УДК 004.4`2 http://doi.org/10.15407/pp2024.01.086

B.I.Dobrianskyi, I.V. Stetsenko

ARCHITECTURAL FRAMEWORK FOR A UNIFIED BLOCKCHAIN INTERACTION LIBRARY

This research addresses the challenges posed by the fragmented nature of blockchain development tools, presenting a comprehensive exploration of the imperative need for a unified architecture. In response to the growing diversity of blockchain networks, a solution in the form of a unified library based on a versatile and interoperable interface that streamlines interactions across various blockchains is proposed. The current state of blockchain development tools, an overview of existing solutions, formulation of design principles, and functions of proposed unified library are provided. By mitigating the complexities associated with disparate tools, the research aims to enhance the accessibility and efficiency of blockchain development, encouraging collaboration and innovation within the blockchain community.

Keywords: Blockchain development, Interoperability, Decentralized applications, Development tools, Blockchain networks, Software development kits (SDKs), Software Architecture

Introduction

The recent explosion of blockchain technology has created a captivating landscape teeming with innovation and brimming with potential. As mentioned in the survey [1], the use of blockchain has already been extended far beyond “electronic cash”, as initially described by Nakamoto S. [2]. The rapid proliferation has birthed a multitude of diverse blockchains, each boasting unique features, consensus mechanisms, and smart contract functionalities. While the abundance presents developers with exciting opportunities to explore and create, it also presents a significant challenge.

The tools currently available to developers for building upon and interacting with these diverse platforms remain largely limited to specific blockchains. This creates a fragmented development environment, forcing developers to juggle a plethora of libraries, frameworks, and SDKs just to navigate the landscape. This fragmented ecosystem poses a significant barrier to entry for newcomers and seasoned developers alike, hindering efficiency and collaboration within the blockchain development community.

To address the pressing need for standardization and interoperability, this research proposes the development of a unified library. The comprehensive library aims to act as a cohesive interface, empowering developers to seamlessly interact with various blockchains. By providing a standardized, adaptable, and interoperable solution, this research seeks to empower developers by streamlining their workflow, fostering collaboration within the community, and ultimately propelling innovation within the realm of blockchain development.

The following sections describe the challenges presented by the fragmented state of blockchain development tools. After that, existing solutions are examined before articulating the foundational principles and features of the proposed unified library. The aim of the above-mentioned exploration is to provide a comprehensive understanding of the motivation behind and the potential impact of our research on the broader blockchain development community. Looking beyond the proposed solution, the research acknowledges the valuable contributions of existing solutions that play a vital role in the ongoing process of improving interoperability and enhancing developer experience within the blockchain landscape.

1. Essential Components of Unified Blockchain Interface

The unified blockchain library development requires a thorough understanding of its core components, independent of the underlying blockchain technology. From managing accounts to facilitating transactions, these

components are the building blocks of the innovative framework.

To ensure a well-designed architecture, the requirements that govern its construction should be carefully analyzed. The analysis delves into the complexities involved creating a unified interface that seamlessly interacts with diverse blockchains, beginning with the fundamental domain of account management, the cornerstone of any blockchain system.

1.1. Account management

Regarding blockchain networks, one common requirement is the solid management of accounts, which emerges from the role of accounts in the process of assuring secure and authenticated connections. Blockchain security heavily relies on private keys. Each user has a unique digital identity formed by a key pair: a public key (like a public address) and a private key (kept secret). Creating an account with a private key is the entry point for user activity, smart contracts, and transactions on the network. These cryptographic keys, which play the role of digital signatures [3], form the basis of the security architecture of the blockchain networks. The algorithms used may vary, for example, widely adopted cryptocurrencies like Bitcoin leverage the Elliptic Curve Digital Signature Algorithm (ECDSA) [4] for key generation and verification, ensuring a robust foundation for secure transactions. In contrast, Ethereum predominantly utilizes ECDSA but also supports the RSA algorithm in certain contexts [5], providing flexibility in cryptographic operations across its network. Additionally, emerging technologies explore more exotic algorithms, such as the Lamport Signature Scheme [6], showcasing the continuous evolution of cryptographic approaches within diverse blockchain ecosystems.

