Rollback Netcode, Implmentation and Adoption

Edward Boulderstone

School of Computing Science, Newcastle University, UK

Abstract

Rollback netcode is a peer to peer networking soultion. It had potential to improve online experiences by minimizing the effects of latency over traditional delay based . However the implementation of rollback netcode in the fighting game industry has been slow and difficult. This project aims to understand understand how the industry has had difficulty implementing rollback, and find any optimizations that can be made to existing public rollback netcode understanding.

Keywords: rollback, netcode, peer to peer, fighting games, networking, industry.

1 Introduction

The goal of networking in video games is to allow people from all over the world to play with each other. However fans of fighting games have historically (Before the pandemic) gone out of their way to organise local tournaments with most major tournaments before the pandemic taking place off-line[2]. This trend started because of the when fighting games were played in arcades in the mid 1990's,[3] and continued until the start of the pandemic, in part because fighting games rely on consistent timing and a low latency environment [7], [8]. The existing networking solutions at the time simply did not provide this environment for competitive play[4]. However the spread of the corona virus in the mid 2020's fighting games found themselves having to fall back onto these "lower quality" online platforms. Games which did not have well optimized netcode found themselves on the back foot, with reduced attention [5], and games with well written netcode found their success [6].

1.1 Delay Based Netcode

The first solution to online play for fighting games was a peer to peer system known as delay based netcode. Peer to peer netcode is important for fighting games because of the aforementioned need for low latency. Because most fighting games are

[1] Email: E.Boulderstone@ncl..ac.uk

between two players [10], introducing a server will increase the ping between the two players, as the information has to travel to a seperate location, before being sent to the opposing player. Delay based netcode works by keeping two players games identical, by waiting for the remote player's input, before running the game update

1.2 Rollback Netcode

Rollback netcode was developed in 2006 as a solution to the problems with delay based netcode. [9]. It works on top of existing delay based netcode by adding rollback frames, predicting the remote users inputs. When the remote user's input arrives if it matches with the predicted input, nothing happens, however if there's a discrepancy the game will "rollback", to the point of the discrepancy and re-simulate the game state back to the real time frame as shown in figure 1.

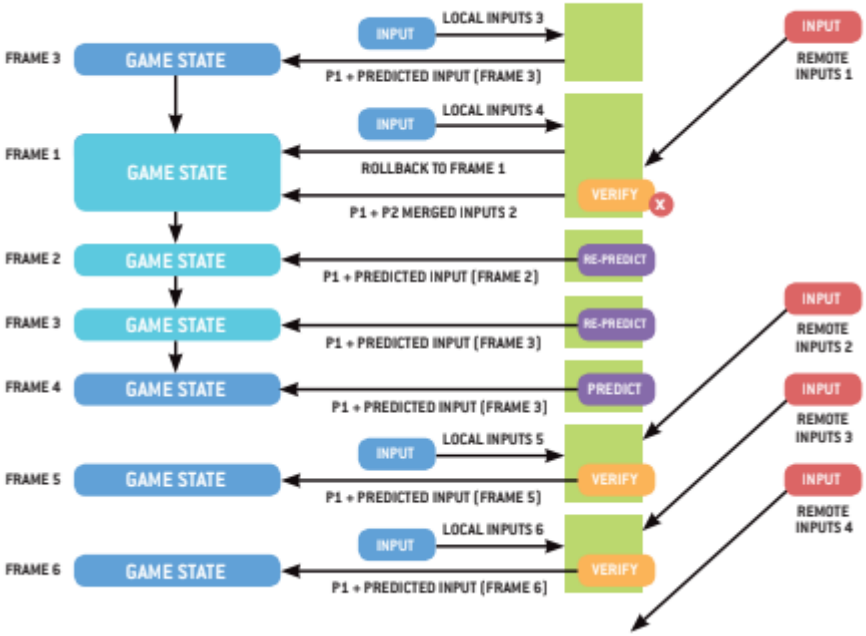

Fig. 1. Rollback netcode [1]

1.3 Difficulties in industry

In today's fighting game market, many games have rollback [13].However there are still notable exceptions such as:

- Super Smash Bros. Ultimate[14]
- Granblue Fantasy Versus[15]
- Under Night In-Birth[16]
- Samurai Shodown[17]
- Soulcalibur VI[18]
- Dead or Alive 6[19]
- EA Sports UFC 4[20]
- Dragon Ball FighterZ[21]

Other fighting games have had difficulty in implementing rollback, such as Street fighter V and Mortal Kombat X. These difficulties with implementing and developing rollback are the basis of the motivation of this paper.

