this dissertation will specifically involve creating a simple multiplayer game that communicates via a hybrid network system which assesses its effectiveness in terms of gameplay experience, network performance, and cost-effectiveness.

I believe this project will throw light onto some of the inner workings of multiplayer systems in online gaming, as well as provide myself and hopefully the reader some insight into how small developers choose between network structures and how the industry as a whole may have to evolve to allow for emerging technologies to take the stage and enable players to experience games like never seen before.

This document will be laid out as follows:

- Methodology: This chapter describes my approach to testing and project planning, as well as some research tools I employed to get an understanding of multiplayer game infrastructures.
- Technology Review: provides a thorough examination of the many network architecture topologies commonly utilised in popular games, as well as new technologies that may impact the future of gaming.

- System Design: Documents the overall design of my project and how each component interacts with the rest. This goes into detail about some of the technologies used to develop the application.
- System Evaluation: Displays the overall effectiveness of the developed system, where strengths and weaknesses have risen and how they may be overcome with further research
- Conclusion: Highlights the findings of the research conducted, experiences encountered through development and any insights that may help with further development.

Chapter 2

Methodology

For my project I used a mix of Waterfall, KanBan and Rapid Application Development approaches. I used these approaches as I found I could quickly iterate development cycles using key features of these approaches without hindering application process too far. I Believe this approach to be valid, as I was quickly able to see merit in my developed application and showed iterative progress throughout.

2.1 Waterfall

For my project I used a General Waterfall approach which follows a linear, sequential approach to software development, requiring that each stage be finished before moving on to the next. The reason it is called a "waterfall" development is because it transitions smoothly from one phase to the next.[1]

The Waterfall model, a traditional method for developing software, is still employed in some circumstances when the project's requirements and scope are clear. It has been criticised, though, for being rigid and for not allowing for adjustments or modifications throughout the development process.[1] However Seeing as some of my requirements were bound to change I decided to employ a personal approach to KanBan boards and Rapid Application Development.

2.2 KanBan

As work items move through different stages of a process, they are managed and tracked using a KanBan board, a visual project management tool. It was first used in the manufacturing sector to streamline production procedures, but software development teams have since adopted it to control their workflows.[2] A KanBan board, in its most basic form, consists of a board with columns labelled "To Do," "In Progress," and "Done," which represent the various stages of the development

process. As tasks advance through the various stages of the process, represented by cards or sticky notes, they are moved across the board. The board offers a visual representation of the status of every task and the project's overall development.[3]

The software development lifecycle can be managed and tracked using kanban boards, which give teams the ability to see their workflow, spot bottlenecks, and continuously improve their procedures.[3] Fortunately, I was developing this application alone so I was able to construct my boards without the use any third party applications.

2.3 Rapid Application Development

Iterative development and rapid prototyping are key components of the rapid application development (RAD) software development methodology. By concentrating on delivering functional software as soon as possible and improving it through feedback and collaboration, it seeks to shorten the time and cost of software development.[1] In RAD, the software development process is divided into smaller, easier-to-manage stages that are then iteratively developed and tested. In order to deliver a high-quality product more quickly, the development team collaborates closely with stakeholders to gather feedback and make continuous improvements to the software.[3]

2.4 Combination

For my approach I designed the underlying system and how the user should interact with the system. Each task was su divided into smaller i.e. User can login and connect to network, User can view current active lobbies, User can join lobbies. Along with this I used GitHub to keep a ledger of tasks completed unfortunately my naming convention wasn't exactly strict and my commit messages didn't fully encapsulate certain tasks which ended up causing confusion on what tasks were appropriately completed leading to further problems down the line.

While developing I continued to use my favourite approach to software development, rapid application development. Which is mainly a prototyping methodology but I have found it incredibly useful when developing and deploying multiplayer network systems fast, so that I can test without compromise as my main focus was to make sure players could connect with one another.

At the start of this year, I had convened with my supervisor on a few occasions to discuss the project when it was under a different idea and structure. Unfortunately, I started having difficulties communicating with supervisor making a disconnect between myself and the supervisor. This disconnect severly hindered

the development of the project as I began having doubts about the project discussed which eventually led to me changing the project idea altogether in order for an easier development cycle, this in fact further threw the project into turmoil as I no longer had any structure to follow, what didn't help either was that I convinced myself that after the disconnect I could not talk to said supervisor, which led to further issues.

