東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

区块链技术与应用

v0.10.18

第四章：P2P 网络（上）

丁烨，网络空间安全学院 副教授
dingye@dgut.edu.cn

目录

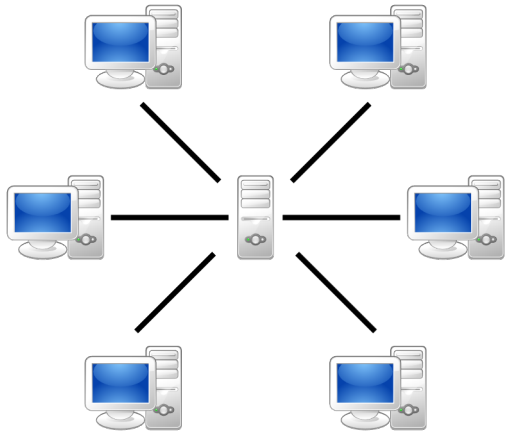
P2P 网络概述

P2P 网络技术简介

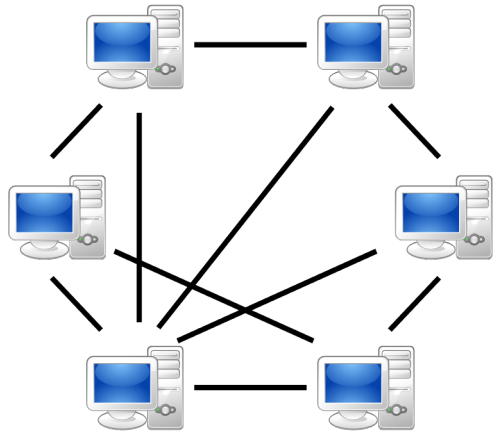
- 对等式网络（Peer-to-peer，P2P）
- 又称点对点技术
- 无中心服务器、依靠用户群（Peers）交换信息的互联网体系
- P2P 网络可以降低因中心节点出错而导致资料丢失的风险
- 与有中心服务器的中央网络系统不同，对等网络的每个用户端既是一个节点，也有服务器的功能，任何一个节点无法直接找到其他节点，必须依靠其户群进行信息交流

P2P 网络概述

P2P 网络技术简介



中心式网络



P2P 网络

P2P 网络概述

BitTorrent

- BitTorrent
- https://www.bittorrent.com/
- 由美国程序员 Bram Cohen 于 2001 年 4 月发布
- 2001 年 7 月 2 日首次正式应用

P2P 网络概述

BitTorrent

- 根据 BitTorrent 协议，文件发布者会根据要发布的文件生成提供一个 .torrent 文件，即种子文件，也简称为“种子”
- 种子文件本质上是文本文件，包含 Tracker 信息和文件信息两部分
- Tracker 信息主要是 BitTorrent 下载中需要用到的 Tracker 服务器的地址和针对 Tracker 服务器的设置
- 文件信息是目标文件的 Hash
- 所以，种子文件就是被下载文件的索引

P2P 网络概述

BitTorrent

- 下载时， BitTorrent 客户端首先解析种子文件得到 Tracker 地址，然后连接 Tracker 服务器，Tracker 服务器回应下载者的请求，提供下载者其他下载者（包括发布者）的 IP
- 下载者再连接其他下载者，根据种子文件，两者分别告知对方自己已经有的文件块，然后交换双方没有的数据
- 下载者每得到一个文件块，需要将文件块的 Hash 与种子文件中对应的 Hash 对比，如果一样则说明数据正确，不一样则需要重新下载这个文件块

P2P 网络概述

BitTorrent

- 磁力链接（Magnet URI Scheme）
- 将种子文件存储在 Tracker 服务器中，使用 Hash 直接从 Tracker 服务器中获取种子文件
- 磁力链接可以避免种子文件过大造成的数据污染问题

```
magnet:?xt=urn:btih:d21f8d9d004d99c2463acb4b3325495bade38693
```

P2P 网络概述

BitTorrent

- 区块链也可以使用 BitTorrent 协议来实现
- 由于区块链可以通过“链”来自行校验
- 大部分时候无需使用“种子文件”

