Blockchain And Games: A Novel Middleware for Blockchain-Based Multiplayer Games

Vladimir Olegovich Shcherba
Innopolis University, Russia
v.shcherba@innopolis.ru

Rasheed Hussain
Innopolis University, Russia
r.hussain@innopolis.ru

ABSTRACT

Multiplayer computer games can be divided into two architectural groups: client-server and Peer-to-Peer (P2P). While the peer-to-peer approach is very promising due to its independence of expensive game servers, a number of issues, such as state synchronization in an un-trusted environment, arise. Blockchain networks are a special case of a P2P network, and they have the potential to efficiently tackle some issues with P2P-based multiplayer games. However, blockchain development is a vast and complicated area, foreign to most game developers. In this paper, we present a framework to connect an existing game client to a blockchain network, friendly towards game developers. More precisely, we develop a middleware solution and a data model therein, to store the game state on a chain and communicate it to the game client.

1 INTRODUCTION

Modern multiplayer computer games are usually played across the Internet. The network architecture in this case poses a number of challenges such as server maintenance, state consistency, account thefts and cheating [6]. Although blockchain is leveraged to solve some of these issues, there are only a handful of available solutions for games that use blockchain (referred to as *blockchain games*) [2]. The most popular of them are restricted to some basic functionality of public blockchain networks [5]. The problem is that to develop blockchain games, a combination of *existing* game development and blockchain tools, and methodologies have to be used, for which a formalized interoperability Application Programming Interface (API) is required. Such API would allow game developers to use blockchain without having to delve into the complicated details of its implementation.

SIGCOMM '21 Demos and Posters, August 23–27, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8629-6/21/08.
https://doi.org/10.1145/3472716.3472845

To fill the gaps, we develop a middleware library that can be integrated into an existing game client. Furthermore, we design a data model to communicate with a blockchain node, thus providing the required interoperability. The novelty of the proposed solution is that it assists the processing of *in-game mechanics* on the blockchain, contrary to the focus on payment and asset management, which is common for existing blockchain-based games.

2 SYSTEM DESIGN

Interoperability between the existing solutions in the game development and blockchain fields is the main goal of the proposed system. Thus, a high level of abstraction and non-intrusiveness is expected from the proposed system. The typical instrument of a game developer is a *game engine* with an editor, and a typical blockchain solution is a network application that serves as a host to some shared runtime and provides Remote Procedure Call (RPC) methods for interaction. Therefore, a middleware library format is chosen for the implementation of the proposed framework. The library should be easily integrated with the game editors and should provide tools to communicate with blockchain nodes through the RPC interface using an abstract data model.

2.1 Game Client Integration

Integration of the middleware with a game engine and its editor depends on a particular game engine. We chose *Unreal Engine* 4 for the proof of concept due to its strong networking capabilities. Given that most of the complexity is hidden inside the middleware library and modern engines have many common features, integration with most major game engines should not significantly differ. *Blueprints*, a visual scripting system in Unreal Engine is the main part of integration with the middleware library. This system allows game designers and artists to implement many parts of the game logic without programmers' help. To provide the middleware functionality to Blueprints, we implemented several wrapper classes in C++ code to connect Unreal C++ reflection system (required for interaction with Blueprints) with the middleware code.

2.2 Middleware

Our designed middleware consists of two major parts: a data model for gameplay logic and a library for interoperability between a game client and a blockchain node. As there is no uniform language for the description of game mechanics, we developed a data model based on the architecture of several existing game engines and theoretical research on game design [1, 4]. In this model, every entity has a list of parameters, restricted by its type. The model is composed of the following entities:

(1) **Actor** - an entity controlled by a player and may perform *actions*. Every actor has a list of parameters with some *constraints* on their state along with associated assets.
(2) **Action** - a command that changes the game state when invoked. An action may be *constrained*. There is a list of pre-defined actions to cover most common cases for the game logic. Custom actions might be implemented via smart contracts or a runtime upgrade.
(3) **Constraint** - a description of an entity's valid state. They can be used to discard transactions that are invalid following the game rules.
(4) **Asset** - a descriptor of a resource in a game. It may be owned by an *actor* and transferred between them.