1.4 Aim

To investigate rollback netcode, it's usage in industry and any short comings of existing public rollback netcode infrastructure

1.5 Objectives

- Understand rollback netcode and the effects on the games it's implemented in.
- Create a visualization for the differences between rollback and delay based netcode.
- Research the difficulties of implementing rollback in existing games.
- Explore optimizations for the existing public rollback structure.
- Investigate further uses of rollback netcode, in the wider video game industry.

2 Background

2.1 Peer to Peer Netcode

Peer to peer networks over the internet communicate data in packets which can run into the following issues.

- Packets take time to reach their destination (Network Latency).
- Packets can get lost on their way (Packet Loss).
- Packets can get there, but have their data corrupted (Corruption).
- Computers can run at different speeds.
- Computers can occasionally get hung up on doing things and skip a frame or two.

In solving these issues, not only does the quality of the user experience for online play, but it also allows for more effective matchmaking soultions, by increasing maximum tolerable connection quality.

2.2 Delay-based netcode

2.2.1 Concept

Delay based netcode works by keeping both players games in lockstep, meaning that each player's simulation of the game are waits for the input of the remote player, before simulating the frame[11]. This system works well when latency is not a major factor, for example, in a turn based game, where the inputs are spread apart by seconds, a pause of half a second may go unnoticed. Input latency frames are added to compensate for the Network Latency, and allow the local inputs to be sent before the frame they would be read on as shown in figure 2.

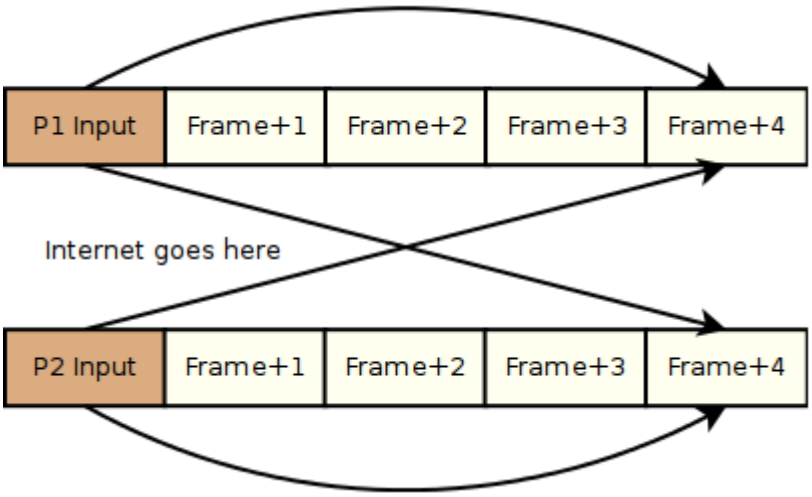

Fig. 2. 4 Frame Input Delay Example [12]

The Input delay to be used can be calculated as follows:

$$InputLatencyFrames = Ceiling(\frac{RoundTripLatency}{2 * FrameDuration}) + Constant$$

Where the constant represents additional frames of delay to compensate for variance in Network Latency.

2.2.2 *Improvements*

However this model does not take into account potential for packet loss/ corrupted packets, which may lead the actual states to progress as shown in figure 3

