東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

区块链技术与应用

v0.11.5

实验五：共识（下）

丁烨，网络空间安全学院 副教授
dingye@dgut.edu.cn

分布式账本回顾

分布式账本工作流程

1. 本地创建交易记录（Transaction）
2. 传送给分布式账本，存放于记录池（Pool）
3. 分布式账本节点之间通过共识不断同步记录池
4. 不断尝试让记录池满足一定条件（例如，完成写入证明）
5. 校验记录池中的交易记录并生成区块（Block）
6. 加入新的区块并生成新的区块链
7. 分布式账本节点之间通过共识不断同步区块链

实现支持共识的分布式账本

优化区块链基础功能

- transaction.py

```
class Transaction(Base):
    def __init__(self,
                 sender: str, receiver: str, amount: int,
                 t: float = None, prev_hash: str = None, transaction_id: str = None):
        assert UUID(sender, version=4)
        assert UUID(receiver, version=4)
        assert transaction_id is None or UUID(transaction_id, version=4)
        ...
```

实现支持共识的分布式账本

优化区块链基础功能

- transaction.py

```
class Transaction(Base):
    def __init__(…):
        …
        self.id = transaction_id if transaction_id is not None else str(uuid4())
        self.sender = sender
        self.receiver = receiver
        self.amount = amount
        self.t = t if t is not None else datetime.now().timestamp()
        self.prev_hash = prev_hash
```

实现支持共识的分布式账本

优化区块链基础功能

- transaction.py

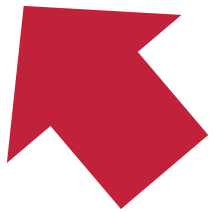
```
class Transaction(Base):
    ...
    def __eq__(self, other):
        return \
            self.id == other.id and \
            self.sender == other.sender and \
            self.receiver == other.receiver and \
            self.amount == other.amount
```

实现支持共识的分布式账本

优化区块链基础功能

- transaction.py

```
class Transaction(Base):
    ...
    def __hash__(self):
        return hash((self.id, self.sender, self.receiver, self.amount))
```



实现支持共识的分布式账本

优化区块链基础功能

- blockchain.py

```
class Block(Base):
    ...
    def add(self, item):
        if item not in self.items:
            item.prev_hash = self.items[-1].hash() if len(self.items) else None
            self.items.append(item)
        else:
            logging.debug('Duplicate item will be dropped: %s', item)
```

实现支持共识的分布式账本

优化区块链基础功能

- blockchain.py

```
class Blockchain(Base):
    ...
    def find(self, item):
        for block in self.blocks:
            if item in block.items:
                return block.items[block.items.index(item)]
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
def on_message(self, message):
    ...
    elif message['op'] == 'blocks':
        blocks = json.loads(str(pickle.loads(db.get('ledger'))))['blocks']
        if 'args' in message and 'start' in message['args']:
            blocks = blocks[int(message['args']['start']):]
        ...
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

❖ app.py

```
def on_message(self, message):
    ...
    elif message['op'] == 'blocks':
        ...
        self.write_message(json.dumps({
            'status': 200,
            'error': 'OK',
            'response': {
                'blocks': blocks
            }
        }))
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
elif message['op'] == 'merge':
    if 'args' in message and 'pool' in message['args']:
        ...
    elif 'args' in message and 'blocks' in message['args']:
        ...
    else:
        self.write_message(json.dumps({
            'status': 500,
            'error': 'Operation "merge" requires following "args": "pool" or "blocks"',
            'response': None
        }))
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
if 'args' in message and 'pool' in message['args']:
    pool = pickle.loads(db.get('pool'))
    ledger = pickle.loads(db.get('ledger'))
    for i in message['args']['pool']:
        pool.add(Transaction(
            transaction_id=i['id'],
            sender=i['sender'],
            receiver=i['receiver'],
            amount=i['amount']
        ))
    ...
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

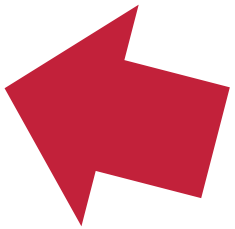
```
if 'args' in message and 'pool' in message['args']:
    ...
    for i in pool.copy():
        if ledger.find(i):
            pool.remove(i)
    db.set('pool', pickle.dumps(pool))
    self.write_message(json.dumps({
        'status': 202,
        'error': 'Accepted'
    }))
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
elif 'args' in message and 'blocks' in message['args']:
    pool = pickle.loads(db.get('pool'))
    ledger = pickle.loads(db.get('ledger'))
    for i in message['args']['blocks']:
        ...
    try:
        ledger.validate()
    except AssertionError:
        ...
    else:
        ...
```



