run...	Week 1: ...	2 weeks ago
tes...	Initial co...	3 weeks ago
tsc...	Initial co...	3 weeks ago

README

Review the assignment due date

Bitcoin Protocol Development - Week 1: Interacting with a Bitcoin Node

Overview

This week, you'll learn how to use Bitcoin Core's RPC to interact with a running Bitcoin node. The tasks involve connecting to a Bitcoin Core RPC daemon, creating, and broadcasting a transaction. You'll need a Bitcoin node running in `regtest` mode on your local machine to test your solution. You can use `bash`, `javascript`, `python`, or `rust` for your implementation.

A docker-compose file is provided to help you launch a Bitcoin node in `regtest` mode locally.

> Tip
>
> [OPTIONAL] You can also use the bitcoin.conf file to start a local regtest node with your locally built bitcoin binaries. Copy the `bitcoin.conf` file in the default bitcoin data-directory `~/.bitcoin/`. If you don't have the data-directory, just create one.

Objective

Successfully send a Payment + OP_Return Transaction.

Your tasks are to:

- Connect to a Bitcoin node in `regtest` mode using RPC.
- Create and load a wallet named “testwallet.”
- Generate an address from the wallet.
- Mine blocks to that address.
- Send 100 BTC to a provided address.
- Include a second output with an OP_RETURN message: “We are all Satoshi!!”
- Set the fee rate to 21 sats/vB.
- Output the transaction ID (txid) to an `out.txt` file.

Place your solution in the appropriate directory based on your chosen language:

- bash
- javascript
- python
- rust

Requirements

Input

Create a transaction with the following outputs:

- **Output 1**:
  - Address: `bcrt1qq2yshcmzdlznnpxx258xswqlmqcxjs4dssfxt2`
  - Amount: 100 BTC
- **Output 2**:
  - Data: "We are all Satoshi!!" (This should be an `OP_RETURN` output with the binary encoding of the string.)

Output

After creating and broadcasting the transaction, save the `txid` to out.txt.

Execution

To test your solution locally:

- Ensure that you have `npm` and `nvm` installed and your system. You will need `node v18` or greater to run the

test script.
- Ensure that there is no `bitcoind` process running in your system.
- Uncomment the line corresponding to your language in run.sh.
- Give execution permission to `test.sh`, by running `chmod +x ./test.sh`.
- Execute `./test.sh`.

If your code works, you will see the test completed successfully.

Submission:

- Create a commit with your local changes.
- Push the commit to your forked repository (`git push origin main`).

Evaluation Criteria

Your submission will be evaluated based on:

- **Autograder**: Your code must pass the autograder test script.
- **Explainer Comments**: Include comments explaining each step of your code.
- **Code Quality**: Your code should be well-organized, commented, and adhere to best practices.

Plagiarism Policy

Our plagiarism detection checker thoroughly identifies any instances of copying or cheating. Participants are required to publish their solutions in the designated repository, which is private and accessible only to the individual and the administrator. Solutions should not be shared publicly or with peers. In case of plagiarism, both parties involved will be directly disqualified to maintain fairness and integrity.

AI Usage Disclaimer

You may use AI tools like ChatGPT to gather information and explore alternative approaches, but avoid relying solely on AI for complete solutions. Verify and validate any insights obtained and maintain a balance between AI assistance and independent problem-solving.

Why These Restrictions?

These rules are designed to enhance your understanding of the technical aspects of Bitcoin. By completing this assignment, you gain practical experience with the technology that secures and maintains the trustlessness of Bitcoin. This challenge not only tests your ability to develop functional Bitcoin applications but also encourages deep engagement with the core elements of Bitcoin technology.