Central Bank Digital Coin(CBDC)

Submitted by

Nikhil Naresh Meshram

To

KBA-CHF

Table of Contents:

Introduction to the CBDC Project

Central Bank Digital Currency (CBDC): Transforming the Financial Landscape

In response to the rapid evolution of digital technologies and changing economic landscapes, central banks worldwide are exploring the concept of Central Bank Digital Currency (CBDC). CBDC represents a digital form of a country's national currency, issued and regulated by the central bank. Unlike cryptocurrencies, CBDC is a centralized and government-backed form of digital currency. The primary aim is to combine the benefits of digital innovation with the stability and security associated with traditional fiat currencies.

Problem Statement

Current Financial System Challenges:

Limited Financial Inclusion: A significant portion of the global population lacks access to basic financial services due to barriers such as geographical constraints, lack of infrastructure, and exclusionary policies.

Inefficiencies in Cross-Border Transactions: Traditional cross-border transactions are often slow, costly, and prone to intermediaries, leading to delays and additional expenses for businesses and individuals.

Counterfeiting and Fraud Risks: Traditional banknotes are susceptible to counterfeiting, leading to financial losses and security concerns.

Operational Inefficiencies: Existing financial systems often involve multiple intermediaries, leading to complex and time-consuming processes, especially in settlements and clearances.

Solution Statement

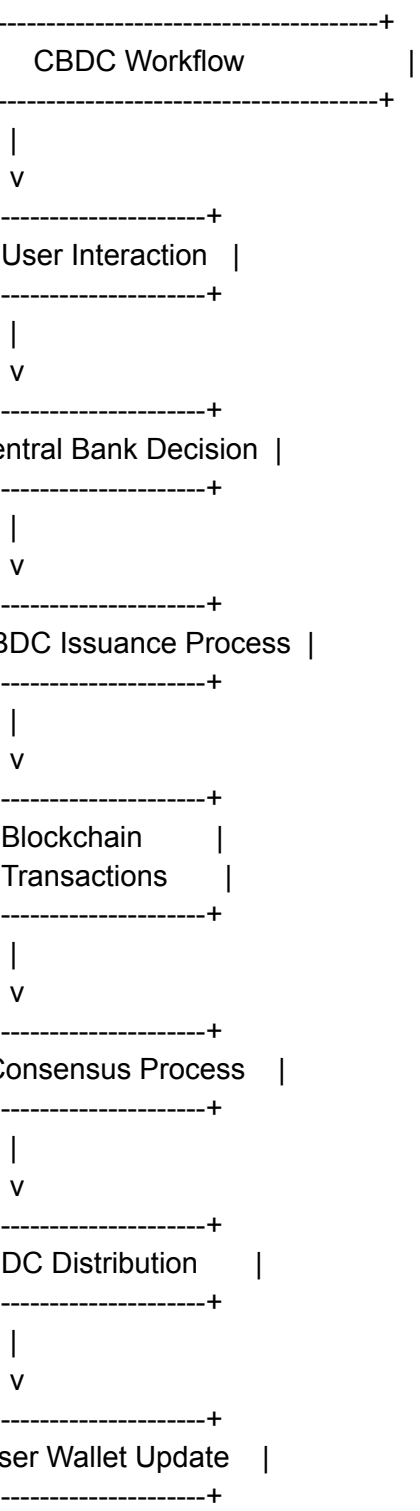
Financial Inclusion: CBDC can facilitate greater financial inclusion by providing a digital payment infrastructure that reaches unbanked and underbanked populations. This empowers individuals who previously lacked access to traditional banking services.