Creating a security key interweaves the formation of a key pair comprised of a public key visible to everyone and a private key known only to the account owner. Thus, creating an account serves as a starting point for involvement in blockchain activity including user activity and smart contract interactions.

Amongst the diverse functionalities encompassed by blockchain account management, two operations emerge as the most generic and essential: account creation and account retrieval. These operations prove critical in laying the groundwork for secure and decentralized participation within blockchain ecosystem, so they are included in the account management.

Having established the significance of account management, this analysis shifts its focus toward the core entity that underpins blockchain interaction: the account. While the specifics of account implementation vary across different blockchains, the library should bridge this gap by presenting a unified perspective. As per the accounts themselves, their significance extends beyond mere placeholders for cryptographic keys; they serve as dynamic entities within the blockchain ecosystem. Within the unified library, the account encapsulates a suite of functionalities tailored to these entities, fostering an environment that not only ensures secure interactions but also facilitates comprehensive account management.

The most basic functional requirement of the account is the ability to retrieve its balance. This fundamental feature provides users with a real-time snapshot of the account's holdings, detailing the quantity of digital assets or tokens it possesses.

Transaction signing equips developers with the capability to sign transactions securely using the account's private key. This cryptographic signature not only ensures the transaction authenticity, but also safeguards against unauthorized alterations, thereby fortifying the integrity of the blockchain network.

Beyond immediate financial insights, the account encompasses the functionality to query transaction history, which provides a historical record of transactions associated with the account. The comprehensive log offers visibility into past interactions, empowering developers to track and analyze the account activity over time.

To augment the versatility of account interactions, one of the account features is the ability to query metadata associated with it. It includes information such as creation timestamps, account types, custom tags, or

any other pertinent details configured. Metadata querying provides a flexible interface for developers to retrieve contextual information, enhancing the richness of the account management experience.

Moreover, recognizing the dynamic nature of blockchain ecosystems and the diversity among different networks, the account is designed to be extensible. It ensures that the account structure can accommodate additional functionalities specific to particular blockchains. Blockchain networks often introduce unique features or token standards that may require additional methods or attributes within the account representation.

Consider the diversity across blockchain networks such as Ethereum, Solana, and Polkadot, each introducing unique functionalities and adhering to specific standards. The account structure within the unified library is designed to be extensible, allowing developers to seamlessly integrate blockchain-specific features. For instance, in Ethereum, renowned for its extensive smart contract capabilities, the Account might extend functionalities for interacting with sophisticated contract deployment and execution processes. Solana, with its high-performance blockchain, might introduce account-related features pertinent to its unique consensus mechanism and token standards. Similarly, Polkadot's multi-chain architecture might necessitate account-related features aligned with its parachain model.

In essence, the Account class emerges as a multifaceted entity within our unified library, aligning with the diverse needs of developers engaged in blockchain interactions. By encompassing balance retrieval, transaction signing, transaction history querying, and metadata retrieval, the Account class serves as a comprehensive toolkit, fostering an environment where developers can seamlessly manage and leverage the full potential of blockchain accounts within their decentralized applications.

1.2. Block management

At the core of every blockchain lies the foundational building block, quite literally known as a "block." A block represents a unit of data that contains a bundle of transactions, a timestamp, and a reference to the preceding block. This sequential chain of blocks forms the bedrock of blockchain technology, enabling the secure and transparent recording of transactions across a decentralized network.

The significance of a block extends beyond its definition; it serves as a crucial mechanism for achieving consensus and immutability within the blockchain. Each block is intricately linked to its predecessor through cryptographic hashes, creating an indelible chain that resists tampering and ensures the integrity of the entire transaction history.

