CloudArcade: A Blockchain Empowered Cloud Gaming System

Juntao Zhao
The Chinese University of Hong Kong, Shenzhen
Shenzhen, Guangdong, China
juntaozhao@link.cuhk.edu.cn

Yuanfang Chi
The University of British Columbia
Vancouver, BC, Canada
yuanchi@ece.ubc.ca

Zehua Wang
The University of British Columbia
Vancouver, BC, Canada
zwang@ece.ubc.ca

Victor C.M. Leung
Shenzhen University
Shenzhen, Guangdong, China
The University of British Columbia
Vancouver, BC, Canada
vleung@ieee.org

Wei Cai*
The Chinese University of Hong Kong, Shenzhen
Shenzhen, Guangdong, China
Shenzhen Institute of Artificial Intelligence and Robotics for Society
Shenzhen, Guangdong, China
caiwei@cuhk.edu.cn

ABSTRACT

By rendering game scenes on the remote cloud and delivering real-time video to end devices via the Internet, cloud gaming enables players to access game services anytime anywhere despite the hardware capacity of their terminals. However, as a commercial service, the state-of-the-art payment models for cloud gaming are still in their preliminary stages. In this paper, we reveal the shortages of existing cloud gaming pricing models and propose CloudArcade, a token-based cloud gaming system that employs blockchain-empowered cryptocurrency as a payment method for the players using the cloud gaming services. By using cryptocurrency, CloudArcade provides a transparent and resource-aware pricing method, it also enables a time irrelevant silent payment on the floating price to protects users' payment. These features eliminate the quality of experience degradation caused by the spot price in the traditional dynamic pricing model on the QoE-aware service pricing. We also employ the payment channel in CloudArcade to improve the system performance. Discussions on service pricing criteria are put forward, open issues about token issuing and malicious resource speculation are also reviewed. We believe the design of CloudArcade can show a good generality on other QoE-aware and human-centered applications.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; • **Applied computing** → *E-commerce infrastructure*; • **Computer systems organization** → Cloud computing;

*corresponding author

BSCI '20, October 6, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7610-5/20/10…$15.00
https://doi.org/10.1145/3384943.3409420

KEYWORDS

Human-centered Computing; Blockchain; Cryptocurrency; Cloud Gaming; Pricing

ACM Reference Format:
Juntao Zhao, Yuanfang Chi, Zehua Wang, Victor C.M. Leung, and Wei Cai. 2020. CloudArcade: A Blockchain Empowered Cloud Gaming System. In *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI '20), October 6, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3384943.3409420

1 INTRODUCTION

Cloud Gaming, service that offloads the game programs from the traditional consoles to the cloud, executes the core game logic and game runtime on the cloud and conveys the game content to the users via video stream, which reduces the hardware resource requirement in the thin clients. We are now getting a more solid version of the cloud gaming future landscape from the recent announcement of several big companies. During the Game Developers Conference (GDC) 2019 conference, Google offered Stadia, a cross-platform cloud gaming platform, aiming to provide cloud gaming service through the browser. Meanwhile, Tencent Cloud releases its cloud gaming solution at ChinaJoy 2019. Recently, Oppo provided cloud gaming experience over 5G in Mobile World Congress (MWC) 2019, while Microsoft is also going to test the xCloud game streaming service in Korea over the 5G soon. Forsaken World, the new massively multiplayer online role-playing game (MMORPG) from Perfect World, has also launched a cloud version on China Telecom's cloud gaming platform in 2020. Worldwide game and tech firms are exploring cloud gaming as a new way to deliver the game services, and the dawn of 5G provided solutions to the pain point of network problems faced with cloud gaming in the past few years, which also fuel up this field.

Extensively studies have been conducted to optimize cloud gaming services, including graphical rendering [9], edge allocation [8], bandwidth allocation [21], server resource management [11], and dynamic streaming [22]. In contrast, few researchers investigated novel cloud gaming pricing strategies, which adopt playing time as their pricing criteria. The existing cloud gaming pricing strategy follows a traditional time granularity pricing in other cloud computing services. For example, PlayStation Now[1], the most popular operating cloud gaming platform, charges their customers with monthly subscription policy. The players need to pay the subscription fee in advance at the beginning of a month to access their cloud gaming services. On the other hand, Parsec[2] applies an hourly spot pricing model, where the players pay $0.5 to $0.8 per hour according to the host. However, none of these pricing strategies are good in practice [3][4].

The coarse time granularity implies a high pre-paid price, which means the players need to play sufficient time to make their payment worthwhile. Therefore, the players with very high service stickiness may benefit from the monthly subscription, while others may suffer from over-pay loss because of their limited playing time. At the same time, the subscription needs resource provision for all the subscribed users on a large time scale, which leaves the cloud computing resource idle and waste.

Spot price solves the previous issues nicely in many discontinuous computing services [20], it is fine-grained and sensible to the market demand, which is helpful to utilize the revenue and computing resource. Under the context of cloud gaming, however, new issues emerged when using a spot price. Users of cloud gaming service have high requirements for the quality of service experience in a long continuous time. Directly forcing users to change service demands may lead to a devastation of users' service experience, resulting in a serious loss of demand and user retention. For example, if a silent payment method is used, which means the players will be charged pay-as-you-go with a floating price. This will lead players' concern on their payment during the gaming session, as they need to estimate both the current service price but also their remain balance to determine how much time they should or will play. If using payment request instead, as the fine-grained pricing model requires a cost based on a certain time unit, the frequent requests of payment will also affect the players' gaming experience. Another problem with the spot price is the floating price in the payment process will be non-transparent, which introduces price discrimination risks to the players. As players can only have a rough estimation of the service price, the transparent problem in payment allows the service provider to arbitrarily control the price with no protection of players' utility.

To mitigate the above issues, a new business model or pricing strategy should be established. The newly proposed model should protect the user's payment, no matter what they already paid or will pay in the future. This requires the new model to keep the paid value while making the payment process transparent. At the same time, to utilize the computing resource, the service price of this new model should be floated with the market demand. To protect the game experience from the worry of price and interruption of the payment request, a silent payment way on floating price without concern should be applied.

