第四周周报

汪子龙

本周用 golang 实现了一个以太坊模块，用于链上交易的发起和查询，在此记录一下学习的过程和遇到的一些问题。

一、环境配置

1. 安装 geth

Geth 是由以太坊基金会提供的官方客户端软件，用 Go 编程语言编写。Geth 提供了一个交互式命令控制台，通过命令控制台中包含了以太坊的各种功能。从官网上下载 Geth，并配置环境变量，即可通过命令行实现相关控制。

2. 创建私链

在私链目录下创建一个初始块配置文件 genesis.json：

```
{
  "gasLimit": "0x8000000",
  "difficulty": "0x20000",
  "alloc": {},
  "config": {
    "chainId": 20,
    "homesteadBlock": 0,
    "epic155Block": 0,
    "epic158Block": 0
  }
}
```

命令行进入该目录中，输入命令：

```
geth --datadir data init genesis.json
```

目录下 data 文件夹中会自动生成区块链信息文件。

3. 启动私链

私链目录下进入命令行，输入命令：

```
Geth --datadir data --networkid 20 --rpc --rpcapi "db,eth,net,web3,personal" --rpcaddr localhost --rpcport "8545" console
```

各参数解释如下：

--networkid	指定链 id
--rpc	开启 rpc
--rpcapi	指定需要调用 rpc 服务的 API 接口
--rpcaddr	指定 rpc 服务地址
--rpcport	指定 rpc 服务监听端口
console	命令行进入私链控制模式

4. 创建账户

在 console 模式下，使用以下命令创建新的账户：

```
personal.newAccount("password")
```

我创建了两个账户，用于后续发起交易，使用以下命令可以查看所有账户的：

```
personal.listAccounts
```

```
> personal.listAccounts
["0×fcd4098b2e815d45223ee1fcf40e115302dd90c5", "0×e78585b2903a0c606d768cfa05f
f84af8fddb8a0"]
```

5. 挖矿

在 console 模式下，使用以下指令进行挖矿（挖矿成功一次后停止）：

```
miner.start(2);admin.sleepBlocks(1);miner.stop()
```

挖矿成功后，可以使用以下命令查看余额：

```
eth.getBalance(eth.accounts[0])
```

```
> eth.getBalance(eth.accounts[0])
5000000000000000000
```

6. 安装 go-ethereum 库

使用 git clone，将 go-ethereum 库下载到 go 环境中。

下载成功后，调用库发现还需要很多其它依赖，用 go get ./...命令下载所有依赖。其中有些依赖被墙了，需要手动使用 git clone 从 github 上下载。另外部分依赖，需要 gcc 环境才能安装，于是又需要下载 mingw-w64 来配置 gcc 环境。

所有依赖都安装好后，即可成功调用 go-ethereum 库，开始以太坊模块的开发！

二、功能实现

golang 可以通过 github.com/ethereum/go-ethereum/ethclient 库与开启 rpc 的私链连接：

```
client, err := ethclient.Dial("http://localhost:8545")
```

上述语句会连接给定路径的私链，并得到一个 ethclient 对象用于管理。

1. 余额查询

使用以下语句先获取账户地址：

```
account := common.HexToAddress("0xfcd4098b2e815d45223ee1fcf40e115302dd90c5")
```

通过账户地址查询地址下的余额：

```
balance, err := client.BalanceAt(context.Background(), account, nil)
```

2. 通过区块 id 获取交易信息

通过 id 获取区块对象：

```
block, err := client.BlockByNumber(context.Background(), blockNumber)
```

使用 block.Transactions()获取区块内所有的交易，返回一个类型为 Transactions 的对象，for range 遍历对象即可得到每个交易的信息。

3. 通过交易哈希获取交易信息

通过哈希获取交易对象：

```
tx, isPending, err := client.TransactionByHash(context.Background(), txHash)
```

4. 发起交易

发起交易需要先获取发起人的私钥，需要输入 keystore 文件路径和账户的密码，然后通过以下语句解析得到私钥（注：此处 GetPrivateKey 函数需要单独编写，见第五点）：

```
var keyValue = GetPrivateKey(&privateKeyFile, &password)
privateKey, err := crypto.HexToECDSA(keyValue)
```