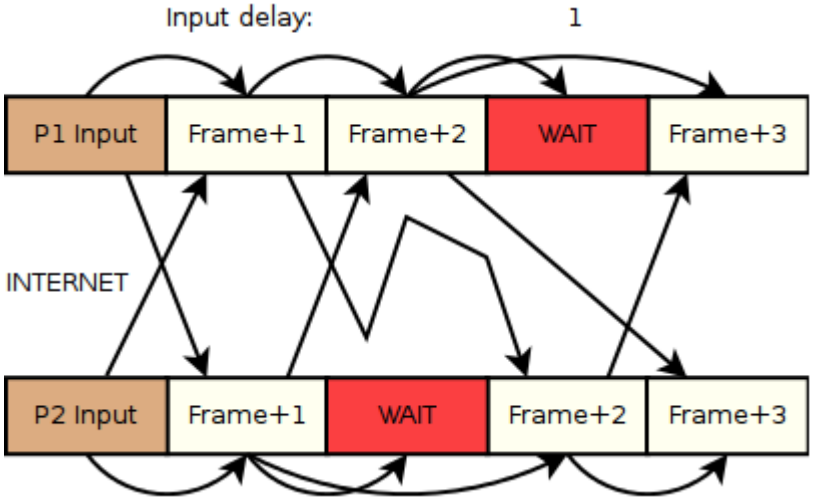

Fig. 3. Packet Loss's Affect on Game State progression [12]

One unexpected side effect of packet loss with single input packets is knock on delays, where by a delay in one game, makes the frame process later, so the inputs for that frame are postponed, delaying the origins game.

To combat packet loss and data corruption multiple inputs can be sent with one packet, extra frames of input delay can be added, and inputs can be sent multiple times, as shown in 4

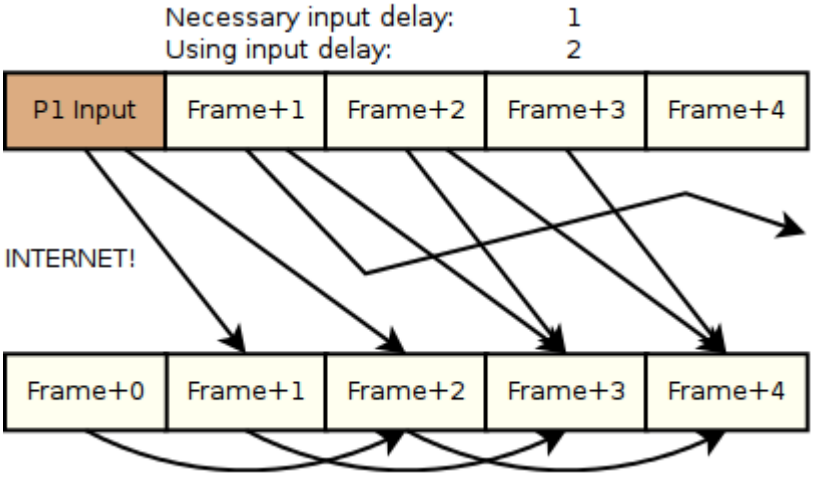

Fig. 4. Potential Solutions to Packet Loss in Delay Based Netcode [12]

Another improvement to made to delay based netcode is to account for Network Latency variance with a dynamic number of delay frames. way to account for Network Latency variance is to have a dynamic number of Delay Frames, varying with average Latency[7]. This can hide stutters, and remedy degrading network conditions causing consistent pauses. However, consistency for fighting games is the one of the highest determiners on network quality[22].

2.2.3 Evaluation

When there is minimal Network Latency variance, delay based netcode functions similarly to a non networked experience, with increased Input Latency. This is the ideal case for delay based networking. However, when the Network Latency variance is high, the game can freeze at seemingly random moments, removing agency from the player in the middle of a game. This breaks user imersion and significantly decreases the quality of the experience[7].

Ultimately Delay based netcode functions as network soultion for high quality connections, but struggles when operating as connection quality degrades. [23].

2.3 Rollback netcode

2.3.1 Concept

Rollback netcode takes the existing framework of delay based netcode and builds on top of it. Instead of waiting for the remote user's input to simulate a frame, the game predicts it. is These predictive frames By not requiring the input of the remote user to simulate a frame, it can tolerate late and lost packets better than delay based [1]. When a packet is lost or late, the rollback frames act as a buffer preventing the from game freezing.

2.3.2 Extra functionality required

Implementing rollback is not as simple as just changing your existing networking solution, these things need to be added

- separate gameplay from rendering
- serializable game state
- loadable game state
- game state simulation for multiple frames in the duration (time) of a single frame

2.3.3 Issues In theory

Because rollback doesn't require the remote user's input to simulate the frame, it comes with it's own associated issues.

