Technical Research on

A Specialized Cross-Chain Communication on Nolus Protocol

Team Collaboration Report

Team Members

1. Mikailu Nadro : Lead Researcher - mikailu29@gmail.com

2. David Emulo : UI/UX designer - Emulodavid@gmail.com

3. Nwogu Victoria : Content writer/SMM - nwoguvictoriachiamaka@gmail.com

4. Okoye Emmanuel : Fullstack developer - okoyeemmanuelobiajulu@gmail.com

5. Olebuezie Chibuzor Damian : Smart contract developer - dev.czdamian@gmail.com

6. Nwofor Gozie Israel : Community manager - gozienwofor@gmail.com

7. Esther Malgwi : Assistant Researcher & Technical copywriter/ SEO - essiemals@gmail.com

8. Okeoma Amaobi : Assistant Researcher - amaobiokeoma@gmail.com

---

Submitted on 28th December, 2024

Table of Contents

A Specialized Cross-Chain Communication on Nolus Protocol i

CHAPTER ONE 1

Technical Overview of Nolus Appchain Communication Architecture 1

1.1 Components of the Nolus Appchain 2

1.2 Interconnections 3

CHAPTER TWO 4

Proposed Solution for Cross-Chain Communication 4

2.1 Existing Technologies 4

2.2 Prerequisites 6

2.3 Potential Challenges.....6

2.4 Proposed Solutions 6

2.5 Integration Steps.....7

CHAPTER THREE 12

Integrated Nolus Blockchain Architecture with Cross-Chain Communication .12

3.1 Architecture Overview 12

3.2 Components 13

3.3 Communication Flow 14

Conclusion 14

CHAPTER ONE

Technical Overview of Nolus Appchain Communication Architecture

In this document, a comprehensive overview of the Nolus appchain architecture, detailing its key components and their interactions/ communication within the ecosystem was provided.