In this work, we propose CloudArcade by borrowing the idea from the traditional amusement arcade, which installs coin-operated machines to provide the game service. Empowered by blockchain [5], the CloudArcade is a novel cloud gaming system that employs cryptocurrency as the coin, a.k.a. token, to start the cloud gaming service. Different from the time-based rental in traditional cloud pricing models, CloudArcade sells gaming content not by the length of gaming periods but by challenging opportunities (e.g. a limited 3 lives in Contra). During the exchange and service purchase process, the payment value is stored by the tokens. Players can consume tokens with their need, thus, the over-paid problem caused by the coarse granularity pricing can be solved. From the perspective of players, because transactions are transparent and traceable on the blockchain, arbitrary price manipulation by cloud game providers can be prevented. At the same time, CloudArcade publishes the price of a game on smart contracts and represents the price with a relatively constant number of tokens, which is determined by the game content and estimated demanded resources. Since the instant price of a blockchain-based token is a direct reflection of the number of tokens in circulation, the actual price for the game will be a dynamic index of the market demand. As the token price will manipulate the players' purchase behavior silently, CloudArcade can leverage it to optimize the resource consumption of the cloud gaming system. The pricing scheme in CloudArcade is not related to the gaming period, for example, if you buy ten tokens in advance, it will still be ten tokens after minutes. Also, as token stores, a floating price, using tokens to pay the service reflects the state of market condition. Thus the time anxiety and disturbing introduced by the spot model can be eliminated, thus, promotes the player's gaming experience.

The major contributions of this paper are:1) Adopt blockchain-empowered token to solve the problem of traditional time granularity pricing in cloud gaming 2) Adopt a special silent payment method to protect players' gaming experience while utilizing computing resources 3) Use transparent blockchain-empowered smart contracts to enhance players' trust on service and protect players' interests. 4) Provide possible approaches to the service pricing of the proposed system.

The remainder of this paper is organized as follows. We review related work in Section 2 and illustrate the overview of the proposed CloudArcade system in Section 3. We then present the design and implementation of CloudArcade in Section 4 and Section 6, respectively. Afterward, we consider players' hybrid payoff and other pricing-related factors, in section 7, to provide several possible pricing approaches. We further discuss the open issues on token issuing and malicious resource speculation prevention and conclude the paper in Section 8 and Section 9, respectively.

2 RELATED WORK

A cryptocurrency is a form of digital currency that is designed to work as a medium of exchange in a blockchain system. Blockchain contributes to cryptocurrency's high security and difficulty of modification[15], thus cryptocurrency can provide a safe but easy way to make transactions under different business situations[1]. Many studies focus on how to implement cryptocurrency in an existing business model or create new business models based on cryptocurrency. Some papers focus on implementing an incentive system. Yamada [25] proposed an activity-based micro-pricing system using cryptocurrency technologies to alter people's behavior

[1] https://www.playstation.com/en-us/explore/playstation-now/
[2] https://parsecgaming.com/

through automatic micro-payments. Ghosh [10] proposed a cryptocurrency based renewable energy dynamic pricing strategy. Xiao proposed a cryptocurrency-related system called Edge-Toll to encourage the public edge resource sharing.[24] Other studies bring the discussion on providing a new embedded infrastructure that supports e-government. [2] Papers proposed a decentralized employment system to handle temporary employment contracts. [16] For the game, Wang designed a music roguelike game[23], Min also proposed a sophisticated survey on it.[14] Here, we apply cryptocurrency to CloudArcade mainly because it fulfills our critical needs: 1) it eliminates problems of using a physical coin; 2) its price reflects the demand in the market; 3) it provide a secure and transparent payment process.

3 SYSTEM OVERVIEW

In this section, we present an overview of our proposed system. The whole system is composed of the game store, cloud gaming service, and blockchain platform. In our system, games are run in the virtual machines (VMs) in the cloud and configured by a cloud gaming service. Players can then access the game service via the service URL. Players purchase game services in the game store using tokens through the blockchain system to unlock the games that run in the cloud. Our design is depicted in Figure 1.

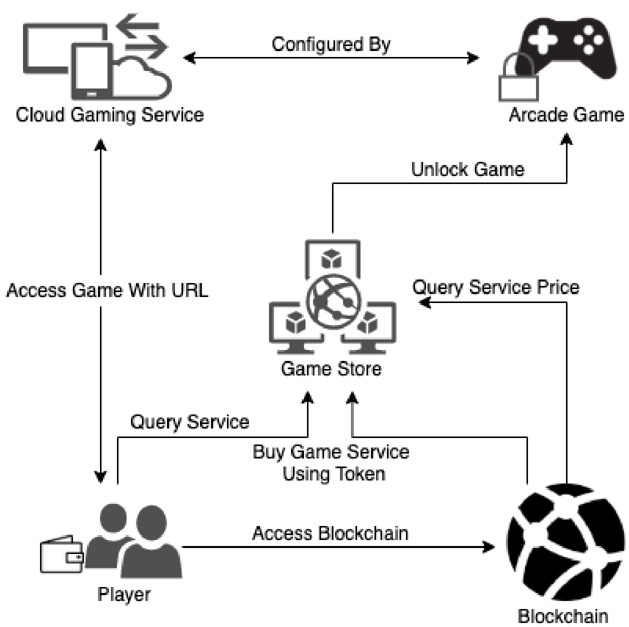

Figure 1: System Overview of CloudArcade

Different from traditional cloud gaming service, here we only choose to serve coin-op arcade games, which require coins to start or continue the gaming sessions. The arcade game is time irrelevant but only related to the avatar life quantity or the time limits in the game design, which means that only if the lifetime or any other finite representative regarding the payment is exhausted, a new payment request is needed. This special character eliminates all the pricing concerns that are relevant to the playing period. In CloudArcade, we set a game service price as a certain number of tokens which were written in a smart contract. This first provides a transparent payment process that helps players to have a clear understanding of the cost they need to pay. Second, though the number of tokens is relatively a constant, the price of tokens can always be a reflection of the real market demand, the game service price is automatically adjusted according to the market condition, and the resource optimization based on dynamic pricing manipulation can be achieved in that sense.

4 DESIGN OF CLOUDARCADE SYSTEM

We illustrate our design of CloudAcade in Figure 2. Video games are executed in the VMs hosted by cloud gaming service and have their corresponding game service URLs. These services are registered in a local database of the cloud server. From the perspective of players, players need to first login to their cryptocurrency wallets to access the game store. Then they can query game prices through the interaction with the smart contract that deployed by the CloudArcade. The game store automatically retrieves other game services information from the cloud server, as specified in Section 4.2.

