A Trustless Low Latency Communication Protocol for Rule-Based Operations on a Shared State

Joshua Handelman

March 2024

Abstract

Peer-to-peer gaming would allow players to stream their data between each other without the need for a centralized game server. It would lower development costs and improve latency, resulting in a better experience for players and developers. In this paper we concentrate on creating a peer-to-peer protocol for a 2-player game in which money is on the line. In this scenario, truth is impossible to prove with current means. Proof-of-work and proof-of-stake would both fail as the computationally superior or wealthier player would always win disputes. We propose a solution utilizing digital signatures and a blockchain. Players communicate in a call and response pattern in which the player who initiates the handshake sets the game rules and the frame rate. Every frame, a block is sent from one user to another. The block contains all the operations done by the user in that time window. This block must be signed by the sender who must also sign the previous block. If the users have a dispute on the true state of the game, this dispute is settled by the decentralized blockchain. Validity of the blocks is proven by signatures. A block signed by both parties becomes true. Any blockchain, with any block time can work, as long as the chain is uncompromised and sufficiently decentralized.

1 Introduction

With current game server technology, players send data to a server which performs corresponding actions and sends the resulting state back to all of the players. In this system, if players want to compete for real-world value, they must trust that the centralized server is not compromised and does not favor their opponent. Additionally the existence of the server adds latency as the route through the server to the other player might not be the fastest route. Current blockchain technology is not capable of running a low-latency game, due to the need to sign transactions, limitations in smart contract architecture, and cost of data storage. This necessitates the creation of P2P software that enables trustless communication between two parties who hope to engage in a

rule-based interaction (game). Since the rules are publically known, the players can both validate each other's actions. Any disputes can be settled by a decentralized blockchain. Since disputes should never happen if both players are honest, the latency penalties incurred by the utilization of a blockchain are practically nonexistent.

2 Protocol Description

1. Two players, Alice and Bob agree to play and share the required data for a P2P connection over a trusted third party or a blockchain (as used in the example).

2. The player who initiates the handshake (Alice) signs and sends the starting state. The state contains the previous state, the block number, the operations performed, the timestamp, and the signatures and public keys of both users. The first state does not include operations or the previous state. All block-critical data - the new state, operations, timestamp, and block number, are hashed and then signed.

3. A new game room is created and stored on a blockchain. The timestamp of this is sent to both players (the blockchain will not be used until the end of the game unless there is a dispute between the two players.)

4. At this time, any player can drop out of the game with no penalty. Bob can drop out if he finds anything wrong with Alice's starting state. He can also drop out if he just doesn't feel like playing with Alice today.

5. Bob reads Alice's state. He verifies it follows from the previous state based on Alice's stated operations (not applicable for the first state), signs it, makes a copy, and performs operations on it to get to a new state. He then signs the new state and sends it back to Alice including all the necessary data.

6. Alice and Bob continue like this until a winning state is reached. At this time, this winning state is sent to the chain for verification. If both players have signed it, then rewards are automatically sent to the winner.

7. If Alice and Bob disagree on the state, a dispute is opened. Both players must send the most recent block they have that is signed by both players. Whichever player opened the dispute must send the state they hope to verify. The chain picks the most recent block signed by both players and attempts to verify the state sent by applying the stated operations to the most recent state. If verification fails, the last state is signed by the contract account and sent to both players. Otherwise, the new state is signed by the contract account and sent to both players. If a player then refuses to sign this new state, a new dispute is opened. The player who failed to sign a state validated by the contract automatically loses the game. (players are given a certain amount of time to respond to disputes)

3 Security

There are potential ways that a nefarious player could attempt to cheat in a game.

1. Let us assume that Alice and Mallory are playing a game. Mallory initiates the handshake, but sends starting data that would make the game unfair towards Alice. Mallory signs her block and sends it to Alice. Alice reads Mallory's block, doesn't like it, and ends the game. Mallory now sends her block to the blockchain, claiming that Alice quit, and so has lost the game. Since Mallory cannot produce a block signed by Alice older than her current block, the chain does not declare a winner and returns the funds to both players.

2. Alice and Mallory are again playing. However, this time Mallory plays fair, at least for the first round. In the next round, Mallory initiates the handshake and sends an unfair start position. She tries to prove that Alice agreed to play by sending one of Alice's blocks from the last game. However, since each block is timestamped, and this timestamp will be before the start timestamp of this round, Mallory's proposal is rejected, and the players are returned their funds

3. Alice and Mallory are again playing. This time, Mallory attempts to make an illegal move. Alice notices this, and rejects her block. Either Alice or Mallory complain, sending the incorrect block to the chain for verification. The chain now requires the most recent block signed by both users but timestamped after the game has started. Since Alice's last block is more recent, the chain verifies that Mallory's block does not follow from Alice's block, declaring Alice the winner.

4. Alice is now playing Quentin. Quentin quits during the game, meaning that he stops sending blocks to Alice. After a certain time, Alice sends the last block signed by both players to the chain, asking it to end the game as Quentin has quit. The chain asks Quentin for a more recent block, gets none, and declares Alice the winner. If Alice does not have a block signed by Quentin, the game is declared a tie and both players receive their funds back.

5. Alice and Bob are playing. Mallory intercepts the data in transit and modifies it. The signature now doesn't match up with the data, and the block will be rejected.

4 Limitations

1. If a player's connection fails, they will automatically lose

2. Latency rises quickly with distance

3. Randomness is possible, but requires the blockchain to produce it. This adds more latency

4. Each player must perform all the calculations on their end. This is not that much of a limitation with current hardware.