Decentralized Online Multiplayer Game Based on Blockchains

Raphael Burkert, Philipp Horwat, Rico Lütsch, Natalie Roth, Dennis Stamm, Fabian Stamm, Jan Vogt, and Marc Jansen(✉)

Institute of Computer Science, Hochschule Ruhr West, Bottrop, Germany
marc.jansen@hs-ruhrwest.de

Abstract. Decentralized apps offer users significant advantages. One main advantage is that the content belongs to the users. Although decentralized apps have been developed, decentralized multiplayer games have still been a challenge. This research work is therefore based on the question: *Is it possible to create a turn-based real-time online multiplayer game based on decentralized applications on blockchain targeting the mobile market?* This question is answered by the successful development of a decentralized online multiplayer game based on a blockchain.

Keywords: dApp · Blockchain · Multiplayer · Gaming · Waves

1 Introduction

The gaming industry is towering over the global box office industry and music industry in terms of revenue, making it by far the most lucrative entertainment industry. In 2018 the global box office for films totaled US$41.7B in revenue, while global music revenues reached US$19.1B. In comparison the global game market alone generated US$152.1B in 2019 [1]. Out of all the segments that make up the gaming market the mobile game segment is by far the biggest, reaching a revenue of US$65.8B in 2019, thus being more than 43% of total revenue [2]. Reason being the huge availability of smartphones, reaching 3.2B worldwide smartphone users in 2019 and the accessibility of smartphone games mostly targeting casual gamers [3]. Additionally online multiplayer games are highly demanded by gamers and as such many developers are focusing on satisfying this demand. Though when it comes to multiplayer games the issue of connecting players arises. One solution to this problem is to decentralize servers using the blockchain. This comes with some major benefits such as cheaper server hosting and more security and privacy regarding user data [4]. All these aspects lead to the research question of this paper.

Is it possible to create a turn-based real-time online multiplayer game based on decentralized applications on blockchain targeting the mobile market?

J. Prieto et al. (Eds.): BLOCKCHAIN 2021, LNNS 320, pp. 44–53, 2022.
https://doi.org/10.1007/978-3-030-86162-9_5

This question is answered by creating a turn based multiplayer game based on the blockchain platform Waves as an easy to use, yet powerful enough technology. The steps necessary for achieving this dictate the structure of this paper which will be briefly described in the following. Firstly related work mainly focusing on blockchain and games based on it is being looked at. This is followed by the architecture and implementation of the proposed blockchain game. The last major chapter deals with the evaluation of the prior described implementation, which in closing leads to the conclusion and future work.

2 Related Work

The blockchain technology offers some advantages in the entertainment industry and especially in game development. The main advantage is that the blockchain technology "Offer[s] better controls for online players over virtual assets and allows them to use these assets across different gaming platforms" [5]. The disadvantages are also decisive for game development. In a blockchain random algorithms are difficult to program and transactions have to happen in a defined order, which makes real-time gaming basically impossible. Furthermore, smart contracts cost real money [6].

A good overview of current blockchain games is given in the study by Min et al. [7]. The authors created categories and assigned games to them based on four advantages of a blockchain for game development. The first category is "Rule Transparency" which mainly includes games of chance such as Satoshi Dice. The second category is "Asset Ownership", these are mainly collection games like CryptoKitties. The third category is "Asset Reusability" but this is not yet mature due to the relatively few games and the fourth is "User-Generated Content", where users can provide their own content via blockchain without commercial middlemen. In the paper by Du et al. [8], a distinction is made between two categories. One is the completely decentralized games, whereby these usually contain only simple game logic and low complexity. An example of this would be the aforementioned collection game CryptoKitties. The second category, a combination of the traditional game process with the blockchain technology, where part of the game process is executed in the blockchain. The paper describes the development of a poker game using the second category. The development of a completely decentralized app could not really be realized so far. The game CryptoKitties, for example, is equipped with a centralized server that stores special identifiers for cats or takes over administrative functions [6].

