An Adaptive and Modular Blockchain Enabled Architecture for a Decentralized Metaverse

Ye Cheng, Yihao Guo, Minghui Xu, *Member, IEEE,* Qin Hu, *Member, IEEE,* Dongxiao Yu, *Senior Member, IEEE,* Xiuzhen Cheng, *Fellow, IEEE*

Abstract—A metaverse breaks the boundaries of time and space between people, realizing a more realistic virtual experience, improving work efficiency, and creating a new business model. Blockchain, as one of the key supporting technologies for a metaverse design, provides a trusted interactive environment. However, the rich and varied scenes of a metaverse have led to excessive consumption of on-chain resources, raising the threshold for ordinary users to join, thereby losing the human-centered design. Therefore, we propose an adaptive and modular blockchain-enabled architecture for a decentralized metaverse to address these issues. The solution includes an adaptive consensus/ledger protocol based on a modular blockchain, which can effectively adapt to the ever-changing scenarios of the metaverse, reduce resource consumption, and provide a secure and reliable interactive environment. In addition, we propose the concept of Non-Fungible Resource (NFR) to virtualize idle resources. Users can establish a temporary trusted environment and rent others' NFR to meet their computing needs. Finally, we simulate and test our solution based on XuperChain, and the experimental results prove the feasibility of our design.

Index Terms—Metaverse, Modular Blockchain, Adaptive, Non-Fungible Resource

arXiv:2309.03502v1 [cs.CR] 7 Sep 2023

1 Introduction

The notion of metaverse was first introduced in the 1992 science fiction novel "Snow Crash" by Neal Stephenson. In recent years, this novel concept has gained increasing attention in various fields, from industry to academia [1]. Researchers and developers have been exploring the potential technologies to recreate the immersive and interactive virtual world described in the novel. The goal is to realize a seamless integration of the virtual and the physical worlds, enabling new forms of communication, collaboration, and creativity [2]. As such, the development of metaverse has significant implications for various industries such as gaming and e-commerce, as well as for social and cultural practices [3], [4].

Blockchain technology [5] has achieved great success in the field of cryptocurrency due to its decentralization, transparency, and immutability, and it has been widely regarded as one of the essential technologies to realize a metaverse [2]. Many studies [6], [7] have investigated the role of blockchain within a metaverse and recognized that blockchain-enabled metaverse can provide a trusted environment for users who do not trust each other in the metaverse. However, besides the aforementioned advantages, a blockchain-enabled metaverse still has two key challenges that need to be addressed.

The first challenge is the incompatibility between the dynamic nature of the metaverse and the static consensus/ledger mechanism inherent to a current blockchain system. Specifically, as the scenes and participants within a metaverse undergo constant changes, the security and system performance requirements for the metaverse also vary accordingly. For example, users may exhibit higher levels of trust when the scenes are limited to a single company, but their trust diminishes when scenes require collaboration among multiple companies. In comparison, the consensus mechanism and ledger in a blockchain are typically static. During initialization of the blockchain, specific consensus algorithm and ledger structure are predetermined, which are suitable for the initial development. However, they are not well-suited for dynamically changing scenarios in the metaverse. XuperChain [8] and Ethereum [9] have attempted to dynamically replace the consensus algorithm of a blockchain, but they are dependent on human interventions and have issues such as high delays and low fault-tolerance.

The *Consensus/Ledger Adaptation (CLA) problem* arises due to the complex coupling design of a blockchain system, making it difficult to dynamically change its consensus algorithm and ledger structure. Human-driven strategies such as those taken by XuperChain and Ethereum face difficulties in terms of time consumption as they need to keep up with the fast-paced changes within the metaverse. Moreover, relying on active human participation introduces artificial behaviors, posing threats to the security of the system. To address this challenge, an adaptive consensus/ledger replacement scheme needs to be designed to eliminate the effect of human factors and cater to the changing demands of the metaverse.

Second, the increasing demand on computing power in a metaverse has raised an entry barrier for users. Numerous metaverse scenes, including gaming [10] and industrial manufacturing [11], require significant computational power to provide immersive experiences and precise models. However, users who lack adequate computational

Corresponding author: Minghui Xu.
Y. Cheng, Y. Guo, M. Xu, D. Yu, and X. Cheng are with the School of Computer Science and Technology, Shandong University, Qingdao, Shandong, China (e-mail: {yech@mail., yhguo@mail., mhxu@, dxyu@, xzcheng@}sdu.edu.cn).
Q. Hu is with the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis, USA (e-mail: qinhu@iu.edu).

This is the author's manuscript of the article published in final edited form as:
Cheng, Y., Guo, Y., Xu, M., Hu, Q., Yu, D., & Cheng, X. (2024). An Adaptive and Modular Blockchain Enabled Architecture for a Decentralized Metaverse. IEEE Journal on Selected Areas in Communications, 42(4), 893–904. IEEE Journal on Selected Areas in Communications. https://doi.org/10.1109/JSAC.2023.3345432

resources face difficulties in participating, hindering the widespread adoption of the metaverse. At present, users can acquire computational power through hardware upgrades or by renting servers. However, the former approach involves significant asset investment, and the latter benefits the giant centralized service providers while limits the opportunity for the small ubiquitous computational power providers. One reason resulting in such a limitation is due to the absence of a trusted distributed trading platform. This problem is known as the *Resource Consolidation (RC) problem*, whose solution can help to utilize idle or under-utilized resources hooked on the Internet. To overcome it, a trusted distributed transaction environment based on blockchain is necessary; nevertheless, blockchain cannot directly trade physical computational resources. Therefore, a resource virtualization solution needs to be developed that can fulfill the on-chain transaction requirements and meet the computational power demands of metaverse participants.

In this paper, we present an adaptive and modular blockchain-enabled architecture for a decentralized metaverse, aiming to effectively tackle the above two challenges. More specifically, to address the CLA problem, we propose the dynamic and adaptive consensus/ledger replacement protocols leveraging a modular blockchain to minimize the costs associated with consensus and ledger replacement. Additionally, we incorporate AI models to ensure adaptability to the constantly changing metaverse environment with human intervention. To solve the RC problem, we introduce the concept of Non-Fungible Resource (NFR) and propose a protocol for NFR virtualization. This protocol allows users to upload their idle resources onto the blockchain for trading. Furthermore, we implement a temporary trusted environment to reduce on-chain resource consumption.

Our contributions are highlighted as follows:

- We introduce an adaptive and modular blockchain-enabled architecture for a decentralized metaverse, highlighting the crucial role of blockchain in its development.
- To solve the CLA problem, we propose adaptive protocols for consensus and ledger replacements. These protocols enable the dynamic adaptation to the constantly-changing metaverse environment.
- To solve the RC problem, we present the concept of NFR via which computing resources can be leased in the metaverse, successfully realizing the resource virtualization and effectively providing support to the nodes in need of computational resources.
- We build a prototype of the mertaverse architecture to demonstrate our design and validate its performance, which can contribute to the metaverse community and help developers in designing practical metaverse applications.

The rest of the paper is organized as follows. Section 2 provides a comprehensive review on the most relevant works. In Section 3, we present the preliminary knowledge necessary to understand the subsequent sections. Section 4 delves into details of the proposed blockchain-enabled metaverse architecture, whose prototype implementation and performance evaluation are discussed in Section 5.

2 Related Works and Motivations

In this section, we first summarize the status quo of metaverse in both academia and industry, then conduct an investigation on the roles that blockchain can play in a metaverse, and finally describe the motivations behind our design on a decentralized metaverse.

2.1 Metaverse in Industry and Academia

Efforts to develop metaverse technologies in industry primarily focus on exploring its feasibility in various fields. In an online office, a metaverse allows users to display 3D physical models in a virtual space and provide strong interaction capabilities, thereby improving user engagement and work efficiency. For examples, Microsoft Mesh [12] and Horizon Workroom [13] enable multi-person meetings, while NVIDIA Omniverse [11] allows real-time collaborative 3D modeling. In the gaming industry, a metaverse offers users an authentic gaming experience, rich game content and diverse gameplay. Users no longer are just participants in a game but also become game designers and developers. Examples include Second Life [14], Minecraft [15], and Roblox [10], which aim to strike a balance between game authenticity and user openness.