实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

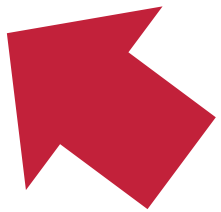
```
for i in message['args']['blocks']:
    if not len(ledger.blocks) or i['prev_hash'] == ledger.blocks[-1].hash():
        block = Block(prev_hash=i['prev_hash'])
        for j in i['items']:
            ...
        ledger.add(block)
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
for j in i['items']:
    transaction = Transaction(
        sender=j['sender'],
        receiver=j['receiver'],
        amount=j['amount'],
        t=j['t'],
        prev_hash=j['prev_hash'],
        transaction_id=j['id']
    )
    ...
```



实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
for j in i['items']:
    ...
    if ledger.find(transaction):
        self.write_message(json.dumps({
            'status': 500,
            'error': 'Duplicate transaction in blockchain: {}'.format(transaction)
        }))
        return
    ...
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
for j in i['items']:
    ...
    if transaction in pool:
        pool.remove(transaction)
    block.add(transaction)
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
try:
    ledger.validate()
except AssertionError:
    self.write_message(json.dumps({
        'status': 500,
        'error': 'Malicious block is detected.'
    }))
else:
    ...
```

实现支持共识的分布式账本

修改 P2P 网络服务并使其支持同步区块链

- app.py

```
try:
    ledger.validate()
except AssertionError:
    ...
else:
    db.set('pool', pickle.dumps(pool))
    db.set('ledger', pickle.dumps(ledger))
    self.write_message(json.dumps({
        'status': 202,
        'error': 'Accepted'
    }))
```

实现支持共识的分布式账本

优化 Tracker 并使其支持同步区块链

❖ tracker.py

```
def spawn(self, url):
    if url not in self.spawning:
        if len(self.peers) < DEFAULTS['max_connection']:
            self.spawning.append(url)
            logging.info('Spawning new peer: %s', url)
            ...
            self.spawning.remove(url)
        else:
            logging.debug('Max # of connections is reached.')
```

实现支持共识的分布式账本

优化 Tracker 并使其支持同步区块链

- tracker.py

```
def on_message(self, ws, message):
    if len(self.peers) >= DEFAULTS['max_connection']:
        return
    try:
        message = json.loads(message)
    except json.JSONDecodeError:
        pass
    ...
```

实现支持共识的分布式账本

优化 Tracker 并使其支持同步区块链

- tracker.py

```
def query(self):
    for peer in self.peers:
        try:
            peer.send(json.dumps({'op': 'peers'}))
            peer.send(json.dumps({'op': 'pool'}))
            peer.send(json.dumps({'op': 'blocks'}))
        except:
            traceback.print_exc()
```

实现支持共识的分布式账本

优化 Tracker 并使其支持同步区块链

- tracker.py

```
def on_message(self, ws, message):
    ...
    if 'response' in message:
        if 'peers' in message['response']:
            ...
        if 'pool' in message['response']:
            ...
        if 'blocks' in message['response']:
            ...
```

实现支持共识的分布式账本

优化 Tracker 并使其支持同步区块链

- tracker.py

```
if 'pool' in message['response']:
    logging.debug('Received pool updates from: %s', ws.url)
    for peer in self.peers:
        if peer != ws:
            logging.info('Announced pool updates to: %s', peer.url)
            peer.send(json.dumps({
                'op': 'merge',
                'args': {
                    'pool': message['response']['pool']
                }
            }))
```

实现支持共识的分布式账本

优化 Tracker 并使其支持同步区块链

- tracker.py

```
if 'blocks' in message['response']:
    logging.debug('Received blocks updates from: %s', ws.url)
    for peer in self.peers:
        if peer != ws:
            logging.info('Announced blocks updates to: %s', peer.url)
            peer.send(json.dumps({
                'op': 'merge',
                'args': {
                    'blocks': message['response']['blocks']
                }
            }))
```

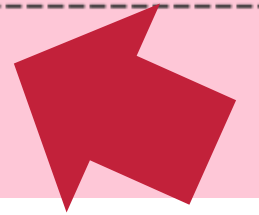
实现支持共识的分布式账本

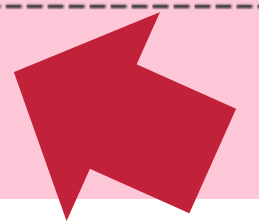
测试支持共识的分布式账本

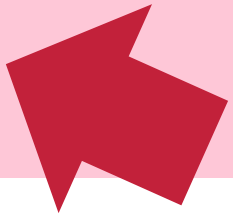
```
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.app -p 9000 -r 9000.db
[2021-05-12 18:47:20,147] Tornado is listening on port: 9000
```