3 Architecture

For this paper a turn-based game inspired by the in 1980 released "Battlezone" was created. As a modern interpretation of this classic game we decided to integrate Augmented Reality (AR) aspects into our approach. Using AR our game does not depend on a generated game world and instead uses the GPS coordinates of the players making the entire globe the playing ground. In this

chapter we will first explain how the game flow works and following that the required components for the game will be outlined.

3.1 Game Flow

Figure 1 shows the game sequence. First the player has to log in to his blockchain account. Since the game uses its own token instead of the blockchain currency, the player must first obtain a certain amount of it. To start the game the stake is transferred to the dApp and the player is placed in a queue. When another player is searching for a game, a new game instance is created with the searching player and the player from the queue. The player from the queue is the first in turn. After that the players take turns until a hit is scored and the game is finished. At the end of the game, the paid wager of both players is paid out to the winner.

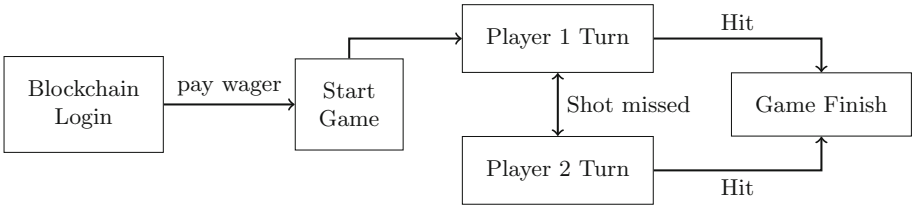


Fig. 1. Game flow

3.2 Components

The components required for the game are shown in Fig. 2. The game is provided to the player as a website, which is hosted on a web server and displayed in a browser on the device. Because of this the application is not completely decentralized. This could be solved through IPFS using a public gateway as access point [9]. The most important component is the dApp, which is hosted in the Waves network. Here the game logic is implemented and executed as well. This includes functions like the calculation of the impact position or the matchmaking. The information about whose turn it is and who the match winner is, if there already is one, is also stored here.

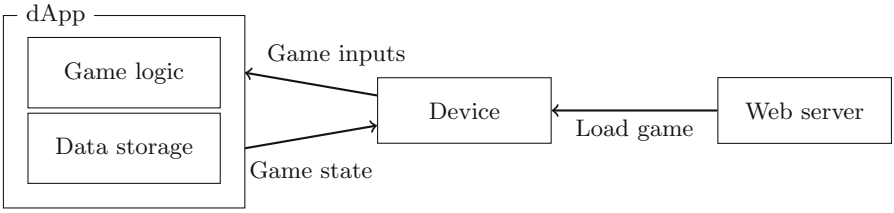


Fig. 2. Data flow between architecture components

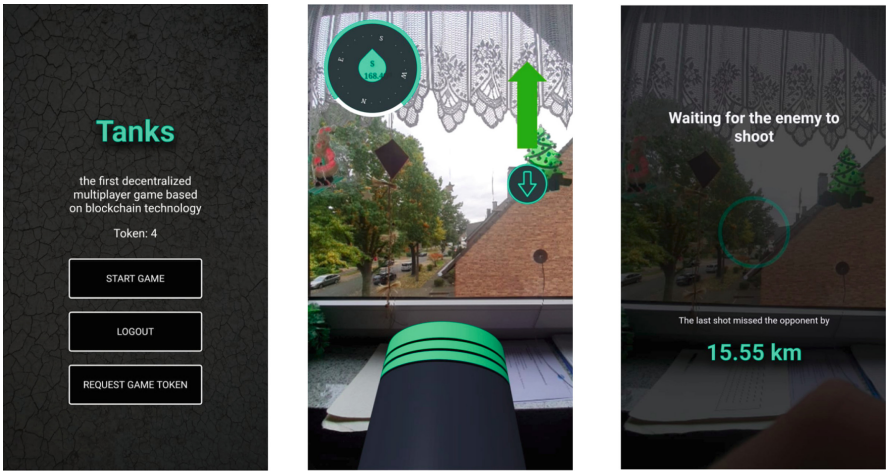
4 Implementation

For the realization of the game flow, the most widely used web framework React.JS is used to provide a state of the art application [10]. Waves is used as blockchain technology. Waves is an open blockchain protocol and a development toolset for decentralized Web 3.0 solutions. The implementation provides an overview of basic React and Waves functionality and provides insight into vulnerabilities and hurdles identified during the development of the application.

