Distributed Computing in VALORANT and How It Maintains Its Competitive Integrity in a Multiplayer Environment

Daniel Ramirez
California State University Long Beach, CECS 327 SEC 05
Daniel.ramirez05@student.csulb.edu

Abstract

Multiplayer online games are currently very popular, some games even establishing their own professional competitive scene. The high stakes competitive nature of these games necessitates consistency, not only in gameplay but in network connectivity and network clarity. The architecture of these games as a distributed system is very important in how it functions for its players. One such game is VALORANT, a relatively new competitive shooter that's system was uniquely developed. VALORANT solves problems of network latency in interesting ways and will be further delved into within this paper.

1. Introduction

1.1 Background

With the ever-expanding amount of technology that surrounds us from day to day, distributed systems are seemingly found everywhere. However, what exactly is a distributed system? Many differing definitions may be found, but it can be loosely defined as a collection of self-sufficient computers (or computing elements) that work together to form one coherent system [1]. The most important takeaway from this definition is that it is composed of multiple pieces (whether that be software or hardware) and it appears to be one single system. This can range from GPS systems, file sharing systems, or mulitplayer online games, which will be the focus of this paper. More specifically this paper will research the competetive first person shooter *VALORANT*, which was officailly released in June of 2020 by Riot Games.

1.2 Overview

This paper will discuss distributed computing in video games. The second section will cover distributed computing in mulitplayer online games, specifically that of first person shooter games (FPS) and the unique issues that arise with the genre. The third section will go over distributed computing specifically in VALORANT and what the developers' reasoning behind their decisions. Lastly, the fourth section will delve into how VALORANT was developed to be fair and competitive, and how that influenced the design process.

2. Distributed computing in multiplayer online games

2.1 Introduction

This section will cover how distributed systems are used in video games, specifically games with online matchmaking/multiplayer. Although there are many ways that one could implement online matchmaking, so this section will go over the topic broadly. When it comes to online games, there are many things that you need to worry about as well, such as latency issues, server problems, etc.

2.2 Distributed server architectures

When developing a multiplayer online game, it is very important to determine which architecture is best for what kind of game is being developed. The two most common architectures used are a client-server and peer-to-peer (P2P) [2]. The client-server model involves a system (the client) connecting to a central system (the server) in which information will be communicated through. With this setup many clients can be connected to a single server, and for this reason this architecture is used for many games, ranging from FPS to massively multiplayer online games (MMO). However, since this system has many clients connected to a single server, the server can undergo a high load, depending on traffic, and bottleneck the system. Furthermore, when a client performs an action, it must be sent to the server and then resent to the rest of the clients connected to the server, which increases the latency amount. Moving on, the P2P model has all the clients (peers) directly communicate simultaneously with one another, meaning that every peer is connected to every other peer. With this architecture there should be a decrease in the amount of latency that the clients would experience. However, since there is no central server node, each peer would need to store a copy of the current game state; and this would then necessitate some form of synchronization between all nodes. It is also worth noting that in this model for each new player/node added the number of links between nodes required would increase. For each new peer, the number of new connections needed would be equal to one less than the total amount of peers in the network, this could become very costly if you were expecting a lot of players in a single network.

2.3 Areas of concern with multiplayer online games

There are multiple areas of concern when it comes to multiplayer online games. One being inconsistencies. Due to servers/clients needing to sometimes communicate over large distances, certain interaction may be had between game states that differ on other servers, which leads to paradoxes [2]. A paradox is essentially when one saved game state (let's call this S_1) holds an action being taken while a different game state (let's call this $S_{2)}$ holds an action that would directly prevent the action on S_1 from taking place to begin with. This issue necessitates some sort of synchroniztion. Another issue lies within geographic scalability, that is to say, a system's ability to hide its communication with another system that is geographically different from its own [1]. This is an interesting obstacle when it comes to creating a video game, since it differs from your typical distributed system. In your more traditional distributed system, you can hide this latency between communication by having your client less relient on information from a server by moving certain calcualtions to the client side or by having the client perform other work while it waits for a reply. When it comes to multiplayer games however, the client relies on information from the server to keep the player up to date with what is currenlty happening, therefore it does not have the luxury to simply wait for a response. One way to tackle this is to set up many servers and have those servers handle the game in that particular region, this make is to player in that region are physically closer to the server lowering the response time — lowering the ping time, or simply ping. Lastly, there lies the issue of tick rate. Tick rate being the rate at which a system produces and send data simulations, this is measured in Hz [4]. Certain games don't require a high tick rate; such as real time strategy games, where players were observed to be content with up to a one second delay on their actions. However, some more fast paced games may require a higher tick rate in order to maintain its players enoyment.