What sets the concept of a block apart is its universality across diverse blockchain networks. While blockchains may differ in their consensus mechanisms, transaction processing speeds, and smart contract capabilities, the fundamental structure of a block remains remarkably consistent [7]. This uniformity provides an opportune avenue for unification, as the shared characteristics of blocks can be harnessed to create a standardized approach to managing and retrieving blockchain data.

This section delves into the universal nature of blocks, investigating their constituent components and exploring the potential for a unified strategy in managing these foundational elements across diverse blockchain networks. By analyzing the shared characteristics and functionalities of blocks, the groundwork can be laid for the development of a standardized approach to their management, potentially fostering greater interoperability and efficiency within the blockchain ecosystem.

First and foremost, a crucial capability in the toolkit is the ability to fetch the latest block from the blockchain. This feature holds paramount importance because it allows users to stay synchronized with the most recent data on the blockchain by ensuring that the applications are always up-to-date and aligned with the dynamic nature of the blockchain.

Equally vital is the capability to retrieve a specific block using its unique numerical identifier, often referred to as the block number. The significance of this features lies in the ability to grants users precision in data retrieval. Consider a scenario where user wants some specific details from a particular

point in the blockchain's history. By being able to get a block by its number, user can precisely pinpoint and extract the required information. This functionality caters to applications that demand targeted insights into specific blocks, contributing to a more refined and efficient data retrieval process.

Regarding the block itself, it serves as a central entity within our unified framework, facilitating a nuanced exploration of blockchain data.

The following sections investigate the inherent features of blockchain blocks, equipping developers with the capability to effortlessly retrieve crucial metadata associated with each block. This metadata encompasses fundamental details, including the block number, a unique identifier within the blockchain sequence, and the mining time, which provides insights into the block's creation timestamp. These metadata elements collectively offer a comprehensive snapshot, elucidating the block position and temporal relevance within the broader blockchain landscape.

Delving further into the capabilities of the block entity, it extends its functionality to enable the extraction of transactions stored within the block. This feature proves pivotal for developers seeking a detailed understanding of the activities recorded in a specific block. By accessing the list of transactions, developers can explore and leverage the intricacies of each transaction within the block, facilitating a more granular analysis of blockchain data.

In essence, the block entity serves as a versatile tool, offering not only a unique identifier and temporal context through metadata retrieval but also providing access to the transactional intricacies encapsulated within individual blocks. These capabilities enrich the developer's toolkit, fostering a more nuanced and detailed exploration of blockchain data at the block level.

1.3. Transaction management

Navigating the intricacies of blockchain transactions, transaction management takes center stage as a crucial orchestrator. Its role is to simplify the complexities associated with transactions, offering developers a set of powerful functionalities for efficient interaction with the blockchain network.

In terms of transaction retrieval, the transaction manager empowers developers to pinpoint and extract specific transactions. Utilizing identifiers such as transaction hashes, transaction indices, or block references, developers gain flexibility in accessing transactions tailored to their needs. This capability is particularly exemplified in networks like Ethereum, where transaction hashes serve as unique identifiers.

Moving beyond retrieval, the Transaction Manager offers a real-time status check for individual transactions. This feature proves invaluable for developers seeking immediate updates on a transaction's processing status - whether it is pending, successfully executed, or encountering issues. This real-time insight into the transaction lifecycle enhances developers' ability to respond effectively to changing conditions on the blockchain.

Additionally, the transaction manager facilitates historical exploration by enabling developers to query the blockchain for a detailed transaction history associated with a specific account. This functionality is instrumental for applications requiring a comprehensive overview of an account's past transactions. For instance, in decentralized finance applications, users can leverage this capability to review their transaction history seamlessly.

In summary, the Transaction Manager emerges as a versatile guide through the blockchain transactions. Its functionalities, ranging from precise retrieval to real-time status checks and historical exploration, equip developers with the tools needed to navigate the dynamic landscape of decentralized data transactions efficiently.