4.1 ReactApp

The game sequence was implemented with a client-side JavaScript Web Application, which was build using React version 16.13.1, that can be accessed via a browser. The application is only available on mobile devices, so the built-in sensors, that are necessary for the game to run, can be used. The game follows an augmented reality approach, in which 3D-objects are fixed in the users field of vision and the environment is visible on the device in real-time. This is done by using the landscape camera of the device in combination with the JavaScript tool Aframe in version 1.0.4. Aframe is a framework for rendering 3D-objects on web browsers [11]. The augmented reality effect is enhanced through animated 3D-objects, which are projected into the environment and disappear after a few seconds.

Furthermore sensors are used to determine the players geographic location and thus the location of the device at the beginning of the game. Additionally the gyroscope is utilized to determine the angle and direction for the shot. By means of an integrated compass, the players receive an indication in which direction the shot is aimed, furthermore the compass includes a rough indication of the direction to the location of the opponent. To communicate with the blockchain, Signer was used, a framework developed by the Waves Community to simplify sign-in and communication with the blockchain. A more detailed description of which data is stored on the blockchain (smart account) and which functions (callables) are executed using Signers is given in Sect. 4.2. The stored data in the smart account is necessary to manage the course of the game. Changes on the smart account are retrieved and processed every 5 s using the Waves node API on the client side. During processing, a check is made whether an opponent has been found or if the player is still waiting for an opponent to join. Within the course of the game, the system checks which players turn it is and whether the game has been decided. If it is a player's turn, he is given the opportunity to adjust the direction, angle and force of the shot. The player whose turn it is not in turn receives feedback that he is waiting for the opponent to shoot. Once the game is decided, the game is ended and the players receive information about the game progress. Figure 3 offers a section of screens from the realized game.



(a) Start Game (b) Player 1 Turn (c) Player 2 Wait

Fig. 3. Representation of the decentralized game based on react

4.2 Waves

Since most of the logic of the game is executed on the blockchain, the input from the players must be passed to the blockchain at the end of each turn. This transfer is done using callables, code functions that are directly executed on the blockchain. Only two callables are needed for the game we created. The first callable contains the logic for creating a game and accepting the wager. It is implemented in such a way that there is always a maximum of one player in the queue. In the second callable the input of the player is processed and it is determined whether the player has hit his opponent. Depending on the results of these calculations the game is ended and the winner is paid his winnings or it is the opponent's turn.

Limitation of Script Complexity. In Ride, to limit the execution time of account scripts, there is a value called complexity. Every line of code and every function call adds to this complexity. The complexity of the callables in a dApp script is limited to 4000 [12]. The final dApp Script of the presented approach has a complexity of 2584, what is below said limit. This was only achieved by optimizing the script with regards to script complexity, while reducing script understandability. However, games with more complex game mechanics would not be possible on the Waves blockchain because of this limitation. In the future this limitation could be bypassed by splitting code execution across multiple script calls.

Other blockchain networks like Ethereum do not have a limit of script complexity. Instead, the transaction fee paid for every script execution increases with the length of the executed code [13]. Developers have to make a trade-off

between the complexity of the game mechanics and the transaction fees players are willing to pay.

Turing Completeness. The script language Ride does not contain while- or goto- operations, recursion, and for- operations are limited to a predefined number of rounds. Therefore this language is not Turing-complete. However, most smart contracts do not require Turing completeness [14], and the implemented functionalities did not require it either.