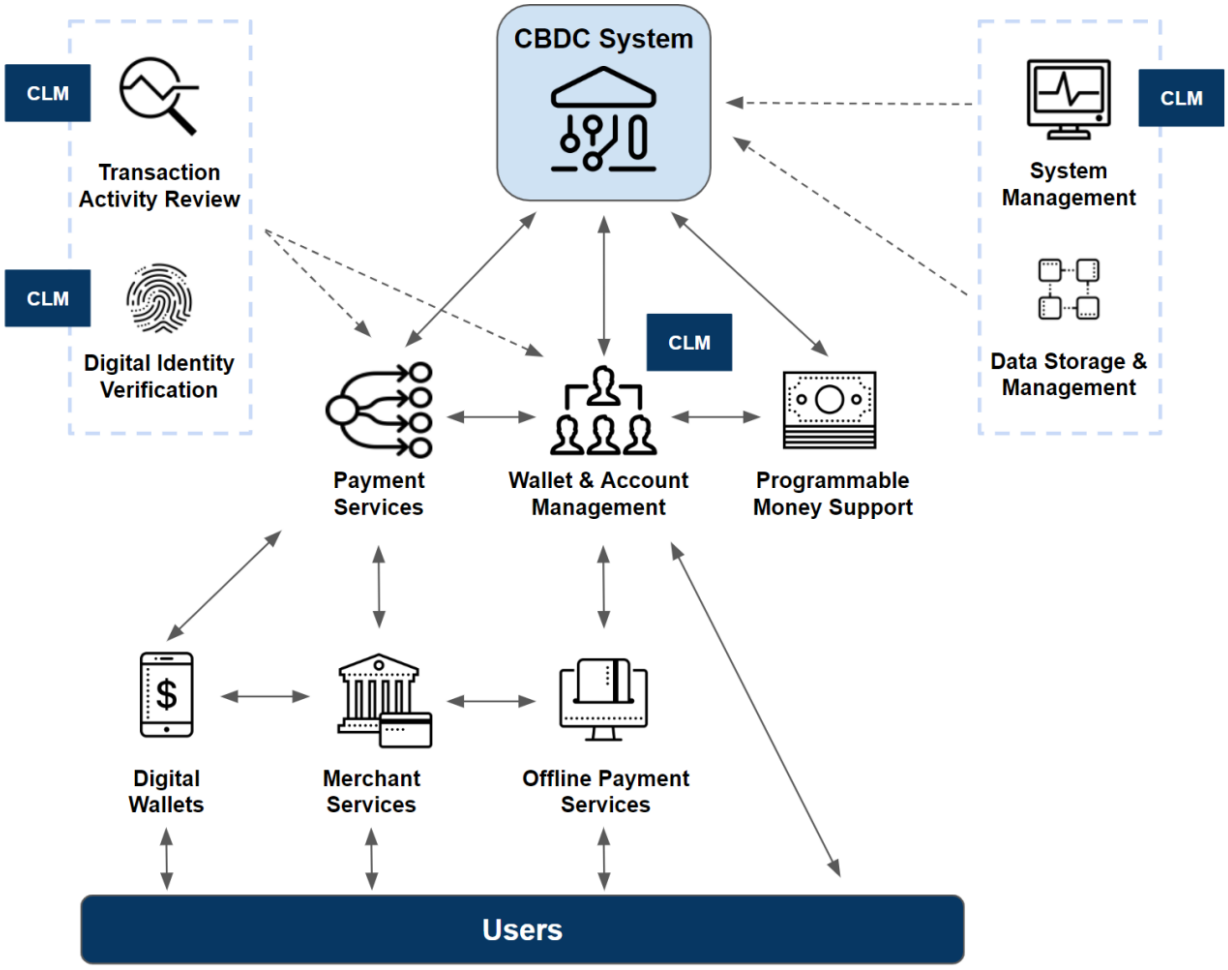
Efficient Cross-Border Transactions: CBDC can streamline cross-border transactions by leveraging blockchain technology for faster, cheaper, and more transparent transfers. This reduces dependency on multiple intermediaries and enhances the overall efficiency of global transactions.

Enhanced Security: The use of blockchain and cryptographic technologies in CBDC mitigates the risks associated with counterfeiting and fraud. The transparent and secure nature of distributed ledger technology ensures the integrity of transactions.