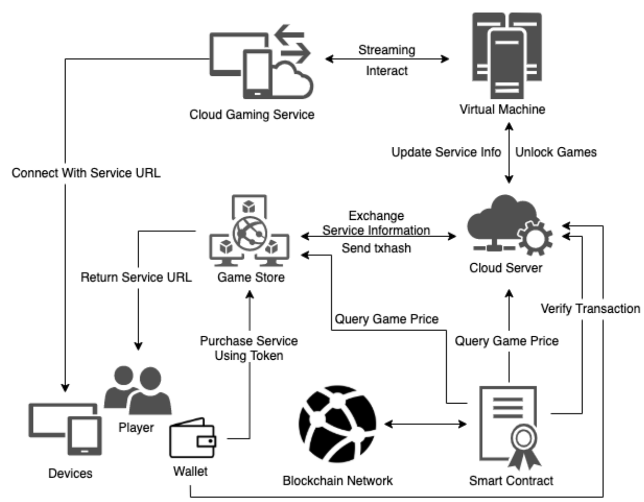

Figure 2: Architectural Design for CloudArcade

Players can invoke the payment process through the smart contract, which will be introduced in Section 4.3. They will receive a transaction hash (*txhash*) from the smart contract whenever a transaction succeeds. When the player wants to play a particular game, he can send his *txhash* together with his account address to the cloud server. The cloud server will fetch the events from the smart contract to validate whether the *txhash* is generated via a valid payment process. The cloud server will also query the local database to check whether the *txhash* has already been used. After the *txhash* is validated, the cloud server will unlock the corresponding game service and return the service URL to the player, which is illustrated in Section 4.4. Players can then use this service URL to access cloud gaming services, as we have discussed in Section 4.5.

4.1 Game Service Setup

Game services provided in the CloudArcade should maintain games that have outside control over their main processes. A game process should be blocked or terminated when there is no activation signal received. Meanwhile, it should also be a non-roguelike genre that can be picked up anytime by players to continue playing. To this end, games run in VMs should be modified to fulfill the following requirements: 1) a game's process can only be run or continued when activated; 2) the game can be only activated by the central server; 3) the game's process should be blocked again when current service is over.

4.2 Game Service Information Fetching

We use a centralized cloud server to store the basic information of the game services and a smart contract deployed on the Ethereum platform to store the price of the games. The game store will automatically query the game service information, like the name and degree of crowdedness, from the cloud server. It will also open up a bi-directional communication channel to receive the latest price and queue information from the server. To completely use the game store, players first need to make an injection of their wallet accounts. Users may use self-hosted wallet plugins in their browser and authorize wallet account in the store. The store will automatically trigger the account address, balance, and other basic information of the wallet, and if the wallet did not login yet, a warning message will be generated. After the wallet information is successfully detected, users can then fetch the game services information by clicking the query button in a certain game card, and the game store will then query the price through the interaction with the smart contract we previously mentioned.

4.3 Service Purchase

After receiving the information from the smart contract and cloud server, the player can choose to make a transaction when there exist available services for a certain game. The player can call the service payment function defined in the smart contract, which will trigger the player's wallet account and the ID of the chosen game to check whether the remaining balance is sufficient for the current game price. If the balance is enough, the transaction will be processed and a GamePayoutSuccess event will be emitted. The player will then receive a *txhash* in return for game service activation.

4.4 Service Allocation

After receiving the *txhash* in the previous step, the player now can use it to exchange the corresponding game service from the cloud server. The player sends his *txhash* together with the account address to the central cloud server, and the cloud server will fetch all the GamePayoutSuccess events from the smart contract. The central server can find the latest event performed by the account address, and check whether the *txhash* is valid. The local database will be also used to make verification, the following checks are performed in the verification process: 1) Whether the *txhash* has been used. That is if the *txhash* has already been recorded in the local database for this account address or not. If the *txhash* already used, the allocation requests will be rejected. 2) Whether the *txhash* is the latest. That is if the *txhash* matches the latest GamePayoutSuccess event that cloud servers retrieved from the smart contract or not. If not, the allocation requests will be rejected. If all checks pass through, the cloud server will derive the game ID from the data part of the GamePayoutSuccess event and check whether there exist available resources to provide the game service for the particular game identified by the game ID. If not, the activation process will be still rejected. If there exist enough resources on a cloud server, then the server will unlock the corresponding game in the local VM by rewriting the lock file and send the service URL back to the player. The latest *txhash* for this account address will be updated. Then the updated service information will be broadcast to all users.

4.5 Game Service Access

After the valid service URL is received by the game player, he can then use it to access the game service. All games are run in the VM with a lock file inside. The game is only runnable when the lock file is set false, and only the cloud server can unlock these files. These files will be reset true after a game service is over like life end or time is up. All games are hosted by the cloud gaming service, and they will have their corresponding configuration file that determines their streaming and control properties as well as the service URLs. And all service URLs and lock files will be registered in the local database of the cloud server. The cloud gaming service will start streaming these game content when a configuration is run, and the service URL can be used to access these games via the client. In this sense, the player can only access the service whose inside game has already been set unlocked. We admit that the service URL can be recorded for malicious attacks, in real practice, the service URL will be generated randomly to ensure game experience's safety. This can be easily done by changing the configuration file of certain game service.

4.6 Summary and Discussions

To better help our reader understand the procedure in using CloudArcade, we depict the sequence diagram in Figure 3. Also, we will still need to address two potential issues in the system.

The first challenge is the malicious occupation detection. In our payment and validation process, there is a problem that if the service was not used, the resource would be occupied and never released. This can easily cause resource stagnation or be brought down by malicious speculators by buying up resources. To prevent this, we designed two different approaches. The first method is to build an over-delay mechanism inside a game. The traditional arcade games will have a certain countdown recover mechanism, resources are automatically released at the end of the countdown. In CloudArcade, this mechanism can be triggered directly when unlocking the game. However, considering that not all games have this built-in countdown mechanism, in our system, we use another way of scanning the database to solve the occupation. After the resource is released to the player, the system records the release time. The database will perform regular tasks of scanning, it will clean up and release long-term unused resources detected.