Trigonometric Functions. Ride does not - at least in Version 4 - support trigonometric functions like sin, cos and tan, so an approximation of them had to be implemented. However these could not be too complex because of the script complexity limitations. In following the approximations used for the trigonometric functions are described:

sin For the sin approximation, the formula invented by Bhaskara I. is implemented. For radians, it is stated in Eq. (1).

$$sin(x) \approx \frac{16x(\pi - x)}{5\pi^2 - 4x(\pi - x)} \tag{1}$$

In the utilized range of -3.15 to 4.72 the approximation has a maximum discrepancy of 0.00163.

cos The cosine function was not approximated itself but was implemented as $cos(x) = sin(\frac{\pi}{2} - x)$.

asin The arcus sinus was implemented as stated in [15]. This was also recommended as reference implementation for shader code by NVIDIA [16].

$$\begin{aligned} asin(x) &\approx sign(x) \cdot (\frac{\pi}{2} - \sqrt{1 - |x|} \cdot p(x)) \\ p(x) &= ((-0.01873 \cdot |x| + 0.07426) \cdot |x| - 0.21211) \cdot |x| + 1.57072 \end{aligned} \tag{2}$$

Inside the definition range from -1 to 1 this approximation has a maximum discrepancy of $9 \cdot 10^{-5}$. For values of x outside the definition range it was assumed that those were rounding errors and the values of $asin(-1)$ or $asin(1)$ were returned.

atan An approximation for arcus tangent was defined as in Eq. (3).

$$atan(x) \approx \begin{cases} -\frac{\pi}{2} - \frac{x}{x^2+0.28} & \text{, if } x < -1 \\ \frac{\pi}{2} - \frac{x}{x^2+0.28} & \text{, if } x > 1 \\ \frac{x}{1+0.28 \cdot x^2} & \text{, otherwise} \end{cases} \tag{3}$$

For any x, the approximation discrepancy does not exceed 0.00489.

atan2 The definition of atan2 uses atan, so this function is implemented as defined and the approximation takes place in atan.

5 Evaluation

Online multiplayer games have many different requirements compared to locally played single-player games especially when it comes to player connection and data storage. In the following sections we evaluated cheat safety in regards to stored data and real-time capabilities of our dApp game.

5.1 Cheat Safety

Because the implemented game is an online multiplayer game where two opponents battle each other about money, special attention has to be paid on the topic cheat safety.

One major strategy to avoid cheating is to implement the game mechanics server-side and to send the inputs directly to the server. Any calculation that is done client-side can be manipulated by cheat software. In contrast, manipulating the code behavior on the server-side is a lot harder. In this case, the server-side is the smart account with the Ride script attached to it. The function to execute a shot requires the aiming pitch and yaw angles as well as the shot force and calculates impact position, distance to the opponent and whether the opponent is hit completely on its own.

To calculate the distance to the opponent, the smart account has to know the exact position of the opponent. In contrast, the player himself must not know the opponents position. Otherwise he could calculate the perfect aiming direction and force and instantly hit his opponent. The problem is that everything the smart account stores can be accessed, even the player positions.

One approach solving this problem is to store the player positions encrypted and to only decrypt them when needed. Unfortunately the Ride scripts executed on a smart account do not have access to the smart account's private key or any other kind of secret [17]. This makes sense because the script execution is part of calculating new blocks on the blockchain, and this should be able to be done by any miner connected to the blockchain. For any miner to have access to the private key, the private key had to be published.

A technique to overcome this is the commit-reveal-scheme. It contains two phases: In the commit phase, every participant uploads a hash of his choice, so no other participant can see his choice during this phase. After every participant uploaded the hash, the reveal phase begins and every participant uploads his choice. The blockchain can check if this choice was not changed by hashing it and comparing to the hash uploaded in the commit phase [18]. After both phases, the results are determined without any participant taking other participants choice into account for his decision.

This technique might be good for elections or games that only take one round, but is not applicable for our use-case. To actually work, the "choice" must include the position of the player. The game takes place over multiple rounds, every one running a separate commit-reveal run. The opponent position would be known after the first round, cheaters would win in the second round.