```
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.app -p 9001 -r 9001.db
[2021-05-12 18:47:40,009] Tornado is listening on port: 9001
```

```
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.app -p 9002 -r 9002.db
[2021-05-12 18:47:47,144] Tornado is listening on port: 9002
```





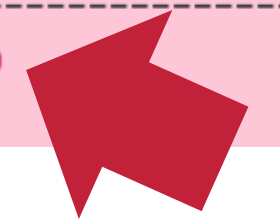


实现支持共识的分布式账本

测试支持共识的分布式账本

```
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.tracker ws://localhost:9000/ws
[2021-05-12 18:48:09,714] Spawning new peer: ws://localhost:9000/ws
[2021-05-12 18:48:09,719] Peer disconnected: ws://localhost:9000/ws
```

```
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.builder -r 9000.db
```



ws://localhost:9000/ws

Send

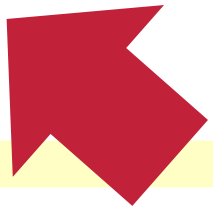
```
{
  "op": "merge",
  "args": {
    "pool": [{
      "id": "00000000-0000-0000-0000-000000000001",
      "sender": "00000000-0000-0000-0000-000000000000",
      "receiver": "00000000-0000-0000-0000-000000000001",
      "amount": 3000
    }]
  }
}
```

```
{
  "status": 202,
  "error": "Accepted"
}
```

ws://localhost:9000/ws

Send

```
{
  "op": "merge",
  "args": {
    "pool": [{
      "id": "00000000-0000-0000-0000-000000000002",
      "sender": "00000000-0000-0000-0000-000000000000",
      "receiver": "00000000-0000-0000-0000-000000000002",
      "amount": 2000
    }]
  }
}
```

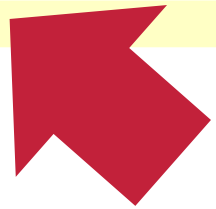


```
{
  "status": 202,
  "error": "Accepted"
}
```

ws://localhost:9000/ws

Send

```
{
  "op": "merge",
  "args": {
    "pool": [{
      "id": "00000000-0000-0000-0000-000000000003",
      "sender": "00000000-0000-0000-0000-000000000000",
      "receiver": "00000000-0000-0000-0000-000000000003",
      "amount": 1000
    }]
  }
}
```



```
{
  "status": 202,
  "error": "Accepted"
}
```

ws://localhost:9000/ws

Send

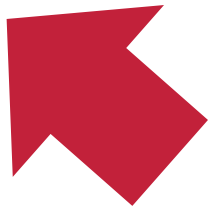
```
{
  "op": "pool"
}
```


```
{
  "status": 200,
  "error": "OK",
  "response": {
    "pool": [{
      "id": "00000000-0000-0000-0000-000000000003",
      "sender": "00000000-0000-0000-0000-000000000000",
      "receiver": "00000000-0000-0000-0000-000000000003",
      "amount": 1000,
      "t": 1620811762.848223,
      "prev_hash": null
    }, {
      "id": "00000000-0000-0000-0000-000000000001",
      "sender": "00000000-0000-0000-0000-000000000000",
      "receiver": "00000000-0000-0000-0000-000000000001",
      "amount": 3000,
      "t": 1620811780.296454,
```

ws://localhost:9000/ws

Send

```
{
  "op": "merge",
  "args": {
    "pool": [{
      "id": "00000000-0000-0000-0000-000000000004",
      "sender": "00000000-0000-0000-0000-000000000000",
      "receiver": "00000000-0000-0000-0000-000000000004",
      "amount": 4000
    }]
  }
}
```



```
{
  "status": 202,
  "error": "Accepted"
}
```

ws://localhost:9000/ws

Send

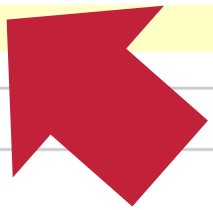
```
{
  "op": "pool"
}
```

```
{
  "status": 200,
  "error": "OK",
  "response": {
    "pool": []
  }
}
```

ws://localhost:9000/ws

Send

