

网络与信息安全课内实验 5： 区块链实验报告

班级：计算机 2101 班

姓名：田濡豪

学号：2203113234

1 实验平台及环境

本次实验使用个人计算机完成。

使用 WSL（Windows Subsystem for Linux）完成，所安装的 Ubuntu 系统版本为：

Ubuntu 22.04.4 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

2 实验步骤

1. 创建区块链
2. 运行区块链
3. 节点一致性
4. 与现实区块链系统的区别

3 实验过程

3.1 创建区块链

区块链类的主体函数与实验文档中链接提供的实现完全一致。一个区块包含 index、timestamp、transactions、proof、previous_hash 等元素。

- Index：区块的序号。需要通过 index 判断区块在链中的位置、链的长度，并在发生冲突时选择最长的链。
- Timestamp：区块的生成时间。提供了区块中交易记录的发生区间，同时也提供了所有矿工在以此区块竞争中的最小生成时间。
- Transactions：区块链作为分布式账本的核心功能。记录了一个区块对应时间窗口内发生的所有虚拟货币交易。

- Proof: 工作量证明, 使用前一个区块的信息和本区块的工作量证明可以构造出一个符合特殊规则的哈希值, 该证明需要的工作量相当大, 并且由所有矿工竞争获取, 使得攻击者若想篡改分布式账本, 在平均情况下需要相当于有所有矿工算力总和一半的算力。
- Previous Hash: 上一个区块的哈希值, 用于验证一个区块的正确性, 根据哈希函数的特点, 一旦区块链经过篡改, 之后的哈希值均会改变。

观察实验文档中链接文章提供的工作量证明机制, 发现该机制有一个漏洞。

```
def proof_of_work(self, last_proof):
    """
    Simple Proof of Work Algorithm:
    - Find a number p' such that hash(pp') contains leading 4 zeroes, where
    p is the previous p'
    - p is the previous proof, and p' is the new proof
    :param last_proof: <int>
    :return: <int>
    """
    proof = 0
    while self.valid_proof(last_proof, proof) is False:
        proof += 1
    return proof

    @staticmethod
    def valid_proof(last_proof, proof):
        """
        Validates the Proof: Does hash(last_proof, proof) contain 4 leading zeroes?
        :param last_proof: <int> Previous Proof
        :param proof: <int> Current Proof
        :return: <bool> True if correct, False if not.
        """
        guess = f'{last_proof}{proof}'.encode()
        guess_hash = hashlib.sha256(guess).hexdigest()
        return guess_hash[:4] == "0000"
```

区块链工作证明的核心思想是保证获取每一个 proof 的计算量都足够大, 以此来构建可信的分布式账本。因此设计必须确保任何一个 proof 都不能以投机取巧的方式取得。在上面的证明机制设计中, 没有考虑到找到的 proof 与上一个工作量验证码 last_proof 是否相等的问题。如果找到了一个 proof 与 last_proof 相同的验证码对, 那么接下来的所有区块都可以递归地立刻生成。推广这种情况, 如果恶意用户发现一个 proofA 与构建若干个块之后的 proofB 相同, 那么 proofA 与 proofB 之间的所有 proof 就形成了一个闭环, 之后的所有块都可以通过重复这个闭环构建。因此必须保证每个块的 proof 各不相同。

对于 sha256 函数, 可以从数值上保证以下特性:

- 只要输入不同, 则输出一定不同。
- 无法从输出反推输入。这可以间接保证输入不会与其对应输出相同。

因此, 对 valid_proof 函数的一种修改办法是: 将上一个 guess_hash 作为 last_proof 与当前 proof 相联计算哈希值, 然后将这个哈希值作为下一个块的 last_proof。连接 guess_hash 与 proof

必定改变计算出的哈希值，所以下一个块的 guess 变量不可能与当前 guess 变量相同——间接保证了下一个哈希也与当前哈希不同，从而确保每个 proof 需要重新计算。

修改区块的内容，添加一个当前 proof 和 previous_hash 产生的 hash。在不输入 curr_hash 时自动生成，因此不需要修改创世区块的构造方式。

```
block = {
    'index': len(self.chain) + 1,
    'timestamp': time.time(),
    'transactions': self.current_transactions,
    'proof': proof,
    'previous_hash': previous_hash or self.hash(self.chain[-1]),
    'hash': curr_hash or hashlib.sha256(f'{previous_hash}{proof}'.encode(
)).hexdigest(),
}
```