5.2 Privacy

Similar problems to cheat safety arise in the player's privacy. Besides the aiming direction and power, the exact geographic location of each player is uploaded at the start of a game with precision in meters. To protect the player's privacy, at least this information should be encrypted and only be decryptable by the smart account. As already stated in Sect. 5.1, this is a major challenge.

5.3 Response Time Evaluation

For online multiplayer games it is important that the delay between turns is as low as possible. In regards to this we measured the response time of the blockchain between executing a shot in the game and getting the results of that shot from the blockchain. Due to the blockchain not being able to respond after executing scripts, it is necessary to permanently poll the blockchain. For testing purposes a poll rate of $t_{\text{poll}} = 50ms$ was chosen so that the time between poll events has negligible influence on the response time.

In Fig. 4 the results of the measurements are shown for a 3G connection, a 4G connection and a WLAN connection with an Android 10 device. The data consists of 40 data points each and has been measured on two different days.

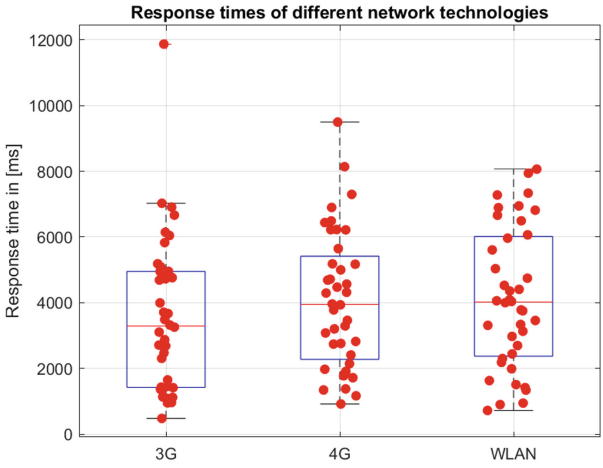


Fig. 4. Measured time between completing a shot and receiving a response from the Waves Testnet blockchain for different network technologies

The results show that there is a negligible difference in the response time between the different network connections. The averages of the different network types are within a 1 s window. The upper quartiles for 3G, 4G and WLAN are $t_{75} = 4950.15ms$, $t_{75} = 5414.05ms$ and $t_{75} = 6015ms$ respectively. Additionally, the measured worst-case has been recorded with a 3G connection with a response

time of $t_{max} = 11872.70ms$. While this is an outlier, it is still an acceptable response time for a turn-based game as presented in this paper.

Surprisingly the fastest average response times were achieved with a 3G connection with an average response time of $t_{\text{avg}} = 3562.96ms$. Since the measurements have been taken in a rather small time frame, the results are prone to errors, like momentary blockchain traffic, which could cause this unexpected result. Furthermore we then performed a two-sample t-test on the data sets to determine if there is a significant difference in the data. The resulting p-values are as follows: $p_{3G,4G} = 0.30397$, $p_{4G,WLAN} = 0.91372$ and $p_{WLAN,3G} = 0.26351$. As a result of the t-tests we confirmed that there is no significant difference in the recorded samples at a 5% significance level.

For games requiring real-time updates in sub-second intervals the results are unsatisfying. While the minimum response times for all three technologies are less than 1 s, the lower quartiles are already above 1 s. On the other hand turn-based games can be implemented with minimal delay as in this type of game the time between moves is determined by the players choices and is not required to be in a fixed interval. Furthermore the results of the measurements indicate that the polling frequency can be lowered to reduce data traffic caused by polling the blockchain.

6 Conclusion and Future Work

For this paper a decentralized turn-based multiplayer game was successfully created. Possible improvements for the future are the complete decentralization, so that there is no central web server necessary, to provide the game to the players. Another possible improvement affects the polling of the smart account data, which would use less traffic, if it was pushed instead of pulled. Furthermore with this approach only games where the connection can take a few seconds are possible. Applications with a higher demand for connection speed need further research.