3. Distributed computing in VALORANT

3.1 Introduction

This section will cover how VALORANT implements distributed computing. Specifically, how the system handles storing of data, handling the distribution of work, and communication with clients/players.

3.2 VALORANT's infrastructure

Some games implement what is referred to as a monolithic service, which is one single unit that handles the entirety of the network's server-side function. This monolithic structure is what League of Legends (Riot Games' first game, released in October of 2009) was initially modeled as. This monolithic structure was found to be problematic; a single bug in one subsystem had the potential to bring down the entire system, and the overall model lacked scalability. VALORANT's infrastructure, rather than a single monolithic contains two major pieces: the game server executable and the platform [5]. The VALORANT platform consists of many different microservices that handle every part of the game, other than the gameplay itself. Each server itself has access to its own instances of these microservices to run the game. Each of these microservices are also stateless, meaning that these services hold no information regarding its client. This acts as a benefit to their overall network since that means these microservices can take request from any players, therefore the servers can load balance the incoming requests by assigning them to the services round-robin [5].

3.3 VALORANT's architecture

VALORANT utilizes the client-server architecture, where the players connect to the server, where the servers are connected to the microservices (mentioned in the previous section) and databases (which will be covered in the next section). The servers themselves work by running different 'scenarios' based on player requests, an example being a 'matchmaking scenario' [5]. Basically, from its pool of online players, the server will take in requests from its players/clients and handle those requests by dispatching them to the corresponding microservice, where that service will run until it reaches a completed state and will return the players to the player pool. Built into the architecture is a way to balance the network load when there is an influx of users. As certain services need resources they will take from a cluster of shares, and as other services need to apply for resources they will "steal" shares from other services until all services have an equal number of resources [5].

3.4 Scalability of their databases and its advantages and disadvantages

Moving on to the databases, the development team built the databases with scalability in mind; specifically, their databases are scaled horizontally [5]. This means that each piece of player data has its own dedicated database, for example, the match history service will hold the player data of their past matches and only that data. This kind of model provides a couple benefits. The first being it allows them to scale up the databases independently. The second benefit is the reduction of issues by isolating data kept by each database, so an issue with one database will not affect the performance of another. However, by scaling horizontally, there is added complexity. The databases can no longer rely on atomic transactions to update several pieces of data. Rather, the use of several operations, lock states, and retry on failure is needed. Another benefit of this horizontal scaling is that it allows them to further break down their stored data into smaller sections by dividing data through player ID, so each database only needs to handle a portion of players. These tables are broken into 64 segments by hashing the player's ID [5]. Although breaking it down into 64 segments is not completely necessary, it enables its scalability to be much easier.

4. Competitive Integrity in VALORANT and the challenges it brings

4.1 Introduction

On release, VALORANT boasted its competitive and fairness, but what exactly did they mean by that? To put it simply, they meant that they took steps to mitigate lag and a problem unique to the competitive shooter genre: 'peeker's advantage' [6]. Peeker's advantage is a repercussion of the client-server model that they use. Mentioned before there is extra latency from communication from one client to the server and finally to the receiving clients. This latency is what makes peeker's advantage. When one player (P_1) is peeking around a corner where another player (P_2) is located, due to the latency on P_2, P_1 would see P_2 first and be able to react before, hence "peeker's advantage". On to how they mitigated the effects of peeker's advantage, VALORANT's servers run 128 tick servers. Most competitive games run servers at 64 ticks, and some at even less. So, with VALORANT servers running at basically double the rate of most other games the amount that peeker's advantage is noticeable is mitigated. How they accomplish such a high tick rate will be further discussed in this section. The next issue is lag, which they mitigate by using their own ISP and multiple servers across their supported regions.