In academia, researchers focus on different aspects of a metaverse such as architecture and security. They also conduct experiments to evaluate scenario usability and propose solutions to address various challenges. Radoff [16] introduced a seven-layer metaverse architecture. Nair *et al.* [17] proposed a privacy-preserving framework to enhance the security of VR devices. Duan *et al.* [18] presented a metaverse prototype for university campuses. Wang *et al.* [2] reviewed the security and privacy issues of major metaverse solutions. Kye *et al.* [19] investigated the opportunities and challenges of a metaverse in education.

2.2 Blockchain for Metaverse

Blockchain can provide a decentralized and trusted interactive environment for a metaverse. Decentraland [20] is a decentralized platform based on Ethereum, enabling users to obtain rewards through NFT creation, trade, and use/consumption. Cryptovoxels [21] provides a highly open virtual world on Ethereum, where players can unleash their creative potential and craft their own custom items. Yang *et al.* [22] emphasized the importance of AI and blockchain in a metaverse. Huynh *et al.* [23] studied the role of blockchain in a metaverse from a technical perspective. Fu *et al.* [6] conducted a comprehensive review on the use of blockchain and intelligent networking in creating immersive metaverse experiences. Huang *et al.* [24] discussed the potential of combining metaverse and blockchain via the building information modeling technology to transform the physical world into an exciting digital one. Metarepo [25] leverages blockchain technologies to provide a secure mechanism for storing the digital assets in a metaverse. By integrating artificial intelligence and blockchain, the metaverse proposed by Ali *et al.* [26] can provide secure and efficient healthcare services. Xu *et al.* [7] presented a blockchain-based metaverse framework to integrate hardware and software resources, stressing blockchain as the underlying layer

that provides a trusted environment with high security and privacy.

2.3 Motivations

Based on the above summary and analysis, one can see that the development of metaverse in industry is still in its early stage and is currently limited to specific application scenarios, such as online collaboration and gaming. In academia, research efforts are largely dedicated to the proposals on innovative concepts and architectures. Even though some studies outlined in Section. 2.2 highlight the importance of blockchain in a metaverse, they often remain at the theoretical level without considering specific system implementations. Furthermore, these existing works overlook the CLA and RC issues introduced in Section. 1, which seriously hinder the development and implementation of a practical metaverse. To address these concerns, we propose an adaptive decentralized metaverse based on modular blockchain technologies. Our metaverse makes use of adaptive consensus/ledger changes and resource virtualization, effectively reducing the entry barriers for users and increasing the system security.

3 Preliminaries

In this section, we present the necessary background information on modularized blockchain, Non-Fungible Token (NFT), and off-chain payment channel.

3.1 Modularized Blockchain

Modularized Blockchain is a type of blockchain architecture that aims to improve scalability, flexibility, and interoperability by partitioning different components of a blockchain into modules or layers. In a modularized blockchain, each module is designed to perform a specific function such as consensus, data storage, or smart contract, and can be developed and updated independently [27].

A modularized architecture enables a blockchain to adapt to different use cases and business requirements by allowing developers to customize and plug-in different modules. For example, a business can choose a specific consensus algorithm or data storage module that suits its own needs, rather than being restricted to the default options of a blockchain platform. Moreover, modularized blockchains facilitate interoperability between different blockchain networks by enabling modules to communicate with each other through well-defined interfaces. This allows for the exchanges of data and assets between different blockchains, which can improve the efficiency and convenience of cross-chain transactions. Several blockchain projects such as XuperChain [8] and Celestia [28] have adopted a modularized blockchain architecture to achieve better scalability and interoperability.

3.2 Non-Fungible Token

Non-Fungible Token (NFT) [29] is a digital asset that represents ownership or proof of authenticity of a unique item or a piece of content. Unlike cryptocurrencies such as Bitcoin, which are interchangeable and have the same value, NFTs are unique and each has its own distinct value. NFTs are based on blockchain technologies and are created using smart contracts that verify the ownership and uniqueness of the asset. The concept of NFT has gained significant attention in recent years, particularly in the art world, where NFTs have been used to authenticate and sell digital arts. NFTs provide a way for artists to monetize their digital creations, as the ownership of an NFT can be transferred between buyers and sellers in a secure and transparent way. Additionally, NFTs have the potential to revolutionize the gaming industry by allowing players to truly own in-game items and assets.

The process of using an NFT involves creating or purchasing the NFT, which is typically done through a blockchain platform such as Ethereum that supports NFTs. An NFT is created via a smart contract that includes details about the item or content being represented, such as its ownership, provenance, and any additional term or condition of the sale or transfer. Once the NFT is created, it can be sold or transferred between individuals, typically through an online marketplace or auction. The transaction is recorded on the blockchain, which ensures that the ownership and authenticity of the NFT are transparent and immutable. Owners of NFTs can display or store them in digital wallets or galleries, and may be able to earn revenue from their NFTs through licensing or resale. The popularity of NFTs has grown rapidly in recent years, with a few high-profile sales reaching millions of dollars [30], [31].

3.3 Off-chain Payment Channel

The concept of off-chain payment channels was first proposed by Joseph Poon and Thaddeus Dryja in their paper talking about Lightning Network [32]. It is a technique used in blockchain technologies to enable faster and cheaper transactions between two parties without having to wait for their transactions to be recorded on a blockchain. This is achieved by creating a temporary payment channel between the two parties off-chain.

Definition 1 (The Process of Off-chain Payment Channel)**.** *An off-chain payment channel involves collaborative interactions between on-chain and off-chain. The whole process can be divided into three steps: Open, Transfer, and Close.*

- Open (on-chain). Both parties initiate a request to a smart contract separately via transactions to apply for opening a channel and deposit a certain amount of funds. The smart contract verifies the correctness of the transaction signatures and deposit amounts, then sets the deposited amounts from both parties as the initial state of the channel. At this point, the channel is opened.
- Transfer (off-chain). Both parties can send transactions back and forth within the channel multiple times. Each transaction needs to be executed by both parties to update the channel state, and the updated state needs to be signed by both parties, which is called a transaction certificate. A transaction certificate includes a timestamp that records the time at which the channel state was updated.
- Close (on-chain). When the interactions between the two parties are complete and the channel needs to be

closed, one party (agreed by both parties) can upload the final transaction certificate to the smart contract. The smart contract then verifies the correctness of the signatures and the final state on the certificate, subsequently updating the corresponding state on the blockchain. If the selected party behaves maliciously and fails to upload the correct final transaction certificate (e.g., one provides a certificate that is biased towards its own interest rather than reflecting the final result), the other party can appeal by providing the correct certificate within a designated time frame. The smart contract would verify the authenticity of the certificates by cross-checking the signatures and timestamps provided by both parties.

Off-chain payment channels only support off-chain exchanges of currency, which limits their application scenarios. To overcome this restriction, various relevant schemes such as off-chain state channels [33], cross-chain channels [34], and privacy-preserving off-chain channels [35] gradually emerged to increase the universality, security, and efficiency of off-chain payment channels.

4 An Adaptive Decentralized Metaverse

In this section, we first provide an overview on our proposed metaverse system, then detail the protocol, and finally describe a specific work scenario of the metaverse.

4.1 Overview

As shown in Fig. 1, our proposed metaverse architecture is designed based on a modular blockchain and it mainly consists of three components to support the operations of a metaverse, in which a user can participate through VR glasses or other devices.