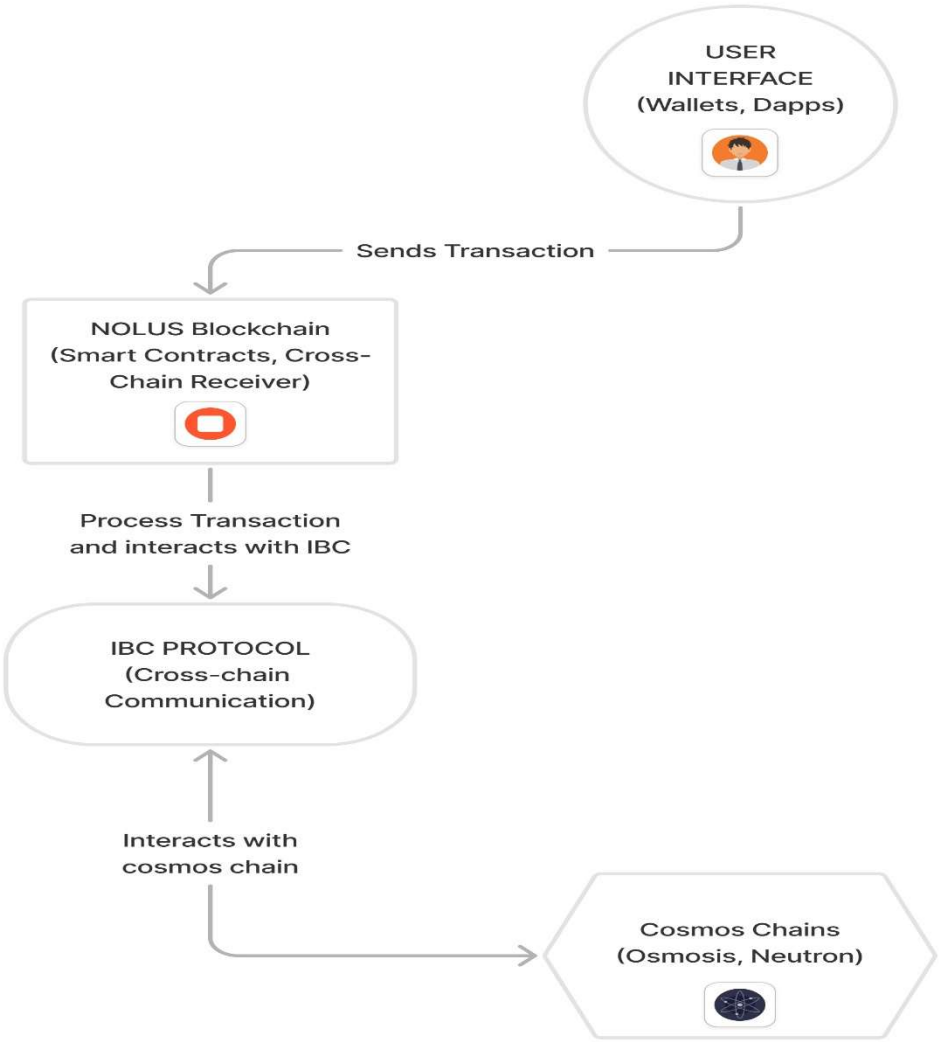

Figure 1: A diagram showing Nolus Appchain communication architecture

1.1 Components of the Nolus Appchain

1. **User Interfaces**:

User interactions occur through decentralized applications (DApps) and wallets, allowing users to send transactions to the Nolus blockchain.

2. **Nolus Blockchain:**

The central component of the Nolus network that facilitates transactions and smart contract execution.

3. **IBC Protocol:**

The Inter-Blockchain Communication (IBC) Protocol enables seamless cross-chain interactions, allowing the Nolus Blockchain to communicate with other blockchain networks.

4. Cosmos Chain:

Nolus has integrated two major cosmos based chains, which are osmosis and neutron.

i. Osmosis:

An IBC-enabled Cosmos chain, Osmosis interacts with the Nolus Blockchain, facilitating liquidity and asset management.

ii. Neutron

Another IBC-enabled Cosmos chain, Neutron connects with Nolus, enhancing cross-chain functionality and interoperability.

1.2 Interconnections

The architecture illustrates clear communication pathways between the components:

- The Nolus Blockchain connects directly with the Lease Smart Contracts.
- The IBC Protocol serves as a bridge between the Nolus Blockchain and external chains.
- Both Osmosis and Neutron interact with the Nolus system via the IBC Protocol, promoting a collaborative ecosystem.

The Nolus appchain architecture presents a robust framework for decentralized finance, enabling seamless interactions and enhancing liquidity across multiple blockchain networks. Through the integration of smart contracts and cross-chain communication, Nolus aims to foster a more interconnected and efficient financial ecosystem.

CHAPTER TWO

Proposed Solution for Cross-Chain Communication

To enhance the interoperability of the Nolus blockchain, I propose implementing a robust cross-chain communication solution that leverages existing technology such as Wormhole. This integration will facilitate seamless interactions between EVM and non-EVM chains, promoting asset transfers and data synchronization across diverse ecosystems.

2.1 Existing Technologies

1. Wormhole: This protocol enables cross-chain communication by allowing EVM and non-EVM chains to exchange messages and assets securely. It acts as a bridge, ensuring that transactions can occur smoothly across different blockchain environments.