Chapter 3

Technology Review

3.1 Introduction

Multiplayer games and applications have grown in popularity in recent years, as has the technology used to support these experiences. When designing a multiplayer system, one of the more important decisions that developers must make is whether to use a centralized or peer-to-peer network architecture. In this technology review, we will assess the advantages and disadvantages of centralized, peer-to-peer and hybrid networks for multiplayer applications. We'll look at scalability, latency, reliability, security, and cost, as well as existing and emerging technologies used in these networks. Our goal is to provide insights into which network type may be best suited for different types of games or applications, as well as to identify trends that may impact the future of multiplayer networking.

3.1.1 Scope

The multiplayer networks used in video games and other real-time interactive applications will be examined in this technology review, with an emphasis on the network architectures and technologies that make these experiences possible. We will compare centralized, peer-to-peer, and hybrid network architectures as well as approaches that combine both. We will analyze the tradeoffs between important features for various network architectures, including scalability, latency, reliability, security, and cost. We will evaluate both LAN (local area network) and WAN (wide area network) environments, with a focus on networks that can accommodate at least 10 players concurrently. To determine how various network architectures and technologies perform across various game genres, we will also take into account first-person shooters, massively multiplayer online role-playing games (MMORPGs), sports games, and strategy games. We will restrict our analysis to middleware and game engines that support multiplayer functionality,

as well as network technologies that are widely used in the sector, such as TCP/IP, UDP, and WebRTC. Turn-based games and applications that don't require real-time communication will not be taken into consideration. Last but not least, our review will take into account recently developed technologies that have the potential to influence multiplayer networking in the future, including cloud gaming, blockchain-based networks, and virtual reality, and we will offer insights into how these technologies may influence multiplayer network design in the future.

3.2 Network Architectures

The planning and structuring of a computer network is known as network architecture. It specifies how networks are constructed, how devices and communication protocols are organised, and how data is transferred. Network architectures come in a variety of forms, including distributed, decentralised, and centralised ones. Peer-to-Peer, Centralised, and Hybrid Network Architectures will all be mentioned in the context of this review.[4][5]

A network architecture that can support a large number of players, quick updates, and real-time synchronisation of game state across all players are typically requirements for multiplayer games.Lag, latency, and reliability of the gaming experience can all be affected by the network architecture.[6] Gaming that is sluggish or unreliable due to a network architecture that is poorly designed can have a negative impact on the player experience.Network architecture is therefore a crucial part of multiplayer games, and game developers must carefully consider and optimise the architecture to guarantee a smooth and enjoyable gaming experience for players.[7]

3.2.1 Centralised Network Architecture

A centralised network architecture in the context of multiplayer games describes a setup where all game data and processing is controlled by a single central server. In this architecture, the central server and each player's device—a computer or gaming console, for example—communicate in order to send and receive data about the game state, such as player positions, game events, and updates to the game world.[8] The central server is in charge of keeping the game state in check, enforcing the rules, and synchronising player data. Due to its simplicity and ease of implementation, this architecture has been widely used in online multiplayer games like World of Warcraft and League of Legends. This architecture does have some disadvantages, though, like the potential for server overload and high latency, which can lead to subpar game performance and a bad user experience. Moreover, the game may be susceptible to attacks or crashes due to the single point of failure

that could be the main server.[8]

That's fine and all, but how exactly does this server architecture operate? Perhaps you're asking. In a centralised network architecture for multiplayer games, all players connect to a central server that manages the game world. When a player first starts the game, they ask to join the game world. The server responds by providing them with a unique ID called a player ID.[8] A player can send and receive data regarding the state of the game once they are connected to the server. For instance, when a player moves their character, their device notifies the server of the new position of that character. The server then sends the updated game state to all other players in the game along with the player's character's new position. Similar to this, whenever a player takes an action, like using a weapon or a spell, their device sends a message to the server indicating what they did. The game state is subsequently updated by the server and sent to every other player in the game.