First, considering the challenges that constantly changing metaverse scenes pose on the blockchain consensus and ledger, we propose adaptive consensus/ledger protocols (introduced in Section. 4.2) that can automatically determine the most suitable consensus/ledger based on the current state of the entire metaverse, and substitute the consensus/ledger through hot plugging, effectively improving the security of the metaverse and conserving system resources. This consideration is based on the modular blockchain design, which allows a low cost consensus change. The hot-pluggable consensus/ledger replacement avoids the deficiency of downtime updates. The introduction of the adaptive protocol with machine learning algorithms reduces the interference of human factors and increases the system's security and robustness, further addressing the CLA problem (as described in Section. 1).

Second, a user can virtualize its idle resources, which could be large computers or small sensors, based on our NFR (Non-Fungible Resource) design (introduced in Section. 4.3), and upload the NFRs into the metaverse blockchain for rent, maximizing the resource utilization and thereby efficiently addressing the RC problem (as described in Section. 1).

Finally, in addition to the adoption of our TME (Trusted Metaverse Environment) design (introduced in Section. 4.4) to purchase and use NFR, we equip the TME with enhanced security, reliability, and parallel processing abilities through the utilization of an improved Local Trust Model (shown in Section. 4.4.1) and an On-Demand Trusted Metaverse Cluster (shown in Section. 4.4.2). Users can join specific metaverse scenes, such as social, NFT, and work, through TME. The transaction records of NFRs are written into the blockchain to ensure the correctness of interactions between untrusted parties. A large amount of computational tasks are carried out off-chain to reduce on-chain resource consumption. It is worth mentioning that NFR enables users with insufficient resources to join a scene, lowering the threshold for entering the metaverse and thereby fully embodying a people-centric metaverse design.

4.2 Adaptive Modular Blockchain

To solve the CLA problem, we propose an adaptive consensus protocol and an adaptive ledger protocol based on a modular blockchain.

4.2.1 Adaptive Consensus Protocol

Hot-plugging is an ability to replace or install a component without shutting down the computer it attaches. In a metaverse, the complexity of the scenario dictates the need for consensus algorithms to be constantly changing. Therefore, when facing new requirements for performance and security, a developer does not need to design a new blockchain to meet the demands but can upgrade the consensus algorithm online instead, as long as the blockchain system supports hot-pluggable consensus.

A simplified process such as the one in XuperChain to support hot-pluggable consensus can be summarized as follows. First, a user initiates a consensus replacement proposal by calling a smart contract, specifying the name of the consensus algorithm to be replaced and that of the new one, the block height at which the voting would be collected, the trigger condition for the consensus replacement, and the block height at which the new consensus should take effect. Then, other users in the blockchain system vote on the proposal. When the number of votes collected in the smart contract meets the trigger condition for the consensus replacement, the smart contract marks the proposal as "success", and all nodes would execute the consensus replacement operation to substitute the consensus, with the new one starting to be effective at the designated block height. If the smart contract fails to collect enough votes, the proposal would be marked as "failure", which signals that the consensus update fails and the nodes do not need to perform any operation.

However, in a metaverse, users pay more attention to their experience rather than the consensus algorithm adopted by the underlying blockchain system. In fact, the general public may not have any knowledge about consensus. This implies that ordinary users may not be able to initiate consensus proposals following the approach the current hot-pluggable consensus mechanism is implemented (which is more likely a developer's behavior). Therefore consensus changes should be transparent to the users of the metaverse. To realize this objective, we design an adaptive consensus mechanism based on the modular blockchain that supports hot-pluggable consensus and obtain a blockchain

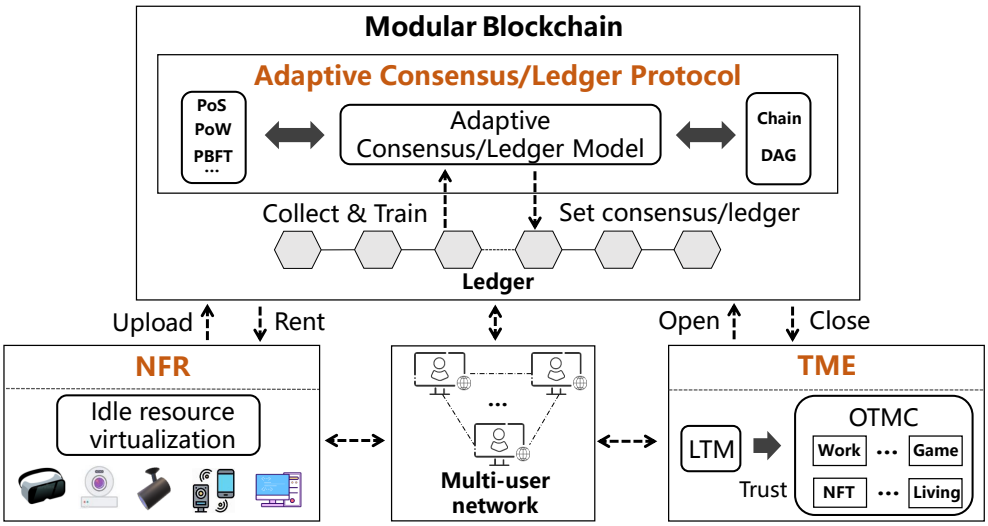


Fig. 1. The architecture of a decentralized metaverse. The adaptive consensus/ledger protocols are responsible for collecting data from the metaverse and learning to set up a consensus/ledger that can better suit the scenario. Users can virtualize idle resources as NFR (Non-Fungible Resources) and rent them on the blockchain. The metaverse supports TME (Trusted Metaverse Environment) to realize a locally trusted working environment.

that can select an optimal consensus algorithm according to the network scale, number of error nodes, network delay, throughput, and other indicators, to replace the consensus that does not work well due to situation changes, thereby improving the performance and security of the underlying blockchain system.

Our adaptive consensus protocol needs a data set that can be obtained by testing the throughput and latency of various consensus algorithms under different network scales, error node ratios, and network delays, over a modular blockchain supporting hot-pluggable consensus. The tags of the data set are various consensus algorithms while the features are indicators such as network scale, error node ratio, and network delay. One can use this data set for training and testing different adaptive consensus models, making use of various machine learning algorithms such as random forest, gradient boosting decision tree (GBDT), XGBoost, and LightGBM [36]. The consensus with the highest accuracy is selected and plugged into the blockchain to take effect in the next.

We summarize the whole process of the adaptive consensus protocol as follows. First, the adaptive consensus protocol obtains the current network state at a fixed interval to identify the optimal consensus algorithm and compares it with the one currently used by the blockchain. If they are the same, the update process stops; otherwise, the protocol broadcasts a transaction (proposal) to the blockchain containing the name of the consensus algorithm to be updated and other relevant information for hot-pluggable consensus change. When a node in the blockchain system receives the transaction, it calls a voting contract, which can count the votes and verify whether the current number of votes has reached the threshold for updating the proposed consensus algorithm. If the threshold is reached, all honest nodes automatically execute the update process of the consensus and broadcast a "success" message after the consensus update is successful. If a node fails to complete the update at the specified block height, the node cannot participate in the subsequent consensus and needs to synchronize its status with other nodes. Note that the consensus algorithm library of the modular blockchain is renewable. When someone proposes a new consensus algorithm, the implementation of the algorithm can be added to the library. Therefore, it is necessary to test and count the indicators of the new consensus off-chain, update the data set, and finally retrain the adaptive consensus model.

4.2.2 *Adaptive Ledger Protocol*

At present, blockchain ledgers can be categorized into two types: chain-based and DAG-based [37]. In a chain-based ledger, each block, except for the genesis one, stores the hash value of the previous block in its block header. The blocks are then linked together by hashes in a chain-like structure. Examples of prominent blockchains that use a chain-based ledger include Bitcoin [5] and Ethereum [9]. In contrast, a DAG-based ledger creates and stores a directed acyclic graph (DAG) of relationships between blocks or transactions. DAG-based ledgers offer higher concurrency than chain-based ones, which results in increased throughput of the blockchain system. However, maintaining consistency in DAG-based ledgers is more challenging, as conflicts take longer time to resolve. Some state-of-the-art DAG-based blockchains include IOTA tangle [38], Conflux [39], and Sui [40]. The choice of ledger implementation should depend on specific application requirements.