修改 proof_of_work 函数的运作方式，在返回值中添加当前 hash 便于创建下一个区块。

```
def proof_of_work(self, previous_hash):
    """
    Simple Proof of Work Algorithm:
    - Find a number p' such that hash(pp') contains leading 4 zeroes, where
    p is the previous p'
    - p is the previous proof, and p' is the new proof
    :param previous_hash: <int>
    :return: <int>
    """
    proof = 0
    while True:
        curr_hash = self.valid_proof(previous_hash, proof)
        if curr_hash[:4] == "0000":
            break
        proof += 1
    return proof, curr_hash
```

修改挖矿函数 mine，传入上一个块的 hash，并在创建当前块时传入当前块的 hash。

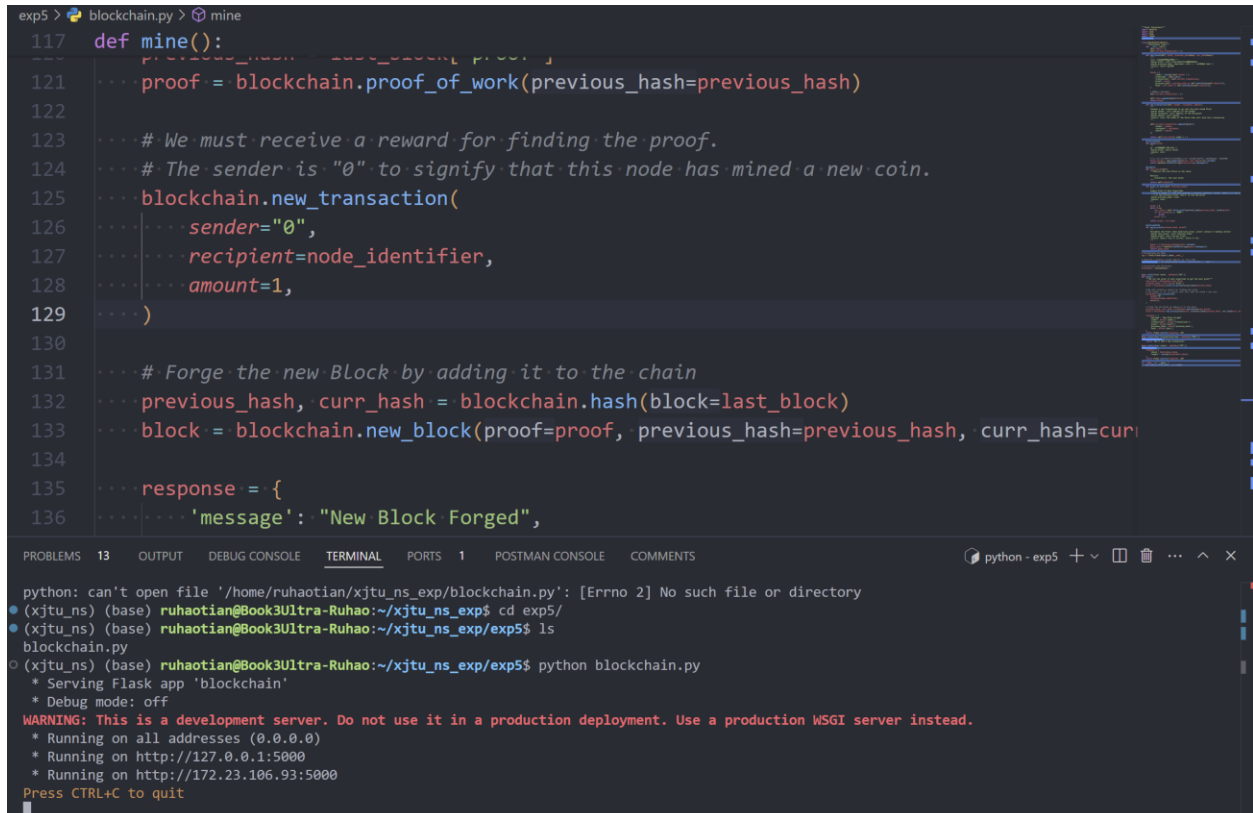
```
def mine():
    """We run the proof of work algorithm to get the next proof"""
    last_block = blockchain.last_block
    previous_hash = last_block['hash']
    proof = blockchain.proof_of_work(previous_hash)
    # We must receive a reward for finding the proof.
    # The sender is "0" to signify that this node has mined a new coin.
    blockchain.new_transaction(
        sender="0",
        recipient=node_identifier,
        amount=1,
    )
    # Forge the new Block by adding it to the chain
    previous_hash, curr_hash = blockchain.hash(last_block)
    block = blockchain.new_block(proof, previous_hash, curr_hash)
    response = {
        'message': "New Block Forged",
    }
```

```

        'index': block['index'],
        'transactions': block['transactions'],
        'proof': block['proof'],
        'previous_hash': block['previous_hash'],
        'hash': block['hash'],
    }
    return flask.jsonify(response), 200

