Examining the Role of Networking in Games

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Logan Ramsey

May 13, 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Logan Ramsey

Capstone advisors: Mark Floryan, Department of Computer Science
Samira Khan, Department of Computer Science

Examining the Role of Networking in Games

Logan Ramsey
University of Virginia
lr8fm@virginia.edu

ABSTRACT

Online gaming began to develop as a part of the gaming experience during the 1990s and 2000s due to the emergence of the World Wide Web making the Internet more accessible. Today, most video games are designed with some online gameplay component. Additionally, in the last decade, the eSports industry has started to see growth, and developers are seeing the potential in designing games targeted to more competitive players. With the projected growth in online gaming in the coming years, it is important that game designers ensure that they can provide a satisfactory gameplay experience over the Internet that does not compromise their design choices. In the past, developers often used delay-based techniques for their online games to maintain a stable online experience. Rollback networking is an alternative that game developers have been adopting to improve the issues presented from delay-based techniques. This paper will analyze each of these techniques, how they impact game design and the online gaming experience, and propose research to further study the impact of rollback networking on gameplay.

1 INTRODUCTION

Online gaming started to see growth in the 1990s due to the Internet becoming more available [1]. While online gaming has become an incredibly popular and commercially successful industry over the years, designing a networked game that satisfies the demands of players is a difficult task. Since transmission of data over a network introduces new variables such as delay and packet loss, the online gameplay experience frequently differs from playing offline. The degree to which this difference is felt by players is dependent on factors such as distance and the network infrastructure which is more often than not, not within their control. For players looking to play online to practice for competitive events or players simply looking for a consistent experience, the contrast between the online and offline experiences is a major issue.

Particularly, players are highly susceptible to performing worse in online games with strict timing. In particular, games such as first-person shooters and sports games have among the "lowest tolerance in terms of delay since players have direct control of their character" [2]. Additionally, cooperative games that require interaction between players are sensitive to network quality with delays beyond 100 milliseconds and jitter higher than 50 milliseconds being unacceptable [2]. For "power gamers" or players that are "instrumental, focused, and goal-oriented in ways that often exceeded the bounds of the game", it is crucial for online games to have responsive inputs and be consistent [3].

There are techniques that exist to deal with networking in online games. However, the techniques that developers use to implement online multiplayer in games is game specific, and the quality at which they are implemented varies between developers. The primary problem that needs to be addressed is determining how to ensure that networking solutions used by developers are more consistent.

2 BACKGROUND

2.1 Client-Server

The client-server architecture is the commonly used structure for networking in multiplayer games. In this architecture, a single authoritative server runs the game logic. The client samples user input and sends that data to the server to execute [4]. Once the server receives user input from the client, it executes the input commands, moves around objects in the game environment, and sends back a list of objects to render to the client [4]. The primary advantage of this architecture is that it is simple way to handle concerns such as game state management, access control, and synchronization of players [5]. However, using a client-server architecture introduces additional costs for deployment, operation, and maintenance, limitations on system scalability and availability, and more delay which makes it a suboptimal solution for some games [5].

In order to prevent long round-trip times from affecting gameplay in client-server games, client-side prediction was introduced. When using client-side prediction, the client's commands are performed locally, and it is assumed temporarily that the server will accept and acknowledge those commands [4]. If the prediction is incorrect, the client's simulation of the game will be corrected once the server responds. The time in which the server's correction

will occur is dependent on the amount of latency. In connections with a lot of latency, the correction can come long after the initial prediction which can cause jarring shifts in the client's simulation [4].

2.2 Peer-to-Peer

Developers looking to avoid the limitations and concerns associated with the client-server model use the peer-to-peer model instead. In a multiplayer game using a peer-to-peer architecture, each peer has control over its game states, and this is handled with overlays. There are two types of overlays: static overlays and dynamic overlays. Static overlays are predefined with each peer getting their own objects, players, or zones to manage regardless of the interests of each player [5]. Dynamic overlays, on the other hand, construct the overlay for peers based on their interests and interactions between players [5]. In ideal conditions, a peer-to-peer model can work well. However, issues with information consistency become apparent when delay is introduced [5].

Peer-to-peer games do not have the benefit of having an authoritative server to handle delay and game state synchronization. Therefore, the choice of implementation for networking into a game becomes a bit less clear. Game developers will use of one of two techniques. The lockstep protocol is a stop-and-wait protocol that waits for the slowest player to send its inputs before updating the game state [5]. This was introduced to mitigate cheating in online games and is the method traditionally used to deal with synchronization [5]. The issue with using this protocol is that the quality of the game will be decided by the player with the slowest connection, and it is subject to freezing if the roundtrip times take too long [5]. The alternative is to use rollback networking which acts similarly to client-side prediction. Instead of waiting for each player's inputs to be sent over the network, local inputs are processed immediately and prediction is used to determine remote inputs [6]. In the event that the remote input is incorrectly predicted, the game rewinds and updates the game state with the corrected inputs [6].

3 RELATED WORK

3.1 GGPO

GGPO is a middleware solution developed by Tony Cannon that provides a framework for implementing rollback networking in arcade style games. The middleware is meant to isolate the developers from the networking details. In order to use GGPO, Cannon suggests separating the game state from rendering.

In a basic game loop for an arcade game, the game will sample controller input, update the game state, and render the scene [7]. With GGPO, the game loop is modified. The game inputs are synchronized, local inputs are transmitted to remote players, and merging predicted and actual remote inputs for all remote players [7]. Once the inputs from each player are collected, GGPO will then compare the predicted and actual inputs and adjust the game state [7].