We present a novel adaptive ledger protocol that leverages the modular blockchain to enable automated conversion between chain-based and DAG-based ledgers. The conversion process relates to consensus replacement since the two ledger types employ different consensus algorithms. In a manner akin to the adaptive consensus protocol, an additional model is trained for the purpose of facilitating decision-making pertaining to the conversion of ledgers. In order to initiate the conversion process, a miner creates

a smart contract as Fig. 2. The LedgerConversion contract comprises of the following elements: function ChainToDAG(), function DAGToChain(), function vote(), solidity mapping isVote, and uint voteCount. The contract LedgerConversion proceeds in two major steps: voting and ledger conversion. The solidity mapping isVote records the miner address that has cast a vote, thus preventing any miner from voting twice. When a miner invokes vote(), voteCount is incremented after verifying isVote; then the function checks whether the current voteCount has surpassed the threshold. Upon reaching the threshold, nodes all execute ChainToDAG() or DAGToChain(). The call to either function requires the two parameters convertHeight (also denoted by $\hat{H}$ in the following context) and consensusName to be passed, where convertHeight specifies the block height at which the new ledger comes into effect, and consensusName represents the name of the consensus algorithm to be adopted following the conversion.

LedgerConversion Contract

```
contract LedgerConversion {
  mapping (address => bool) public isVote;
  uint voteCount;
  function vote() public {
    require(isVote[msg.sender] == false);
    voteCount++;
    if (count >= threshold) {
      execute either ChainToDAG() or DAGToChain();
    }
  }
  function DAGToChain (uint convertHeight, string
  consensusName) public {···}
  function ChainToDAG (uint convertHeight, string
  consensusName) public {···}
}
```

Fig. 2. A simplified example of smart contract for ledger conversion.

A large number of transactions in a miner's transaction pool can be processed more efficiently by converting a chain-based ledger to a DAG-based structure, thereby increasing concurrency. If a strongly-consistent consensus algorithm such as PBFT is used before ledger conversion, there will be no fork before $\hat{H}$. However, if a weakly-consistent consensus algorithm is used, the block at $\hat{H}-1$ may not be determined when all nodes are in the process of $\hat{H}$. As shown in Fig. 3, proposed blocks or transactions from different miners at height $\hat{H}$ may point to different branches (forks) before $\hat{H}$, causing inconsistency. To ensure that all miners record the same hash pointer in the proposed blocks or transactions at $\hat{H}$, a distributed randomness beacon is used to conduct leader election between $\hat{H}-1$ and $\hat{H}$. Then the leader selects one path to be committed, and all proposed blocks or transactions at $\hat{H}$ point to the same block, i.e., the leader, ensuring ledger consistency after conversion.

When a miner encounters a few pending transactions in its transaction pool, it can initiate a transaction to invoke DAG-Chain conversion, which can reduce the likelihood of transaction conflicts and strengthen the consistency of the ledger. In the case when the blockchain system operates on a strongly-consistent DAG-based consensus algorithm, there will be no fork at height $\hat{H}$. In the case when the blockchain

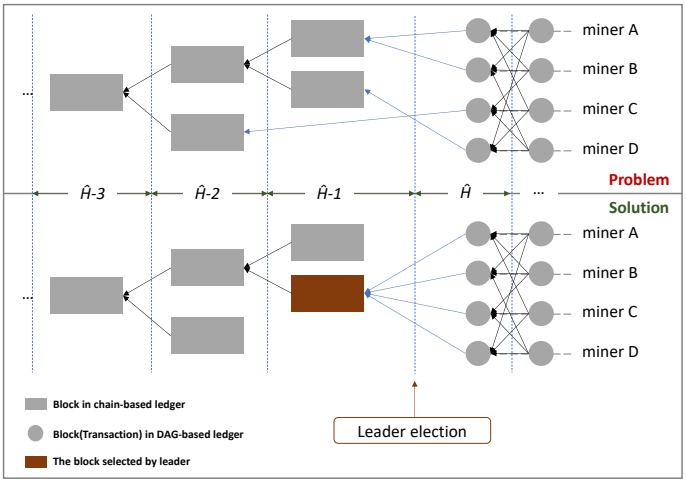


Fig. 3. The challenge and solution concerning the chain-DAG conversion.

system utilizes a weakly-consistent DAG-based consensus algorithm, there is a probability that forks might occur in the ledger starting from $\hat{H}$ as depicted in Fig.4. Nevertheless, such a fork does not impact the eventual consistency of the ledger after $\hat{H}$ since a chain-based consensus algorithm can ensure the consistency through the longest chain rule or the GHOST rule.

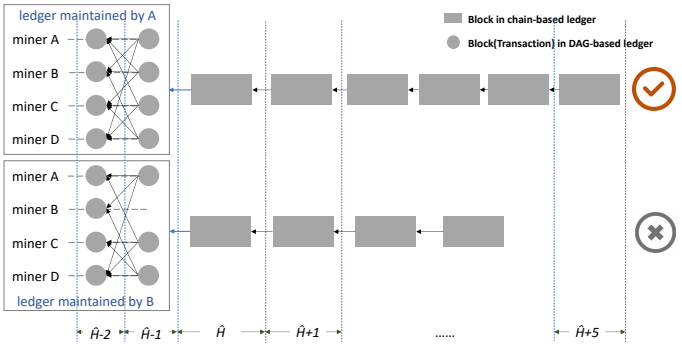


Fig. 4. The challenge and solution concerning the DAG-chain conversion.

4.3 Non-Fungible Resources

A metaverse needs to integrate the computing capabilities of various devices hooked on the Internet. An NFT (Non-Fungible Token) can be understood as a certificate of virtual or physical asset, which is indivisible, irreplaceable, and unique [41]. Recently, NFT has been developed rapidly, expanding from games and collectibles to music, real estate, art, and finance [42], [43]. However, NFT focuses on digital ownership and uniqueness, and it does not support functionalities such as virtualization, access control, and fair-trading of hardware resources, making it incapable of addressing the RC problem. To overcome this issue, we propose Non-Fungible Resource (NFR), which can virtualize computational resources including CPU, GPU, and Disk, and map them to tokens recorded by a blockchain.

We implement NFR using the NFR contract as presented in Fig. 5. An NFR is described as a solidity struct NFR, includ-

ing uint tokenID, string resourceOwner, string resourceType, and uint price. The string tokenID is the unique identifier of an NFR. The contract includes the following functions: Registration(), Rental(), Liquidation(), and Cancellation(). The function Registration() is to register the resources from users and returns NFRs to the them. An NFR can only be rented by one user at a time. When a resource-constrained node wants to complete a computing task, it needs to call the function Rental() to rent some NFRs on demand. When calling Rental(), nodes need to provide the rent time and pay a deposit for preventing malicious rentals. If a resource-constrained node completes the computing task, it returns the NFR by calling the function Liquidation(), which automatically settles the cost of rental and return the balance of the deposit to the node. If the node fails to liquidate until the timeout, Liquidation() deduct the deposit from the node and retracts the NFR which is rented by the node. If the owner of a resource no longer rents out its resource forever, it calls the function Cancellation() in the contract to cancel the NFR corresponding to the resource.

NFR Contract

```
contract NFR {
  struct NFR {
    uint tokenID;
    string resourceOwner;
    string resourceType;
    uint price;
  }
  function Registration (string resourceType, uint price)
  public {···}
  function Rental (uint tokenID, uint deposit) public {···}
  function Liquidation (uint tokenID) public {···}
  function Cancellation (uint tokenID) public {···}
}
```

Fig. 5. An example of simplified smart contract of NFR.