P2P 网络概述

P2P 网络的必要性

- 无中心的 P2P 网络可以防止中心服务器数据污染
- 从而达到区块链的安全性目的
- 真实的区块链系统中，P2P 网络协议也算“共识”的一种

P2P 网络概述

P2P 网络的必要性

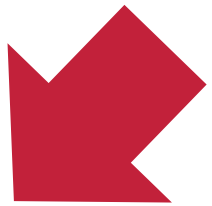
- P2P 网络的一个重要的目标就是让所有的客户端都能提供资源
- 包括带宽，存储空间和计算能力
- 因此，当有更多的节点加入时，整个系统的容量也随之增大
- 这是具有一组固定服务器的 Client-Server 结构不能实现的

目录

实现简单的 P2P 区块链网络

简易 P2P 节点

- app.py



```
import argparse
import json
import logging
import pickle

from redislite import Redis
from tornado import websocket, web, ioloop
from wrenchbox.logging import setup_log

DEFAULTS = {'port': 9000}
clients = []
db = None
```

实现简单的 P2P 区块链网络

简易 P2P 节点

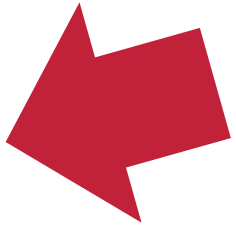
- `argparse`
- https://docs.python.org/3/library/argparse.html
- 一个优雅的命令行参数处理工具

实现简单的 P2P 区块链网络

简易 P2P 节点

- app.py

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--debug',
        action='store_true',
        default=False, help='show debug information'
    )
```



实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

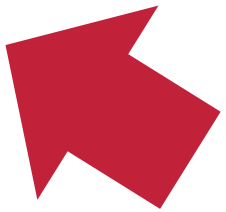
```
if __name__ == '__main__':
    ...
    parser.add_argument(
        '-p', '--port',
        type=int,
        default=DEFAULTS['port'],
        help='listening port, default: {}'.format(DEFAULTS['port'])
    )
```

实现简单的 P2P 区块链网络

简易 P2P 节点

- app.py

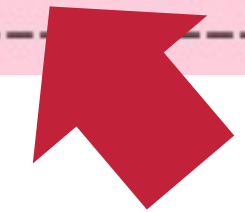
```
if __name__ == '__main__':
    ...
    parser.add_argument(
        '-r', '--redis',
        type=str,
        default='redis.db',
        help='redis database file, default: redis.db'
    )
    args, _ = parser.parse_known_args()
    print(args)
```



实现简单的 P2P 区块链网络

简易 P2P 节点

```
------------------------------------------------------------------------
~/Documents/Workspace/foxchain(master) » python3 -u -m foxchain.app.app
Namespace(debug=False, port=9000, redis='redis.db')
------------------------------------------------------------------------
```



实现简单的 P2P 区块链网络

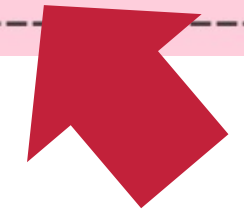
简易 P2P 节点

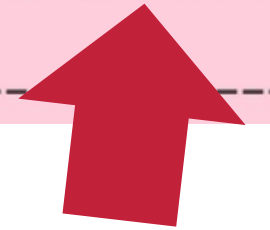
```
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.app -h
usage: app.py [-h] [--debug] [-p PORT] [-r REDIS]

optional arguments:
  -h, --help            show this help message and exit
  --debug               show debug information
  -p PORT, --port PORT  listening port, default: 9000
  -r REDIS, --redis REDIS
                        redis database file, default: redis.db
```

实现简单的 P2P 区块链网络

简易 P2P 节点

```
--------------------------------------------------------------------------------
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.app -p 3000 -r test.db
Namespace(debug=False, port=3000, redis='test.db')
--------------------------------------------------------------------------------
```





实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

```
import argparse
import json
import logging
import pickle

from redislite import Redis
from tornado import websocket, web, ioloop
from wrenchbox.logging import setup_log