```
{
  "op": "blocks"
}
```




```
{
  "status": 200,
  "error": "OK",
  "response": {
    "blocks": [{
      "items": [{
        "id": "00000000-0000-0000-0000-000000000003",
        "sender": "00000000-0000-0000-0000-000000000000",
        "receiver": "00000000-0000-0000-0000-000000000003",
        "amount": 1000,
        "t": 1620811762.848223,
        "prev_hash": null
      }, {
        "id": "00000000-0000-0000-0000-000000000002",
        "sender": "00000000-0000-0000-0000-000000000000",
        "receiver": "00000000-0000-0000-0000-000000000002",
        "amount": 2000,
        "t": 1620811774.80847,
        "prev_hash": "cca2b52f7ba2d64086a20d8764b600f688539c55"
```

ws://localhost:9001/ws

Send

```
{
  "op": "blocks"
}
```

```
{
  "status": 200,
  "error": "OK",
  "response": {
    "blocks": []
  }
}
```

ws://localhost:9000/ws

Send

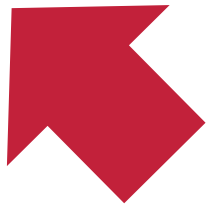
```
{
  "op": "register",
  "args": {
    "addr": ["ws://localhost:9001/ws", "ws://localhost:9002/ws"]
  }
}
```

```
{
  "status": 202,
  "error": "Accepted"
}
```

ws://localhost:9000/ws

Send

```
{
  "op": "peers"
}
```



```
{
  "status": 200,
  "error": "OK",
  "response": {
    "peers": ["ws://localhost:9000/ws", "ws://localhost:9002/ws", "ws://localhost:9001/ws"]
  }
}
```

ws://localhost:9001/ws

Send

```
{
  "op": "blocks"
}
```


```
{
  "status": 200,
  "error": "OK",
  "response": {
    "blocks": [{
      "items": [{
        "id": "00000000-0000-0000-0000-000000000003",
        "sender": "00000000-0000-0000-0000-000000000000",
        "receiver": "00000000-0000-0000-0000-000000000003",
        "amount": 1000,
        "t": 1620813710.01229,
        "prev_hash": null
      }, {
        "id": "00000000-0000-0000-0000-000000000001",
        "sender": "00000000-0000-0000-0000-000000000000",
        "receiver": "00000000-0000-0000-0000-000000000001",
        "amount": 3000,
        "t": 1620813722.948254,
        "prev_hash": "81ba993aa21a5ecd197f400ae29b517d650369eb"
```

ws://localhost:9000/ws

Send