Maintaining consistency between digital NFRs and their corresponding physical resources is of utmost importance. Inconsistencies can lead to vulnerabilities, particularly resource fraud attacks. Resource fraud occurs when a user pays for NFRs, but the malicious owners of the NFRs do not provide the corresponding physical resources, resulting in unreliable resources for the user despite payment. To address this issue, we propose a scheme that guarantees the availability and reliability of digital resources corresponding to NFRs in the physical world. We utilize Proofs of Storage (PoS) technique to ensure the validity of storage space when an NFR corresponds to storage resources. PoS allows a user uploading data to a server to repeatedly verify if the server is storing data correctly. For NFRs corresponding to CPUs/GPUs, we employ the Proof of Work (PoW) technique. The server responds to users with a random challenge along with a proof every fixed period of time. If the proof is valid, the resource owner has idle CPU/GPU to serve computing tasks.

In conclusion, our scheme ensures the consistency between digital NFRs and physical resources, thereby mitigating the risk of resource fraud attacks.

4.4 Trusted Metaverse Environment

In this section, we demonstrate the process of enabling parallel computing tasks in a metaverse where users are mutually untrusted. The entire platform consists of two components, with the first assessing the trust of the users involved and the second constructing an on-demand trusted metaverse cluster.

4.4.1 *Trust Evaluation of Computing Nodes*

Trust is an important prerequisite for cooperation, which is influenced by historical behaviors and the current state of the nodes. The crisis of trust between nodes has seriously affected the development of distributed systems [44], [45], and a metaverse is facing similar problems. For example, a node in a metaverse may make a false description of its shared resources to seek improper benefits.

In a typical metaverse, nodes form different groups according to their computing tasks, and each group works independently. The participation of a large number of heterogeneous devices in the metaverse makes the relationship among all nodes complex and challenging to describe. To overcome this difficulty, we adopt the Local Trust Model (LTM) mentioned in [7], which satisfies the *locality* property and can effectively solve the above trust evaluation problem. To be specific, the locality property of LTM implies that the nodes irrelevant to a metaverse task do not need to join in the trust evaluation of the task. In fact, these irrelevant nodes themselves are unwilling to participate in the task to avoid consuming excess resources.

Based on the LTM model, we further refine the evaluation criteria and assessment algorithm for the trustworthiness in an LTM. To provide a symbolized expression of the LTM, we denote the metaverse network as a weighted hypergraph [46] $\mathsf{H} \stackrel{\text{def}}{=} (\mathsf{V}, \mathsf{E}, \mathsf{W})$, where V represents the set of nodes, E the set of hyperedges, and W the set of weights. Fig. 6 shows an example, which illustrates four LTMs in a metaverse network formed by six nodes (marked as $\{\mathsf{v1}, \mathsf{v2}, \cdots, \mathsf{v6}\}$). One can see that LTM 3 consists of three nodes $\{\mathsf{v2}, \mathsf{v3}, \mathsf{v4}\}$ connected by a hyperedge. The trust value of LTM 3 is determined by the degree of trust among $\{\mathsf{v2}, \mathsf{v3}, \mathsf{v4}\}$, and the factors considered according to specific tasks can include the network size, the message latency, and the historical behaviors of the nodes. We adopt the Oracle mechanism [47] to aggregate the metaverse network data to calculate trust. After obtaining the data, nodes in the LTM can choose a trust evaluation algorithm, e.g., Powertrust [48], PET [49], and those in [50], [51], from the trust evaluation module and obtain trust values. Significant elements of the trust evaluation process, such as the participating nodes, the selected trust evaluation algorithm, and the trust values, are recorded on the blockchain, and the smart contract allocates appropriate amount of NFRs to the LTM according to its computing task and the trust values.

4.4.2 *On-Demand Trusted Metaverse Cluster*

Considering that there are many independent and parallel computing tasks in a metaverse, we adopt the On-Demand Trusted Computing Environment technique presented in [7]. However, due to limitations of the computing

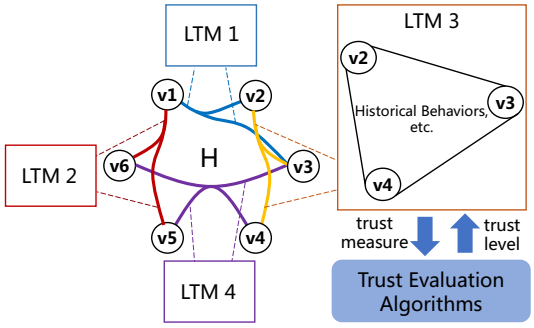


Fig. 6. An example of trust evaluation.

power and storage, some nodes are unable to participate. Therefore, we propose the concept of On-Demand Trusted Metaverse Cluster (OTMC) based on NFR (introduced in Section. 4.3), which allows users to rent NFRs on the chain to acquire the necessary computing resources, thereby alleviating the resource limitation problem. The main idea of OTMC is to create a temporary trusted environment for nodes participating in a task. Note that most of the metaverse tasks are put off-chain to complete, and only some state information of an OTMC needs to be recorded on-chain, such as $\mathsf{Open}, \mathsf{Run}$, and Close (shown in Fig. 7). Multiple computational tasks in the metaverse may occur simultaneously, therefore all OTMCs could be executed in parallel. The participating nodes of each computing task only need to form an OTMC off-chain and rent NFRs on demand to perform the computing task. Since the computing process is off-chain, the process of each OTMC is independent and does not occupy the resources on the blockchain.

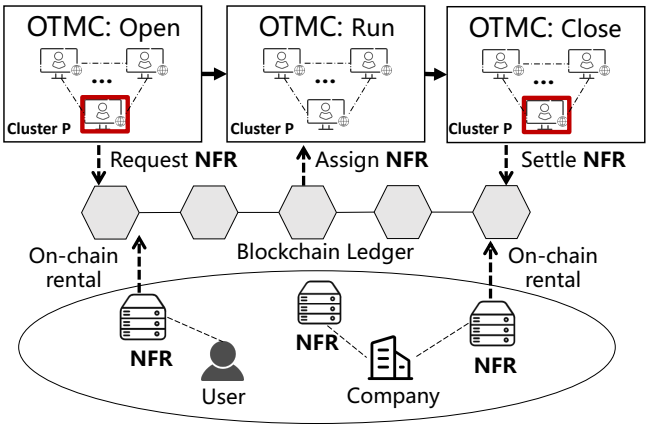


Fig. 7. The lifecycle of an OTMC, which contrains three stages: Open, Run, and Close. Nodes with insufficient resources (marked by the red boxes) can participate in a task by renting NFRs.

Figure 7 presents the lifecycle of an OTMC, denoted by P. We take P as an example to illustrate the details of an OTMC. Formally, we denote an OTMC by a vector $\vec{C}^{\mathsf{State}}_{\mathsf{Cluster}} \stackrel{\mathsf{def}}{=} (\mathsf{CID}, \mathsf{State}, \mathsf{G}, \Delta\mathsf{T}, \mathsf{NFR}, \mathsf{Results})$, where CID is a unique identifier of the OTMC, State represents the current state of the OTMC ($\mathsf{State} \in \{\mathsf{Open}, \mathsf{Run}, \mathsf{Close}\}$), G represents the participating nodes, $\Delta\mathsf{T}$ is a time duration that regulates how long $\vec{C}$ can last, NFR represents the resource borrowing records of the participating nodes, and Results is the set of results that needs to be uploaded when $\vec{C}$ attempts to be closed. Therefore, we can use $\vec{C}^{\mathsf{Open}}_{\mathsf{P}} = (\mathsf{CID}_{\mathsf{P}}, \mathsf{Open}, \mathsf{G}_{\mathsf{P}}, \mathsf{t}+\delta, \mathsf{NFR}_{\mathsf{P}}, \perp)$ to denote the initial state of the cluster P, which states that multiple nodes form a group G_{P} and send $\vec{C}^{\mathsf{Open}}_{\mathsf{P}}$ to the blockchain to establish an on-demand trusted metaverse cluster P at time t with a duration δ. Note that, we use the block height to measure δ to prevent the clock out-of-sync problem among the participants in G_{P}.