DEFAULTS = {'port': 9000}
clients = []
db = None
```

实现简单的 P2P 区块链网络

简易 P2P 节点

- `pickle`
- https://docs.python.org/3/library/pickle.html
- 将 Python 的对象序列化与反序列化

实现简单的 P2P 区块链网络

简易 P2P 节点

- 序列化（Serialization）
- 将数据结构或对象状态转换成可保存的二进制格式（例如存成文件，存于缓冲，或经由网络发送）
- 以留待后续在相同或另一台计算机环境中，能恢复原先状态的过程

实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

```
import argparse
import json
import logging
import pickle

from redislite import Redis
from tornado import websocket, web, ioloop
from wrenchbox.logging import setup_log

DEFAULTS = {'port': 9000}
clients = []
db = None
```

实现简单的 P2P 区块链网络

简易 P2P 节点

- Redis
- https://redis.io/
- Redis 是一个使用 ANSI C 编写的开源、支持网络、基于内存、分布式、可选持久性（Durability）的键值对存储数据库
- Redis 目前是最流行的键值对存储数据库

实现简单的 P2P 区块链网络

简易 P2P 节点

- Redislite
- https://github.com/yahoo/redislite
- 由 Yahoo 开发的 Redis 的 Python 精简版
- 无需安装 Redis，直接在内存中创建类似 Redis 的服务
- 支持数据持久化

实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

```
if __name__ == '__main__':
    ...
    setup_log(level=logging.DEBUG if args.debug else logging.INFO)
    db = Redis(args.redis)
    if b'peers' not in db.keys():
        db.set('peers', pickle.dumps(set([])))
```

实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

```
import argparse
import json
import logging
import pickle

from redisli    port Redis
from tornado import websocket, web, ioloop
from wrenchbox.logging import setup_log

DEFAULTS = {'port': 9000}
clients = []
db = None
```

实现简单的 P2P 区块链网络

简易 P2P 节点

- Tornado
- https://www.tornadoweb.org/
- 全称 Tornado Web Server
- 用 Python 语言写成的 Web 服务器兼 Web 应用框架
- 由 FriendFeed 公司在自己的网站 FriendFeed 中使用，被 Facebook 收购以后以开源软件形式开放给大众
- Tornado 对 WebSocket 异步双向通信的支持性非常好

实现简单的 P2P 区块链网络

简易 P2P 节点

- WebSocket
- 一种网络传输协议，可在单个 TCP 连接上进行全双工通信，位于 OSI 模型的应用层，在 2011 年由 IETF 标准化为 RFC 6455，后由 RFC 7936 补充规范
- WebSocket 使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据
- 在 WebSocket 中，浏览器和服务器只需要完成一次握手，两者之间就可以创建持久性的连接，并进行双向数据传输

实现简单的 P2P 区块链网络

简易 P2P 节点

- app.py

```
if __name__ == '__main__':
    ...
    web.Application([
        (r'/', IndexHandler),
        (r'/ws', SocketHandler)
    ]).listen(args.port)
    logging.info('Tornado is listening on port: %d', args.port)
    ioloop.IOLoop.instance().start()
```

实现简单的 P2P 区块链网络

简易 P2P 节点

- app.py