To increase cheat safety, a technique is required to secure the players data—in this case his exact location—while the dApp is still able to calculate with it. The commit-reveal-scheme is already discussed in Sect. 5.1, other techniques could be alterations of zero knowledge proof or multiple hashing procedures. The most promising approach might be homomorphic encryption [19]. However, all of the listed techniques need further research whether they could increase cheat safety or not.

References

1. Stewart, S.: Video game industry silently taking over entertainment world, 22 October 2019. https://www.ejinsight.com/eji/article/id/2280405/20191022-video-game-industry-silently-taking-over-entertainment-world
2. Jeni, C.: Mobile gaming industry analysis & trends in 2020 (June 2019). https://medium.com/@christinaalex/mobile-gaming-industry-analysis-trends-in-2020-c2a1b2df86d6

3. O'Dea, S.: Smartphone users worldwide 2016–2021, 20 August 2020. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
4. riki28: Networkunits: The decentralized multiplayer platform, 03 January 2018. https://medium.com/@pusinggg22/networkunits-the-decentralized-multiplayer-platform-a81261ce76a9
5. Al-Jaroodi, J., Mohamed, N.: Blockchain in industries: a survey. IEEE Access **7**, 36500–36515 (2019). https://doi.org/10.1109/ACCESS.2019.2903554
6. Kingdoms Beyond: How decentralized does blockchain gaming need to be? 4 April 2019. https://medium.com/kingdomsbeyond/how-decentralized-does-blockchain-gaming-need-to-be-10862685b610
7. Min, T., Wang, H., Guo, Y., Cai, W.: Blockchain games: a survey (2019). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8662573
8. Du, M., Chen, Q., Liu, L., Ma, X.: A blockchain-based random number generation algorithm and the application in blockchain games (2019). https://doi.org/10.1109/SMC.2019.8914618
9. Benet, J.: Ipfs - content addressed, versioned, p2p file system (2014)
10. Stackoverflow: 2020 developer survey, 10 October 2020. https://insights.stackoverflow.com/survey/2020
11. Kumar Ahir: How to create a virtual tour using a-frame—by kumar ahir—medium, 21 December 2018. https://medium.com/designerrs/how-to-create-a-virtual-tour-using-a-frame-164941fea573
12. Limitations—waves documentation, 24 September 2020. https://docs.waves.tech/en/ride/limits/
13. Valson, P.: Transaction fee estimations: How to save on gas? Part 1, 21 September 2020. https://upvest.co/blog/transaction-fee-estimations-how-to-save-on-gas
14. Jansen, M., Hdhili, F., Gouiaa, R., Qasem, Z.: Do smart contract languages need to be turing complete? In: J. Prieto, A.K. Das, S. Ferretti, A. Pinto, J.M. Corchado (eds.) Blockchain and Applications, Advances in Intelligent Systems and Computing, vol. 1010, pp. 19–26. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-23813-1_3
15. Abramowitz, M., Stegun, I.A. (eds.): Handbook of Mathematical Functions: With Formulas, Graphs and Mathematical Tables. Dover Publ., New York (1965). Page 81, equation 4.4.45 ; [outgrowth of a Conference on Mathematical Tables, Cambridge, Mass., Sept. 15–16, 1954, unabridged and unaltered republ. of the 1964 ed. edn. Dover books on intermediate and advanced mathematics
16. asin—nvidia developer zone, 03 May 2012. https://developer.download.nvidia.com/cg/asin.html
17. Mark Muskardin: Mastering the fundamentals of ethereum (for new blockchain devs) part iii – wallets, keys, and accounts—by mark muskardin—medium, 08 December2019. https://medium.com/@markmuskardin/mastering-the-fundamentals-of-ethereum-for-new-blockchain-devs-part-iii-wallets-keys-and-4cd3175b535b
18. Griffith, A.T.: Commit reveal scheme on ethereum—gitcoin—medium. Gitcoin, 14 December 2018. https://medium.com/gitcoin/commit-reveal-scheme-on-ethereum-25d1d1a25428
19. Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C.A., Strand, M.: A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192 (2015). https://eprint.iacr.org/2015/1192