```

3.2 运行区块链



```

exp5 > blockchain.py > mine
117 def mine():
121     proof = blockchain.proof_of_work(previous_hash=previous_hash)
122
123     # We must receive a reward for finding the proof.
124     # The sender is "0" to signify that this node has mined a new coin.
125     blockchain.new_transaction(
126         sender="0",
127         recipient=node_identifier,
128         amount=1,
129     )
130
131     # Forge the new Block by adding it to the chain
132     previous_hash, curr_hash = blockchain.hash(block=last_block)
133     block = blockchain.new_block(proof=proof, previous_hash=previous_hash, curr_hash=curr_hash)
134
135     response = {
136         'message': "New Block Forged",

```

```

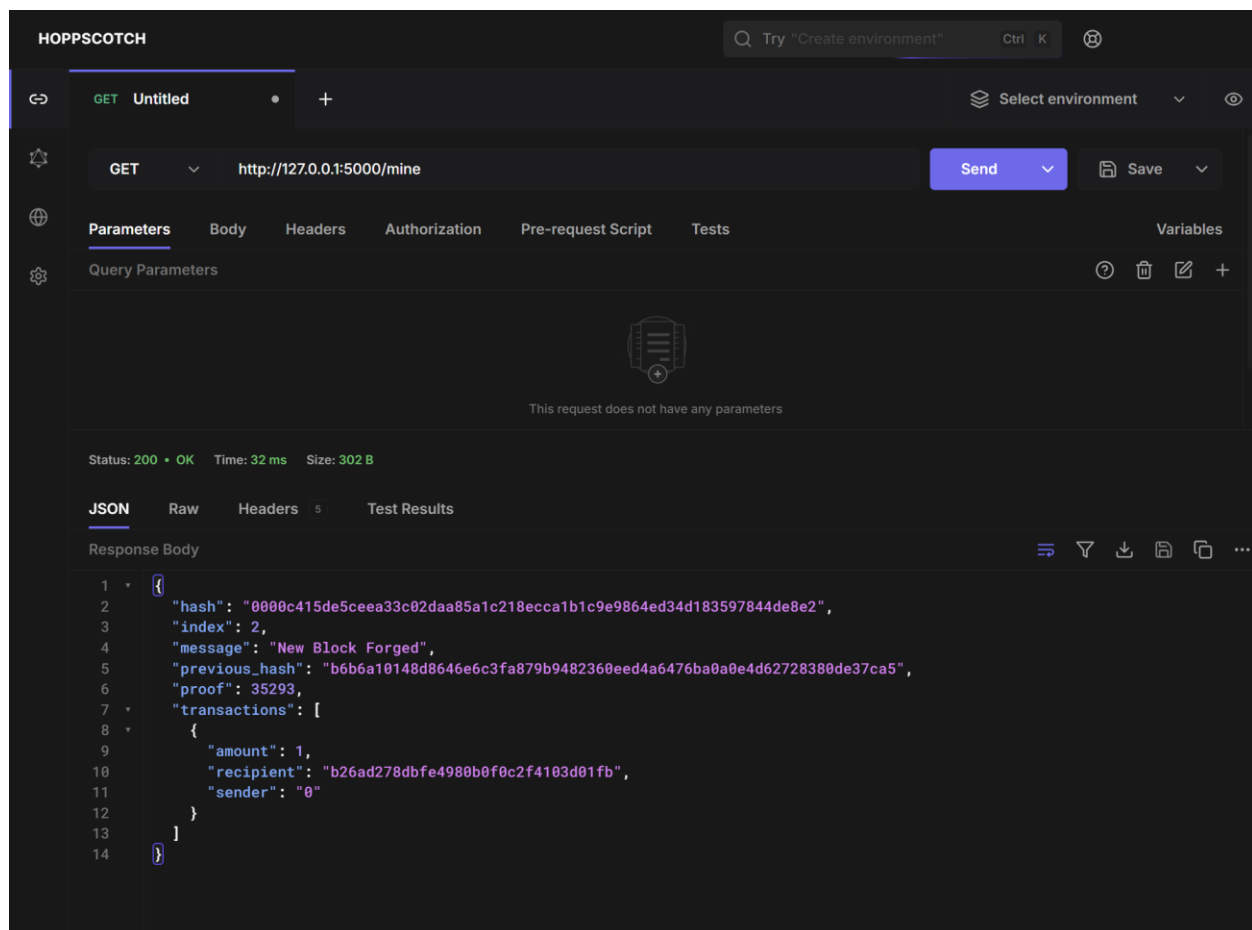
python: can't open file '/home/ruhaotian/xjtu_ns_exp/blockchain.py': [Errno 2] No such file or directory
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp$ cd exp5/
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp/exp5$ ls
blockchain.py
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp/exp5$ python blockchain.py
* Serving Flask app 'blockchain'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.23.106.93:5000
Press CTRL+C to quit