```
class IndexHandler(web.RequestHandler):
    def get(self):
        self.render("index.html")
```

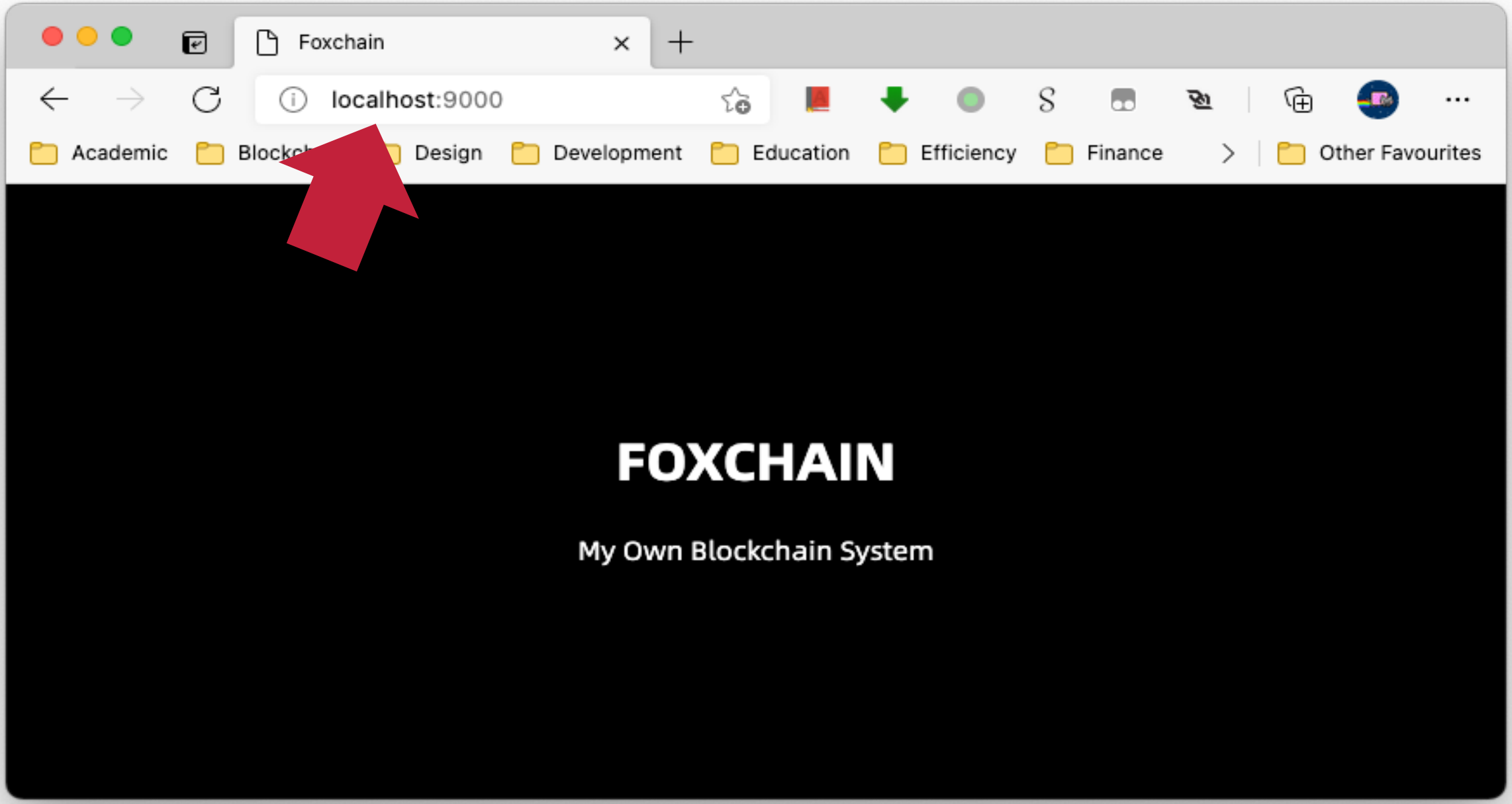
- index.html

```
<div class="container">
    <h1 class="pixel">FOXCHAIN</h1>
    <p>My Own Blockchain System</p>
</div>
```

实现简单的 P2P 区块链网络

简易 P2P 节点

```
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.app
[2021-04-01 17:10:26,386] Tornado is listening on port: 9000
```


Foxchain
localhost:9000
Academic
Design
Development
Education
Efficiency
Finance
Other Favourites
FOXCHAIN
My Own Blockchain System

实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

```
class SocketHandler(websocket.WebSocketHandler):
    def check_origin(self, origin):
        return True

    def open(self):
        logging.info('Client connected: %s', self.request.remote_ip)
        if self not in clients:
            clients.append(self)

    def on_close(self):
        if self in clients:
            clients.remove(self)
```

实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

```
def on_message(self, message):
    try:
        message = json.loads(message)
    except json.JSONDecodeError:
        logging.warning('Cannot parse request message: %s', message)
        self.write_message(json.dumps({
            'status': 500,
            'error': 'Cannot parse request message.',
            'response': None
        }))
    else:
        ...
```

实现简单的 P2P 区块链网络

简易 P2P 节点

- app.py

