

Leveraging blockchain technique for predicting and transmitting data over pervasive environment

Aditya Tripathi, Anurag Choudhury, Karan Sharma

Dr. Tapas Kumar Maiti, Dr. Rahul Mishra

***Abstract*--- This document describes the workings and implementation of blockchain across three nodes using smart contracts. This setup of the blockchain can be tweaked to include any number of devices (more than 1). Tools such as Python, Flask, and Postman have been used to implement the same.**

***Keywords*--- Blockchain, Smart contracts, Decentralization, Python, Flask, Postman**

1. INTRODUCTION

Transparency and trust are essential in today's digital world to carry out safe and effective transactions. On the other hand, trust is frequently severely hindered by old centralized systems, which can result in problems like fraud, data manipulation, and inefficiency. By establishing a decentralized, immutable record that facilitates safe and transparent transactions without the need for middlemen, blockchain technology represents an evolutionary change.

Using the Python programming language, we will create a "smart blockchain." In a "smart blockchain" network, there is a Block Producer Smart Contract (BPSC). The sender must first transfer the cryptocurrency to the BPSC wallet for the transaction to be successful in the network; the BPSC will then automatically send the same cryptocurrency to the wallet of the final recipient.

* 1. How does Smart Blockchain work?

Any successful transaction in a smart blockchain is dependent upon the presence of one or more Block Producer Smart Contracts (BPSC) within the decentralized network. For a transaction to be successful on the network, the sender must first send the cryptocurrency to BPSC's wallet, after which BPSC will automatically transmit it to the recipient. With the help of this straightforward action, the BPSC was able to obtain accurate and flawless data about every network transaction. As a result, the BPSC was able to record this data into a new block and add it to the end of the blockchain, eliminating the need to assign tasks to block producers and miners.

Smart blockchain also offers the ability to categorize transactions according to their subject. For instance, it permits the trade of multiple coin types within a network made up of separate blockchains. The network's BPSC count will be increased to achieve this. The fact that transactions might be assigned to different BPSCs with unrelated subjects is both a drawback and an important factor. For instance, if we want a smart contract to provide a reward for each of the 100 successful transactions, then the cryptocurrency transactions and the total number of successful transactions for that project should be recorded, registered, and maintained by a single BPSC and added to a single blockchain.

* 1. What are the benefits of Smart Blockchain?

The correct and precise transaction details are recorded and saved at the time of the transaction. This eliminates the need to create several alternatives from a block, as is the case with Bitcoin, Ethereum, and other networks, and allows miners to select the right and legitimate block with minimal financial expense, energy consumption, and the use of multiple proof methods.

* 1. What is the use of Smart Blockchain?

Every new smart contract can be easily registered and deployed on a separate blockchain, which is what we can do with networks like Ethereum, EOS, and others. In this manner, a BPSC will be launched simultaneously with the new smart contract, and all the smart contract's transactions will be recorded and registered into the BPSC's ledger right away. In this scenario, the dedicated blockchain will function independently and won't require any data from the Ethereum blockchain, EOS, etc. This BPSC can be coded independently and launched in between, or its codes can be imported into the appropriate smart contract codes.

1. METHODOLOGY

For demo purposes, we are going to implement 3 nodes on 1 device on ports 5000, 5001, and 5002 using 3 Python script files for each node, respectively. We can also implement this on different devices simply by executing scripts on different devices and making minor changes to them.

We will use Flask to make each node act as a server and connect them with each other. Since there is no "agreement" in this smart blockchain, there is no need for a proof-of-work (PoW) or proof-of-stake (PoS) mechanism.

Technologies such as Python, Flask, and tools such as VS Code for execution and Postman for viewing and testing the features of the smart blockchain have been used. We have created a smart blockchain based on the concept of smart contracts spread across multiple nodes.

This system supports features such as mining a new block, attaching the block to the chain, inserting custom data into the chain, and registering new nodes.

Since all the nodes are interconnected, changes made on either of the nodes are synced across all the nodes. All basic blockchain features have also been implemented in this system.

A smart contract gains permanence once it is recorded on the blockchain network. That is, no human being will be able to alter the contract's original text in any way if the blockchain network is operational. The same applies to block producer smart contracts (BPSCs).

Working proof of each of these features has been presented in the next section.

1. IMPLEMENTATION

A screenshot of a computer

Description automatically generated

Figure 1: Mining of Block on Node 1

A screenshot of a computer

Description automatically generated

Figure 2: Sync of Blocks on Node 2

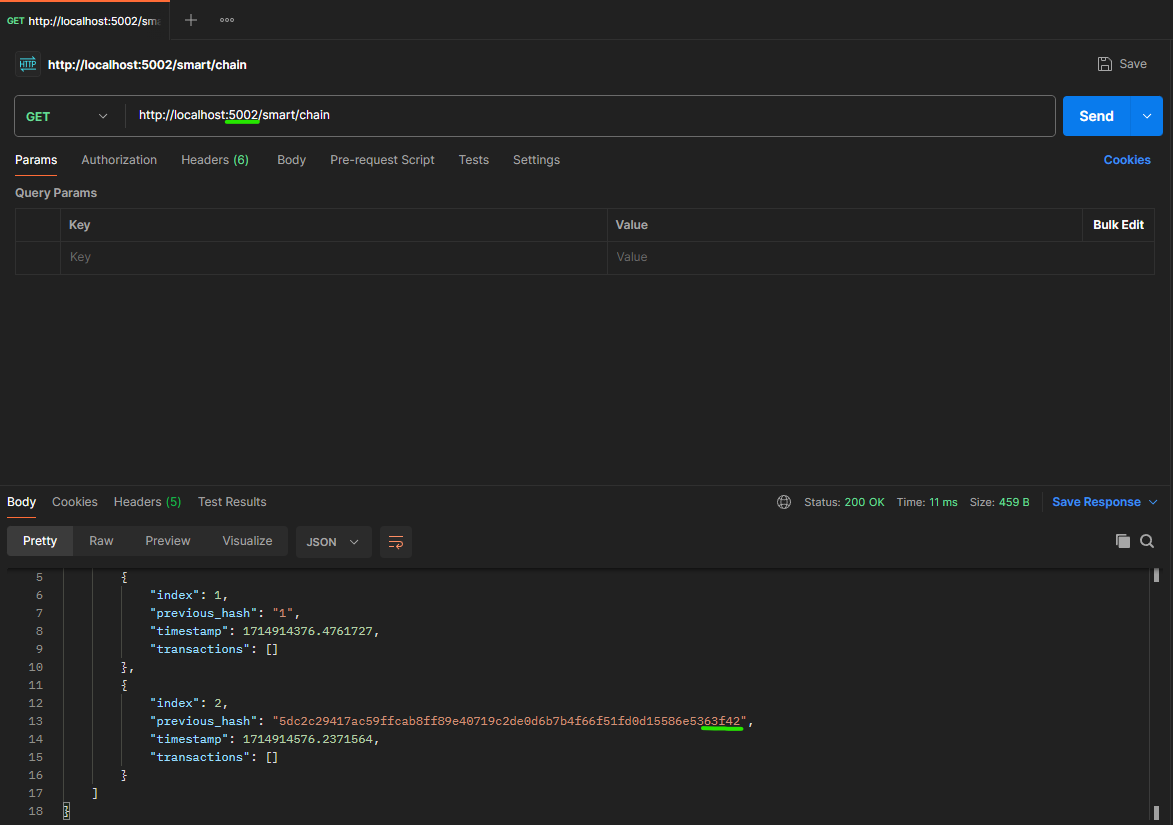


Figure 3: Sync of Blocks on Node 3