1.4. Smart contract management

In the realm of blockchain development, smart contracts shine as unique entities injecting programmable capabilities into the decentralized landscape. Unlike blocks, which vary but retain some uniformity, smart contracts stand out distinctly. It's worth noting that not all blockchains, such as Bitcoin and Ripple, natively support smart contracts.

Despite this, smart contracts play a crucial role in blockchain development. They act as architects for self-executing agreements, allowing developers to encode rules directly into the blockchain. This capability, though not universally supported, remains a fundamental tool for developers crafting decentralized applications, which is why smart contract management is included in the library.

The smart contract manager, a pivotal component within the proposed architectural framework, plays a crucial role in facilitating seamless smart contract interactions across diverse blockchain networks. This manager offers two fundamental operations, catering to various blockchain environments.

First of all, it provides a unified interface for retrieving existing smart contracts from the underlying blockchain. This functionality enables developers to interact with deployed smart contracts, including tasks such as reading data, verifying states, and executing predefined operations. By abstracting the complexities of smart contract retrieval, the manager enhances developer efficiency and ensures a consistent experience across different blockchain networks.

In addition to retrieving existing smart contracts, the manager empowers developers to deploy new contracts onto the blockchain. This essential feature facilitates the initiation of decentralized applications and the implementation of custom smart contract logic. The manager abstracts the intricacies of the deployment process, offering developers a streamlined mechanism to bring their smart contract creations to life on the blockchain, regardless of the underlying blockchain's specifications.

Building upon the foundation laid by the smart contract manager, the smart contract entity inherits the ability to read data directly from the smart contract it represents. This continuity ensures that developers maintain real-time access to crucial information stored within the decentralized ledger. The entity becomes a conduit for understanding the evolving state and conditions of the contract, fostering an environment of informed decision-making.

As an evolution of transactional capabilities, the smart contract entity empowers developers to submit smart contract transactions to the blockchain. This functionality signifies an active engagement in the execution of predefined functions within the smart contract. Developers navigate the decentralized landscape with a refined ability to participate in the decision-making process encoded within the blockchain.

Extending the narrative of event-driven architecture, the smart contract entity allows developers to seamlessly subscribe to events emitted by the represented smart contract. This continuity in observability enhances transparency, ensuring external entities, for example other smart contracts or off-chain systems, can monitor and react to changes within the contract. The smart contract entity becomes a vital component in fostering a collaborative and responsive blockchain ecosystem.

1.5. Plugin system

A key component of the architecture is a robust plugin system. Recognizing the dynamic and evolving nature of blockchain ecosystems, the plugin system allows for adaptability and extensibility. It enables:

- Streamlined integration of new features: plugins can be added without modifying the core library, keeping development efficient.
- Empowerment for the developer community: developers can contribute and enhance the library's capabilities, fostering collaboration and innovation.

However, achieving true unification requires acknowledging the inherent diversity of blockchain networks. The proposed solution addresses this by emphasizing the use of blockchain-specific plugins. These plugins cater to the unique characteristics and functionalities of each blockchain, ensuring:

- Preservation of individual blockchain design: developers can leverage the full potential of each network without compromising its integrity.
- Optimal performance: plugins allow fine-tuning the library's functionality for specific blockchains, maximizing efficiency.

Balancing standardization and customization are crucial within this framework. Blockchain-specific plugins acknowledge the value of individual network characteristics while maintaining a unified approach through the core library interface. This embrace of diversity reflects the richness and innovation present in the blockchain landscape.

Moving forward, the proposed architecture provides a flexible and adaptable foundation for blockchain development. It empowers developers with a unified interface while acknowledging the unique needs of individual blockchain networks, paving the way for a more collaborative and future-proof development landscape.

2. Results

Continuing upon the established conceptual foundations, this section unveils the architectural design of the proposed unified library. Figure 1 presents a structure diagram of this architecture.