The second issue is regarding the game information modification. When CloudArcade is going to modify the information of games, they need to modify information in both the smart contract and database. All games' price is declared transparently in the smart

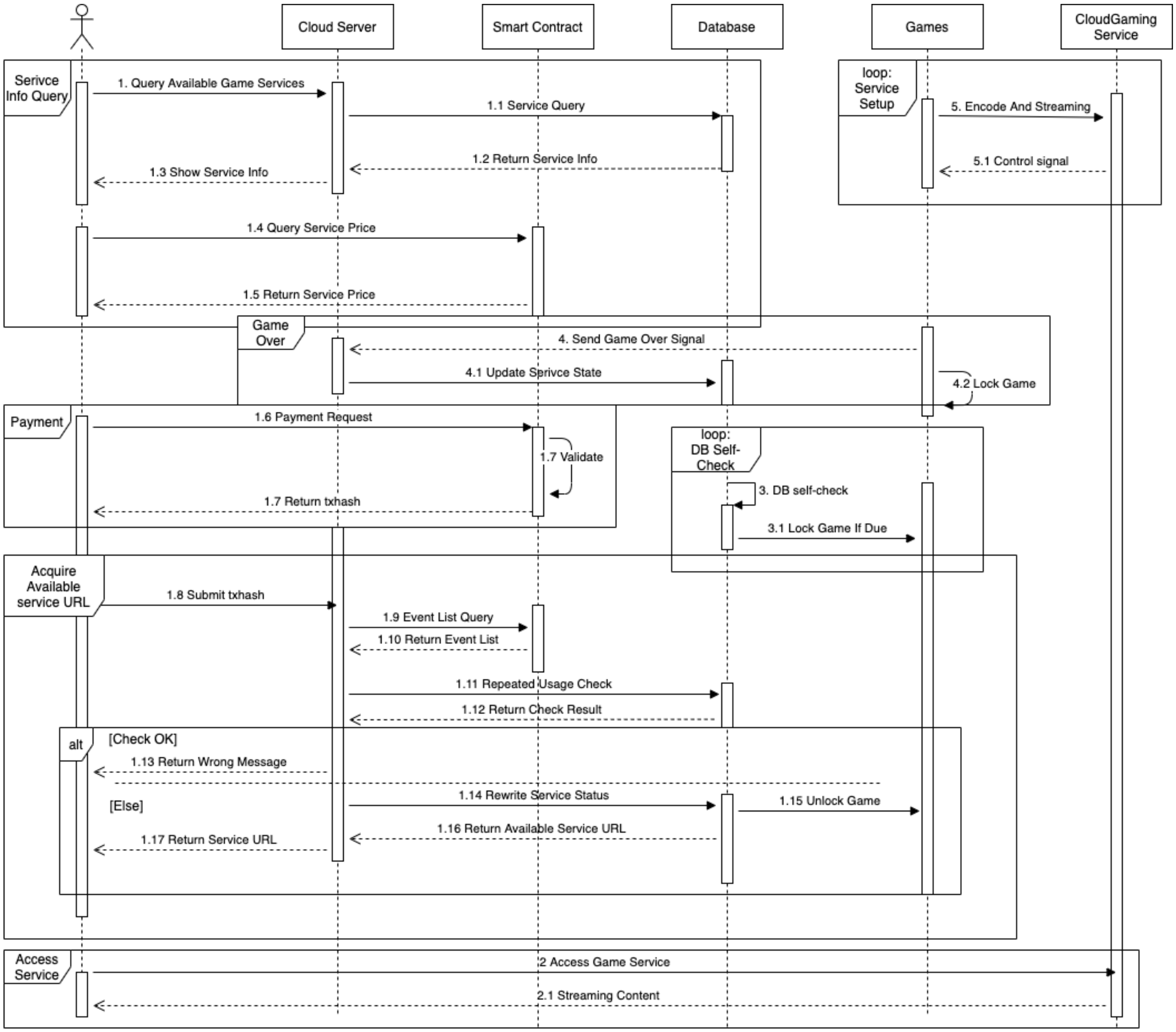

Figure 3: Sequence Diagram For CloudArcade

contract, CloudArcade provider can call a certain interface of the smart contract to reset the price for a game. During the price modification process, the smart contract will first check whether the API caller is the CloudArcade provider, and then perform the game price reset procedure. For other game information modification, the central cloud server provides APIs that help providers quickly modify the game information like names and picture URLs which are stored in the local database.

5 CHANNEL-BASED CLOUDARCADE

The service acquisition process is clear in CloudArcade. However, this system has several shortages. First, the pending time for a transaction will be pretty high. In our test, it will cost at least more than 16 seconds on average for a miner to pack the transaction, sometimes even larger than 30 seconds. Because the average playing time for an arcade game is relatively low, the purchase requests will be very frequent. For a service like that, this pending time will result in too much waiting. Second, because there exists an emission for every transaction, finally there will exist a large number of total events for all the payment processes, which add difficulty for the cloud server to search the latest block that matched up with the need, the searching process will become slower as the event blocks grow up. Also, in this system, there exists a problem for frequent interactions of smart contracts, which also results in some delay.

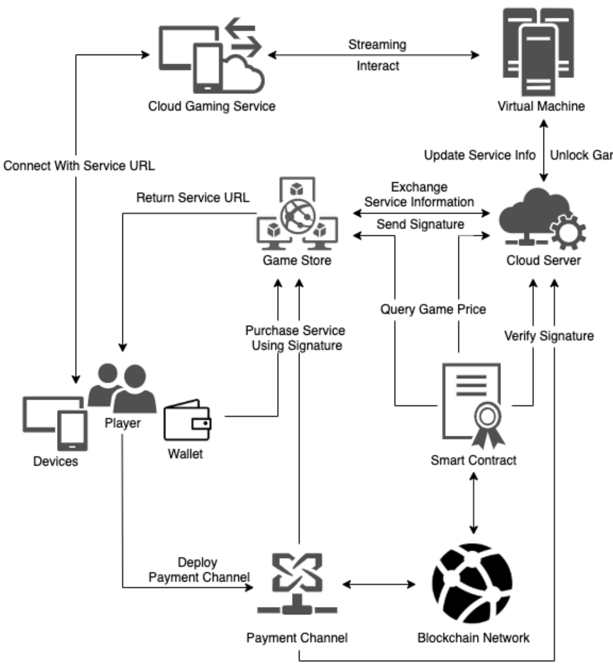

Figure 4: Channel-based CloudArcade Architecture

To solve the problems mentioned above, we put forward a new CloudArcade system that implements the payment channel[3], as illustrated as Figure 4. The payment channel is widely studied and utilized by researchers in solving such problem[13][7]. Players now use the payment channel instead of directly commit transactions to the blockchain to perform purchase actions. A payment channel is a pre-payment offline transaction model designed to allow users to make multiple transactions without committing these transactions to the blockchain. Here, instead of directly call the smart contract that deployed by the CloudArcade to make a transaction, the player first deploys a smart contract by himself, which is then called the payment channel. The player need to attach enough tokens to the contract to make further transactions. Every time a payment channel is created, the game store will send its address together with the player's account address to the central cloud server. The server will then update the record in the local database if there already exists a payment channel in the local database for the corresponding player's wallet account. The old address will be replaced with the new one and CloudArcade will record the address for further claim and fund release.