In this way, the central server assumes the role of the game's supreme authority and makes sure that everyone is in sync and following the same set of rules. Although this architecture gives the game a high degree of control and consistency, it also places a heavy burden on the server by requiring it to process and manage all of the game data.[8] The server may employ a number of strategies, including load balancing, data compression, and effective message passing protocols, to handle the substantial amount of data and traffic produced by numerous players. As the number of players connected to the game world changes, the server may also need to be scaled up or down.[6]

Larger game publishers or developers with the financial means and technical know-how to run a centralised server infrastructure may benefit more from a centralised network architecture for game servers. This is due to the fact that configuring and running a centralised game server can be costly and technically difficult, requiring sizable hardware and infrastructure investments in addition to a group of knowledgeable server administrators.[9] The advantages of a centralised architecture, such as improved security and simpler management of game data, outweigh the disadvantages in games with large player bases or resource-intensive gameplay, so centralised networks may be a better fit in these cases. Examples include games such as: Grand Theft Auto Online, League of Legends and Fortnite.[8]

3.2.2 Peer-to-Peer Network Architecture

Peer-to-peer (P2P) networks are decentralised computer networks in which each computer or node functions as both a client and a server, sharing resources and data with other nodes directly without the need for a centralised server. A P2P network enables players to connect and play with one another in multiplayer games without relying on a centralised server to control the game.[10] Each player man-

ages their own game state and communicates directly with other players to exchange game data in a P2P network, where each player's computer functions as both a client and a server. As it can lessen the load on centralised servers and increase the responsiveness of the game overall, this method can be especially helpful for games that only require a few players or low-latency connections.[11] An access request is sent directly to the other node when a network node needs to use a resource or piece of data that is kept there. The two nodes then agree on the terms of the transaction and transfer the data or resource among themselves directly, cutting out all middlemen and centralised servers.[12] P2P networking is still a popular option for many multiplayer games, especially those that involve small player bases or demand low-latency connections, despite these difficulties. P2P networks are able to provide gamers with a more adaptable and responsive gaming experience by utilising the power of decentralised computing.

P2P systems come in a variety of forms, each with its own special features and architecture. Nodes connect to each other in unstructured P2P networks randomly or independently, without any centralised organisation or control. File sharing and other P2P applications that don't require rigid organisational or management structures are frequently used in these networks.[10] Nodes are arranged into hierarchical or structured systems in structured P2P networks, frequently based on distributed hash tables (DHT) or other indexing systems. These kinds of P2P networks are frequently employed for applications that call for the storage and retrieval of more structured and organised data.[12]

3.2.3 Hybrid Network Architecture

Peer-to-peer (P2P) and centralised network models are both used in hybrid network architectures in the gaming industry. Games that demand scalability and in-game communication between players frequently employ this architecture. In a hybrid network, certain game data and logic are controlled by centralised servers, while other data and logic are dispersed among players in a P2P network. This makes player communication faster and more effective, relieving pressure on centralised servers and enhancing the overall gaming experience.[13]

A hybrid network architecture can offer a more scalable solution than a purely centralised or purely P2P network, which is one of its benefits. Matchmaking and leaderboard management are two examples of tasks that centralised servers can handle that are challenging or impossible to distribute among players. P2P networking, on the other hand, can offer a more effective and low-latency method of handling player-to-player interactions, such as movement and combat.[13][14] Using a hybrid network architecture, Sea of Thieves enables seamless player interactions in a shared open-world setting. While player interactions and combat occur in a P2P network, the game uses centralised servers to manage matchmaking,

progression, and rewards.[15]

Hybrid network architectures do, however, come with some difficulties. One problem is that since a P2P network depends on players being able to communicate with each other directly, the quality of individual players' connections can have an impact on the game's performance and stability. As a result, it's possible for some players to experience more lag or disconnections than others, which can be annoying and create an unfair playing environment.[16][17] Making sure that the data and logic of the game remain secure and uniform across all network users is another challenge. Due to the fact that the data is spread across multiple network nodes, a hybrid network may make it more challenging to accomplish this than a purely centralised or purely P2P network. To prevent cheating or unintentional data manipulation in games, game developers must carefully plan the architecture of their games and put in place the necessary security measures.[16] Destiny 2 is one game that makes use of a hybrid network architecture. While cooperative play and player-vs-player combat are handled by centralised servers in this game, matchmaking and social spaces are handled by peer-to-peer networks. This enables the game's developers to prevent cheating while also enabling smooth gameplay in situations where it is crucial to minimise lag and latency issues.[18]

3.3 Network Technologies

To enable player communication, simplify data transfer, and guarantee a seamless gaming experience, multiplayer networks rely on a number of network technologies. TCP/IP, UDP, WebRTC, and middleware are some of the network technologies that are most frequently utilized in multiplayer networks.[7]

3.3.1 Transmission Control Protocol/Internet Protocol (TCP/IP)

One of the foundational protocols of the Internet Protocol (IP) family, the Transmission Control Protocol (TCP), is widely employed for secure data transmission over networks. TCP delivers data between applications in an orderly, dependable, and connection-oriented manner at the transport layer of the OSI (Open Systems Interconnection) paradigm[19].TCP employs a number of strategies to guarantee dependable data transmission, these include:

- Sequence Numbers: Each data segment that TCP transfers has a sequence number assigned to it. The position of the segment within the overall data stream is indicated by the sequence number. These sequence numbers are used by the receiver to recreate the initial data stream and guarantee that the segments are received in the correct order.[20]

- Acknowledgements: A TCP receiver sends an acknowledgment (ACK) back to the sender after successfully receiving a data segment, letting them know that it was done so. The sender retransmits the segment if it believes it was lost and does not receive an ACK after a predetermined amount of time.[20]

- Retransmission: TCP assumes that a delivered data segment was lost and retransmits it if it does not receive an ACK for it within a predetermined amount of time. TCP employs a number of algorithms to regulate the rate of retransmission, prevent network congestion, and guarantee that segments are only retransmitted when necessary.[20]

- Flow Control: To stop the sender from immediately overloading the recipient with data, TCP employs a flow control mechanism. The amount of data that the receiver is willing to accept at any one time is indicated by the window size that the receiver advertises. In order to ensure that the receiver can handle the data flow, the sender then modifies its transmission rate to coincide with the receiver's window size.[21]

- Congestion Control: TCP also has a congestion management system that limits the rate at which data is transferred, assisting in preventing network congestion. To prevent overloading the network and resulting in packet loss, TCP reduces its transmission rate when it notices that the network is becoming crowded.[21]

3.3.2 User Datagram Protocol (UDP)

A fundamental protocol of the Internet Protocol (IP) family that is frequently used when speed is more crucial than reliability is the User Datagram Protocol (UDP). Because UDP is a connectionless protocol, data is sent between endpoints without first establishing a secure channel of communication. Instead, it sends datagrams directly to the target without establishing any connections or waiting for any acknowledgements.[4] The source and destination port numbers, as well as the datagram's length, are all included in the fixed-size header of UDP datagrams. A UDP datagram can have a payload that is between 0 and 65,535 bytes long, which makes it practical for sending brief bursts of data quickly.[22]

Small data packets containing details about the game state, such as player positions and movements, weapon fire, and other game events, are frequently transmitted in games using UDP. Games can give players real-time feedback without the delays brought on by TCP by using UDP. However, there is a chance of packet loss and out-of-order delivery because UDP lacks the TCP reliability mechanisms.[4] Like TCP, UDP developers employ a variety of techniques to ensure reliable data

transmission. Sequencing, Acknowledgments, and Retransmission are are included in the list as well as:

- Interpolation: To estimate the current state of the game, Interpolation uses data from the last known state. For instance, if a packet containing a player's position and velocity is lost, the game engine may estimate the player's current position using the player's last known position and velocity. This can make sure that the game can still be played even if there are packet losses.[23]

- Extrapolation: The process of predicting the game's future state using information about its current state. The game engine, for instance, might use information such as a player's direction and speed to predict where the player will be in the future. This can be helpful for predicting where other players or game objects will be in the future.[23]

3.3.3 Web Real-Time Communication(WebRTC)

A set of technologies and protocols are provided by the open-source Web Real-Time Communications (WebRTC) project, which enables real-time communication between web browsers and other devices. It enables programmers to create web applications that can send audio, video, and data streams directly between browsers without the use of plugins or other third-party applications.[24] For real-time multiplayer games like first-person shooters or racing games, where low latency is essential for a seamless gameplay experience, WebRTC can be especially helpful. WebRTC makes it possible for players to communicate directly with one another without the use of intermediary servers, which can help to lower latency and increase the game's overall responsiveness.[25]

Even though Web Real-Time Communications (WebRTC) is a strong technology for enabling real-time communication between web browsers, developers should be aware of some of its limitations. These consist of:

- Browser Compatibility: Although most popular web browsers support WebRTC, not all of its features are supported by all browsers, and some may need additional configuration or implementation workarounds. To ensure compatibility, developers should thoroughly test their applications on various browsers.[26]

- Bandwidth Requirements: When transmitting audio and video streams, WebRTC can use up a lot of bandwidth. Developers should plan their applications to operate efficiently within these limitations and take into account how bandwidth restrictions will affect user experience.[24]

- Network Firewall Issues: WebRTC establishes direct connections between browsers using a variety of network ports, which can occasionally be blocked by network firewalls or security regulations. To ensure compatibility, developers should test their applications in various network environments.[24]

- Latency: Even though WebRTC is intended for low-latency communication, users may still experience latency depending on a variety of factors, including network performance and browser efficiency. Particularly for real-time applications like online games, developers should carefully consider the effects of latency on their applications.[26]

- Security: Although WebRTC protects data transmitted between browsers with encryption and other security measures, developers must still take precautions to ensure the security of their applications, especially when transmitting sensitive data like personal information or payment details.