```
{
  "op": "merge",
  "args": {
    "pool": [{
      "id": "00000000-0000-0000-0000-000000000005",
      "sender": "00000000-0000-0000-0000-000000000000",
      "receiver": "00000000-0000-0000-0000-000000000005",
      "amount": 5000
    }]
  }
}
```

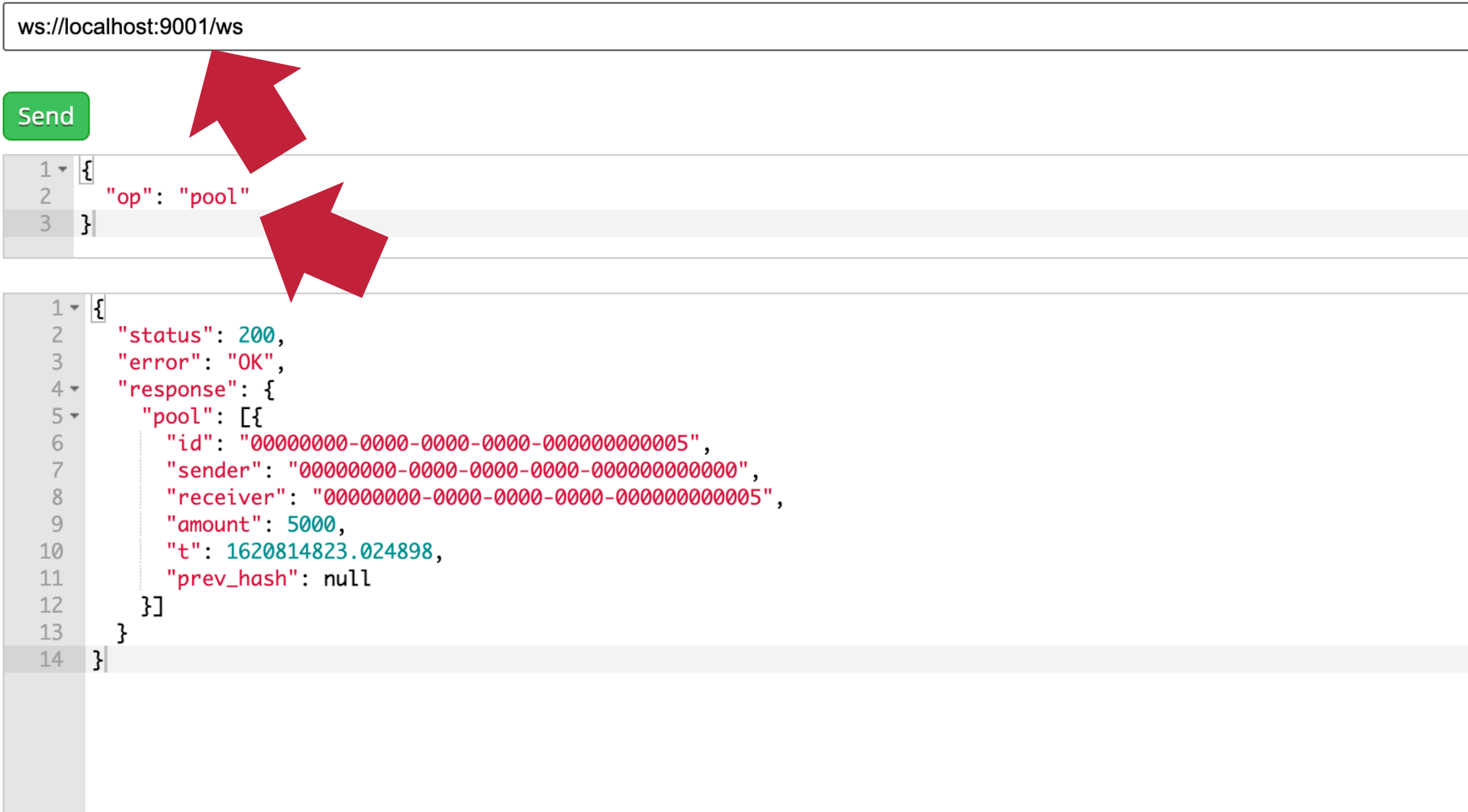
```
{
  "status": 202,
  "error": "Accepted"
}
```

ws://localhost:9000/ws

Send

```
{
  "op": "pool"
}
```

```
{
  "status": 200,
  "error": "OK",
  "response": {
    "pool": [{
      "id": "00000000-0000-0000-0000-000000000005",
      "sender": "00000000-0000-0000-0000-000000000000",
      "receiver": "00000000-0000-0000-0000-000000000005",
      "amount": 5000,
      "t": 1620814822.199175,
      "prev_hash": null
    }]
  }
}
```


ws://localhost:9001/ws
Send
{
"op": "pool"
}
{
"status": 200,
"error": "OK",
"response": {
"pool": [{
"id": "00000000-0000-0000-0000-000000000005",
"sender": "00000000-0000-0000-0000-000000000000",
"receiver": "00000000-0000-0000-0000-000000000005",
"amount": 5000,
"t": 1620814823.024898,
"prev_hash": null
}]
}
}

作业

- 参考实验教程，实现一个支持共识的分布式账本
- 在 3 个以上节点的 P2P 网络中测试分布式账本

- 提供完整的测试代码及截图

作业

- 在作业系统中下载并完成本实验课对应实验报告
- https://hw.css.dgut.edu.cn/
- 注意：所有标识为 * 的地方都需要填写
- 请务必在截止时间之前提交实验报告

课程名称：区块链技术与应用　　学期：2021 年春季

实验名称	分布式账本			实验序号	1
姓　名	***	学　号	***	班　级	***
实验地点	***	实验日期	***	指导老师	丁烨
教师评语	-			实验成绩	-
				百分制	100
同组同学					

四、　实验作业及分析

4.1　实验过程

1) *** 请将详细实验过程填写在此处 ***

4.2　实验结果

*** 请将实验结果截图填写在此处 ***

五、　实验总结

*** 请撰写一段 200 字左右的实验总结 ***

实验课安排

- 实验六：期末大作业
- 2021-12-01
- 10:25-12:00
- 9A-206-1

- 请务必自带电脑
- 电脑需要预装好 Python 3.10


GOOD
LUCK!