Worlds: A protocol for a distributed MMORPG

Ryan Walker
Ryan.cjw@gmail.com
w̃lkr

MistyWest

Abstract. A protocol defining how anyone can join or contribute to a completely unbounded universe could allow the flexibility to organically grow an MMO faster and more efficiently then any proprietary closed system can. This paper will overview a truly limitless yet fair protocol that allows developers to bolt their code into in common universe.

1 Introduction

For an open software ecosystem to grow organically there should be outlets for people to contribute. In the context of a massively multiplayer online game these currently do not exist. There are a couple reasons for this, one being the proprietary nature of the game, another being (Insert something here), but most prominently, the lack of consensus forming on a distributed scale. The system variables the network must agree on are a function of the required fairness and desired gameplay. Some examples are: player location, player health, players skills, player level or player items. It's trivial to solve this using conventional methods, a server maintains a secure connection with a player and the player pipes his actions in the server, the server then reacts, providing ground truth for the network.

A decentralised system will require a different approach. Blockchains present clear deficiencies for solving this problem, they are too slow, the chain would get too large and they required additional consensus protocols that are still in heavy development.

A completely alternate consensus method will now be presented...

2 Worlds

2.1 Overview

A world, defined as w_x, is a node that serves content to players, defined as P_x, and forms consensus among it's own domain, it's possible for any node in the network to be a world. A world has four adjacent worlds that is decided by itself. Players can enter a world one of two ways, the first being a **player genesis**(2.5), and the second being a **world transfer** (2.4).

2.2 Action Ledger

A player's action ledger, defined as $P_{x,yAL}$ (where x is a unique player identifier and y is the world the action ledger was generated in), contains all the cryptographically signed actions that he wishes to commit in a world, the list in held in chronological order. In order to commit an action a player must concatenate the worlds current nuance with the action sign the action and sent it to the world, if the action is legal, it is entered into the worlds action ledger for that player. Players are required to keep their actions ledger dating back to their genesis, worlds are only required to keep **transport hashes** (2.3).

2.3 Transport Hash

A transport hash is simply a secure hash of an action ledger, $h_s(P_{xyAL})$, these are secured by the worlds are used to prove a player is presenting an honest action ledger. The transport hash is formed when a player leaves the world. A **Forward Transport Hash** is just a transport hash kept in a special location. This is explained more in the world transfer section, it is defined as $h_{sf}(P_{xyAL})$

2.4 World Transfer

There needs to be a secure and fair way for players to transcend worlds. If a player, defined as P_1, wants to move from w_1 to w_2 (Figure 1) and then to w_3, there is a detailed state machine that honest worlds must adhere to in order to maintain consensus. The system values in this scenario are defined as...

Fig. 1. P_1 moving from w_1 to w_2

$$\begin{tikzcd} w_1 \arrow[r, "P_1"] & w_2 \arrow[r, dash] & w_3 \end{tikzcd}$$

	w_1	w_2	w_3
$h_s(P_{1,yAL})$	$NULL$	$NULL$	$NULL$
$h_{sf}(P_{1,yAL})$	$NULL$	$NULL$	$NULL$
Neighbor	w_2	w_1 & w_3	w_1

1. First w_2 must verify that P_1 currently resides in w_1, this is done by ensuring $h_{sf}(P_{1,yAL}) = NULL$ this is found by sending a signed data request message from w_2 to w_1
2. w_2 insures that w_1 is adjasent to itself
3. The player presents a $P_{1,1AL}$ to w_2
4. w_1 calculates $h_s(P_{1,1AL})$ using the AL on the serverside

5. w_2 calculates $h_s(P_{1,1AL})$ using the AL provided by P_1, the hashes must match
6. (Optional) The Action ledger traceback can be complete
7. P_1 is now granted access to w_2 and can submit action
8. w_1 Must store $h_s(P_{1,1AL})$, $P_{1,1AL}$ can be deleted

Fig. 2. P_1 in w_2

$$\begin{tikzcd} w_1 \arrow[r, dash] & w_2,\ P_1 \arrow[r, dash] & w_3 \end{tikzcd}$$

	w_1	w_2	w_3
$h_s(P_{1,yAL})$	$h_s(P_{1,1AL})$	$NULL$	$NULL$
$h_{sf}(P_{1,yAL})$	$NULL$	$NULL$	$NULL$
Neighbor	w_2	w_1 & w_3	w_1

Fig. 3. P_1 in w_3

$$\begin{tikzcd} w_1 \arrow[r, dash] & w_2 \arrow[r, dash] & w_3,\ P_1 \end{tikzcd}$$

	w_1	w_2	w_3
$h_s(P_{1,yAL})$	$h_s(P_{1,1AL})$	$h_s(P_{1,2AL})$	$NULL$
$h_{sf}(P_{1,yAL})$	$h_s(P_{1,2AL})$	$NULL$	$NULL$
Neighbor	w_2	w_1 & w_3	w_1

2.5 Player Genesis

Player Genesis is the creation of a new player, for this to occur a player must digitally sign a genesis package with the current nuance of the world. The player is then instated into the world with all the initial player values set to zero.

3 Trust

The system is not entirely trustless, the neighboring worlds need to trust each other. It's possible for neighboring worlds to have disagreements and still function. For instance, w_1 might introduce an item w_2 consider to be too powerful, in this case the world can just neglect it's existence, any action presented using that item would be considered illegal and not entered into the action ledger of the w_1.

Worlds that have rules that are considered to be completely egregious can be neglected, consider Figure 3.

Fig. 4. Three adjacent worlds

$$\begin{tikzcd} w_1 \arrow[r, dash] & w_2 \arrow[r, dash] & w_3 \end{tikzcd}$$

It's entirely possible that w_1 might not agree with the rules of w_3. There are two ways of dealing with a dispute like this, either a **World Disconnect** (3.2) or **Action Ledger Traceback**(3.1).

3.1 Action Ledger traceback

Depending on the required security of the world, it may required an Action ledger traceback. If this is requested, the player must provide a list of action ledgers that date back to either the player genesis or to the last entry of the world committing the audit. A world may wish to trust the consensus of the adjacent world, but this is a security question to be answered by the world itself. If the world conducts an action ledger traceback, and the traceback contains actions in a world that are not considered legal by the world doing the audit, the rewards and actions committed in the offending world are neglected in the world.

3.2 World Disconnects

It's possible a world may issue a disconnect of an adjacent world, this means that players can no longer travel back and fourth through these worlds and they are no longer considered adjacent. If the player resided in the disconnected world this could leave the player stranded in the offending world. It would be logical for the world that issued the disconnect to accept the player back into the world in an earlier state before the player moved into the offending world.