[25]

3.3.4 Middleware

Software known as middleware serves as an abstraction layer between various system parts, such as various network technologies, to make system development and integration easier. Middleware can be used in the context of game networks to offer a variety of network-related functionality, including matchmaking, player authentication, and real-time communication.[27] By offering a variety of ready-made components that can be quickly integrated into their games, middleware can assist game developers in overcoming the difficulties associated with creating intricate, distributed game networks. A real-time communication middleware, on the other hand, might offer an API for sending and receiving game state updates between players. As an example, a matchmaking middleware might offer a server that assists in matching players based on skill level or location.[28]

By utilising middleware, game developers can concentrate on creating the essential gameplay mechanics for their games rather than spending time and resources creating and maintaining complex network technologies. Furthermore, middleware by offering optimised network protocols and algorithms can assist in enhancing the overall performance and stability of game networks. These may include:

- Graphics and Audio Engines: Tools for game development can easily integrate pre-built graphics and audio engines provided by middleware. This can help independent game developers focus on creating the essential gameplay mechanics for their games rather than spending time creating and optimising their own graphics and audio engines.[27]

- Artificial Intelligence: To make non-player characters (NPCs) in games smarter and more dynamic, middleware can offer libraries and pre-built AI algorithms. Because of this, independent game developers can make games that are more immersive and interesting without having to invest a lot of time in creating their own AI algorithms.[28]

- Analytics and Monetization: Tools for game data analysis and monetization strategy optimisation can be found in middleware. As a result, they can improve their understanding of player behaviour and preferences and optimise their games for both high player engagement and profit.[28]

- Cross-Platform Development: Games that can be played on desktop, mobile, and console platforms can be created with the help of middleware. As a result, independent game developers may be able to reach a larger audience and potentially boost their games' profitability.[27]

The tradeoffs of middleware, such as the potential for less flexibility and control over network functionality, as well as the possibility of extra expense and licencing fees, should be carefully considered by developers.

3.4 Emerging Technologies

Due to the introduction of new technology, the multiplayer gaming industry has seen tremendous change in recent years. Some of the most promising technologies that have the potential to revolutionise multiplayer networking include cloud gaming, blockchain-based networks, and virtual reality. With new ways for players to interact with one other and the game world, these technologies can dramatically improve their gaming experience. In this response, we will look at how each of these new technologies may affect multiplayer networking in the future and how those networks may need to change to take full advantage of their potential.[29]

3.4.1 Cloud Gaming

The gaming business could undergo a transformation due to cloud gaming technology. With cloud gaming, more players are able to take part in the online gaming community since they can play high-quality games with little latency on low-end devices. However, networks will need to adapt to match the high bandwidth and low-latency needs if they are to fully realise the potential of cloud gaming.[30] Cloud gaming needs networks that can supply high-speed, low-latency connectivity to guarantee a flawless gaming experience. This is due to the fact that the player is streaming the game's audio and video output to their device from

a remote server where the game is running. Any latency or delay can ruin the player's game experience and make them angry. Networks must therefore provide the required infrastructure to make cloud gaming a practical choice for players.[30]

For networks to accommodate the massive volumes of data needed for cloud gaming, there must be enough bandwidth. This pertains to both download and upload speeds, as players will also need to download the video and audio output in addition to uploading their inputs to the server. Additionally, networks must have low latency connections to guarantee that there is little to no lag between player inputs and game responses.[30] Networks must be optimised for the unique needs of cloud gaming in addition to being fast and low latency. For instance, networks must offer Quality of Service (QoS) guarantees to assure that traffic for cloud gaming is given priority over other network traffic. This will help to prevent network congestion and guarantee that other network users won't have a negative impact on the player's gaming experience.[30] Overall, the ability of networks to adapt to the high-bandwidth and low-latency requirements of this developing technology will determine the success of cloud gaming. Networks must continue to invest in their infrastructure to accommodate gamers' expectations as cloud gaming's popularity grows.[30]

