Worlds: A distributed MMO *(DRAFT)*

Ryan Walker
ryan.cjw@gmail.com

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. This paper overviews a protocol that enables developers to bolt their game into in common universe.

1 Introduction

For an open software ecosystem to grow organically there should be outlets for contribution. In the context of a massively multiplayer online game the outlets become more complicated then a simple code repository. Fair game mechanics are built on a fragile ecosystem that have negative consequences if managed improperly. The fairness and security of the game are entirely dependent on the network forming consensus on what is considered to be the truth. This is trivially done with conventional methods. A server maintains a secure connection with a player and the player pipes his actions to the server. The server then provides ground truth for the network. Distributed systems require a very different approach. Blockchains present clear deficiencies, they are too slow and the chain would get too large. A completely alternate method will now be presented.

2 Worlds

2.1 Overview

A world, defined as w_k, is a node that forms consensus among it's domain. It is possible for any node in the network to be a world. A world can have up to four adjacent worlds determined by itself. Players, defined as P_k, can enter a world one of three ways, the first being a **player genesis**(2.5) and the second being a **world transfer** (2.4). A **refuge packages**(2.6) is a special third method for world entry, it can be issued by orphaned players. Worlds and players both have their own RSA key pair.

2.2 Action Ledger

An action ledger, defined as $AL(P_k, w_k)$, is a chronological list of all the signed actions and reactions that has happened to a player in a world. In order to commit an action a player must concatenate current unix time with the action, then sign and send to the world. If the action is legal, it is entered into the world's action ledger for that player. If reactions are to occur to a player, the world must concatenate current unix time with the reaction then sign and sent to the 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 a hash of an action ledger, $h_s(AL(P_k, w_k))$, these are secured by the worlds and 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 a transport hash kept in a special location. This is explained more in the world transfer section, it is defined as $h_{sf}(AL(P_k, w_k))$

2.4 World Transfer

A detailed state machine outlines how players can move about neighboring worlds. Honest worlds must follow this procedure to maintain fairness, if they misbehave they could be risking a **World Disconnect** (3.2) from their neighboring worlds.

Eg: Lets say a player, defined as P_1, wants to move from w_1 to w_2 (Figure 1) and then to w_3, The network values in this scenario are defined as...

Fig. 1. P_1 moving from w_1 to w_2

$$w_1 \xrightarrow{P_1} w_2 \text{ ——— } w_3$$

	w_1	w_2	w_3
$h_s(AL(P_1, w_k))$	$NULL$	$NULL$	$NULL$
$h_{sf}(AL(P_1, w_k))$	$NULL$	$NULL$	$NULL$
Neighbor	w_2	w_1 & w_3	w_2

1. P_1 sends a signed world entry packet to w_2
2. w_2 insures that w_1 is adjasent to itself
3. w_2 must verify that P_1 currently resides in w_1, this is done by ensuring $h_{sf}(AL(P_1, w_k)) = NULL$. This is found by sending a signed data request to w_1
4. P_1 presents $AL(P_1, w_1)$ to w_2
5. w_1 calculates $h_s(AL(P_1, w_1))$ using the AL on the serverside
6. w_2 calculates $h_s(AL(P_1, w_1))$ using the AL provided by P_1, the hashes must match
7. (Optional) An **Action Ledger Traceback** (3.1) can be complete

8. P_1 is now granted access to w_2 and can submit actions
9. w_1 must store $h_s(AL(P_1, w_1))$, $AL(P_1, w_1)$ can be deleted

Fig. 2. P_1 in w_2

$$w_1 \text{ ——— } w_2,\ P_1 \text{ ——— } w_3$$

	w_1	w_2	w_3
$h_s(AL(P_1, w_k))$	$h_s(AL(P_1, w_1))$	$NULL$	$NULL$
$h_{sf}(AL(P_1, w_k))$	$NULL$	$NULL$	$NULL$
Neighbor	w_2	w_1 & w_3	w_2

Fig. 3. P_1 in w_3

$$w_1 \text{ ——— } w_2 \text{ ——— } w_3,\ P_1$$

	w_1	w_2	w_3
$h_s(AL(P_1, w_k))$	$h_s(AL(P_1, w_1))$	$h_s(AL(P_1, w_2))$	$NULL$
$h_{sf}(AL(P_1, w_k))$	$h_s(AL(P_1, w_2))$	$NULL$	$NULL$
Neighbor	w_2	w_1 & w_3	w_2

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 unix time. The world must ensure there are no existing values entered for that player. The player is then instated into the world with all the initial player values set to zero.

2.6 Refuge Package

A refugee package is an action ledger that dates to the time that the package is sent. This allows worlds to reinstate players without the need of a world transfer. This is useful for player that have been orphaned into a world. This typically happens during either a **world disconnect** (3.2) or **world outage** (6.3). This has the effect of splitting the player's history into two forks, which opens the possibility of an **action ledger conflicts** (6.4).

3 Worlds Engine Mechanics

The mechanics below are simply suggestions. As the engine is completely open, worlds are free to impose whatever mechanics they wish. Worlds with drastically different game mechanics will probably not be bordering, this limits gameplay but maintains fairness.