4.2 Server tick rate

Other than peeker's advantage, there are other issues with having low tick rate servers. The effects of low tick rate are only exaggerated by peeker's advantage, but the same principle behind it is consistent throughout the game. An example being with hit registration with weapons fired by a player. When a player fires their weapon, the server is ahead of the player and needs to rollback to a previous state to match that of the players to check if the shot would hit. With slow updates it can seem as if on the client side a player is hitting another player when the server does not register this hit. So, the developers pushed for 128 tick servers [6], but how was this done? To reach a server tick rate of 128 they would need to process an entire frame in 7.8125ms [7]. However at this rate it would take one CPU core to run a single game, which on the hardware side would become extremely expensive. They needed a way to run at least three games per core, which would mean they needed to get one frame to be processed every 2.6ms, but this number is very conservitive since this processing time does not count the overhead for OS scheduling and other software that is running on the host server. This meant the actual number that they needed to shoot for was closer to 2.34ms per frame [7]. They accomplisehd this by optimizing their net code in order to cut unecessary processes and calculations. But one solution they found involved a unique solution involving the network. When the server was updating it needed to synchronize with its clients, and a part of this process involved checking every variable that needed to be replicated, was slow to compute. To solve this the developers levereges remote procedure calls (RPCs) in order to speed things up. So rather than synchronizing the game state, comparing it to all the clients connected to the game, and sending that information back to the clients, they would instead send RPCs to the clients based on the actions it received from the other clients.

4.3 Server latency issues

To combat latency and lag Riot Games set up many servers within the regions they supported as well as created their own ISP, Riot Direct, to support their games. First, Riot Games set up many servers to reduce the distance that players' data would have to travel to the servers. VALORANT is region locked, each account is only able to play within the region the account was made in; regions include North America, Latin America, Europe, Asia, etc. However, some of these regions span large physical spaces, so there may be instances were one player who is physically close to the server would have an advantage over a player that is physically far from the server. To combat this, each region has subregions where there are servers to handle a smaller section of the region (it is worth noting that players can choose to play on any subregion withing their region, but their subregion is defaulted to the closest physical one); subregions include, for example in North America, US West (Oregon 1), US West (Oregon 2), US Central (Texas), etc. By doing this they effectively reduced the amount of time that it takes for their players' data takes to travel to their servers. Riot Games' decision to build their own ISP was a unique, albeit costly, solution to latency. Riot found that sometimes the route that certain ISP took to route player's connection to Riot servers was longer than the physical distance from the player to the Riot servers. So, to combat the difference between the physical and actual difference, Riot worked with many different ISPs to connect a piece of fiber from their servers to their own Riot Direct servers, so they could better route player's connections [8]. With this the latency for some players was reduced significantly.

5. Conclusion

To summarize, multiplayer online games work as distributed systems, which can be implemented in different ways. However, the need to keep players updated consistently provides unique challenges. VALORANT, which works using the client-server architecture, found unique solutions to the issues with multiplayer online games. Riot Games developed this game in such a way as to create a competitive and fair environment and this was done through finding solutions to network latency issues and lag.

6. References

[1] A. S. T. Maarten van Steen, Distributed Systems Third Edition, Pearson Education, Inc., 2018.

[2] D. Liang, "An implimentation of multiplayer online game with distributed server architecture," Wollongong, 2006.

[3] M. A. Rubio, "What do 128-tick servers mean for Valorant's competitive play?," Daily ESports, 20 March 2020. [Online]. Available: https://daily.upcomer.com/what-do-128-tick-servers-mean-for-valorant-competitive-play/#:~:text=High%2Dfrequency%20servers%20(128%2D,luxury%20in%20most%20competitive%20games.&text=Implementing%20128%2Dtick%20servers%20means,on%20a%2064%2Dtick%20server.. [Accessed 5 May 2021].

[4] E. R. Tveteras, "Evaluating Loss and Letency Mitigation Techniques in a Tick-Based Game Server," University of Olso, Olso, 2018.

[5] K. Gunning, "Scalability and Load Testing For Valorant," Riot Games, 15 December 202. [Online]. Available: https://technology.riotgames.com/news/scalability-and-load-testing-valorant. [Accessed 6 May 2021].

[6] D. S. Dave Heironymus, *Netcode & 128-Servers // Dev Diaries – VALORANT,* Los Angeles: Riot Games, 2020.

[7] B. Randall, "VALORANT's 128-Tick Servers," Riot Games, 31 August 202. [Online]. Available: https://technology.riotgames.com/news/valorants-128-tick-servers. [Accessed 7 May 2021].

[8] Riot Games, "Riot Direct: Video," Riot Games, 2 May 2016. [Online]. Available: https://technology.riotgames.com/news/riot-direct-video. [Accessed 8 May 2021].