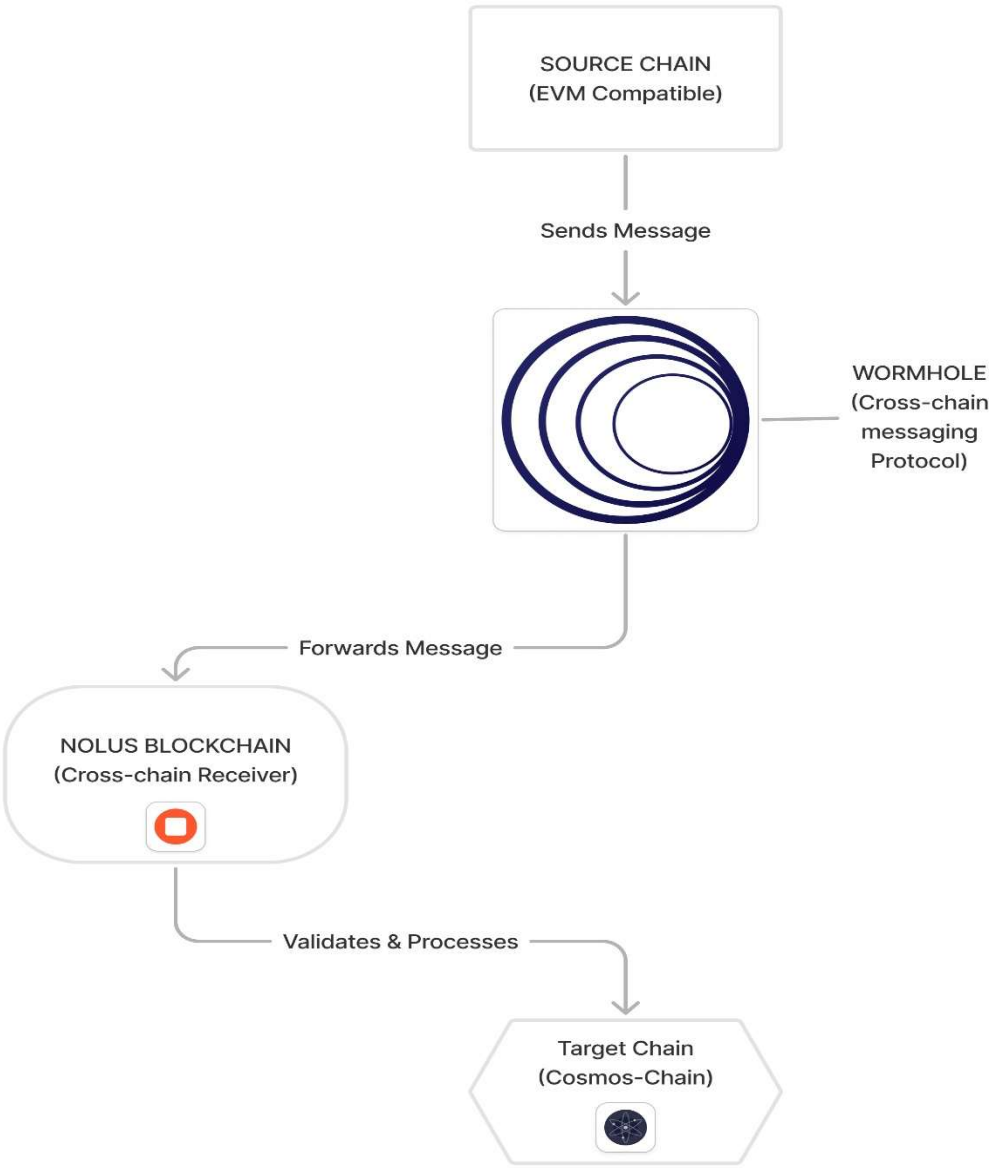

Figure 2: Wormhole cross-chain messaging protocol forwarding message from EVM to Nolus blockchain

2. LayerZero: An omnichain interoperability protocol designed to facilitate messages across various blockchains. LayerZero provides a lightweight solution for building decentralized applications that span multiple chains.

3. Axelar: This platform offers secure cross-chain communication and is compatible with various ecosystems, including EVM chains. Axelar prioritizes security and scalability, making it a strong candidate for integration.

2.2 Prerequisites

Before proceeding with the integration, it is crucial to ensure that the Nolus architecture supports these technologies. The architecture must be adaptable to accommodate semi-permissioned appchain requirements, ensuring that it can handle the nuances of cross-chain interactions.

2.3 Potential Challenges

1. Security: One of the primary concerns in cross-chain communication is maintaining trust-minimized interactions while ensuring the security of assets. The solution must leverage robust security measures to protect against potential vulnerabilities.

2. Complexity of Integration: Different consensus mechanisms between EVM and Cosmos chains may complicate the integration process. A clear strategy is required to navigate these complexities.

3. Data Consistency: Ensuring synchronized states between chains is critical for maintaining the integrity of data across the ecosystem. Solutions must be in place to handle potential discrepancies.

2.4 Proposed Solutions

1. Oracles for Reliable Data Feeds: To address data consistency issues, I propose utilizing oracles that provide reliable data feeds between chains. Oracles can

facilitate the accurate transfer of information, ensuring that all parties involved have access to the same data.

2. Multi-Signature Wallets: Implementing multi-signature wallets will enhance security for asset transfers. By requiring multiple signatures for transactions, we can significantly reduce the risk of unauthorized access and fraud.

2.5 Integration Steps

1. Select a Cross-Chain Communication Technology: I will begin by selecting a suitable technology, such as Wormhole, for its proven capabilities in cross-chain interactions.