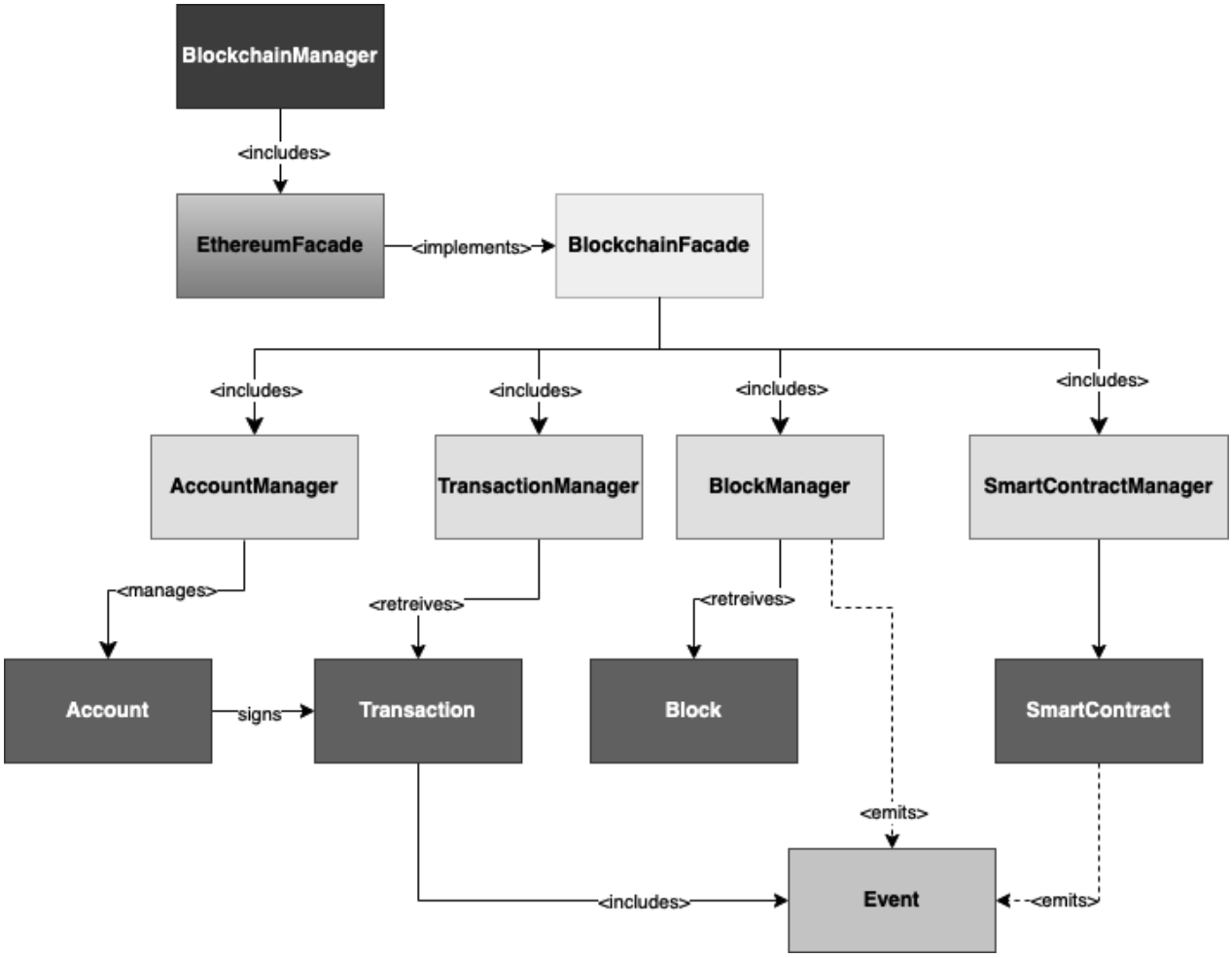

Fig.1. Structure Diagram of the library

The architecture encapsulates key components, allowing for seamless interactions with various blockchain networks. A brief overview of the major components is provided below.

The blockchain manager serves as an entry point to the library. It can include one or more blockchain facades, that are implemented as separate libraries (plugins) for each blockchain.