然后从私钥派生账户的公钥，映射到 ECDSA 数字签名：

```
publicKey := privateKey.Public()
publicKeyECDSA, ok := publicKey.(*ecdsa.PublicKey)
```

通过公钥获取账户的地址：

```
fromAddress := crypto.PubkeyToAddress(*publicKeyECDSA)
```

之后需要用户输入交易的具体信息，包括 nonce、toAddress、value、gasLimit 和 gasPrice，用下面语句生成一个新交易：

```
tx := types.NewTransaction(nonce, toAddress, value, gasLimit,
```

```
gasPrice, data)
```

用以下语句签署交易，需要提交链 id 和发起人私钥：

```
signedTx, err := types.SignTx(tx, types.NewEIP155Signer(chainID), privateKey)
```

最后将签署的交易发送：

```
client.SendTransaction(context.Background(), signedTx)
```

5. 获取私钥

先读取 keystore 文件的 json：

```
keyJSON, err := ioutil.ReadFile(*privateKeyFile)
```

通过 json 和账户密码得到加密后的私钥：

```
unlockedKey, err := keystore.DecryptKey(keyJSON, *password)
```

解析加密后私钥得到私钥：

```
privKey := hex.EncodeToString(unlockedKey.PrivateKey.D.Bytes())
```

三、结果演示

1. 获取账户余额

```
Please choose your option:
0: Check the balance of a certain account.      1: Check transactions in a certain block.
2: Check the tansaction with a certain hash.    3: Start a transaction.
4: Exit.
0
Please input your account address:
0xfcd4098b2e815d45223ee1fcf40e115302dd90c5
Balance:  10000000000000000000
```

2. 根据区块 id 获取交易

```
Please choose your option:
0: Check the balance of a certain account.      1: Check transactions in a certain block.
2: Check the tansaction with a certain hash.    3: Start a transaction.
4: Exit.
1
Please input the block id:
13
Transaction 0:
{
  Hash: 0x01e69308e69c1eb5300ca09220a491e0f639e24e4ebeebc0c2da7300fdc7594a,
  Value: 10000000000000000000,
  Gas: 90000,
  GasPrice: 18000000000,
  Nonce: 0,
  To: 0xe78585B2903A0c606D768CFA05Ff84AF8fdDb8a0,
  From: 0xfCd4098B2E815d45223EE1Fcf40E115302dd90C5,
  Status: 0,
}
```

3. 根据交易哈希获取交易

```
Please choose your option:
0: Check the balance of a certain account.      1: Check transactions in a certain block.
2: Check the tansaction with a certain hash.    3: Start a transaction.
4: Exit.
2
Please input the hash of transation:
0x01e69308e69c1eb5300ca09220a491e0f639e24e4ebeebc0c2da7300fdc7594a
{
  Hash: 0x01e69308e69c1eb5300ca09220a491e0f639e24e4ebeebc0c2da7300fdc7594a,
  Value: 10000000000000000000,
  Gas: 90000,
  GasPrice: 18000000000,
  Nonce: 0,
  To: 0xe78585B2903A0c606D768CFA05Ff84AF8fdDb8a0,
  From: 0xfCd4098B2E815d45223EE1Fcf40E115302dd90C5,
  IsPending: false,
}
```

4. 发起交易

```
Please choose your option:
0: Check the balance of a certain account.      1: Check transactions in a certain block.
2: Check the tansaction with a certain hash.    3: Start a transaction.
4: Exit.
3
Please input your key file path:
D:\Projects\GethProgram\data\keystore\UTC--2020-07-25T17-56-33.557856600Z--fcd4098b2e815d45223ee1fcf40e115302dd90c5
Please input your password:
tredson
(Transaction Config) Please input nonce (if skipped, nonce will be set as the default):

(Transaction Config) Please input value:
10000000000000000000
(Transaction Config) Please input gas limit:
90000
(Transaction Config) Please input gas price:
20000000000
(Transaction Config) Please input recipient account address:
0xe78585B2903A0c606D768CFA05Ff84AF8fdDb8a0
Transaction has been sent, hash:  0x354fc72b328eb1b16c1c08a056d9361f521acb33e86e867677ad64345d6c5b51
```