- Particle Simulation. Because particles are often expensive computationally to process, so preventing deletion of the particles until the roll-back buffer expiries, and only generating them until their existence is confirmed from a remote input
- Sound Effects Typically sound effects are loaded as files and sent to audio drivers, with no ability to pause the effect or play from a specific time stamp. To allow for rollback without jarring audio effects they will have to be handled differently.
- Desync Detection Naturally rollbacks occur because the remote input state (and thus game state) disagrees with the local game state. Desyncs in rollback netcode happen when confirmed (non-rollback) frames differ.
- Optimization Because rollbac can require the simulation of multiple game state frames in the space of a normal frame duration, the game will likely have to have the capability to process game state updates faster than delay based solutions.

2.4 Evaluation

Ultimatly Rollback on paper seems like a superior solution for fighting games as shown in the figure 5. table from https://youtu.be/7jb0FOcImdg?t=623

	Rollback	Delay Based
Simple		X
Visually Smooth		X
Performant		X
Robust	X	X
Low Bandwidth	X	X
Responsive	X	
Single Frame Latency	X	

Fig. 5. Table comparing differences between Rollback and Delay Based Netcode[25]

Made even easier to implement given an open source solution under the MIT licence [**?**].

3 Industry Issues

3.1 Street Fighter V

https://www.youtube.com/watch?v=OtSveL7X6xg

Dynamic Network latency frames can cause issues with rollback. Didn't design some game mechanics around rollback, so movement is hard to react to. Mashing causes latency, even though inputs don't matter (Don't send them)

https://www.youtube.com/watch?v=qYPBasy5STg One sided rollback got patched out

https://www.youtube.com/watch?v=9ENocn-x0Ws Syncing between systems didn't take latency into account = one sided rollback guaranteed

https://github.com/AltimorTASDK/SFVNetcodeFix Netcode fix Constantly resyncs if drift

https://github.com/fluffysheap/SFVNetcodeFix Montiors connection and evaluates if resync is needed, takes into account wifi jitter can hide resync when the game is paused

3.2 Mortal Kombat X

Identify QOS cut off (max ping) Added after lauch, huge performance issues solved with: Multithreading, improved object handling (deferred deletion), evaulation of fixed cost, profiling tools !, optimizing game play loop as it in the worst case is

going to be replayed multiple times. Turn off predictable(?) things each frame (by predictable I mean, f(sum(inputs)) = sum(f(inputs)). Need a seperate resimulated gameplay loop (desync detection). Only need to save confirmed frame (good for worst, bad for average) (save simulation mid point calculate performance based on rollback frame and particles generated this frame) precompute! Really fuzzy matching on non gameplay effecting effects i.e. audio, particles with no hitboxes Game rollback max really infrequently. Humans input speed max = find sc2 max apm Possible to slow fps of game if struggling to get performance instead of pausing Design systems to update with variable time steps defer processing until guaranteed response i.e. camera cuts, particle spawning 7/8 man years to implement (4/5 optimization, 1/2 serialization,

3.3 Third party Rollback

- Slippi - Fightcade (GGPO)

4 Rollback Visualization tool

To demonstrate the effectiveness of Rollback vs Delay based netcode, a simulation of the two different styles of netcode was made.

4.1 Design

The high level design of this visualization tool was to have two side by side game clients, communicating to each other through simulated latency and netcode. Because the visualization tool was just supposed to be a demonstration, no overly complex mechanics would be implemented. It would just be two avatars that could move and jump.

Unity was chosen as the engine to develop the simulation, as designing from an existing game engine, would make development faster, and the developer had experience in using unity before.

The user would have the options to change the effect of network latency on the simulation(s) in real time, and the properties of the netcode, such as Input Latency, Rollback frames and packet loss %. The overall design of the visualization tool is shown in figure 6.