```
if message is not None:
    if 'op' in message:
        if message['op'] == 'register':
            if 'args' in message and 'addr' in message['args']:
                ...
            else:
                ...
        elif message['op'] == 'peers':
            ...
        else: ...
    else: ...
```

实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

```
if 'args' in message and 'addr' in message['args']:
    peers = pickle.loads(db.get('peers'))
    if not isinstance(message['args']['addr'], list):
        message['args']['addr'] = [str(message['args']['addr'])]
    for addr in message['args']['addr']:
        if addr.startswith('ws://') or addr.startswith('wss://'):
            peers.add(addr)
    db.set('peers', pickle.dumps(peers))
    self.write_message(json.dumps({
        'status': 202,
        'error': 'Accepted'
    }))
```

实现简单的 P2P 区块链网络

简易 P2P 节点

❖ app.py

```
if 'args' in message and 'addr' in message['args']:
    ...
else:
    self.write_message(json.dumps({
        'status': 500,
        'error': 'Operation "register" requires the following "args": "addr"',
        'response': None
    }))
```

实现简单的 P2P 区块链网络

简易 P2P 节点

- app.py

```
elif message['op'] == 'peers':
    self.write_message(json.dumps({
        'status': 200,
        'error': 'OK',
        'response': {'peers': list(pickle.loads(db.get('peers')))}
    }))
else:
    self.write_message(json.dumps({
        'status': 404,
        'error': 'Operation "{}" is not supported.'.format(message['op']),
        'response': None
    }))
```

实现简单的 P2P 区块链网络

简易 P2P 节点

- app.py

```
else:
    logging.warning('Message body is not supported: %s', message)
    self.write_message(json.dumps({
        'status': 500,
        'error': 'Message body is not supported.',
        'response': None
    }))
```

实现简单的 P2P 区块链网络

简易 P2P 节点

```
~/Documents/Workspace/foxchain(master*) » python3 -u -m foxchain.app.app
[2021-04-01 17:10:26,386] Tornado is listening on port: 9000
```

实现简单的 P2P 区块链网络

测试 P2P 节点

- WebSocket 客户端
- http://www.websocket-test.com/
- https://chrome.google.com/webstore/detail/smart-websocket-client/omalebghpgejjiaoknljcfmglgbpocdp

HISTORY

Clear

- Request #72
- Request #71
- Request #70
- Request #69
- Request #68
- Request #67
- Request #66

Everything is ok, ready to go!

ws://localhost:9000/ws

Disconnect

Send

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

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

HISTORY

Clear

- Request #73
- Request #72
- Request #71
- Request #70
- Request #69
- Request #68
- Request #67
- Request #66

Everything is ok, ready to go!

ws://localhost:9000/ws

Disconnect

Send

```
{
  "op": "register",
  "args": {
    "addr": "ws://localhost:9000/ws"
  }
}
```

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

HISTORY

Clear

- Request #74
- Request #73
- Request #72
- Request #71
- Request #70
- Request #69
- Request #68
- Request #67
- Request #66

Everything is ok, ready to go!

ws://localhost:9000/ws

Disconnect

Send

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

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

实现简单的 P2P 区块链网络

实验课安排

- 实验二：P2P 网络（上）
- 2021-11-05
- 10:25-12:00
- 9A-206-1

- 请务必自带电脑
- 电脑需要预装好 Python 3.9

总结

- 袁煜明《区块链技术进阶指南》，机械工业出版社
- http://product.dangdang.com/28538836.html

- 安德烈亚斯·安东诺普洛斯《精通区块链编程：加密货币原理、方法和应用开发》第二版，机械工业出版社
- http://product.dangdang.com/27877333.html

总结

- Bitcoin 比特币
- https://bitcoin.org

- Ethereum 以太坊
- https://ethereum.org

東莞理工學院
DONGGUAN UNIVERSITY OF TECHNOLOGY

Thank You!

丁烨，网络空间安全学院 副教授
dingye@dgut.edu.cn