If G_{P} needs to rent resources through NFR, it should deposit certain amount of money in the smart contract. Then, it can send requirements, such as computing power consumption and time, to the smart contract. The smart contract would allocate the NFR resources to the corresponding users and record this in $\mathsf{NFR}_{\mathsf{P}}$. After the blockchain consensus, the state of P changes from $\vec{C}^{\mathsf{Open}}_{\mathsf{P}}$ to $\vec{C}^{\mathsf{Run}}_{\mathsf{P}} = (\mathsf{CID}_{\mathsf{P}}, \mathsf{Run}, \mathsf{G}_{\mathsf{P}}, \mathsf{t}+\delta, \mathsf{NFR}_{\mathsf{P}}, \perp)$, indicating that the participants can start to interact. P needs to be closed by the time $\mathsf{t}+\Delta\mathsf{T}$, otherwise it would be punished (deducting money deposited in the smart contract). Participants upload the results of the interactions to the blockchain, and the miners close P after successful verification via $\vec{C}^{\mathsf{Close}}_{\mathsf{P}} = (\mathsf{CID}_{\mathsf{P}}, \mathsf{Close}, \mathsf{G}_{\mathsf{P}}, \mathsf{t}+\Delta\mathsf{T}, \mathsf{NFR}_{\mathsf{P}}, \mathsf{Results})$. The smart contract settles down the NFR according to the content of $\mathsf{NFR}_{\mathsf{P}}$ and returns the excess amount of the deposit.

5 IPerformance Evaluation

In this section, we present a concrete implementation of the major components of our proposed metaverse architecture.

5.1 Implementation and Experiment Setup

We choose XuperChain[1] and Solidity[2] as building blocks, and implement the adaptive modular blockchain on top of them. Our blockchain system supports three types of consensus algorithms, namely PoW, PoA, and TDPoS. The smart contracts of NFR and OTMC, which are the key components to enable our decentralized metaverse, are realized with Solidity. The adaptive consensus protocol relies on a machine learning model, which intends to choose the most appropriate consensus algorithm based on the number of nodes, the ratio of faulty nodes, and the hardware resources.

We test the performance of our adaptive modular blockchain using up to 50 instances distributed in various regions on TencentCloud. In order to compare the impact of different computing power resources on the performance of the consensus algorithms, we set up two networks, which only differ in the amount of computational resources. The first one includes up to 50 S6.MEDIUM4 instances, with each having a 2-Core CPU (Intel Ice Lake 2.7/3.3 GHz), 4GB memory, and 50GB SSD, and running Ubuntu 20.04 LTS; while the second one contains up to 50 C6.LARGE8 instances, with each having a 4-Core CPU (Intel Ice Lake 3.2/3.5 GHz), 8GB memory, and 50GB SSD, and running Ubuntu 20.04 LTS. The bandwidth of each instance is 100 Mbps. Only one XuperChain node is set up on each instance.

1. https://github.com/xuperchain/xuperchain
2. https://github.com/ethereum/solidity

We carry out the following three experiments:

- Throughput and latency of the modular blockchain.
- Latency of the adaptive consensus protocol.
- Latency and gas cost of NFR and OTMC operations.

In the first experiment, we build an XuperChain blockchain system considering three consensus algorithms (PoW, PoA, and TDPoS). We vary the number of instances added to the blockchain network (from 10 to 50), and send 5000 requests to test the performance of the blockchain system. We record the start and end timestamps of each request and check the blockchain to determine the number of on-chain messages during that period. Using this data, we calculate the TPS (transactions per second) and latency of each request. In the second experiment, we use the results of the first experiment to create a dataset; then we train and test five learning models, i.e., Decision Tree, Random Forest, AdaBoost, XGBoost, and GBDT (all implemented with sklearn and xgboost), and record their accuracy. We select the model with the highest accuracy to build an adaptive consensus protocol. Next we send 20 requests to test the latency of the consensus protocol with varied number of nodes. In the third experiment, we deploy the NFR and OTMC smart contracts on the second blockchain network. Then we designate the nodes on 5 randomly-selected instances as contract invokers to call 8 contract functions 200 times each. We record the start and end timestamps of each contract function call, as well as the gas consumption, and use the data to calculate the latency and gas of each function in the two contracts.

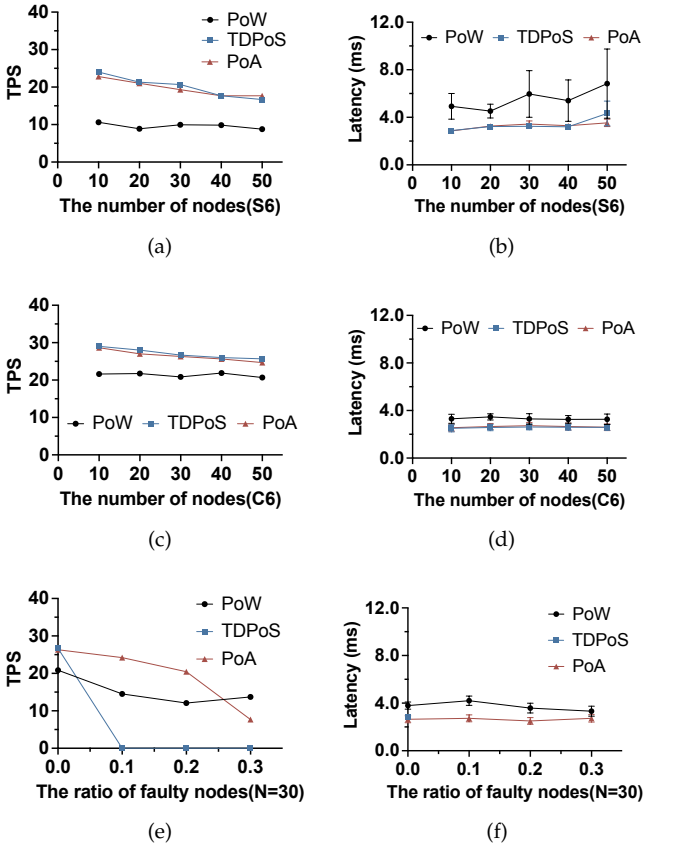


Fig. 8. TPS and latency of different consensus on cloud servers (S6.MEDIUM4) tests (a) (b), TPS and latency of different consensus on cloud server (C6.LARGE8) tests (c) (d), TPS and latency of different faulty nodes when the number of nodes is 30 on cloud servers (C6.LARGE8) tests (e) (f).

5.2 Performance Evaluation Results

5.2.1 *Throughput and Latency of the Modular Blockchain*

This experiment is to test the performance of the blockchain system with respect to different consensus algorithms and to provide training data for the adaptive consensus protocol. In this experiment, we compare three consensus algorithms, i.e., PoW, PoA, and TDPoS, varying the number of nodes, the ratio of faulty nodes, and the hardware resources. The unit for throughput is TPS, which refers to the number of transactions per second. Latency refers to the time delay between initiating a transaction and confirming it on the blockchain.

As shown in Fig. 8 (a), the TPS of PoA and TDPoS is better than that of PoW. Specifically, the TPS of the first two consensus algorithms decreases as the number of nodes increases, but that of PoW does not follow this trend. This is because TPDoS and PoA rely on the approvals of sufficient validators to add new blocks to the blockchain, while PoW requires miners to solve a complex mathematical problem. The communication complexity of PoW is relatively low compared to the other consensus algorithms. As the number of nodes increases, the communication time during the consensus process increases, resulting in a TPS decrease of PoA and TDPoS, but the TPS of PoW remains relatively stable. Fig. 8 (c) shows that the gap between PoW and the other two consensus algorithms is getting narrower. We find that the node deployed on S6 has less computing power. When the nodes use PoW, the CPU usage tends to reach 100%, which becomes the bottleneck of performance limitation. When we switch C6 with a more powerful CPU, the bottleneck disappears. Fig. 8 (b) demonstrates that the latency of PoA and TDPoS remains stable, but the latency of PoW fluctuates wildly. This is also because the weaker CPU limits the performance of PoW. As expected, Fig. 8 (d) demonstrates that the latency of PoW remains stable.