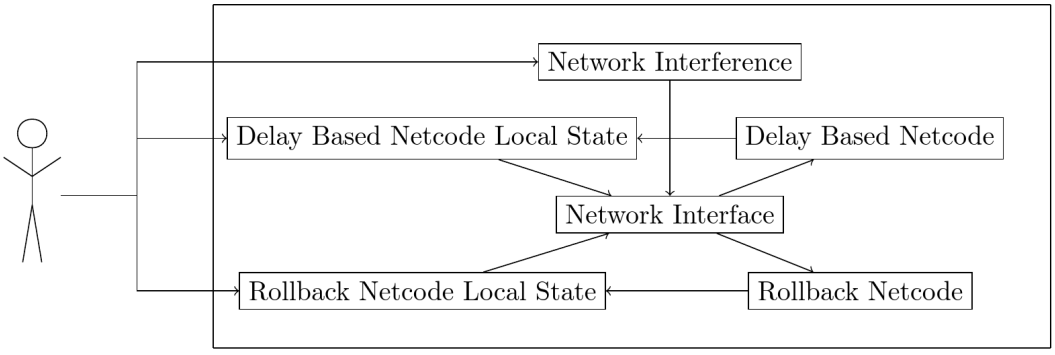

Fig. 6. Data Flow Diagram for Rollback Visualization tool

4.2 Implementation

Initially the packet structure was chosen to be a single frame's input, however during development I realized that the packet structure improvements mentioned in **??** could be applied to combat packet loss.

State serialization was surprising simple, as the only two dynamic elements of my game were the two players. Meaning that the only properties I had to save would be their location and velocity. If the game had more functionality, such as a health system, or audio effects, which are present in most modern games, extra systems would have had to be implemented on top to support dynamic saving and loading of state.

Experiment With packet loss solutions

How seperate rendering from game state in unity? = Don't allow render texture to update

If a players inputs cannot change the state, then the simulation can continue without them. This can be used to slightly reduce the average amount of rollbacks and freezes in both netcode implementations. To implement this idea, a frames till actionable variable was added to the packet data, and the delay based netcode would not freeze for the remote user's inputs / rollback would not re-simulate frames if the guessed inputs were different whilst the frames till actionable value was larger than 0. This could also have further adaption in rollback netcode, where only a certain number of the remote's inputs could matter, minimizing potential rollbacks.

I noticed desyncs happening, between my adjacent simulations, despite inputs being identical. In order to try to fix this, I made more thorough re-jump detection, reducing parallelization in the code, manually ran physics updates for a set duration in stead of relying upon unity's fixed updates. Ultimately to no prevail. So I had to chose between redesigning unity's physics engine (the presumed source of lack of determinism) for deterministic behaviour or re-designing my visualization tool. I chose redeisgn as I could felt I could demonstrate the difference in delay based and rollback netcode from a users experience, if they were allowed to toggle between the different types of netcode.

reworked design

Networked state — / V Netcode handler (recives input from game states and handles communication between them) - Network interfence customiser, Rollback, delay based, remote input 1 game states / 1 local player 1 remote player local input remote input subclass of local input

made own fixed size buffer class to prevent excess memory being used Rollback always saving state, so if swap, can load prior states Delay based code explained Rollback code explained pause game when rollbacks happen

having issues with frames taking longer than 16ms (Turns out unity overhead) simulate max rollbacks 100% of time needed extra for game state from rollback explained

Buffer sizes?

Implement ideas from delay based netcode packet soultions = expanded packet size

4.3 Evaluation

Unity no determinsim (redo engine?) Rollback hard even for simple projects Didn't implement Audio or particle systems Better ping modelling This simulation could also allow for testing of various improvements of the netcodes, such as multiple packet sending, increasing frames sent, etc.

5 Improvements for Rollback

I propose a theory for rollback frames

$$rollbackframes = ceilingmaxacceptableping2 * frameduration - delayframes)$$

for dynamic: dynamically changing rollback = minimal effect, (Problem with maxing rollback? = Divergence and hardware doesn't support rolling back that far in 1 frame (16ms)) however try to keep delay static

5.1 Prediction Quality

https://www.diva-portal.org/smash/record.jsf?pid=diva2

-Naive is 90% accurate https://www.youtube.com/watch?v=k9JTIn1SVQ4&t=1888s hard to run expensive algorithms alongside large rollbacks (cite MKX talk)

5.2 Packet Contents Optimization

Delay based strategies] How many frames to send, request for frame? = delay frames? + constant?