Transaction Manager stands as a central hub for retrieving, monitoring, and exploring transactions within the blockchain.

Transaction represents individual transactions, this entity encapsulates details, signing mechanisms, verification, and any additional metadata associated with a transaction.

Block Manager serves as a linchpin for managing blocks, the Block Manager of-

fers functionalities such as retrieving specific blocks by number, obtaining the latest block, and navigating through blocks with pagination and filtering. It also emits related events, for example mining of a new block.

The Block entity encapsulates metadata retrieval, facilitating the extraction of transactions, and ensuring extensibility for blockchain-specific features.

Account Manager is responsible for simplifying account-related interactions, the Account manager provides methods for creating and retrieving accounts, signing transactions, and managing account-specific functionalities.

The Account entity represents individual accounts within the blockchain, offering functionalities like balance retrieval, transaction signing, checking transaction history, and querying metadata.

Smart contract manager is responsible for deployment and retrieval of smart contracts. The Smart Contract entity itself is responsible for reading data from the smart contract, calling methods on smart contracts and listening to the events emitted by the smart contract.

This architecture is designed with versatility and extensibility in mind. Its strength lies in providing developers with a unified interface for blockchain interactions, reducing the complexities associated with navigating diverse blockchain networks. The modular structure allows for easy integration and adaptation to specific blockchain characteristics, ensuring a robust foundation for blockchain development. With a foundational understanding of the architecture now established, the following analysis covers the advantages and potential applications that make this design a compelling choice for blockchain development.

2.1. Implementation

The unified blockchain interaction library has been successfully implemented using JavaScript and TypeScript to support the diverse needs of developers engaged in blockchain interactions. The decision to employ these languages is rooted in their widespread adoption within the blockchain development community, serving as the go-to choices for both backend and frontend development. This approach ensures a seamless integration of the library into various projects, maintaining compatibility across different layers of the software stack.

The Ethereum and Solana plugins were also crafted in TypeScript using web3.js and @solana/web3.js libraries correspondingly, ensuring consistency and ease of use across the entire ecosystem. The language choice aligns with the prevalent practices in blockchain development, where JavaScript and TypeScript stand out as the primary languages for facilitating comprehensive blockchain interactions across different platforms.

2.2. Pros and potential use cases

Versatility across blockchains. One of the notable strengths of this architecture lies in its adaptability to a variety of blockchain networks. The modular design allows developers to seamlessly integrate the library into diverse blockchain ecosystems, mitigating the challenges posed by the heterogeneity of blockchain technologies.

Reduced development complexity. By providing a unified interface for blockchain interactions, our architecture significantly reduces the development complexity associated with navigating different blockchain networks. Developers can leverage consistent methods and entities across various blockchains, streamlining the development process and fostering code reusability.

Facilitated cross-blockchain development. The modular structure and unified interface empower developers to engage in cross-blockchain development more efficiently. Whether working with Ethereum, Solana, Polkadot, or other blockchains, the proposed design simplifies the development of applications that seamlessly interact with multiple blockchain networks.

Extensibility for blockchain-specific features. The architecture's extensibility ensures that it can accommodate blockchain-specific features seamlessly. For instance, if a blockchain introduces new metadata or unique transaction types, the library's extensible entities, for example the Transaction and

Account, can readily adapt without requiring a major overhaul of the existing codebase.

2.3. Challenges and considerations

While our architecture presents numerous advantages, it is essential to acknowledge potential challenges and considerations.

Blockchain-specific limitations. Certain blockchain networks may have unique limitations or characteristics that might not align perfectly with the unified interface. Developers should be aware of such nuances and handle them appropriately when working with specific blockchains.

Performance considerations. The architecture's adaptability across various blockchains may introduce slight performance trade-offs. Developers should carefully assess performance requirements and optimize implementations accordingly. However, this also heavily depends on the implementation itself.