2. Develop a Proof-of-Concept Smart Contract: I will create a proof-of-concept smart contract on the Nolus blockchain to demonstrate the feasibility of cross-chain communication. Below is a simple example of a smart contract that utilizes Wormhole for asset transfer:

```
1.    Solidity
2.    // Example Solidity contract for asset transfer via Wormhole
3.    contract CrossChainTransfer {
4.        event TransferInitiated(address indexed from, address
indexed to,
5.        uint256 amount);
6.        function transferToOtherChain(address to, uint256 amount)
external {
7.            // Logic to initiate transfer using Wormhole protocol
8.            emit TransferInitiated(msg.sender, to, amount);
9.            // Call Wormhole function to complete the transfer
10.       }
11.   }
```

3. Implement Messaging Protocol: To implement a messaging protocol specifically for sending messages between EVM-compatible chains and the Nolus blockchain

using IBC (Inter-Blockchain Communication), we'll focus on how to structure the smart contracts to facilitate cross-chain communication with IBC.

i. Smart Contract for Sending Messages to Nolus:

This contract will be used to send messages from an EVM-compatible chain to the Nolus blockchain via IBC.

Solidity

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";

contract CrossChainMessenger {
    event MessageSent(address indexed sender, string message, uint256 nonce);

    // IBC endpoint
    address public ibcEndpoint;

    constructor(address _ibcEndpoint) {
        ibcEndpoint = _ibcEndpoint;
    }

    function sendMessageToNolus(string calldata message, uint256 nonce) external {
        // Here you would typically create a payload for IBC
       // bytes memory payload = abi.encode(msg.sender, message);

     // Logic to send the payload via IBC, using the specific IBC functions
     // This is a placeholder; actual implementation would depend on the IBC library used
     // ibc.sendMessage(ibcEndpoint, payload, nonce);

        emit MessageSent(msg.sender, message, nonce);
    }
}
```

ii. Smart Contract for Receiving Messages on Nolus:

This contract will handle incoming messages from EVM-compatible chains.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract CrossChainReceiver {
    event MessageReceived(address indexed sender, string message);

    // Function to handle incoming messages via IBC
    function receiveMessage(bytes calldata messagePayload)
external {
        (address sender, string memory message) =
abi.decode(messagePayload, (address, string));

        // Process the received message
        emit MessageReceived(sender, message);
    }
}
```

Explanation of the Code

a. CrossChainMessenger Contract:

- **Event**: An event MessageSent is emitted whenever a message is sent.
- **IBC Endpoint**: The address of the IBC endpoint that will handle the message routing to Nolus.
- **sendMessageToNolus Function**: This function constructs a payload containing the sender's address and the message. It is designed to interact with IBC to send messages to the Nolus blockchain.

b. CrossChainReceiver Contract:

- **Event**: An event MessageReceived is emitted when a message is received.

- **receiveMessage Function**: This function decodes the incoming message payload and emits an event with the sender's address and the message content.

Integration Steps

a. Deploy Contracts:

First deploy the CrossChainMessenger contract on an EVM-compatible blockchain, then deploy the CrossChainReceiver contract on the Nolus blockchain.

b. Establish IBC Connection:

Ensure that the Nolus blockchain is set up to accept IBC messages from the EVM-compatible chain by configuring the appropriate IBC channels.

c. Send Messages:

Users can call the sendMessageToNolus function from the CrossChainMessenger contract to send messages to the Nolus blockchain.

d. Receive Messages:

The receiveMessage function in the CrossChainReceiver contract will be invoked when messages are sent via IBC.

4. Conduct Extensive Testing: Finally, I will conduct extensive testing across various scenarios to validate the integration. This will include testing for security vulnerabilities, data consistency, and the overall functionality of the cross-chain communication.

By implementing this proposed solution, Nolus will enhance its interoperability with other blockchains, allowing for seamless asset transfers and data synchronization. The integration of technologies like Wormhole with robust

security measures and reliable data feeds, will position Nolus as a versatile and secure platform in the blockchain ecosystem.

CHAPTER THREE

Integrated Nolus Blockchain Architecture with Cross-Chain Communication

The Nolus appchain communication between Appchain, Cosmos chains and EVM. Wormhole supports cross-chain messaging across EVM to Nolus.

3.1 Architecture Overview

The Nolus blockchain ecosystem is designed to facilitate seamless interactions between various blockchain platforms, specifically targeting both Cosmos-based chains and EVM-compatible chains. This architecture leverages the IBC (Inter-Blockchain Communication) protocol for communication within the Cosmos ecosystem and the Wormhole protocol for cross-chain messaging between EVM-compatible blockchains and Nolus.