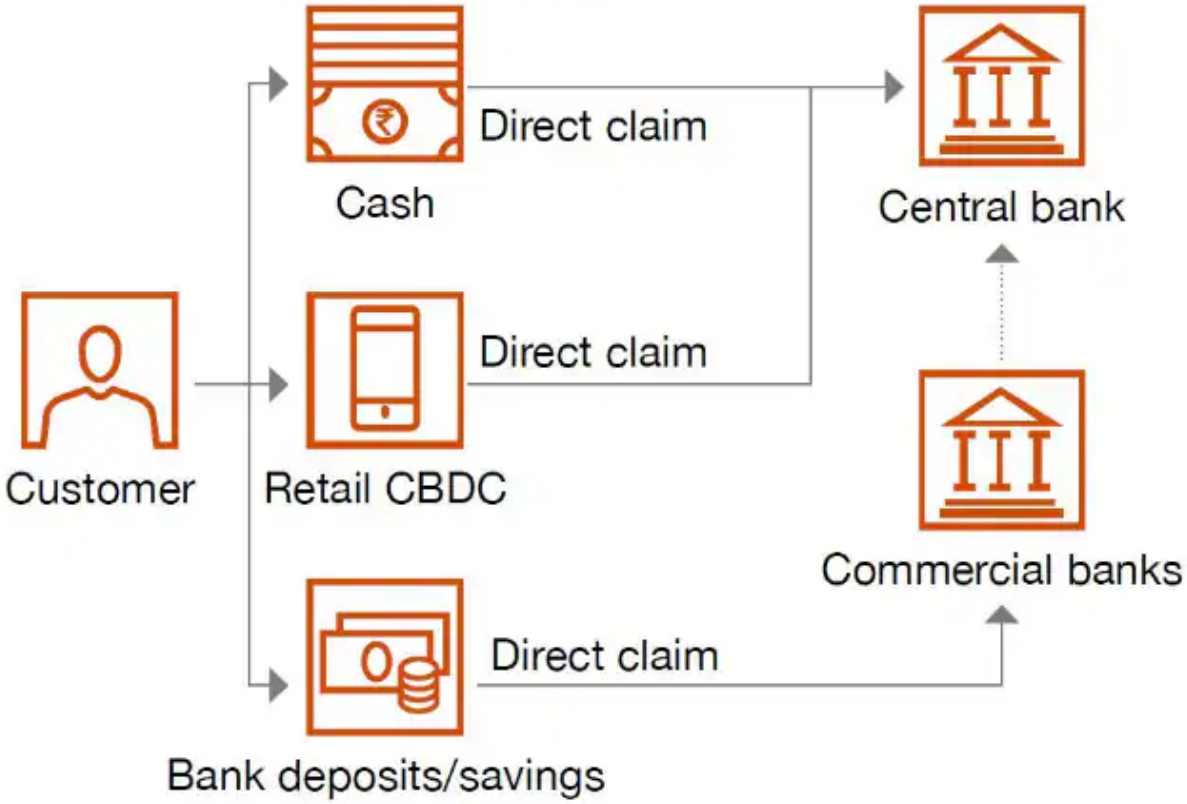
Operational Streamlining: CBDC introduces a more efficient and direct mechanism for financial transactions, reducing the need for intermediaries and accelerating settlement processes. This results in operational cost savings and increased overall efficiency.

Workflow


CBDC Workflow
User Interaction
Central Bank Decision
CBDC Issuance Process
Blockchain Transactions
Consensus Process
CBDC Distribution
User Wallet Update


CBDC System
CLM
Transaction Activity Review
CLM
Digital Identity Verification
CLM
System Management
Data Storage & Management
CLM
Payment Services
Wallet & Account Management
Programmable Money Support
Digital Wallets
Merchant Services
Offline Payment Services
Users

Retail CBDC and monetary system



Prerequisites:

- Hardware Requirement:
- Operating System: Ubuntu 22.04 LTS
- Minimum 8 GB RAM
- 40 GB Free Space
- High-Speed internet connection
- sudo access
- Project: Supply-Chain
- Step-by-Step Instructions

Hardware Requirement:

Operating System: Ubuntu 22.04 LTS

Minimum 8 GB RAM

40 GB Free Space

High-Speed internet connection

sudo access

Project: CBDC

Step-by-Step Instructions Installation:

Update Packages

In case of a fresh Ubuntu 22 installation, use the following command to update the

packages before installing other dependencies.

sudo apt update

1. cURL

Step1. Install curl

sudo apt install curl –y

2. NodeJS (Ver 18.x)

Step1. Install NVM (Node Version Manager), open a terminal and execute the following

command.

wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | Bash

Step2. Now make sure to close the previous terminal and open a new one, then execute

the following command to install NodeJs

nvm install 18

Step3. Check the version of nodeJS installed

node -v

Step4. Check the version of npm installed

npm -v

3. Docker

Step 1. Download the script

curl -fsSL https://get.docker.com -o get-docker.sh

Step 2. Execute permission for the script

sudo chmod +x get-docker.sh

Step 3. Execute the script

./get-docker.sh

Step 4. Remove the script

rm get-docker.sh

Step 5. To manage Docker as a non-root user

sudo usermod -aG docker $USER

4. JQ

Step 1. Installing JQ

sudo apt install jq –y

5. Build Essential

Step 1. Installing Build Essential

sudo apt install build-essential

6. Hyperledger Fabric binaries

Step 1. Download the script

curl -sSLO
https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/
install-f
abric.sh && chmod +x install-fabric.sh

Step2. Execute the script

./install-fabric.sh -f '2.5.4' -c '1.5.7'

Step3. Copy the binaries to /usr/local/bin
sudo cp fabric-samples/bin/* /usr/local/bin

Step4. Check the binary versions
peer version
orderer version
fabric-ca-client version

7. Go

Step1. Download Go
wget https://go.dev/dl/go1.21.3.linux-amd64.tar.gz

Step2. Extract
tar -C /usr/local -xzf go1.21.3.linux-amd64.tar.gz

Step3. Add /usr/local/go/bin to the PATH environment variable.