```

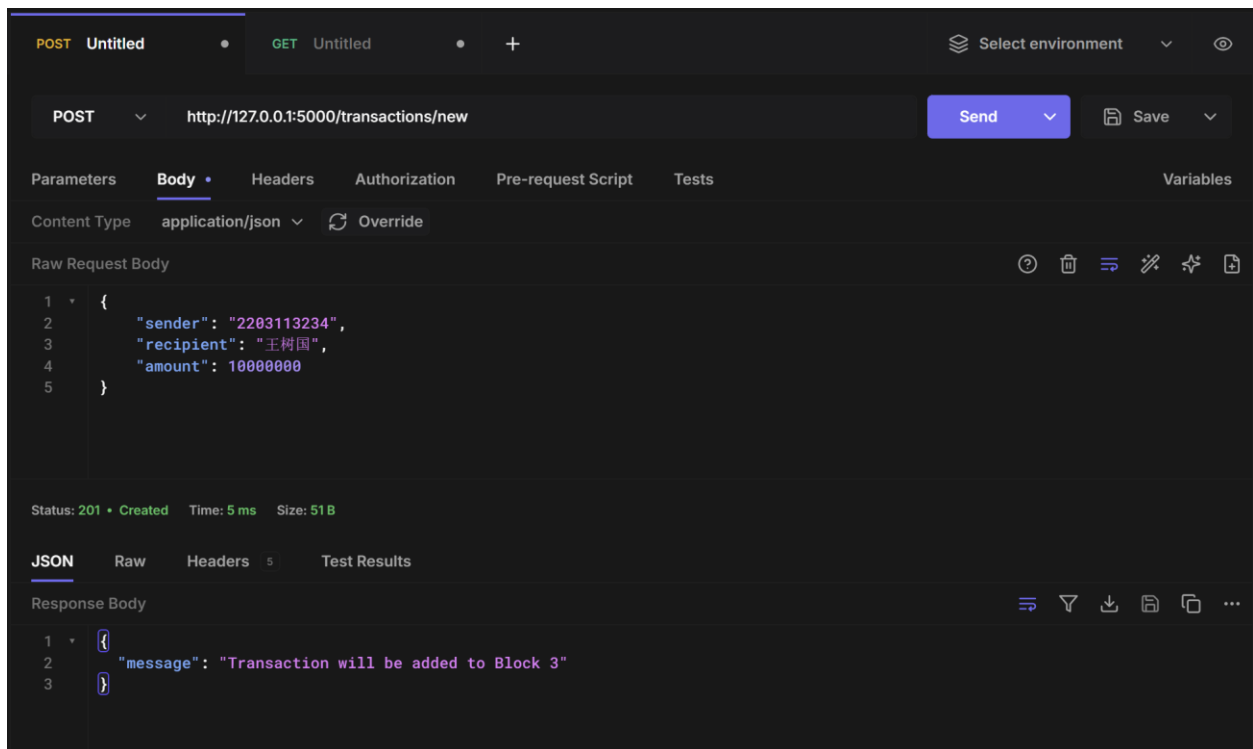
首先在本地启动区块链服务。

因为本人之前用过 API 测试平台，个人认为 Postman 系统占用过多，对本简单实验能力过于冗余。因此本实验使用轻量级 API 管理平台 Hoppscotch 完成。