GGPO utilizes speculative prediction to remove the perceived input delay from each player. This is achieved through the concept of frame advantage. Frame advantage is the number of frames that the local player is ahead of the remote player [7]. The formula used to calculate the frame advantage is shown below.

$$frame\ advantage = (last\ remote\ frame\ + \left(ping * \frac{frame\ frequency}{2}\right)) - (last\ local\ frame)$$

GGPO ensures that the frame advantage between the two players is within a one-frame tolerance by slowing down the game for the player with a higher frame advantage [7].

3.2 Mortal Kombat X

While integrating GGPO could be a viable option, many developers are looking to craft solutions specifically designed for their games. In a presentation at the 2019 Game Developers Conference (GDC), Michael Stallone, software engineer at NetherRealm Studios discusses the process of retrofitting rollback networking in Mortal Kombat X [6]. He estimates that this process took around two man-years to implement. Based on response from players, the rollback system seems to work comparatively better than the delay-based system that was used previously. The rollback system has 3 frames of input latency and 10 frames of network latency (333 milliseconds) with a 7-frame rollback window.

From a game design perspective, implementing the system posed unique challenges that did not exist with the delay-based system. In order to implement rollbacks, a previous state of the game must be saved. Therefore, optimizing serialization systems proved to be a significant part of implementing rollback networking. Initially, the developers used a singly-threaded solution for performing rollbacks. However, since the goal was to handle rollbacks within a single frame, this method was not efficient enough. Consequently, they switched to parallelized serialization and only saved mutable data. If no actions happened within the rollback window, then no data was saved. Additionally,

Stallone stresses the importance of handling object creation and destruction when serializing data. He mentions that a naïve solution would result in objects constantly crossing the construction and destruction edges.

Ensuring that audio and visual effects were consistent during rollbacks posed another challenge for developers since these effects can be non-deterministic. Rolling back audio and visual effects could cause "pops" during gameplay. To avoid this, Stallone found that it was better to hash these objects. Every time that the game would roll back and play through the creation edge, it would check to see what was hashed. If the situation was the same, then the game would reuse that object.

4 IDEA

While GGPO and rollback networking have proven to be an effective way to develop some online games, much of the known implementations have been limited to fighting games. Considering that prediction techniques such as dead reckoning are used in other games, the methods for prediction presented by fighting games may be applicable to games in other genres that use a peer-to-peer architecture. However, since every video game genre has different design requirements and constraints, the prediction schemes will likely not be a fully compatible solution for online multiplayer. In fighting games, there is an element of consistency with every attack having startup, active, and recovery frames [7]. This means that lag can be masked into the startup frames of an attack [7]. The process of concealing the effects of lag in games with other variables may be different. Therefore, implementing prediction will likely require making adjustments to the prediction algorithm.

5 HYPOTHESIS

The main hypothesis is that implementing rollback networking in games other than fighting games will require a modified prediction scheme compared to the one used in GGPO and fighting games like Mortal Kombat X. The other hypothesis that will be tested is that using more than the previous frame to perform prediction will improve consistency.

6 METHODOLOGY

To test the correctness of my hypotheses, I will develop a simple 2D platformer that supports rollback networking. This will be implemented using Unity, a cross-platform game engine that supports peer-to-peer networking by default.

7 PLAN

Below is a plan of proposed work:

- Design a 2D platformer that uses a peer-to-peer architecture and implement rollback networking using a prediction algorithm similar to GGPO
- Sample individuals to test the implementation and describe how it feels
- Modify the prediction algorithm to the use more frames to perform a prediction
- Sample individuals to test the implementation and describe how it feels
- Analyze the results and determine the effectiveness of both implementations

REFERENCES

[1] Katie Jones. 2020. Online Gaming: The Rise of a Multi-Billion Dollar Industry. Retrieved November 8, 2020 from https://www.visualcapitalist.com/online-gaming-the-rise-of-a-multi-billion-dollar-industry/

[2] Anastasiia Beznosyk, Peter Quax, Karin Coninx, and Wim Lamotte. 2011. Influence of network delay and jitter on cooperation in multiplayer games. In Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI '11). Association for Computing Machinery, New York, NY, USA, 351–354. DOI:https://doi.org/10.1145/2087756.2087812

[3] T. L. Taylor. 2020. The Rise of Massive Multiplayer Online Games, Esports, and Game Live Streaming. American Journal of Play 12, 2 (Jan. 2020), 107–116, https://files.eric.ed.gov/fulltext/EJ1255270.pdf

[4] Yahn Bernier. 2003. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. Retrieved from https://web.cs.wpi.edu/~claypool/courses/4513-B03/papers/games/bernier.pdf

[5] Christoph Neumann, Nicolas Prigent, Matteo Varvello, Kyoungwon Suh. 2007. Challenges in Peer-to-Peer Gaming. Retrieved November 8, 2020 from http://ccr.sigcomm.org/online/files/p2p_gaming.pdf

[6] Ricky Pusch. 2019, Explaining how fighting games use delay-based and rollback netcode. Retrieved from https://arstechnica.com/gaming/2019/10/explaining-how-fighting-games-use-delay-based-and-rollback-netcode/

[7] Tony Cannon. 2012. Fight the Lag! The Trick Behind GGPO's Low-Latency Netcode. Game Developer. September 2012. 7-13

[8] Michael Stallone. 2018. 8 Frames in 16ms Rollback Networking in Mortal Kombat X and Injustice. Video. (Mar 11, 2019). Retrieved November 8, 2020 from https://youtu.be/7jb0FOcImdg