5.3 Input Locking

No need for rollbacks when input doesn't matter, i.e. hit stun, hit pause, cutscene, jumping, in a move past a cancel window

5.4 Conculsion

These require further testing

6 Wider use case of rollback netcode

Rollback ideas that only apply to fighting games? (Visual teleportation?, Expensive to implement?,) RTS (Including sports games) FPS - https://ieeexplore.ieee.org/abstract/document/79 Prediction quality: - Connection 'quality'. This doesn't mean bandwidth, but rather how much a connection's response time fluctuates. This has improved in modern times, making rollback a much more viable option than it once was. - The type of game. Because rollback needs you to be able to revert the state of the game in some cases, trying to implement it in a game with hundreds of different units all acting at once is extremely challenging (not to mention very offputting for the player). As a result, it's more suited to fighting games and other games with fewer player-controlled assets flying around at any given time.

7 Conclusions and further work

Rollback hard to implement, but not too hard to be impossible, newer games not releasing without it. Lessons learnt. Probably not applicable for wider use case.

furture work: Non-game applications? Expand upon demo

References

[1] https://drive.google.com/file/d/1nRa3cRBQmKj0-SEyrT_1VNOkPOJWNhVI/view or https://web.archive.org/web/20220101162600/https://drive.google.com/file/d/1nRa3cRBQmKj0-SEyrT_1VNOkPOJWNhVI/view Tony C., GGPO Game Developer Magazine's article. 2012. (Visited 25/05/2022)

[2] https://liquipedia.net/fighters/Tier_1_Tournaments or https://web.archive.org/web/20210121113554/https://liquipedia.net/fighters/Tier_1_Tournaments Liquidpedia list of major fighting game tournaments. (Visited 25/05/2022)

[3] https://www.usgamer.net/articles/the-oral-history-of-evo or https://web.archive.org/web/20220321000753/https://www.usgamer.net/articles/the-oral-history-of-evo John L., The Oral History of EVO: The Story of the World's Largest Fighting Game Tournament (Visited 25/05/2022)

[4] https://esportsinsider.com/2021/10/can-fighting-games-become-a-mainstream-esport/ or https://web.archive.org/web/20220516060805/https://esportsinsider.com/2021/10/can-fighting-games-become-a-mainstream-esport/ Elizbar R., Can fighting games become a mainstream esport without abandoning grassroots?. (Visited 26/02/2022)

[5] https://www.ssbwiki.com/COVID-19_pandemic_and_its_impact_on_competitive_Smash or https://web.archive.org/web/20220516020551/https://www.ssbwiki.com/COVID-19_pandemic_and_its_impact_on_competitive_Smash List of smash tournaments affected by the pandemic. (Visited 26/02/2022)

[6] https://techraptor.net/gaming/opinions/guilty-gear-strive-changed-fighting-games-rollback-netcode or https://web.archive.org/web/20210713144527/https://techraptor.net/gaming/opinions/guilty-gear-strive-changed-fighting-games-rollback-netcode Davi B. Guilty Gear Strive Might Have Changed Fighting Games Forever. (Visited 26/02/2022)

[7] https://arstechnica.com/gaming/2019/10/explaining-how-fighting-games-use-delay-based-and-rollback-netcode/ or https://web.archive.org/web/20220425211823/https://arstechnica.com/gaming/2019/10/explaining-how-fighting-games-use-delay-based-and-rollback-netcode/ Ricky P. Explaining how fighting games use delay-based and rollback netcode. (Visted 26/05/2022)

[8] https://www.polygon.com/2020/3/25/21192522/netcode-samurai-showdown-fighting-games-rollback-delay or https://web.archive.org/web/20220401231646/https://www.polygon.com/2020/3/25/21192522/netcode-samurai-showdown-fighting-games-rollback-delay David C. Bad netcode is killing many of your favorite fighting games. (Visted 26/05/2022)

[9] https://gamasutra.com/view/news/34050/Interview_How_A_Fighting_Game_Fan_Solved_Internet_Latency_Issues.php or https://web.archive.org/web/20220325062609/https://gamasutra.com/view/news/34050/Interview_How_A_Fighting_Game_Fan_Solved_Internet_Latency_Issues.php Kyle O. Interview: How A Fighting Game Fan Solved Internet Latency Issues. (Visted 26/05/2022)