When a player is going to make a transaction, he needs to authorize a payment by signing the message with the newest cumulative payment and the payment channel address, then send it to the cloud server. After receiving the signature, the cloud server will deconstruct the signature to get the information inside of it, make verification and send back the available service URL if all checks pass through. When CloudArcade decides to withdraw money, it only needs to present a signed message to the smart contract, after the authenticity of the message is verified, the fund will be released. Because the payment is offline and not operate on the blockchain network, it eliminates the pending problems in CloudArcade.

[3]https://solidity.readthedocs.io/zh/stable/solidity-by-example.html#micropayment-channel

5.1 New Validation Process

Instead of committing transactions to the contract that deployed by the CloudArcade, the transactions in channel-based CloudArcade are made via the signed messages. So the validation process of these two systems will be very different.

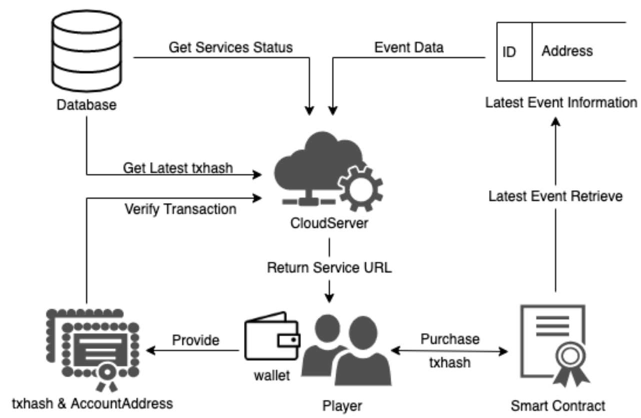

Figure 5: Validation In CloudArcade

As mentioned above, for the CloudArcade, it requires the player to offer account address and *txhash*, and query the smart contract independently to get the latest event, as well as query local database to get the latest *txhash* that used by the given account address, which is illustrated in Figure 5. If the event is found and successfully matched, as well as *txhash* record is empty or different, the corresponding game service will be allocated if it is available.

For the channel-based CloudArcade, the player first needs to deploy a payment channel. When a player wants to send a payment request, he needs to use the payment channel address and accumulated price to sign a message with his wallet account. The signed message will then be sent together with the payment channel address and wallet address to the cloud server. The cloud server will deconstruct the signed message to check whether the signature is valid. The following checks are performed: *1)Address verification:* That means the contract address and user address inside the signature will be validated. The player's address will be confirmed to see whether it is matched up with the player that sends this signature to the cloud server. The contract address will be validated to avoid a replay attack. The request will be rejected if there exists any wrong in the previous check process. *2) New Total Amount Verification:* The cloud server can get the newly paid fees by comparing the new accumulated price inside the signature and the record new verified accumulated price to the local database. If the newly added price is not matched up with the current game price demonstrated in the smart contract, the request will be rejected. *3) Total Amount Exceeding Verification:* As the payment channel always has a ceiling for the pre-paid ether amount, the cloud server needs to check whether the new total amount already exceeds the maximum. If so, the request will be rejected. The check process can be graphed as Figure 6. If all checks are correct, the service allocation process will be conducted. And the latest signature and the total amount of the given wallet address will be updated.

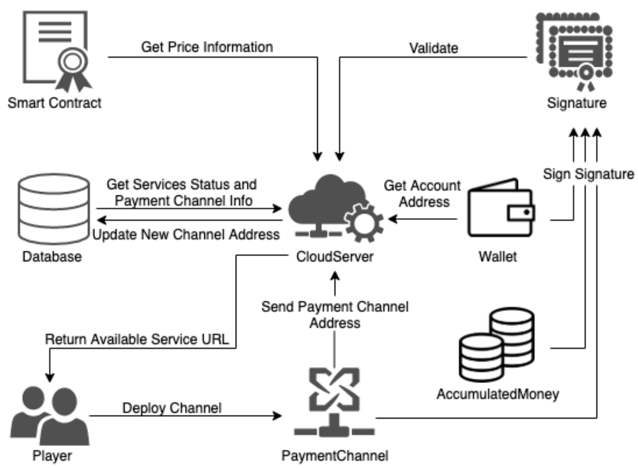

Figure 6: Validation In Channel-based CloudArcade

5.2 Summary and Discussions

By using the payment channel, the CloudArcade reduces the interaction with the blockchain, which greatly speeds up the payment process. Since game resources are often limited, queues need to be implemented to both systems in real practice. Long pending time can seriously affect the service delivery process. With the payment channel, the pending time for a transaction is saved, the system can work more smoothly. This will greatly increase the user experience. Also, instead of pay a commission charge every transaction, the only fee for the channel-based CloudArcade is in the deployment of the payment channel. The player's transaction fee will be significantly reduced.

However, there still exist some problems with the payment channel. As the payer needs to attach a certain amount of tokens to the payment channel when making transactions. If the volume of tokens is set too large, the player may not have enough balance for the channel deployment. A large amount of pre-paid tokens may also affect the user's payment process for other activities. But if the payment is set too low, the player may need to frequently deploy the channels, which lead to the increment of the total commission charge. So a proper amount of attached ether should be well-defined in the payment channel. To promote the circulation of the cryptocurrency, the CloudArcade also need to release abandoned channel at once to release the funds inside the contract. If the pending problem in blockchain transactions can be solved in the future, the waiting time difference between these two systems will be pretty small. Another problem is that a more complex validation process increases the cost of validation, which may affect the concurrency of the platform.

6 SYSTEM IMPLEMENTATION

In this section, we present the implementation of a prototype to demonstrate our proposed CloudArcade system.

6.1 Enabling Technologies

We select a series of packages to fulfill the prototype development requirements. For the blockchain platform, we employ Ethereum[4] due to its popularity in the decentralized application community. To this end, solidity[5] becomes our smart contract programming language. For the client, we adopt vue-cli[6] and webpack[7] framework to supports the fast development of the front-end. And we make a wallet injection in the game store with the support of the Metamask[8], a web browser plug-in to run Ethereum dApps without running a full Ethereum node. The smart contract is invoked by web3.js[9], which is a JavaScript interface for contract interaction.