在 Hoppscotch 中输入一个新的挖矿请求，可见挖出了一个新的区块。



尝试 post 请求，可见也能“成功转账”。



3.3 节点一致性

参考实验文档中链接的文章补全相关函数。

```
def valid_chain(self, chain):  
    """  
    Determine if a given blockchain is valid  
    :param chain: <list> A blockchain  
    :return: <bool> True if valid, False if not  
    """  
    last_block = chain[0]  
    current_index = 1  
    while current_index < len(chain):  
        block = chain[current_index]  
        print(f'{last_block}')  
        print(f'{block}')  
        print("\n-----\n")  
        # Check that the hash of the block is correct  
        if block['previous_hash'] != self.hash(last_block):  
            return False  
        # Check that the Proof of Work is correct  
        if not self.valid_proof(last_block['proof'], block['proof']):  
            return False  
        last_block = block  
        current_index += 1  
    return True  
def resolve_conflicts(self):  
    """
```

```

This is our Consensus Algorithm, it resolves conflicts
by replacing our chain with the longest one in the network.
:return: <bool> True if our chain was replaced, False if not
"""

neighbours = self.nodes
new_chain = None
# We're only looking for chains longer than ours
max_length = len(self.chain)
# Grab and verify the chains from all the nodes in our network
for node in neighbours:
    response = requests.get(f'http://{node}/chain', timeout=5)
    if response.status_code == 200:
        length = response.json()['length']
        chain = response.json()['chain']
        # Check if the length is longer and the chain is valid
        if length > max_length and self.valid_chain(chain):
            max_length = length
            new_chain = chain
# Replace our chain if we discovered a new, valid chain longer than ours
if new_chain:
    self.chain = new_chain
    return True
return False

```

接下来对区块链的冲突解决进行测试。首先构建两个分布式账本实例，分别映射到调试地址的 5000 和 5001 端口。

```

PROBLEMS 13 OUTPUT DEBUG CONSOLE TERMINAL PORTS 2 POSTMAN CONSOLE COMMENTS python - exp5 + v
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp$ cd exp5/
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp/exp5$ python blockchain.py --port 5001
* Serving Flask app 'blockchain'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://172.23.106.93:5001
Press CTRL+C to quit
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp$ python blockchain.py 5000
python: can't open file '/home/ruhaotian/xjtu_ns_exp/blockchain.py': [Errno 2] No such file or directory
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp$ cd exp5/
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp/exp5$ python blockchain.py 5000
usage: blockchain.py [-h] [--port PORT]
blockchain.py: error: unrecognized arguments: 5000
(xjtu_ns) (base) ruhaotian@Book3Ultra-Ruhao:~/xjtu_ns_exp/exp5$ python blockchain.py --port 5000
* Serving Flask app 'blockchain'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.23.106.93:5000
Press CTRL+C to quit

```

在两个账本上注册自身地址及彼此的地址，此处以暴露到 5001 端口的账本为例。

POST http://127.0.0.1:5001/nodes/register Send

Parameters Body Headers Authorization Pre-request Script Tests

Content Type application/json Override

Raw Request Body

```
1 {
2   "nodes": ["http://127.0.0.1:5000", "http://127.0.0.1:5001"]
3 }
```

Status: 201 • Created Time: 23 ms Size: 90 B

JSON Raw Headers 5 Test Results

Response Body

```
1 {
2   "message": "New nodes have been added",
3   "total_nodes": [
4     "127.0.0.1:5000",
5     "127.0.0.1:5001"
6   ]
7 }
```