3. Related works

From specialized libraries tailored for specific blockchain networks to comprehensive frameworks designed for cross-chain compatibility, the variety in the approaches mirrors the continuous evolution of blockchain technology. The research through these related works unveils the multifaceted efforts that shape the current and future landscapes of blockchain development.

In the following exploration of related works, 3 distinctive solutions that have made significant strides in simplifying and unifying blockchain interactions are spotlighted. These solutions, each with their unique approach, contribute to the broader narrative of enhancing developer experiences within the blockchain landscape.

3.1. Web3.js

Web3.js stands as a JavaScript library specifically tailored for EVM blockchains, providing developers with a streamlined interface for interacting with the Ethereum blockchain. It abstracts away the complexities of Ethereum's native RPC (Remote Procedure Call) and JSON-RPC protocols, offering a simplified programming model.

Designed with a focus on Ethereum, Web3.js provides comprehensive functionalities for smart contract interactions, transaction management, and blockchain data retrieval specific to the Ethereum network.

Being a widely adopted library, Web3.js benefits from an active and vibrant community, ensuring continuous updates, bug fixes, and the availability of extensive documentation.

It is important to note, however, that Web3.js is primarily applicable for EVM-compatible networks, and its features are optimized for the Ethereum ecosystem.

3.2. Cosmos SDK

The Cosmos SDK takes a broader approach by offering a framework for building multi-blockchain applications. It allows developers to create their customized blockchains (Zones) with specific functionalities, while the Cosmos Hub acts as a secure and interoperable hub connecting different Zones [9].

Cosmos SDK focuses on enabling interoperability between blockchains. The Hub-and-Zone architecture allows for secure communication between different blockchains, fostering a vision of an interconnected blockchain ecosystem.

Developers using Cosmos SDK have the flexibility to design and launch their blockchains with tailored features, ensuring versatility in addressing specific use cases.

3.3. Polkadot and Parachains

Polkadot introduces a novel concept of parachains, offering a framework for building customized blockchains that connect to the Polkadot relay chain. Parachains run in parallel, fostering scalability and interoperability across the Polkadot network [10].

The use of parachains in Polkadot enables horizontal scalability, allowing multiple blockchains to run concurrently. This approach addresses scalability concerns and promotes efficient resource utilization.

By connecting to the Polkadot relay chain, parachains benefit from enhanced security and interoperability, creating a cohesive ecosystem where different blockchains can seamlessly interact.

3.4. Challenges

This survey reveals a common thread – interoperability is often confined to specific blockchain ecosystems, lacking a universal solution that spans the entirety of the blockchain landscape. While these existing libraries and frameworks excel within their designated environments, the quest for a unified architecture for blockchain interaction persists.

A constantly developing landscape of blockchain innovation necessitates a comprehensive solution. Analyzing the process of designing a unified library for blockchain interaction revealed both challenges and opportunities. By envisioning an architecture that transcends individual blockchain ecosystems, the goal is to foster an interoperable development landscape, empowering developers across diverse networks. With wider adoption and expansion of blockchain usage, the need for a universal framework becomes increasingly evident, paving the way for a more interconnected and collaborative future in decentralized development.

Conclusions

This research addresses the challenge of fragmented blockchain development tools by proposing a novel unified library architecture. This architecture aims to bridge the gap by providing developers with a standardized and interoperable interface, facilitating seamless interactions across diverse blockchain networks.

The proposed solution presents a step forward in blockchain development, offering the following advantages:

− Enhanced developer efficiency. By providing a unified interface, the architecture significantly reduces the complexity caused by navigating and interacting with various blockchain ecosystems.

− Promoted cross-chain collaboration. Whether building applications that interact with Ethereum, Solana, Polkadot, or other emerging networks, the architecture simplifies communication and data exchange, fostering collaboration and innovation across the broader blockchain landscape.

− Future-proof design. Blockchain-specific features can be seamlessly integrated through adaptable entities within the library, ensuring it remains relevant and adaptable as new protocols and functionalities emerge.