6.2 System Deployment

We deploy our smart contract on Rinkeby Testnet[10] to conduct empirical experiments. The smart contract is deployed on Etherscan[11]. We designed two different smart contracts using solidity. The first smart contract provides the interface for the price query of the game services, it also provides the ability to directly using Ethereum as the payment method for the cloud gaming services. The second smart contract is the payment channel contract provided by the solidity. Its bytecode and API will be stored in the front-end, and users can use them to deploy the payment channel with the help of the Metamask. After successful deployment, the user can sign the transaction using the address of the smart contract.

To deploy the CloudArcade system, we set up the open-source GamingAnywhere [12] platform as our cloud gaming service provider. Three open-source games, including Mario[12], Bubble shooter[13], and Pacman [14], are retrieved from the GitHub repositories to be executed in the CloudArcade. We design a simple lock and unlock procedure for our game services. We define the status of the game service: true for unlocked and false for locked and store the status inside the JSON file lockfile.json under the root of the game directory. The service ID will be also stored in it. All lock files' file paths and their corresponding service identifiers will be stored in the database. The game will continually scan the lock file, only when the status is true, the game process can be run normally. So we initialize all the game status as false. When a successful transaction is confirmed, the server will unlock the allocated game service, and give back service URL as the response, players can use it to access the game service. The server also needs to mark the given service as occupied in the database. When a service is over, for example, the lifetime comes to zero, the game process will rewrite the status

[4]https://www.ethereum.org/
[5]https://github.com/ethereum/solidity
[6]https://cli.vuejs.org/
[7]https://webpack.js.org/
[8]https://metamask.io/
[9]https://web3js.readthedocs.io/en/v1.2.0/
[10]https://www.rinkeby.io/
[11]https://rinkeby.etherscan.io/
[12]https://github.com/justinmeister/Mario-Level-1
[13]https://github.com/justinmeister/bubbleshooter
[14]https://github.com/CharlesPikachu/Games/tree/master/Game14

to false and send the modification request to mark the status of the service in the database as available.

6.3 Demonstration

Figure 7 illustrates a screenshot of the CloudArcade game store. After the injection of the Metamask account, the account information will be demonstrated in the sidebar. And the game store will then get the information about games like name, avatar, and crowdedness from the central server, they will be shown in the center of the game store in a card form.

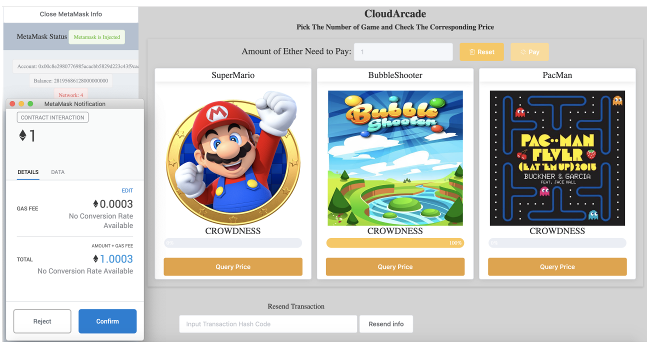

Figure 7: Payment In CloudArcade

If a player wants to purchase a game service, they need to first click the button inside the card to query the price of a certain game service through the interaction with the smart contract. The price will be shown at the top of the game store. If there exist available services, players can then click the button near the price to send a transaction request. A service URL will be notified after a successful payment. If the transaction pends too long time, the user will receive a *txhash* notification, and the *txhash* will be automatically copied into the resend place for players' retry.

After getting the service URL from the server, the game process can be visited and controlled via the support of the GamingAnywhere clients as demonstrated in Figure 8.

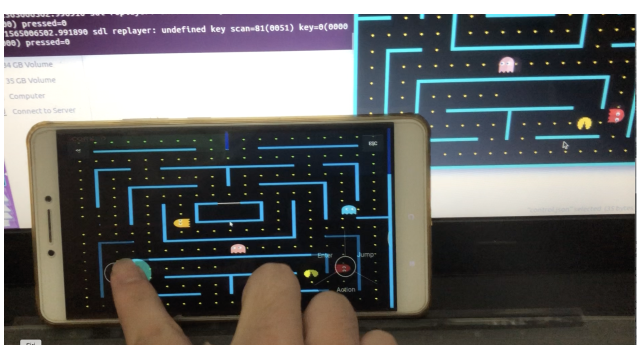

Figure 8: Demonstration of Game Play with CloudArcade

For channel-based CloudArcade, the service price will be automatically shown in the game card. Different from the transaction process of the CloudArcade, players need to first deploy a payment channel. Players can click the button at the bottom of the page to deploy new payment channel, or click the other button to get the payment channel address they created in the past. After a channel is selected, players now can click the button on the game card to send transaction requests. Paid request validation triggers a signature warning from Metamask, as shown in Figure 9. Upon confirmation, the signature will be sent to the server, and if the signature is verified, the game store will notify the player with the service URL. The cumulative cost within the payment channel will appear at the bottom of the page.

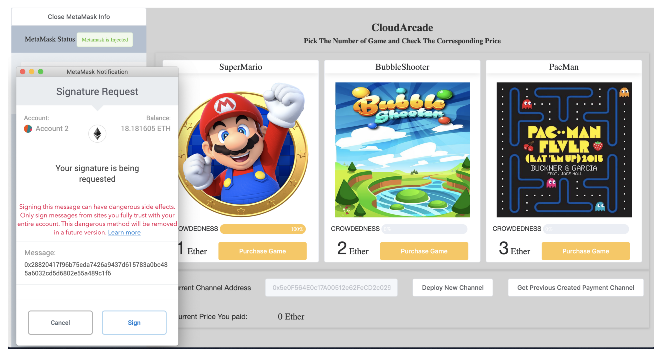

Figure 9: Payment in Channel-based CloudArcade

We design an experiment to compare the performance of two CloudArcade systems in terms of gas fee and time cost. For the gas fee, we look up the fee history recorded by Metamask. The channel-deployment fee is the only fee to be considered for channel-based CloudArcade, while all the transaction gas fees are accumulated for the CloudArcade without the channel support. For time cost, the smart contract hosted by the Rinkeby test net will emit an event to record the precise time-stamp of block-creation, while our client uses function console.time() to note the start time of the payment for CloudArcade. We repeated the transaction experiment 100 times and the system performance data is plotted as follows.