3.4.2 Blockchain

Blockchain technology is a decentralised and distributed ledger that can enable secure transactions without middlemen, as well as providing transparent and safe data storage. Blockchain technology is becoming more and more prevalent in the game industry, especially in areas like asset ownership, digital identity, and payment systems. However, networks will need to adjust to accommodate the specific needs of this developing technology in order to fully realise the potential of blockchain technology.[31]

The enormous bandwidth and processing power needed to run the decentralised network is one of the major difficulties with blockchain technology. Because blockchain networks are decentralised, data is spread among a large number of nodes, and each node is responsible for processing and validating transactions.[31] In order to manage the computational demands of blockchain technology, networks will need to have enough processing capacity. Furthermore, smaller gaming companies may find it difficult to enter the market due to the high bandwidth requirements of blockchain technology. Networks will therefore need to offer affordable means of transmitting data between blockchain networks, such as by data compression or by improving data transfer protocols.[32]

The security requirements of blockchain technology provide another difficulty. In order to safeguard data integrity and avoid hacking, blockchain networks must be extremely secure. This necessitates networks to have robust encryption mech-

anisms and to constantly check for security flaws.[31] Networks will also need to offer ways for various blockchain networks to communicate with one another. This is due to the possibility of several gaming platforms using various blockchain networks, and the possibility that players may seek to transfer assets between these platforms. Networks that support several blockchain protocols and offer rapid and safe transmission mechanisms are necessary for interoperability solutions.[32]

3.4.3 Virtual Reality

With the use of immersive technologies like virtual reality (VR), players can engage in more authentic interactions with other players and the game world. The social aspect of multiplayer gaming can be improved by enabling participants to take part in virtual events and competitions. By enabling players to engage with virtual objects and settings in ways that weren't before feasible, VR can also open up new possibilities for multiplayer gaming. Although multiplayer networking may be difficult with VR, it does require high bandwidth and low latency connections.[33] For VR to give an immersive experience without lag or motion sickness, high-speed and low-latency networks are necessary. The current level of VR technology makes it difficult for existing network infrastructures to handle the amount of data that must be transferred in real-time. Networks must have enough bandwidth, be low latency, and be able to handle the additional traffic demand that comes with VR apps in order to enable VR.[33]

Mobile Edge Computing (MEC), which seeks to bring processing power closer to the end-user to decrease latency and enhance network performance, is one strategy for overcoming this difficulty. MEC makes it possible to transmit VR content over cellular networks and may open the door to 5G and the next iterations of VR technology [34][35]. Use of Field Of View (FOV) rendering technologies, which are created to optimise the bandwidth and latency needed for VR, is another strategy. By rendering only the portion of the scene that the user is now viewing rather than the complete screen, these solutions decrease the amount of data that needs to be transferred [34]]. Network statistics and visualisation can be extremely important in enabling VR. Networks are the best choice for visually and analytically examining massive data because they may offer a potent depiction of how interconnected parts of complex systems interact [34]. The quality of VR experiences may be impacted by bottlenecks and other problems that can be found by analysing network data.

In conclusion, the ability of networks to manage the increased traffic needs, lower latency, and provide high-speed performance is crucial to VR's success. These problems can be addressed by solutions like MEC and FOV rendering, while network visualisation and analytics can offer information for improving network performance.

3.5 Conclusion

The technological landscape is always changing, and the development of new technologies has increased the complexity of networks. Network designs must therefore change to meet the needs of contemporary applications.

For enterprises with little technological know-how, centralised architectures provide a highly safe and manageable solution with a single point of control. The drawback of centralised networks is that they are attackable, and a single point of failure can take down the entire network. Peer-to-peer architectures, on the other hand, offer a more decentralised and scalable solution by enabling users to interact directly with one another and lessening the load on central servers. Peer-to-peer networks can, however, be difficult to maintain and protect, and the absence of a single point of control can make it challenging to oversee network activities. By combining the advantages of centralization and decentralisation, hybrid architectures aim to provide the most flexibility and scalability. Hybrid networks, on the other hand, need professional specialists to configure and maintain them because they are more complicated and might be harder to administer.

TCP/IP is the most used protocol for establishing connections between devices on the internet in terms of network technologies. While WebRTC is a well-liked option for real-time communication in web browsers, UDP is frequently used for applications that require minimal latency, such as online gaming. Enterprise environments frequently use middleware, a layer of software that simplifies communication between several applications.