Potential challenges requiring further investigation include:

− Addressing blockchain-specific nuances. While the unified interface offers significant advantages, particular blockchain networks may possess unique limitations or characteristics that it does not cover. Developers should know these nuances and implement appropriate workarounds interacting with specific blockchains.

− Balancing performance and versatility. The architecture adaptability across various blockchains might necessitate slight performance trade-offs.

Based on the proposed architectural framework, the future improvements are the following:

– conducting rigorous testing and performance evaluations across diverse blockchain networks to identify and address potential limitations;

– expanding the library's capabilities by developing and integrating plugins for a wider range of blockchain networks;

– investigating and integrating emerging interoperability protocols and solutions to further enhance seamless communication across blockchain ecosystems;

– conducting in-depth research on optimizing the library's internal data structures and algorithms to ensure efficient and scalable interactions across diverse blockchain networks.

By taking these steps, the proposed unified library architecture can evolve into a robust and versatile solution, empowering developers to navigate the ever-evolving landscape of blockchain development with greater efficiency and fostering a more collaborative and innovative future for decentralized technologies.

References

1. Al-Jaroodi J., Mohamed N. (2019), 'Blockchain in Industries: A Survey', *IEEE Access*, vol. 7, pp. 36500-36515.
2. Nakamoto S. 'Bitcoin: A Peer-to-Peer Electronic Cash System', available at https://bitcoin.org/bitcoin.pdf

3. Zhang, R., Xue, R., Liu, L. (2019), 'Security and Privacy on Blockchain', *ACM Computing Surveys*, 52, 3, Article 51.
4. Johnson, D., Menezes, A., Vanstone, S. (2001), 'The Elliptic Curve Digital Signature Algorithm (ECDSA) ', *International Journal of Information Security*, 1, 36–63, https://doi.org/10.1007/s102070100002
5. Shamir A., Rivest, R., Adleman L. (1978), 'A method for obtaining digital signatures and public-key cryptosystems', *Communications of the ACM*, 21 (2), 120–126.
6. Pandey, S., Behl, R. & Sinha, A. (2023), 'Decentralized blockchain-based security enhancement with lamport merkle digital signature generation and optimized encryption in cloud environment', *Multimedia Tools and Applications,* doi: 10.1007/s11042-023-17365-8.
7. Clincy, V., Shahriar, H. (2019), 'Blockchain Development Platform Comparison', *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, doi: 10.1109/COMPSAC.2019.00142.
8. Web3.js framework. Available at https://github.com/web3/web3.js (Accessed 28 February 2024).
9. Kwon, J., Buchman, E. (2016). Cosmos: a network of distributed ledgers. Available at https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md (Accessed 28 February 2024).
10. Wood, G. (2016). Polkadot: Vision for a heterogenous multi-chain framework. Available at https://assets.polkadot.network/Polkadot-whitepaper.pdf (Accessed 28 February 2024).

Received: 01.03.2024

***Про авторів*:**

Добрянський Богдан Ігорович,
магістрант другого року навчання
Національного Технічного Університету
України «КПІ імені Ігоря Сікорського».
Кількість наукових публікацій
в українських виданнях – 1.
Кількість наукових публікацій
в зарубіжних виданнях – 0.
Індекс Гірша – 0.
https://orcid.org/0009-0003-4797-1947

Стеценко Інна Вячеславівна,
доктор технічних наук, професор,
професор кафедри інформатики
та програмної інженерії НТУУ
“КПІ імені Ігоря Сікорського”.
Кількість наукових публікацій
в українських виданнях – понад 140.
Кількість наукових публікацій
в зарубіжних виданнях – 16.
Індекс Гірша – 11.
http://orcid.org/ 0000-0002-4601-0058

Місце роботи авторів:

Національний технічний університет
України «Київський політехнічний
інститут імені Ігоря Сікорського»,
03056, м. Київ,
проспект Берестейський 37.
Тел.: (044) 236-9651
e-mail: bogdan.dobryanskiy@jcloud.com,
stiv.inna@gmail.com