在 5000 号账本上挖 3 个节点，可见最后一个节点 index 为 4。在 5001 号账本挖 5 个节点，构造冲突的区块链。

GET http://127.0.0.1:5000/mine Send Save

Parameters Body Headers Authorization Pre-request Script Tests Variables

Query Parameters

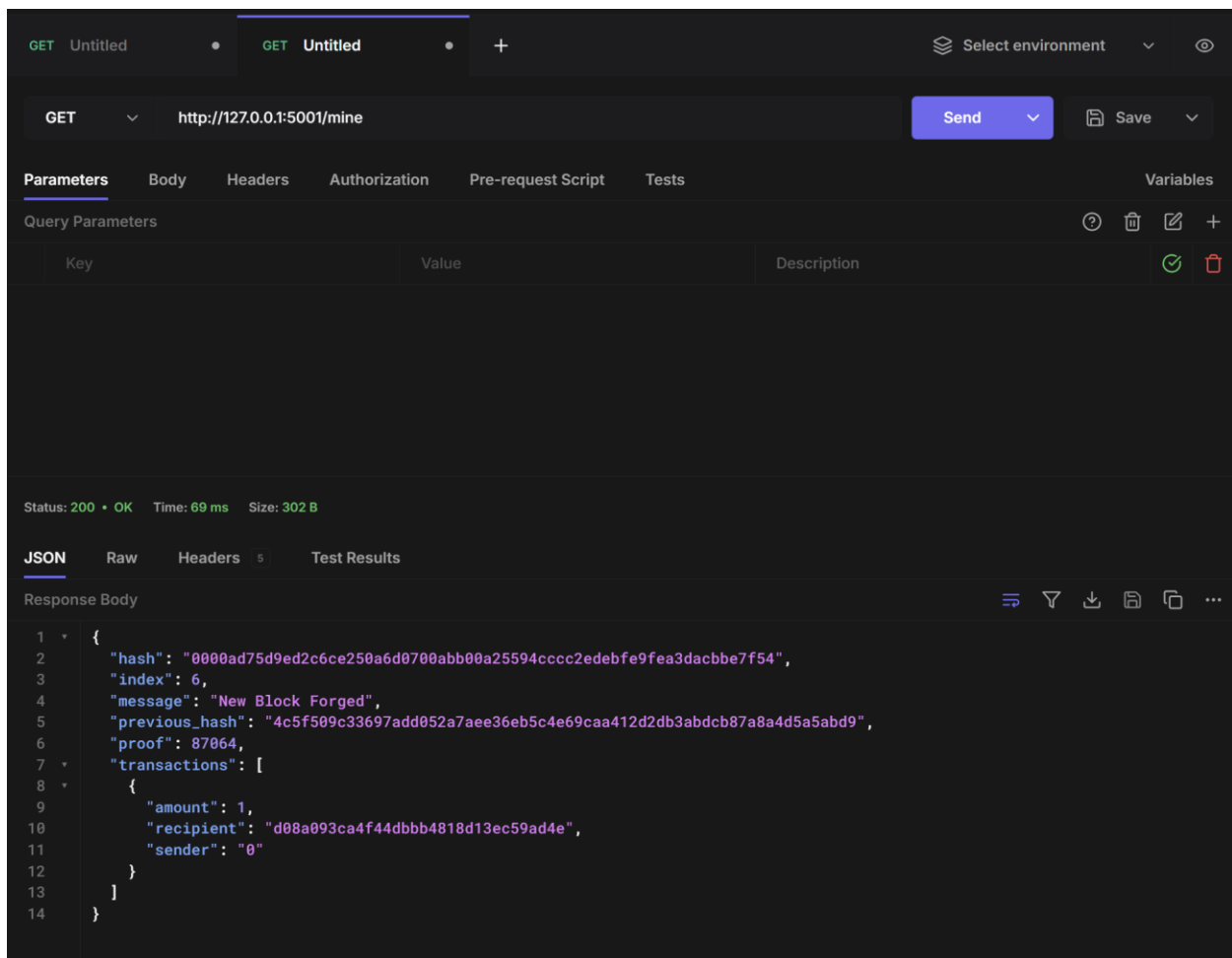
Key	Value	Description
-----	-------	-------------

