Low-Latency Gaming In An Ever-Expanding World  
John Miner

Long Beach, CA

Abstract- The continued advancements in computing have made it so that even milliseconds of latency issues are noticeable. Current infrastructure of the internet often leads to latency issues and gives people in more advanced areas of the world an unfair advantage over another gamer. Ticks has become the measurement for how well a server is performing and of is the leading factor in determining how serious a company is about being eSport ready. Esports, sponsorships, and clout are the three biggest concerns when it comes to making money and without a stable network you have none of these. Gaming companies once considered to be AAA rated program by their communities are quickly losing ground and new informal ratings are being applied to those that are considered top tier: AAA+ is now considered the highest rating a company is able to obtain. Time has shown that hackers and those that are extremely network/computer savvy will leverage weaknesses in code and other networks to gain every advantage.  
  
Index Terms- Distributed Architectures, Infrastructure, Neighbors, Machine Learning

I. Introduction

Many large companies use systems that are customized to fit that companies needs at the time of conception and often times get stuck utilizing that technology well passed it’s expected shelf life. The latest and greatest technology as far as networks are concerned are those that utilize distributed architectures. In order to improve the gaming experience we must be able to “playout latency independent of network conditions and adapt[s] to network congestion to optimize performance”[1] . The required data is not necessarily available or even noticeable to people, but through machine learning techniques it may be.

Since machine learning algorithms alone cannot solve all of our problems in computer networking we have to come up with ways to improve interactions between the users and their online environment. Some companies choose to let problems run rampant on their platforms because they refuse to upgrade servers while others charge a maintenance fee to keep theirs up and running. Utilization of unused resources through proper routing techniques can alleviate this problem.

II. The Algorithms

With machine learning algorithms “…what is the content of training data and what are the key factors decide quality and speed.”[2] . Because games are not manufactured for just one platform anymore there is a need for constant updates that will help to fix issues on the capabilities of each system. Utilizing something referred to as Neighbors we can help to speed up content delivery.

*A. Neighbors*

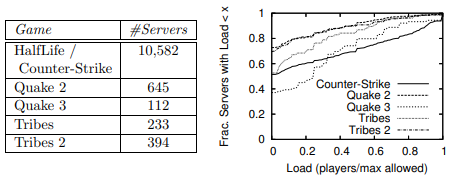
A neighbor would be considered someone that is in close proximity to the user and has or is accessing similar content. In the case of gaming if many people are playing the same games machine learning could use the “Top N Neighbors”[3] approach to find enough nodes to validate the data being used to improve system performance. The flipside would be if no one is really playing a game, the person is in a remote/rural area, or if the game has a fairly small user base there is little to no gain in this method. This is where threshold filtering is used to continue to find neighbors until enough qualified ones are found.

*B. Collaborative Filtering*

Collaborative filtering is a method that allows the machine learning algorithm to find similarities in the gamers that are not necessarily apparent. For instance someone that built their own PC and someone that ordered one online may have similar components and not know it. These similarities are good candidates for test data because they allow the algorithm to expand beyond any one specific system or platform. Traditionally collaborative filtering is used to provide recommendations to users on platforms like Netflix, Hulu, etc. but by looking at training data such as geographical region, peek usage days and times, and the components in each users platforms predictions can be made as to the max recommended settings each user should play at.

III. The Architecture

Each instance of a match within a game shares similar qualities. Multiple users are matched based on some sort of filters (whether user or manufacturer made). When the requirements of the match are met a “lobby” is created and now all of the gamers are locked into one server together until an exit condition is met. That exit condition can be anything from being kicked for inactivity to the user losing and exiting the match. No matter what once the user, node, leaves the lobby, distributed network, they are not allowed back in. Where most of these games differ is when it comes to who or what is hosting the servers and how do we ensure that the playing field is even. Video games like the ever popular CS-GO use user-based servers to provide their matches. In this case the network uses a centralized system to make lobbies and play matches which is great if you are the host. The problem is that the further you are in the chain the more latency issues you are bound to encounter. If a more distributed model were used to connect these servers better results could be reached as suggested by the following figure.



Figure(1) [3]

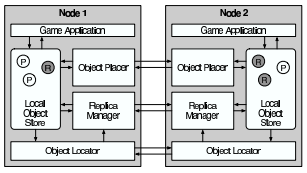
The question now becomes how do large companies allow users to connect using distributed architecture no matter how far apart they may be. Using the Top Neighbors method allows users to discover nodes along the way and can calculate average latency between users and match those with similar connection speeds to each other. When a node connects to its lobby it can be assigned a name that it will carry throughout this instance. As players, nodes, are removed updates are continuously provided to each server that has been created until there is only one instance of a node remaining. Throughout each lobby there are two main concerns that will constantly be in need of an update, what is left in the game (I will call this the state) and the actual interface.