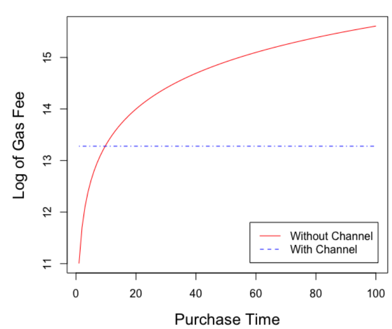

Figure 10: Gas Fee Cost Comparison of two CloudArcades

As depicted in Figure 10, the channel-based CloudArcade consumes more gas when the number of purchase times is less than 10. This phenomenon can be explained by a relatively higher gas fee for channel deployment. However, since the following transactions in channel-based CloudArcade are conducted by signing off-chain signatures, its overall cost is constant. Hence, the accumulated gas fee for the CloudArcade without payment channel will increase proportionally to the times of payments, and eventually higher than that of the channel-based CloudArcade when the players purchase for more than 10 times.

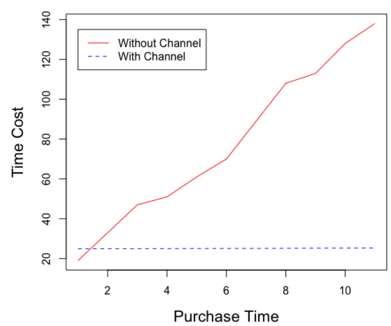

Figure 11: Time Cost Comparison of two CloudArcades

Similar results can be observed from Figure 11, which shows the comparison on the time cost. The payment channel will significantly reduce the average time cost if a player purchases token more than two times. In summary, the channel-based CloudArcade system provides a cheaper and faster payment solution for the repeating players.

7 DISCUSSION ON SERVICE PRICING STRATEGY

In the CloudArcade, the challenging opportunity price is defined as a certain amount of tokens. Though the amount of tokens is relatively stable as we mentioned above, we still need to determine the price for different challenging opportunities. For example, games have diverse resource requirements, which lead to various QoE responses for a single player. At the same time, video quality is always a crucial part of players' gaming experience under different gaming scenarios, pricing of the opportunity cost should also consider this factor.

The price set for the CloudArcade is highly related with user's payoff. To derive a sound price, the most needed information is the user-percieved payoff in CloudArcade. Here we give an analysis of the hybrid user-perceived payoff in the CloudArcade and proposed other possible considerations for the challenging opportunity pricing criteria.

7.1 Hybrid Payoff

CloudArcade owns a composite player pattern of both cloud gaming and traditional amusement arcade. As we mentioned in previous sections, players in CloudArcade simultaneously require high-quality gaming experience. They are customers who are very sensitive to the QoS level but also take expenses into consideration [3]. Thus the simulation of the user QoE should be correlated with both the price and QoS level they received. Saturation phenomenon also exists, which means the enhancement of configuration won't always lead to a better game experience. Another study [6] found that player churn is substantial and increases over time, which means the marginal utility of a certain player continuously decreases with the growth of the accumulated in-game playing time. This finding demonstrated that the players' QoE model should be also linked with the accumulated playing time, as players will choose to quit a game when they cannot receive enough QoE from their game experience.

In sum, a player's payoff function for a certain game should be correlated with the quality level of game experience, game service price, and accumulated playing times.

7.2 Cost Of CloudArcade

The cost of CloudArcade can be separated into fixed operational costs and variable service costs. The fixed cost mainly comes from the license fee for games and the expenditure of the platform maintenance. The variable cost comes from the dynamic resource consumption for different game service.

7.3 Possible Pricing Method

With the payoff and cost function we derived above, various methods can be applied to derive the opportunity price. Based on the parties involved, a game-theoretical approach like the sequential game or probabilistic simulation using time-series approaches like HMM and LSTM can be applied.

8 OPEN ISSUES

8.1 Token Issuing Algorithm

We implement the smart tokens to maintain the users' payment and respond to the market condition, however, the traditional smart token design is not very suitable for CloudArcade. Many tokens are designed to generate at a fixed rate, and the stable supply means the token's price can be a direct reflection of the market demand. However, this nature of blockchain-based token is vulnerable and prone to speculative behavior, which leads to the bubble and price shocks. If blockchain tokens are used as a direct payment method, the violent price fluctuations caused by speculation will seriously affect the quality of game service. In fact, since players' tendency is easy to be tracked during a day, speculation can be more purposeful and destructive. Therefore, it is necessary to design a stable currency mechanism to balance the token's supply and demand to reduce the currency price fluctuation caused by speculation. Solutions in modifying the supply and demand, including collateral, interest rates, currency rate, open market operation, and coin depreciation are widely studied [17]. According to previous studies [19], changing the design of blockchain and modifying the transaction fee considering the time stamp effect can cause the currency price automatically stabilized by absorbing the positive and negative demand shocks. Another problem is that the value stored in the token is threatened by the risk of bankrupt of the project, which means token should also have a valued endorsement. In sum, what we proposed is to use smart contract to issue tokens in a way that the token price can withstand a certain degree of speculation, while the token price has a valued endorsement against another asset.

8.2 Malicious Resource Speculation Precaution

The game service resources are always scarce and limited. Some speculators may use multiple accounts to purchase game services to raise the price for profit. This may disturb the market order, affect the pricing and service experience of other players. Blockchain itself can be used to prevents malicious bots. Inspired by the previous research [18], we can apply a similar centralized registration server to verify the user before they use our system. The requirements can be illustrated as below: 1) no user can register more than one account. 2) each user is an actual human being. 3) each user has his/her Etheruem account. After verification, the verified users' addresses will be then recorded in a smart contract - account bank. Only addresses in the account bank can make further operations in CloudArcade.

9 CONCLUSION

We present CloudArcade, a new cloud gaming business model based on the blockchain-empowered token. It provides a new landscape of the commercial cloud gaming business model, which tackles the various problem of the current cloud gaming business model and pricing strategy.

The service on CloudArcade is paid by token, whose price is a reflection of the market demand. By purchasing and using tokens, customers pay the floating price in a silent and time irrelevant way, which protects the user's utility as well as their service experience. On the other hand, the floating token price also utilizes cloud computing resources via the manipulation of consumers' behaviors. By exploiting the smart contract, we also ensure the transparency of the payment process. The transparency payment builds up user's trust on the platform, implicitly increase in the number of users.