Then we compare the three consensus algorithms with respect to the faulty node ratio (FR), the ratio of the number of existing faulty nodes over the total number of nodes (N). The behaviors of the faulty nodes defined here mainly include stopping communications with other nodes, quitting or joining the network at any time, and proposing no blocks when working as a miner. We set $N = 30$, and $\mathsf{FR} \in \{0\%, 10\%, 20\%, 30\%\}$, considering $\mathsf{FR} < 33\% = \mathrm{f/N}$, where $\mathsf{FR} = 0$ represents the non-faulty case. As Fig. 8 (e) shows, the TPS of PoA and TDPoS decreases obviously, and the TPS of TDPoS even decreases to 0 in the presence of $10\%N$ faulty nodes. In Fig. 8 (f), as FR approaches to 10%, the TPS of TDPoS approaches to 0, and threfore the latency of TDPoS approaches to infinity. When FR is 30%, the TPS of PoW surpasses that of PoA. This is because PoA relies on authoritative nodes taking turns to become the leader and propose blocks. When a faulty node becomes the leader, it can cause the blockchain process to stall or slow down. Therefore, as the proportion of faulty nodes increases, the

performance of PoA significantly decreases, while that of PoW is not affected. It is clearly shown that the PoW scheme has stronger fault tolerance property.

TABLE 1
The accuracy of five models

Models	Decision Tree	Random Forest	Adaboost	XGBoost	GBDT
Accuracy	80.0%	68.3%	70.0%	80.0%	**96.7%**

5.2.2 The Performance of the Adaptive Protocol

In this experiment, we construct a data set according to the results of the previous experiments. The data set is characterized by the number of nodes and the ratio of the faulty ones. The labels of this dataset include PoA, PoW, and TDPoS. Then, we use this dataset for model training and testing. We use the hold-out method to randomly divide the data set into the training set and the test set according to the ratio of 7:3, and repeat it 10 times, using the average accuracy of all tests as the final evaluation result. We first test the accuracy of Decision Tree, Random Forest, AdaBoost, XGBoost, and GBDT, and the results are shown in Table 1. One can see that the GBDT model has the highest test accuracy; thus we adopt GBDT as the learning model in our adaptive consensus protocol. Note that the accuracy of these models is measured based on the selected features in the experiment. If more feature information is added to the dataset in future, such as network latency, the model we ultimately select may be different. Then we test the latency of the adaptive consensus protocol. The latency here is defined to be the time duration from when a user inputs parameters into the model, the model outputs results, the input and output are sent as a transaction to the blockchain, until finally the transaction is written into a block. The transaction specifies the type of consensus to be changed and the block height (H) at which the change should occur. Once such information is written into a block, all nodes update their local consensus mechanism at H and notify other nodes of the successful update. As shown in Fig. 9, when the current consensus is PoA, PoW, or TDPoS, the latency of the adaptive replacement to other consensus is similar. This is because, under our experimental conditions, the time to output the model results far exceeds the transaction latency. Therefore regardless of the current consensus, the entire consensus change latency is almost the same.

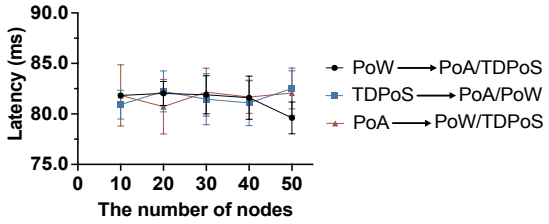


Fig. 9. The latency of updating consensus algorithms.

5.2.3 Latency and Gas Cost of NFR and OTMC Operations

In this experiment, we test the latency of each function with respect to a variable number of nodes, and the gas consumption of each function in NFR and OTMC. As shown in Fig. 10(a) and (b), one can see that the latency of each function is at the millisecond level, and it increases slightly with the increasing number of nodes. The latency of Tx_{Open} is the longest, roughly from 3.3ms to 4.5ms, which is reasonable because it involves more uploaded data, e.g., multiple signatures and addresses. We test the Ethereum gas cost and XuperChain fee for each function. As our experiment is based on XuperChain, we also deploy these contracts on Ethereum to facilitate comparisons and provide readers with a better understanding of the specific cost of each contract function. However, as Ethereum and XuperChain utilize different fee calculation methods, the results are not proportional. From Fig. 11(a) and (b), one can see that the gas cost of each function fluctuates between 50,000 and 250,000, and Tx_{Open} consumes the most gas (about 250,000). The reason lies in that this operation requires more input data and a more complex calculation process.

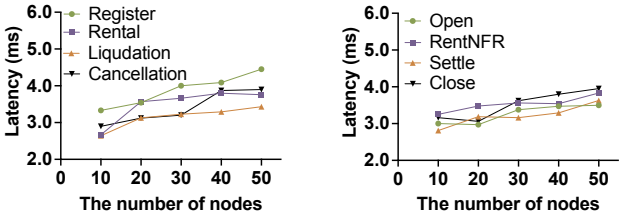


Fig. 10. The latency of NFR and OTMC.

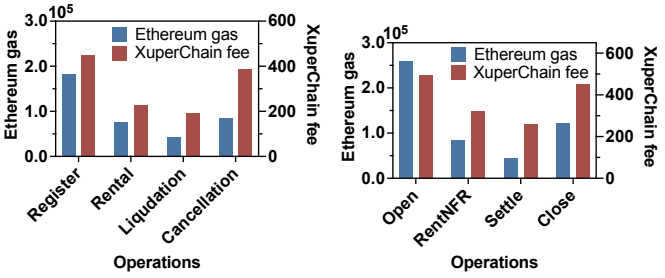


Fig. 11. The gas cost of NFR and OTMC.

6 Conclusion

In this paper, we present an adaptive and modular blockchain-enabled architecture for a decentralized metaverse. Our architecture tackles two key challenges: the Consensus/Ledger Adaptation (CLA) problem and the Resource Centralization (RC) problem. To address the CLA problem, we propose an adaptive consensus/ledger protocol that maintains consensus across a distributed network while adapting to changing conditions. To solve the RC

problem, we introduce the concept of Non-Fungible Resource to enable resource virtualization and consolidation within the metaverse without risking centralization. Additionally, we design the On-Demand Trusted Metaverse Cluster to support parallel computing and promote fair trading of the Non-Fungible Resources. Finally, we test the performance of our design using three consensus algorithms, and the experimental results demonstrate the feasibility of our approach.

7 Acknowledgement

This study was partially supported by the National Key R&D Program of China (No.2022YFB4501000), the National Natural Science Foundation of China (No.62232010), Shandong Science Fund for Excellent Young Scholars (No.2023HWYQ-008), and the Shandong Science Fund for Key Fundamental Research Project (ZR2022ZD02).

References

[1] H. Ning, H. Wang, Y. Lin, W. Wang, S. Dhelim, F. Farha, J. Ding, and M. Daneshmand, "A survey on metaverse: the state-of-the-art, technologies, applications, and challenges," *arXiv preprint arXiv:2111.09673*, 2021.

[2] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Communications Surveys & Tutorials*, 2022.

[3] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui, "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda," *arXiv preprint arXiv:2110.05352*, 2021.

[4] J. D. N. Dionisio, W. G. B. III, and R. Gilbert, "3d virtual worlds and the metaverse: Current status and future possibilities," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, pp. 1–38, 2013.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[6] Y. Fu, C. Li, F. R. Yu, T. H. Luan, P. Zhao, and S. Liu, "A survey of blockchain and intelligent networking for the metaverse," *IEEE Internet of Things Journal*, 2022.

