



School: Campus:

Academic Year: Subject Name: Subject Code:

Semester: Program: Branch: Specialization:

Date:

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiment : Gas Race – Optimizing Smart Contract Efficiency

* **Coding Phase: Pseudo Code / Flow Chart / Algorithm**

Introduction

In this experiment, the objective is to deploy and test smart contracts on **multi-chain environments**, focusing on **Stacks** bar (Bitcoin then Layer for Clarity smart contracts) and **Arbitrum** (Ethereum Layer-2 scaling solution). This helps developers understand **cross -chain deployment, gas optimization, and Layer-2 interoperability** in blockchain ecosystems.

- **Stacks** integrates directly with the Bitcoin network, enabling smart contracts using the **Clarity language** for secure and predictable execution.
- **Arbitrum** is an **Ethereum Layer-2** solution that uses **Optimistic Rollups** to reduce gas costs and improve scalability while maintaining EVM compatibility.

The deployment demonstrates how a single DApp or contract logic can be adapted to work across **two distinct blockchain architectures** — one Bitcoin-linked and one EVM-based — achieving **cross-chain adaptability and performance efficiency**.

Algorithm

1. **Set up development environments** ○ Install **Stacks CLI** and **Clarinet** for Stacks contract deployment.
 - Install **Hardhat** or **Remix + MetaMask (Arbitrum Testnet)** for EVM-based deployment.
2. **Write Smart Contracts** ○ Create a **Clarity contract** for Stacks (e.g., token or storage logic).
 - Create a **Solidity contract** for Arbitrum implementing similar logic.
3. **Compile Contracts** ○ Use **Clarinet test** and **build** commands for Stacks.
 - Use **Hardhat compile** or **Remix compiler** for Arbitrum.
4. **Deploy on Testnets** ○ Deploy the Clarity contract on **Stacks Testnet**.
 - Deploy the Solidity contract on **Arbitrum Sepolia Testnet** using **MetaMask** and **Hardhat**.
5. **Verify Deployment** ○ Confirm deployment transaction on **Stacks Explorer** and **Arbitrum Explorer**.
 - Test function execution (e.g., token minting, data update).
6. **Cross-Chain Observation** ○ Analyze performance difference, transaction fee, and confirmation time between Stacks and Arbitrum.

* **Softwares used**

1. **Clarinet** – for developing and deploying Clarity smart contracts on the Stacks blockchain.
2. **Hardhat** – for compiling, testing, and deploying Solidity contracts on Arbitrum.
3. **MetaMask** – for wallet connection and contract interaction on Arbitrum Testnet.

* Implementation Phase: Final Output (no error)

Multi-Chain Deploy – Stacks & Arbitrum Environment Setup

Objective:

To set up and configure development environments for deploying smart contracts on **Stacks (Bitcoin Layer)** and **Arbitrum (Ethereum Layer 2)**, enabling multi-chain dApp testing and interoperability.

Steps / Algorithm

- **Environment Setup – Stacks (Clarity):**

1. **Install Clarinet:**

Use npm install -g @hirosystems/clarinet to install the Stacks development tool.

2. **Initialize Project:**

Run clarinet new project_name to create a new Clarity smart contract environment.

3. **Write Smart Contract:**

Create .clar files in the /contracts folder.

4. **Test Contracts:**

Execute clarinet test to run unit tests locally.

5. **Deploy to Testnet:**

Use clarinet integrate or **Stacks Explorer** for contract deployment on Testnet.

- **Environment Setup – Arbitrum (Solidity):**

1. **Install Hardhat:**

Use npm install --save-dev hardhat to create a Solidity project for Arbitrum.

2. **Configure Network:**

Add Arbitrum RPC in hardhat.config.js:

```
networks: {  
  arbitrumSepolia: {  
    url: "https://sepolia-rollup.arbitrum.io/rpc",  
    accounts: [PRIVATE_KEY]  
  }  
}
```

3. **Compile Contracts:**

Run npx hardhat compile.

4. **Deploy Contract:**

Use a deploy script like npx hardhat run scripts/deploy.js --network arbitrumSepolia.

5. **Verify on Explorer:**

Check the deployed contract on **Arbiscan Testnet**.

Softwares Used

1. **Clarinet** – for Stacks smart contract development and testing.

2. **Hardhat** – for Solidity contract compilation and Arbitrum deployment.

3. **MetaMask** – for connecting wallet and interacting with both networks.

Output / Learning

Successfully configured and deployed smart contracts on both **Stacks** and **Arbitrum** environments, understanding differences in **Clarity (non-Turing complete)** and **Solidity (EVM-compatible)** ecosystems.

Page No.....

**As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.

• Cross-Chain Deployment & Configuration:

Set up and configured smart contract environments on **Stacks (Bitcoin Layer)** and **Arbitrum (Ethereum Layer 2)**, enabling seamless multi-chain deployment and interoperability testing.

• Network Integration & Optimization:

Configured RPC connections, wallets, and deployment scripts in **Hardhat** and **Clarinet**, ensuring efficient cross-network communication and optimized transaction flow.

• Smart Contract Compatibility & Validation:

Adapted Solidity and Clarity contracts to their respective environments, validating performance and ensuring proper function execution across both ecosystems.

• Performance Monitoring & Final Testing:

Deployed and verified contracts on **Stacks Testnet** and **Arbitrum Sepolia**, analyzing transaction confirmation speed, gas efficiency, and contract stability for consistent cross-chain behavior.

* Observations

- Observed smoother and faster transaction execution on **Arbitrum** due to Layer-2 scalability enhancements.
- Verified strong **security and clarity in contract logic** during deployment on the **Stacks** blockchain.
- Noted effective **cross-chain interoperability** and consistent contract behavior across both testnets.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	50		

Signature of the Student:

Name :

Regn. No. :

Page No.....

Signature of the Faculty:

***As applicable according to the experiment.**
Two sheets per experiment (10-20) to be used.