Status: 200 • OK Time: 49 ms Size: 302 B

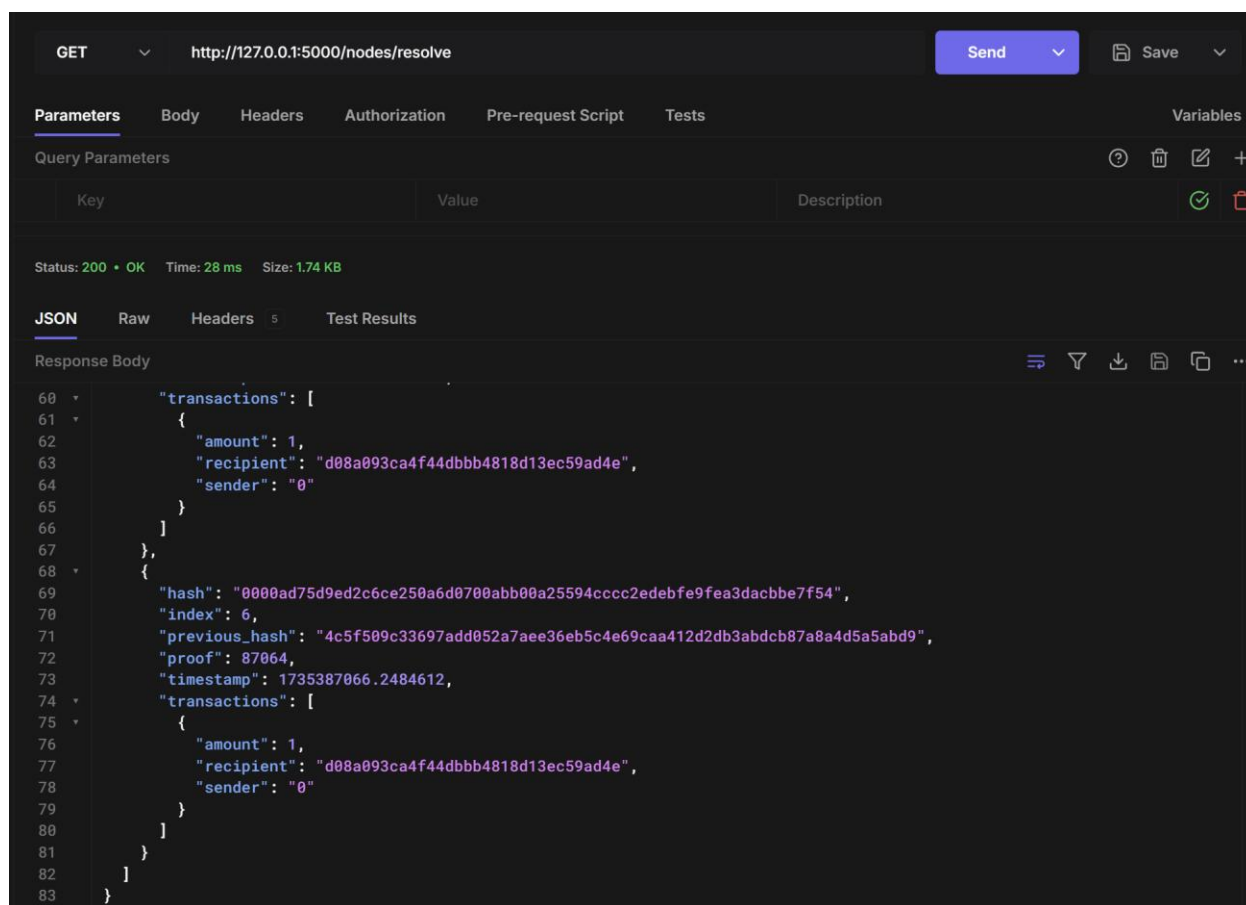
JSON Raw Headers 5 Test Results

Response Body

```
1 {
2   "hash": "000076b073bd700deaa30d61ee68adf18b7c71c7bc56322fb05db22aca50ec04",
3   "index": 4,
4   "message": "New Block Forged",
5   "previous_hash": "a5caef81b2bb07e5e02f1479e68b9084c26992c984340c63d44a8b2b79b6b394",
6   "proof": 70275,
7   "transactions": [
8     {
9       "amount": 1,
10      "recipient": "9429c91110ff4b7fbe0dd80742a8eab9",
11      "sender": "0"
12    }
13  ]
14 }
```

在 5000 号账本上调用 `resolve`，从返回的报文中可见 5000 号账本的区块长度已经变为了 7（最后一个区块的 `index` 为 6）



3.4 与现实区块链系统的区别

本系统与比特币等实际区块链应用的一个明显不同是挖矿没有虚拟货币奖励。对矿工的虚拟货币奖励是比特币等虚拟货币维持账本不可篡改的一个重要保障。奖励激励矿工不断提高算力挖矿，从而使伪造者赢得算力竞争的难度越来越高。本次实验中的区块链系统并没有这一经济体系。

4 心得体会

我个人认为，尽管目前区块链的主要应用还是虚拟货币交易，区块链的去中心化和不可篡改特性可以延伸到虚拟货币和经济系统之外。比如通过分布式账本的难以篡改特性进行投票统计，实现政府和企业重大决策透明化。通过将各种证书和版权加入区块，可以助力不可篡改的证书验证。将各类难以可信传播的数据，比如各个医疗机构的数据使用区块链技术统一管理，可以在解决安全问题的同时加速信息流动。