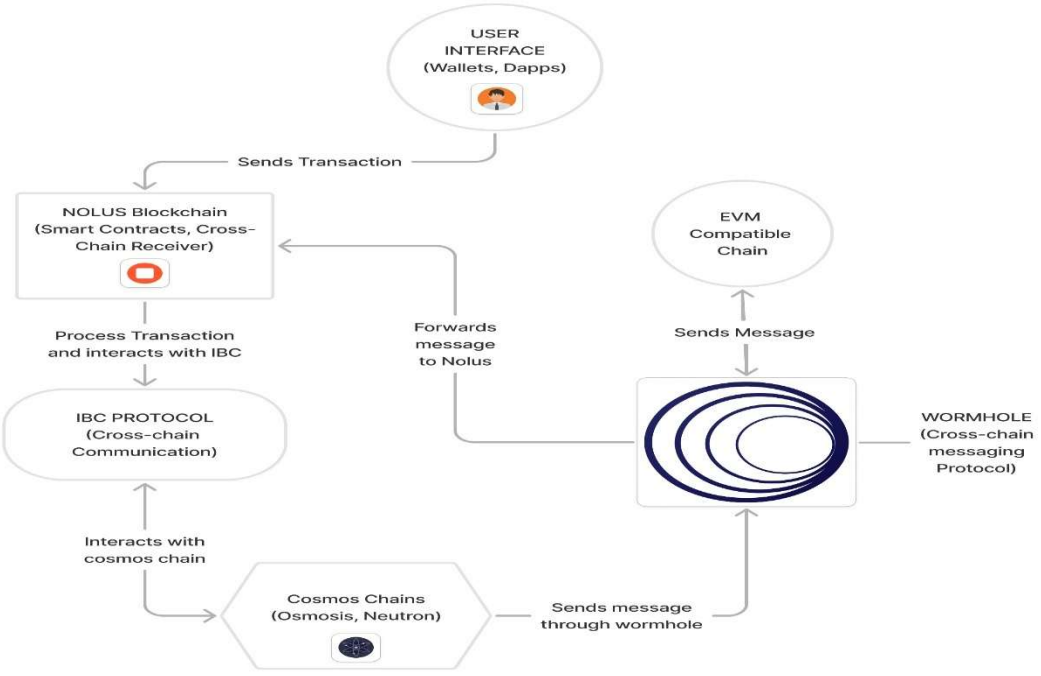

Figure 3: Nolus appchain messaging integrated with wormhole cross-chain protocol

3.2 Components

1. **User Interfaces**:

User interactions occur through decentralized applications (DApps) and wallets, allowing users to send transactions to the Nolus blockchain.

2. **Nolus Blockchain**:

The core component of the architecture, responsible for processing transactions via smart contracts and managing incoming messages through its CrossChainReceiver.

3. **IBC Protocol**:

Facilitates cross-chain communication specifically between Nolus and other Cosmos-based blockchains, enabling the transfer of tokens and data.

4. **Cosmos Chains**:

These chains (e.g., Osmosis, Neutron) interact with Nolus through the IBC protocol for native transactions and messaging.

5. **Wormhole Protocol**:

A cross-chain messaging protocol that enables Cosmos chains and EVM-compatible chains to communicate with the Nolus blockchain. Wormhole acts as a bridge, securely relaying messages between these ecosystems.

6. **EVM-Compatible Blockchains**:

Blockchains like Ethereum that can send messages to the Wormhole protocol and receive data from Nolus through this intermediary.

3.3 Communication Flow

1. **EVM-Compatible Chain to Wormhole**:

An EVM-compatible chain initiates a transaction or message, which is sent to the Wormhole protocol. This message may include token transfers, state changes, or other data relevant to the Nolus blockchain.

2. **Wormhole to Nolus**:

Upon receiving the message, Wormhole processes and wraps it, ensuring integrity and security, before forwarding it to the Nolus blockchain.

3. **Nolus Processes the Message**:

The Nolus blockchain receives the message through its CrossChainReceiver. It validates the incoming data, executing the corresponding smart contracts and actions based on the message content. This may involve updating state, transferring assets, or triggering other functions within the Nolus ecosystem.

Conclusion

This integrated architecture allows for rich interactions between the Nolus blockchain, Cosmos chains, and EVM-compatible chains. By leveraging both the IBC protocol for Cosmos interactions and the Wormhole protocol for EVM communications, Nolus enhances its interoperability and expands its ecosystem, enabling a diverse range of decentralized applications and services across multiple blockchain networks. This seamless cross-chain capability fosters a more connected and efficient blockchain environment.