Future-looking technologies like Blockchain, Virtual Reality, and Cloud Gaming will need considerable network design upgrades to attain their full potential. For instance, low latency and high bandwidth connections are needed for cloud gaming, whereas a secure, decentralised network is needed for blockchain to maintain the integrity of its data. Virtual reality is a challenging application for networks since it needs extremely low latency and large bandwidth to give an immersive experience.

In summary, network design is a crucial factor to take into account for contemporary networks, and various architectures and technologies will be more appropriate for certain use cases. Network design will need to change as technology progresses in order to satisfy the requirements of new applications, and organisations will need to weigh the trade-offs between security, scalability, and usability when selecting a network architecture.

Chapter 4

System Design

As mentioned previously the applied project portion of this dissertation was to build a simple multiplayer game. For this I had decided to build a simple racing game that allowed players to connect through a lobby system and would host cloud-based servers that connected clients to servers. For development purposes I chose to use Unity as my development platform, as well as Unity Lobby and Relay services for connecting client to servers.

4.1 Unity

Unity is a robust game engine that has grown in popularity among developers in recent years. Unity, which was first launched in 2005, has grown to become one of the world's most popular game engines, with a market share of more than 60% among the top 1,000 mobile games [36]. Unity's flexibility to produce games for numerous platforms, including PC, Mac, mobile devices, and consoles, is one of the primary reasons for its popularity. Unity, in addition to its cross-platform capabilities, provides a number of features that make it an excellent choice for producing games of all types, including multiplayer games. Unity's networking capabilities give creators a variety of options for generating online multiplayer games, while its user-friendly drag-and-drop interface and powerful scripting tools make it simple to create games of many types, from small mobile games to big 3D open-world titles. Furthermore, Unity has a huge and active developer community that is constantly generating and sharing game development resources. This includes tutorials, code samples, and other valuable resources that can assist developers of all skill levels in learning new abilities and improving their productivity.

4.1.1 Lobby

Unity Lobby is a powerful multiplayer framework that provides developers with a set of tools and APIs for creating online multiplayer games. It is developed on top of Unity's networking technology, which supports both peer-to-peer and client-server topologies, allowing developers to construct large-scale multiplayer games. The Unity Lobby lobby system allows players to build or join gaming lobbies and manage members within them. Lobbies can be configured with many parameters such as game mode, player count, and matchmaking criteria, allowing players to select the ideal game to play. The matchmaking mechanism matches players with others who have comparable skill levels or preferences, providing fair and entertaining gameplay for everybody. Developers can use the player management system to track player behaviour and data such as game progress, achievements, and stats. This data can be used to personalise player experiences such as custom matchmaking and leaderboards.

I will be using Unity's Lobby Functionality to connect players together into lobbies and allow them to connect together and play games

4.1.2 Relay

Unity Relay is a networking solution developed by Unity Technologies that enables developers to construct quick and stable network connections between multiplayer gaming players. Unity Relay is a cloud-based service that ensures players can connect to one another fast and effortlessly, even if they are on separate networks or behind firewalls. One of the primary advantages of Unity Relay is that it reduces latency in multiplayer games. The delay between a player's actions and the time those actions are reflected in the game is referred to as latency. High latency can make a game feel sluggish or unresponsive, which is annoying for players. Unity Relay reduces latency by establishing a quick and stable connection between players, ensuring that actions are reflected in the game in real-time. Another advantage of Unity Relay is that it secures the connection between participants. This is especially critical for games involving sensitive information or requiring a high level of security, such as online banking or gambling. Unity Relay contributes to the security and encryption of all connections between players, thereby preventing unauthorised access or hacking attempts.

I have used Unity Relay to connect players together after forming a lobby, this allows for a persistent connection to be handled by the host

4.1.3 Core Components

My application is structured as follows

- Input Controls
- Kart Controls
- Lobby Handler
- Player Handler
- Relay Handler

Chapter 5

System Evaluation

Evaluate your project against the objectives set out in the introduction. This chapter should present results if applicable and discuss the strengths and weaknesses of your system. This is a clear opportunity for you to demonstrate your critical thinking in relation to the project.

Chapter 6

Conclusion

Briefly summarise your context and objectives. Remind the reader about the overall rationale and goals of the project. Highlight your findings from the System Evaluation chapter.

Chapter 7

GitHub Repo

https://github.com/MurphyUan/final-year-project

Bibliography

[1] T Al-Maghrabi and O Zwikael. Project management methodologies: A review of the literature. *Project Management Journal*, 48(3):100–116, 2017.