3.1 Action Ledger traceback

The system is not entirely trustless, the neighboring worlds need to trust each other. It is 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 a case like this w_1 world can just neglect it. Actions presented using that item would be considered illegal and not entered into the action ledger of the w_1. Worlds that are not entirely trusted can be audited, consider Figure 4

Fig. 4. Three adjacent worlds

$$w_1 \text{———} w_2 \text{———} w_3$$

It is possible that w_1 might have conflicts with the rules of w_3, however w_1 did not choose be adjacent to w_2. During the worlds transfer, w_1 would not get the action ledger from w_3. Completely illegal action could have been committed in w_3 (eg. free money). In this case w_1 can request an **Action Ledger Traceback**(3.1). 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. The world then needs to obtain $h_s AL(P_k, w_{k:n})$ from w_k to w_n. If the hashes match then the players history has been confirmed.

If $AL(P_k, w_{k:n})$ contains illegal actions, the world has some options.

- Deny the player entry
- Neglect rewards. This will cause **Action Ledger Fragmentation** (6.1)

If illegal options are seen, the world should consider a disconnect from the offending world.

3.2 World Disconnects

It is possible a world may issue a disconnect of an adjacent world, this means that players may no longer travel between these worlds. This could leave players orphaned in the disconnected world, see **Orphaned Players** (**??**).

3.3 One Way Gates

Worlds can only control their adjacent neighbors, they have no way of controlling the neighbors of their neighbors. Consider Figure 4, w_1 might have serious issues with w_3. There are three ways of dealing with this. The first is to issues a disconnect from w_2, this has the disadvantage of players now not being able to travel to w_2. The second is to use negotiation tactics with either w_3 or w_2 to change the rules or disconnect respectively. The third way to is implement a one-way-gate. This is a method of ensuring that illegal rules don't make their way into the issuing world. If w_1 issues a one-way-gate against w_3 player are no longer allowed to travel back into w_1 if they have traveled through w_3.

3.4 World Economic Caps

It's possible that worlds may require neighboring worlds to employ a *world economic cap*. This sets a hard cap to the amount of resources that can be distributed to players from the environment in a given time period. This can cap experience points, coins, items, etc...

4 Action Listing

4.1 Construct

An *action listing* is a data construct containing possible player actions. It's possible for a world to make their own action listing, defining an unlimited amount of possible player actions. World reactions are also reside on this list and start at address code: *0x7F*. There are three types of actions. **Legal Actions** are action that a world will accecpt into a players action ledger if presented, they reside in the worlds action listing. **Acceptable Actions** are action that are to be performed in other worlds only, results from these actions are respected. **Illegal Actions** are actions that a world does not consider to be fair. Illegal actions will simply not show up on a worlds action listing. Action ledger tracebacks (3.1) can deal with the occurrence of illegal actions on players action ledgers.

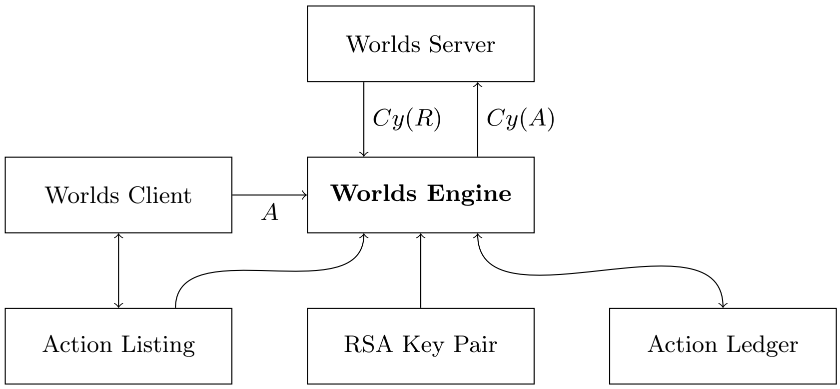
Fig. 5. Section of an Action Listing

4.2 Action Listing Translation

Worlds might not share the exact same action listing. An action listing translation is a data construct used to translate actions from one worlds actions listing to another worlds action listing.

5 System Architecture

A: Action
$Cy(A)$: Encrypted Action
R: Response
$Cy(R)$: Encrypted Response



6 Outstanding Issues

6.1 Action Ledger Fragmentation

Action ledger fragmentation is when a significant portion of a players action ledger has become 'neglected' due to worlds with action disagreements. This should be scarce, as if there are worlds with disagreements they should be considering a disconnect or changing rules.

6.2 Malicious Worlds

Traditionally players would have no recourse against malicious worlds. But it possible that after a world disconnect adjacent worlds might allow players to rejoin, in addition they have the option to neglect the malicious activity.

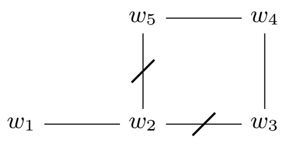
6.3 World Outages

In the event of a server outages players may become stranded. The first line of defense for this should be server mirrors, these do no have to be as capable as the main servers but should allow players to exit. In the even of a permanent world outage, adjacent worlds should accept signed **refugee packages**(2.6).

6.4 Action Ledger Conflicts

Figure 6 depicts a possible scenario in which w_2 disconnects from w_3 and w_5. If a player resides in $w_{3:5}$ there is now no way to return to $w_{1:2}$. $w_{1:2}$ may accept a refugee package and reinstate the player. The player now has the option to play on either forks. In the event of a future path creation between $w_{3:5}$ and $w_{1:2}$ it is possible that that player forks might collide.

Fig. 6. Action Ledger Conflict



This is called an action ledger conflict. Without an action ledger traceback being done it is possible for items or funds to be duplicated. To avoid situations like this is should be common practice to avoid joining worlds that have player history that collides with other attached worlds player history.