[10] https://archive.org/details/nextgen-issue-015/page/n33/mode/2up "The Next Generation 1996 Lexicon A to Z: Fighting Game". (Visited 26/05/2022)

[11] https://www.gamasutra.com/view/feature/3094/1500_archers_on_a_288_network_.php or https://web.archive.org/web/20220506231117/https://www.gamasutra.com/view/feature/3094/1500_archers_on_a_288_network_.php Mark T. 1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond. (Visted 26/05/2022)

[12] https://web.archive.org/web/20210228051849/https://mauve.mizuumi.net/2012/07/05/understanding-fighting-game-networking.html Mauve M. Understanding Fighting Game Networking. (Visted 26/05/2022)

[13] https://attackofthefanboy.com/guides/rollback-netcode-games-list/ or https://web.archive.org/web/20211122034207/https://attackofthefanboy.com/guides/rollback-netcode-games-list/ Andron S. Rollback Netcode Games List (March 2022). (Visted 26/05/2022)

[14] https://en.wikipedia.org/wiki/Super_Smash_Bros._Ultimate or https://web.archive.org/web/20220519200835/https://en.wikipedia.org/wiki/Super_Smash_Bros._Ultimate Super Smash Bros. Ultimate. (Visited 26/05/2022)

[15] https://en.wikipedia.org/wiki/Granblue_Fantasy_Versus or https://web.archive.org/web/20220512125807/https://en.wikipedia.org/wiki/Granblue_Fantasy_Versus Grandblue Fantasy Versus (Visited 26/05/2022)

[16] https://en.wikipedia.org/wiki/Under_Night_In-Birth or https://web.archive.org/web/20220516020108/https://en.wikipedia.org/wiki/Under_Night_In-Birth Under Night In-birth (Visited 26/05/2022)

[17] https://en.wikipedia.org/wiki/Samurai_Shodown_(2019_video_game) or https://web.archive.org/web/20220504020927/https://en.wikipedia.org/wiki/Samurai_Shodown_(2019_video_game) Samurai Shodown (Visited 26/05/2022)

[18] https://en.wikipedia.org/wiki/Soulcalibur_VI or https://web.archive.org/web/20220421074040/https://en.wikipedia.org/wiki/Soulcalibur_VI Soulcalibur VI (Visited 26/05/2022)

[19] https://en.wikipedia.org/wiki/Dead_or_Alive_6 or https://web.archive.org/web/20220130144906/https://en.wikipedia.org/wiki/Dead_or_Alive_6 Dead or Alive 6 (Visited 26/05/2022)

[20] https://en.wikipedia.org/wiki/EA_Sports_UFC_4 or https://web.archive.org/web/20211019192531/https://en.wikipedia.org/wiki/EA_Sports_UFC_4 EA Sports UFC 4 (Visited 26/05/2022)

[21] https://en.wikipedia.org/wiki/Dragon_Ball_FighterZ or https://web.archive.org/web/20220424200258/https://en.wikipedia.org/wiki/Dragon_Ball_FighterZ Dragon Ball FighterZ (Visited 26/05/2022)

[22] https://www.youtube.com/watch?v=0NLe4IpdS1w or https://web.archive.org/web/20220529231214/https://www.youtube.com/watch?v=0NLe4IpdS1w Core A Gaming. Analysis: Why Rollback Netcode Is Better. (Visted 31/05/2022)

[23] https://www.youtube.com/watch?v=1RI5scXYhK0 or https://web.archive.org/web/20220330062542/https://www.youtube.com/watch?v=1RI5scXYhK0 Hold Back To Block. Talking Netcode With Adam "Keits" Heart (Visited 31/05/2022)

[24] https://displaylag.com/video-game-input-lag-database/ or https://web.archive.org/web/20220519183335/https://displaylag.com/video-game-input-lag-database/ Display Lag. Video Game Input Lag Database. (Visted 02/06/2022)

[25] https://www.gdcvault.com/play/1025021/8-Frames-in-16ms-Rollback or https://web.archive.org/web/20210819020226/https://www.gdcvault.com/play/1025021/8-Frames-in-16ms-Rollback Michael S. 8 Frames in 16ms: Rollback Networking in 'Mortal Kombat' and 'Injustice 2'. (Visted 08/06/2022)