[2] Hiren Patel and Sanjay Kumar Sahu. Kanban based software development-a systematic literature review. *International Journal of Computer Science and Information Security (IJCSIS)*, 16(4):100–107, 2018.

[3] Roger S Pressman and Bruce R Maxim. *Software engineering: A practitioner's approach*. McGraw Hill, 8th edition, 2015.

[4] Josh Glazer and Sanjay Madhav. *Multiplayer Game Programming: Architecting Networked Games*. Addison-Wesley, Boston, MA, 2015.

[5] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach*. Pearson, 2017.

[6] Mark Claypool and Kate Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, 2006.

[7] Larry L. Peterson and Bruce S. Davie. Computer networks: A systems approach. In *Computer Networks: A Systems Approach*, chapter 3. Morgan Kaufmann, San Francisco, CA, 5 edition, 2011.

[8] Zhihong Zhang and Jie Wu. Design and analysis of a centralized network architecture for multiplayer online games. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1078–1091, 2012.

[9] Jake Birkett. Centralized vs distributed network architectures for indie games, 2021.

[10] Jason Gregory. Peer-to-peer gaming: A technical overview. *Gamasutra*, 2005.

[11] Sangho Shin KyoungSoo Park and Sue Moon. Peer-to-peer networks for multiplayer games. *ACM Digital Library*, 2019.

[12] Oracle. Understanding peer-to-peer networking. *Oracle Documentation*, 2012.

[13] Andrew Ferguson and Eddie B Fernandez. *Multiplayer Game Programming: Architecting Networked Games*. Addison-Wesley Professional, 2015.

[14] Mirko Blattner and Simon Schubert. Network performance evaluation of hybrid multiplayer games. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*, pages 1–6. ACM, 2016.

[15] Rare Ltd. Sea of thieves: A shared-world adventure game, 2021.

[16] Arash Hadavi and Emmanouil Panaousis. Security challenges in hybrid peer-to-peer networks: A systematic literature review. *IEEE Access*, 8:223804–223818, 2020.

[17] Julian Amann and Robin Wolff. The benefits and challenges of hybrid multiplayer games. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, pages 219–225. ACM, 2016.

[18] Bungie. Destiny 2 networking, 2021.

[19] W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, Boston, MA, 2011.

[20] J. Postel. Transmission Control Protocol. RFC 793, September 1981.

[21] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 5681, September 2009.

[22] Glenn Fiedler. *Networking for Game Programmers*. Gaffer On Games, Santa Monica, CA, 2013.

[23] Christer Ericson. *Real-Time Collision Detection*. Morgan Kaufmann Publishers, 2005.

[24] WebRTC. WebRTC Homepage.

[25] WebRTC for the Curious. Real-Time Gaming with WebRTC.

[26] Tsahi Levent-Levi. WebRTC and Gaming: The Perfect Match, accessed 24 Apr. 2023.

[27] Jim Haas. The Pros and Cons of Game Middleware, 2017.

[28] Weihao Luo, Tatiana Noronha, and Kathy Qian. Game Networking Middleware: A Practical Guide to Choosing Solutions, 2013.

[29] Gartner. 4 Emerging Technologies You Need to Know About. https://www.gartner.com/en/articles/4-emerging-technologies-you-need-to-know-about, 2023.

[30] Sergio Lopez. Cloud gaming: A revolution in gaming technology.

[31] David LEE Kuo Chuen Chuen and Linda Low. Blockchain Gaming and NFTs: The Future of Entertainment, 2021.

[32] Xinxin Zhang, Guoqing Li, Jing Li, Yuan Fang, Zhe Zhang, Wei Jiang, and Gang Li. Blockchain and Its Applications in the Gaming Industry: A Review. *IEEE Access*, 8:58194–58209, 2020.

[33] World Economic Forum. Connecting more people to ar and vr technologies could transform everything from education and healthcare to mining and tourism. *World Economic Forum*, February 2021.

[34] Amit Navon... Simone Mangiante, Guenter Klas. Optimizing Field Of View in VR Streaming using Mobile Edge Computing. *ACM SIGCOMM Computer Communication Review*, 47(5):12–18, 2017.

[35] Fan Zhang, Bingxin Wang, Xiaojun Lu, Rong Chen, Ningwei Liu, and Jingyu Zhang. Edgefov: Enabling immersive video streaming in cellular networks. In *Proceedings of the 2017 Workshop on Mobile Edge Communications*, pages 15–20. ACM, 2017.

[36] Stepico. Why is unity the best game engine: Pros and cons, 2021.