[7] M. Xu, Y. Guo, Q. Hu, Z. Xiong, D. Yu, and X. Cheng, "A trustless architecture of blockchain-enabled metaverse," *High-Confidence Computing*, p. 100088, 2022.

[8] X. Lab. What is xuperchain. [Online]. Available: https://github.com/xuperchain/xuperchain

[9] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[10] R. Corporation. (2006) Roblox. [Online]. Available: https://developer.roblox.com/en-us/

[11] N. Team. (2021) Nvidia omniverse. [Online]. Available: https://www.nvidia.com/en-us/omniverse/

[12] M. Team. (2022) Microsoft mesh. [Online]. Available: https://www.microsoft.com/en-us/mesh

[13] A. Heath. (2004) Inside facebook's metaverse for work. [Online]. Available: https://www.theverge.com/2021/8/19/22629942/facebook-workroomshorizon-oculus-vr

[14] L. Labs. (2003) Second life. [Online]. Available: https://secondlife.com/

[15] M. Team. (2009) Minecraft maps. [Online]. Available: https://www.minecraftmaps.com/tags/real-cities-in-minecraft

[16] J. Radoff. (2021) The metaverse value-chain. [Online]. Available: https://medium.com/building-the-metaverse/the-metaverse-value-chain-afcf9e09e3a7

[17] V. Nair, G. M. Garrido, and D. Song, "Going incognito in the metaverse," *arXiv preprint arXiv:2208.05604*, 2022.

[18] H. Duan, J. Li, S. Fan, Z. Lin, X. Wu, and W. Cai, "Metaverse for social good: A university campus prototype," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 153–161.

[19] B. Kye, N. Han, E. Kim, Y. Park, and S. Jo, "Educational applications of metaverse: possibilities and limitations," *Journal of Educational Evaluation for Health Professions*, vol. 18, 2021.

[20] Decentraland. (2017) Welcome to decentraland. [Online]. Available: https://decentraland.org/

[21] Cryptovoxels. (2021) Welcome to voxels - a user-owned virtual world. [Online]. Available: https://www.cryptovoxels.com/

[22] Q. Yang, Y. Zhao, H. Huang, Z. Xiong, J. Kang, and Z. Zheng, "Fusing blockchain and ai with metaverse: A survey," *IEEE Open Journal of the Computer Society*, vol. 3, pp. 122–136, 2022.

[23] T. Huynh-The, T. R. Gadekallu, W. Wang, G. Yenduri, P. Ranaweera, Q.-V. Pham, D. B. da Costa, and M. Liyanage, "Blockchain for the metaverse: A review," *Future Generation Computer Systems*, 2023.

[24] H. Huang, X. Zeng, L. Zhao, C. Qiu, H. Wu, and L. Fan, "Fusion of building information modeling and blockchain for metaverse: a survey," *IEEE Open Journal of the Computer Society*, vol. 3, pp. 195–207, 2022.

[25] M. Ersoy and R. Gürfidan, "Blockchain-based asset storage and service mechanism to metaverse universe: Metarepo," *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 1, p. e4658, 2023.

[26] S. Ali, T. P. T. Armand, A. Athar, A. Hussain, M. Ali, M. Yaseen, M.-I. Joo, H.-C. Kim *et al.*, "Metaverse in healthcare integrated with explainable ai and blockchain: Enabling immersiveness, ensuring trust, and providing patient data security," *Sensors*, vol. 23, no. 2, p. 565, 2023.

[27] M. Xu, Y. Guo, C. Liu, Q. Hu, D. Yu, Z. Xiong, D. Niyato, and X. Cheng, "Exploring blockchain technology through a modular lens: A survey," *arXiv preprint arXiv:2304.08283*, 2023.

[28] C. Lab. The first modular blockchain network. [Online]. Available: https://celestia.org/

[29] O. Ali, M. Momin, A. Shrestha, R. Das, F. Alhajj, and Y. K. Dwivedi, "A review of the key challenges of non-fungible tokens," *Technological Forecasting and Social Change*, vol. 187, p. 122248, 2023.

[30] K. Ito, K. Shibano, and G. Mogi, "Predicting the bubble of non-fungible tokens (nfts): An empirical investigation," *arXiv preprint arXiv:2203.12587*, 2022.

[31] S. B. Far, S. M. H. Bamakan, Q. Qu, and Q. Jiang, "A review of non-fungible tokens applications in the real-world and metaverse," *Procedia Computer Science*, vol. 214, pp. 755–762, 2022.

[32] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[33] S. Dziembowski, S. Faust, and K. Hostáková, "General state channel networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 949–966.

[34] Y. Guo, M. Xu, D. Yu, Y. Yu, R. Ranjan, and X. Cheng, "Cross-channel: Scalable off-chain channels supporting fair and atomic cross-chain operations," *arXiv preprint arXiv:2212.07265*, 2022.

[35] W. Yu, M. Xu, D. Yu, X. Cheng, Q. Hu, and Z. Xiong, "zk-pcn: A privacy-preserving payment channel network using zk-snarks," in *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2022, pp. 57–64.

[36] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[37] H. Pervez, M. Muneeb, M. U. Irfan, and I. U. Haq, "A comparative analysis of dag-based blockchain architectures," in *2018 12th International Conference on Open Source Systems and Technologies (ICOSST)*, 2018, pp. 27–34.

[38] W. F. Silvano and R. Marcelino, "Iota tangle: A cryptocurrency to communicate internet-of-things data," *Future Generation Computer Systems*, vol. 112, pp. 307–319, 2020.

[39] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 {USENIX} Annual Technical Conference ({USENIX} {ATC} 20)*, 2020, pp. 515–528.

[40] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias, "Bullshark: Dag bft protocols made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2705–2718.

[41] M. Nadini, L. Alessandretti, F. Di Giacinto, M. Martino, L. M. Aiello, and A. Baronchelli, "Mapping the nft revolution: market trends, trade networks, and visual features," *Scientific reports*, vol. 11, no. 1, pp. 1–11, 2021.

[42] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (nft): Overview, evaluation, opportunities and challenges," *arXiv preprint arXiv:2105.07447*, 2021.

[43] C. Liu, M. Xu, H. Guo, X. Cheng, Y. Xiao, D. Yu, B. Gong, A. Yerukhimovich, S. Wang, and W. Lv, "Tokoin: a coin-based accountable access control scheme for internet of things," *arXiv preprint arXiv:2011.04919*, 2020.
[44] W. Jiang, G. Wang, M. Z. A. Bhuiyan, and J. Wu, "Understanding graph-based trust evaluation in online social networks: Methodologies and challenges," *Acm Computing Surveys (Csur)*, vol. 49, no. 1, pp. 1–35, 2016.
[45] X. Wu, R. Zhang, B. Zeng, and S. Zhou, "A trust evaluation model for cloud computing," *Procedia Computer Science*, vol. 17, pp. 1170–1177, 2013.
[46] A. Bretto, "Hypergraph theory," *An introduction. Mathematical Engineering. Cham: Springer*, 2013.
[47] Chainlink. (2021) What is a blockchain oracle. [Online]. Available: https://chain.link/education/blockchain-oracles
[48] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*, 2003, pp. 640–651.
[49] Z. Liang and W. Shi, "Pet: A personalized trust model with reputation and risk evaluation for p2p resource sharing," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. IEEE, 2005, pp. 201b–201b.
[50] A. Josang and J. Haller, "Dirichlet reputation systems," in *The Second International Conference on Availability, Reliability and Security (ARES'07)*. IEEE, 2007, pp. 112–119.
[51] G. Theodorakopoulos and J. S. Baras, "On trust models and trust evaluation metrics for ad hoc networks," *IEEE Journal on selected areas in Communications*, vol. 24, no. 2, pp. 318–328, 2006.