*A. Node State*

On March 23, 2017 an indie game by the name of PUBG was released. People instantly flocked to it because of it’s amazing interface and realism, but as the number of players grew the servers became overloaded and would often crash. The amazing thing was that if you were disconnected your state was saved and anything that did or did not happen would continue as long as the server continued to run. This is because the creators of the game utilized a distributed architecture so when the node left the lobby their state was already replicated and they were allowed back into their position. This also created a problem. If a server’s state and a players state is always being replicated then eventually memory becomes an issue and performance issues are felt my all nodes connected. The solution was, and will continue to be for all of eternity, to ensure that as things become irrelevant to the lobby’s current state they are removed properly from each individual instance. In PUBG’s case they were removing the node once it was no longer in the game, but due to the nature of a distributed network with gossip capabilities the removed states were being placed back in and resources were being used to now update these “new” things into the game rather than being used to make the game continue to run smoothly. To fix this PUBG pushed an update to all of it’s users that would issue what is known as a death certificate once things were no longer relevant to the current state of the lobby. This certificate would visit and remove whatever needed to go until it confirms that the item or node no longer exists and would in turn remove itself.

*B. Interface*

How each player is able to interact with each game once they are in a lobby is extremely important. If a distributed model were to be used it would require constant updates. Going back to PUBG, this game requires constant updates to the player’s health, location, kills, damage, speed, and distance from one another just to name a few.



Figure(2)[3]

Figure 2 is a proposed model of how the Colyseus architecture is used to interact between the player nodes. A system like this allows the game to continually update and share things like the player’s location, their inventory, their health, and any items that the players may interact with.

IV. Ticks

For every player the most important thing is that if you were to lose it was because the other player was better. For most this is not the case. No matter how good of a system you have and no matter how good the internet you pay for is, ultimately it comes down to how much processing power the servers have. The unit of measurement for how many times the server is updated per second is referred to as a tick. A game like CS-GO typically runs off of a 64-tick server and can move one a 128-tick server for competition. This is fine because there are a small amount of players at any given time in a lobby, but for a game like CoD’s Warzone it is a whole different story. Activision Blizzard has reported that their game only uses 22-tick servers and 12-tick for a custom made match. The main reason being the financial difference to run CoD’s 150 player per match battle royal. Ticks are something that need to be considered when determining how many nodes are allowed to connect at a time. This becomes an issue in gaming and can be confused with latency issues because the server is only able to update a certain amount of times per second and can make it seem like your shots did not register. What actually happens is the server takes in what it can from the server and puts itself into a block state until it is able to receive again and will miss the input completely so the other person doing something as jumping or sliding while shooting can hog all of the resources and block the other player from completing their action.

So the solution for something like this would be to utilize the distributed network to receive replies from users. Theoretically, in a distributed system you could hash out the data and distribute it to peer nodes and bring them back on the other end from multiple connections. The difficult part would be how do you choose which peers get what data and how do they keep their own data separate from other player’s.

V. Conclusion

Although the effort would be a multi-disciplined one the direct solution to reduced latency in online gaming would involve establishing better decentralized networks. Distributed networks allow for otherwise catastrophic events like loss in internet connection to do little damage to a network and makes it so that the lost node can now rejoin their server state.

Hardware costs can be significantly reduced if the software and middleware is upgraded now utilizing the beauty that are distributed networking, processes, threads, and advanced algorithms.

VI. Abbreviations

CS-GO: Counter Strike Global Offensive, PUBG: Player Unknown’s Battle Grounds, CoD: Call of Duty

References

[1] GauthierDickey, C., Zappala, D., Lo, V. and Marr, J., 2004. *Low Latency And Cheat-Proof Event Ordering For Peer-To-Peer Games | Proceedings Of The 14Th International Workshop On Network And Operating Systems Support For Digital Audio And Video*. [online] Dl.acm.org. Available at: <https://dl.acm.org/doi/10.1145/1005847.1005877> [Accessed 10 August 2020].

[2] Wu, G., 2016. *A Continuous Dataflow Pipeline For Low Latency Recommendations*. [online] Diva-portal.org. Available at: <https://www.diva-portal.org/smash/get/diva2:896151/FULLTEXT01.pdf> [Accessed 10 August 2020].

[3] Bharambe, A., Pang, J. and Seshan, S., 2005. *A Distributed Architecture For Interactive Multiplayer Games*. [online] Cs.cmu.edu. Available at: <https://www.cs.cmu.edu/~ashu/papers/cmu-cs-05-112.pdf> [Accessed 10 August 2020].