Considering the pricing problem in the CloudArcade, we discussed the service pricing problem, token issuing problem, and malicious resource speculation precaution. The service pricing should consider the payoff of users, including factors such as player group characteristics, video quality level, and price factors into consideration. Several possible approaches are further discussed to provide a reasonable price of the CloudArcade. A desired token issue algorithm in CloudArcade should withstand a certain degree of speculation, also maintained a valued endorsement, and the malicious recourse speculation could be prevented by centralized control.

Considering the good generality of our design, the promotions made by CloudArcade could be extended to other user-oriented and QoE-aware cloud computing services.

ACKNOWLEDGMENTS

This work was supported by Project 61902333 supported by National Natural Science Foundation of China, by the Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), by the National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, China, by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels, and B. Amaba. 2017. Blockchain technology innovations. In *2017 IEEE Technology Engineering Management Conference (TEMSCON)*. 137–141.

[2] P. Andrea and I. Simona. 2019. A Blockchain-Based Decentralized System for Proper Handling of Temporary Employment Contracts. In *Intelligent Computing*, Kohei Arai, Supriya Kapoor, and Rahul Bhatia (Eds.). Springer International Publishing, Cham, 1231–1243.

[3] W. Cai, R. Shea, C. Huang, K. Chen, J. Liu, V. C. M. Leung, and C. Hsu. 2016. The Future of Cloud Gaming [Point of View]. *Proc. IEEE* 104, 4 (April 2016), 687–691.

[4] W. Cai, R. Shea, C. Huang, K. Chen, J. Liu, V. C. M. Leung, and C. Hsu. 2016. A Survey on Cloud Gaming: Future of Computer Games. *IEEE Access* 4 (2016), 7605–7620.

[5] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung. 2018. Decentralized Applications: The Blockchain-Empowered Software System. *IEEE Access* 6 (2018), 53019–53033.

[6] C. Chambers, W. Feng, S. Sahu, D. Saha, and D. Brandt. 2010. Characterizing Online Games. *IEEE/ACM Transactions on Networking* 18, 3 (June 2010), 899–910. https://doi.org/10.1109/TNET.2009.2034371

[7] D. Christian and W. Roger. 2015. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*. Springer, 3–18.

[8] Y. Deng, Y. Li, R. Seet, X. Tang, and W. Cai. 2018. The Server Allocation Problem for Session-Based Multiplayer Cloud Gaming. *IEEE Transactions on Multimedia* 20, 5 (May 2018), 1233–1245.

[9] H. E. Dinaki and S. Shirmohammadi. 2018. GPU/QoE-Aware Server Selection Using Metaheuristic Algorithms in Multiplayer Cloud Gaming. In *2018 16th Annual Workshop on Network and Systems Support for Games (NetGames)*. 1–6.

[10] A. Ghosh, V. Aggarwal, and H. Wan. 2018. Exchange of Renewable Energy among Prosumers using Blockchain with Dynamic Pricing. *CoRR* abs/1804.08184 (2018). arXiv:1804.08184 http://arxiv.org/abs/1804.08184

[11] Y. Han, D. Guo, W. Cai, X. Wang, and Victor C. M. Leung. 2020. Virtual Machine Placement Optimization in Mobile Cloud Gaming through QoE-Oriented Resource Competition. *IEEE Transactions on Cloud Computing* (2020).

[12] C. Huang, C. Hsu, Y. Chang, and K. Chen. 2013. GamingAnywhere: An Open Cloud Gaming System. In *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13)*. ACM, New York, NY, USA, 36–47. http://doi.acm.org/10.1145/2483977.2483981

[13] P. Joseph and D. Thaddeus. 2015. The bitcoin lightning network: scalable off-chain instant payments (2016). *DRAFT Version 0.5* 9 (2015).

[14] T. Min, H. Wang, Y. Guo, and W. Cai. [n.d.]. Blockchain Games: A Survey. In *IEEE Conference on Games (CoG 2019), London, United Kingdom, Aug 20-23*.

[15] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, and R. Brooks. 2016. A brief survey of Cryptocurrency systems. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. 745–752.

[16] G. Perboli, S. Musso, and M. Rosano. 2018. Blockchain in Logistics and Supply Chain: A Lean Approach for Designing Real-World Use Cases. *IEEE Access* 6 (2018), 62018–62028.

[17] I. G. A. Pernice, S. Henningsen, R. Proskalovich, M. Florian, H. Elendner, and B. Scheuermann. 2019. Monetary Stabilization in Cryptocurrencies – Design Approaches and Open Questions. In *2019 Crypto Valley Conference on Blockchain Technology (CVCBT)*. 47–59.

[18] C. Pietro, L. Giovanni, and R. Marina. 2019. TickEth, a Ticketing System Built on Ethereum. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)*. ACM, New York, NY, USA, 409–416. http://doi.acm.org/10.1145/3297280.3297323

[19] K. Saito and M. Iwamura. 2019. How to make a digital currency on a blockchain stable. *Future Generation Computer Systems* 100 (2019), 58 – 69. http://www.sciencedirect.com/science/article/pii/S0167739X18300475

[20] R. Shiftehfar, K. Mechitov, and G. Agha. 2014. Towards a Flexible Fine-Grained Access Control System for Modern Cloud Applications. In *2014 IEEE 7th International Conference on Cloud Computing*. 966–967.

[21] I. Slivar, L. Skorin-Kapov, and M. Suznjevic. 2019. QoE-Aware Resource Allocation for Multiple Cloud Gaming Users Sharing a Bottleneck Link. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 118–123.

[22] B. Tandianus, H. S. Seah, T. D. Vu, and A. T. Phan. 2018. Cloud-Based Dynamic Streaming and Loading of 3D Scene. In *2018 International Conference on Cyberworlds (CW)*. 409–414.

[23] T. Wang, S. Zhang, X. Wu, and W. Cai. 2019. Rhythm Dungeon: A Blockchain-based Music Roguelike Game. In *Foundation of Digital Games (FDG 2019), San Luis Obispo, California, USA, August 26-30*.

[24] B. Xiao, X. Fan, S. Gao, and W. Cai. 2019. EdgeToll: A Blockchain-based Toll Collection System for Public Sharing of Heterogeneous Edges. In *2019 IEEE Conference on Computer Communications Workshops (INFOCOM 2019 WKSHPS), Paris, France, April 29-May 2*. Paris, France.

[25] Y. Yuki, N. Tatsuo, and S. Mizuki. 2016. Blockchain-LI: A Study on Implementing Activity-Based Micro-Pricing Using Cryptocurrency Technologies. In *Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media (MoMM '16)*. ACM, New York, NY, USA, 203–207. http://doi.acm.org/10.1145/3007120.3007151