8. Visual Studio Code
Download and install the latest version of VS code from here

Chaincode Functions

PauseTokenTransfers

- Description: Pauses token transfers. Only authorized entities can call this function.
- Parameters: `ctx` (TransactionContextInterface)
- Returns: `(bool, error)`

UnpauseTokenTransfers

- Description: Unpauses token transfers. Only authorized entities can call this function.
- Parameters: `ctx` (TransactionContextInterface)
- Returns: `(bool, error)`

GetAllAccountsWithOrgs

- Description: Returns a list of all accounts along with their respective organizations on the network.
- Parameters: `ctx` (TransactionContextInterface)
- Returns: `(map[string]string, error)`

BondExists

- Description: Returns true when an asset with the given ID exists in the private data collection.
- Parameters: `ctx` (TransactionContextInterface), `BondID` (string)
- Returns: `(bool, error)`

CreateCBDCBond

- Description: Creates a new instance of Bond in the private data collection.
- Parameters: `ctx` (TransactionContextInterface), `bondID` (string)
- Returns: `(string, error)`

GetCBDCBond

- Description: Retrieves an instance of Bond from the private data collection.
- Parameters: `ctx` (TransactionContextInterface), `bondID` (string)
- Returns: `(*Bond, error)`

RemoveCBDCBond

- Description: Deletes an instance of Bond from the private data collection.
- Parameters: `ctx` (TransactionContextInterface), `bondID` (string)
- Returns: `(error)`

Mint

- Description: Creates new tokens and adds them to the minter's account balance.
- Parameters: `ctx` (TransactionContextInterface), `amount` (int)
- Returns: `(error)`

Burn

- Description: Redeems tokens from the minter's account balance.
- Parameters: `ctx` (TransactionContextInterface), `amount` (int)
- Returns: `(error)`

Transfer

- Description: Transfers tokens from the client's account to the recipient's account.
- Parameters: `ctx` (TransactionContextInterface), `recipient` (string), `amount` (int)
- Returns: `(error)`

BalanceOf

- Description: Returns the balance of the given account.
- Parameters: `ctx` (TransactionContextInterface), `account` (string)
- Returns: `(int, error)`

ClientAccountBalance

- Description: Returns the balance of the requesting client's account.
- Parameters: `ctx` (TransactionContextInterface)
- Returns: `(int, error)`

ClientAccountID

- Description: Returns the ID of the requesting client's account.
- Parameters: `ctx` (TransactionContextInterface)
- Returns: `(string, error)`

TotalSupply

- Description: Returns the total token supply.
- Parameters: `ctx` (TransactionContextInterface)
- Returns: `(int, error)`

Approve

- Description: Allows the spender to withdraw from the calling client's token account.
- Parameters: `ctx` (TransactionContextInterface), `spender` (string), `value` (int)
- Returns: `(error)`

Allowance

- Description: Returns the amount still available for the spender to withdraw from the owner.
- Parameters: `ctx` (TransactionContextInterface), `owner` (string), `spender` (string)
- Returns: `(int, error)`

TransferFrom:

- Transfers a specified amount of tokens from one address ("from") to another address ("to").
- Checks if the contract is initialized, retrieves the spender's allowance, and updates it after the transfer.
- Emits a "Transfer" event.

Name:

- Returns the name of the token.

Symbol:

- Returns the symbol of the token.

Initialize:

- Sets initial information for the token (name, symbol, decimals).
- Checks the authorization of the client to initialize the contract and ensures that the contract is not already initialized.

GetCBDCHistory:

- Retrieves the transaction history of CBDC (Central Bank Digital Currency) transfers.

GetAllCBDCBondsHistory:

- Retrieves the transaction history of CBDC bonds, possibly from a private data collection.

GetAllAssets:

- Combines the transaction history of CBDC transfers and CBDC bonds.

TransferHelper:

- A helper function for transferring tokens, ensuring that the "from" and "to" addresses are not the same, the transfer amount is non-negative, and the sender has sufficient funds.

Add and Sub Functions:

- Helper functions for addition and subtraction with overflow checks.

CheckInitialized:

Checks whether the contract options have been initialized.

List of the docker up containers