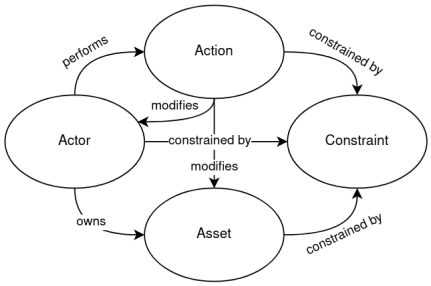

Figure 1: Data Model relations

Implementation of the networking part is based on JSON RPC protocol on top of HTTP/WebSocket, as this protocol is used in Polkadot runtime hosts. SCALE codec is a required part of Polkadot specification, therefore, it is used for communication with Polkadot nodes.

2.3 Blockchain Runtime

For the blockchain part, we chose Polkadot protocol [7] due to its featured interoperability between different blockchain networks, available industrial-quality implementations, and large community. Conformance to this protocol will allow our solution to operate with both completely custom blockchain networks and large public networks (e.g., Bitcoin and Etherium).

An example of how different game applications across different chains can interact is demonstrated in [3]. Polkadot WebAssembly Runtime allows implementation of a custom logic with comparably low overhead. Due to these reasons, the on-chain part of the framework was implemented as an extension of the Polkadot runtime. As the most developed Polkadot host is based on the Rust framework Substrate, the latter was chosen for the reference implementation.

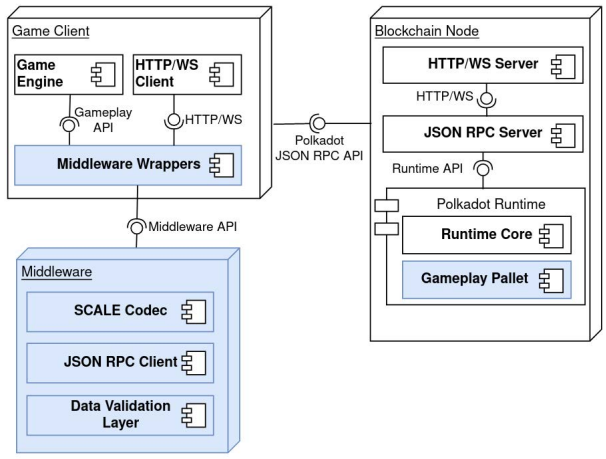

Figure 2: Component diagram of the proposed middleware (proposed and developed parts in blue colour)

3 EVALUATION

The resulting solution can be integrated into the existing game engines to provide a developer-friendly API coherent with the engine subsystems. For example, an integration with Unreal Engine 4 exposes the middleware functionality to the reflection and garbage collection systems. The proposed data model allows describing rules of an arbitrary game using a composition of primitive entities and then storing and synchronizing the game state on a blockchain. The performance of this solution heavily relies on the choice of game engine and blockchain solution, as the middleware itself performs only lightweight operations of SCALE serialization and JSON RPC messaging. The on-chain part optimization for particular practical scenarios is left for the future work.

4 CONCLUSION

In this work, we proposed a framework for interoperability between a game client and a blockchain network, which includes a data model for an abstract description of arbitrary game rules and a middleware library that can be integrated into an existing game client. Additionally, we suggested to use Polkadot protocol for flexible implementation of the on-chain part of the game logic.

REFERENCES

[1] Ernest Adams and Joris Dormans. 2012. *Game mechanics: advanced game design.* New Riders.

[2] L. Besançon, C. F. D. Silva, and P. Ghodous. 2019. Towards Blockchain Interoperability: Improving Video Games Data Exchange. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 81–85.

[3] W. Cai and X. Wu. 2019. Demo Abstract: An Interoperable Avatar Framework Across Multiple Games and Blockchains. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 967–968.

[4] Jesse Schell. 2008. *The Art of Game Design: A book of lenses.* CRC press.

[5] Alesja Serada, Tanja Sihvonen, and J. Tuomas Harviainen. [n. d.]. CryptoKitties and the New Ludic Economy: How Blockchain Introduces Value, Ownership, and Scarcity in Digital Gaming. *Games and Culture* ([n. d.]). https://doi.org/10.1177/1555412019898305 arXiv:https://doi.org/10.1177/1555412019898305

[6] Marcus Toftedahl and Henrik Engström. 2019. A Taxonomy of Game Engines and the Tools that Drive the Industry. In *DiGRA 2019, The 12th Digital Games Research Association Conference, Kyoto, Japan, August, 6-10, 2019*. Digital Games Research Association (DiGRA).

[7] Gavin Wood. 2016. Polkadot: Vision for a heterogeneous multi-chain framework. *White Paper* (2016).