**MATRX 定时交易设计方案**

**版本: V1.0**

**作者：MATRIX 基金会北京研究院**

**日期：2018.07.25**

任何情况下，与本系统及其衍生产品，以及与之相关的全部文件（包括本文件及其任何附件中的全部信息）相关的全部知识产权（包括但不限于版权、商标和技术秘密）皆属于 MATRIX 基金会所有。

本文件中的信息是保密的，且仅供用户指定的接收人内部使用。未经 MATRIX 基金会事先书面同意，本文件的任何用户不得对本产品和本文件中的信息向任何第三方（包括但不限于除用户指定接收人以外的管理人员、员工和关联公司）进行披露、出借、许可、转让、出售、分发、传播或进行与本产品和本文件相关的任何其他处置，也不得使该等第三方以任何形式使用本产品和本文件中的信息。

未经 MATRIX 基金会事先书面允许，不得为任何目的、以任何形式或任何方式对本文件进行复制、修改或分发。本文件的任何用户不得更改、移除或损害本文件所使用的任何商标。

本文件按“原样”提供，就本文件的正确性、准确性、可靠性或其他方面，MATRIX 基金会并不保证本文件的使用或使用后果。本文件中的全部信息皆可能在没有任何通知的情形下被进一步修改，MATRIX 基金会对本文件中可能出现的任何错误或不准确之处不承担任何责任。

在任何情况下，MATRIX 基金会均不对任何因使用本软件产品和本文件中的信息而引起的任何直接损失、间接损失、附带损失、特别损失或惩罚性损害赔偿（包括但不限于获得替代商品或服务、丧失使用权、数据或利润；或商业中断），责任或侵权（包括过失或其他侵权）承担任何责任，即使 MATRIX 基金会事先获知上述损失可能发生。

MATRIX 基金会保留所有权利

# 目 录

[目 录 1](#_Toc3193)

[变更记录 3](#_Toc31518)

[1 引言 4](#_Toc2876)

[1.1 文档用途 4](#_Toc29741)

[1.2 术语 4](#_Toc9515)

[1.3 使用对象 4](#_Toc26543)

[2 设计说明 5](#_Toc5344)

[2.1 定时交易 5](#_Toc3972)

[2.1.1 需求描述 5](#_Toc14241)

[2.1.2 设计内容 5](#_Toc23328)

[2.1.3 流程图 10](#_Toc12622)

[2.1.4 涉及的接口 11](#_Toc28724)

[2.1.5 相关逻辑 11](#_Toc9437)

[2.1.6 参数配置 11](#_Toc20931)

[3 测试用例 14](#_Toc19875)

[3.1 用例名称（定时交易状态测试\_01） 14](#_Toc26811)

[3.2 用例名称（定时交易状态测试\_02） 15](#_Toc26619)

[3.3 用例名称（定时交易参数测试\_01） 16](#_Toc17039)

[3.4 用例名称（定时交易参数测试\_02） 17](#_Toc3082)

[3.5 用例名称（定时交易容量测试） 17](#_Toc15715)

# 变更记录

| 修订日期 | 设计内容 | 作者 | 评审者 | 评审日期 |
| --- | --- | --- | --- | --- |
| 2018-07-18 | 交易定时发送 | 郑贺 |  |  |
| 2018-07-24 | 定时交易 | 郑贺 |  |  |

# 引言

## 文档用途

本文档针对定时交易的实现做了详细说明，目的是让研发人员快速了解定时交易实现逻辑。

## 术语

| **术语** | **备注** |
| --- | --- |
| 定时交易 | 指定在固定区块高度上线该交易 |

## 使用对象

阅读对象主要针对熟悉相关业务、技术和应用知识的人员，包括：研究院全体同事

# 设计说明

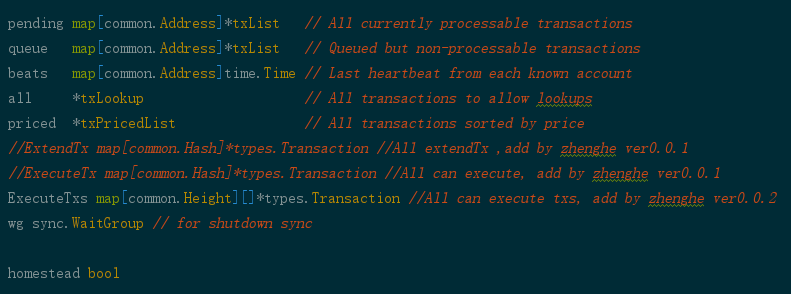
## 定时交易

### 需求描述

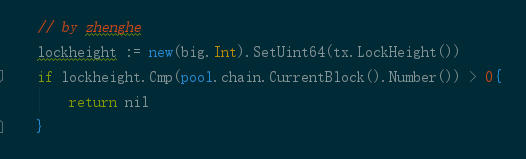
定时交易需要在交易发出时对该笔交易做简单的验证（数据大小、签名正确性）后便可以对该交易打包上链并扣除gas费用，但在指定区块高度到达后真正执行交易（from账户扣款，to账户增加同样的金额）。

### 设计内容

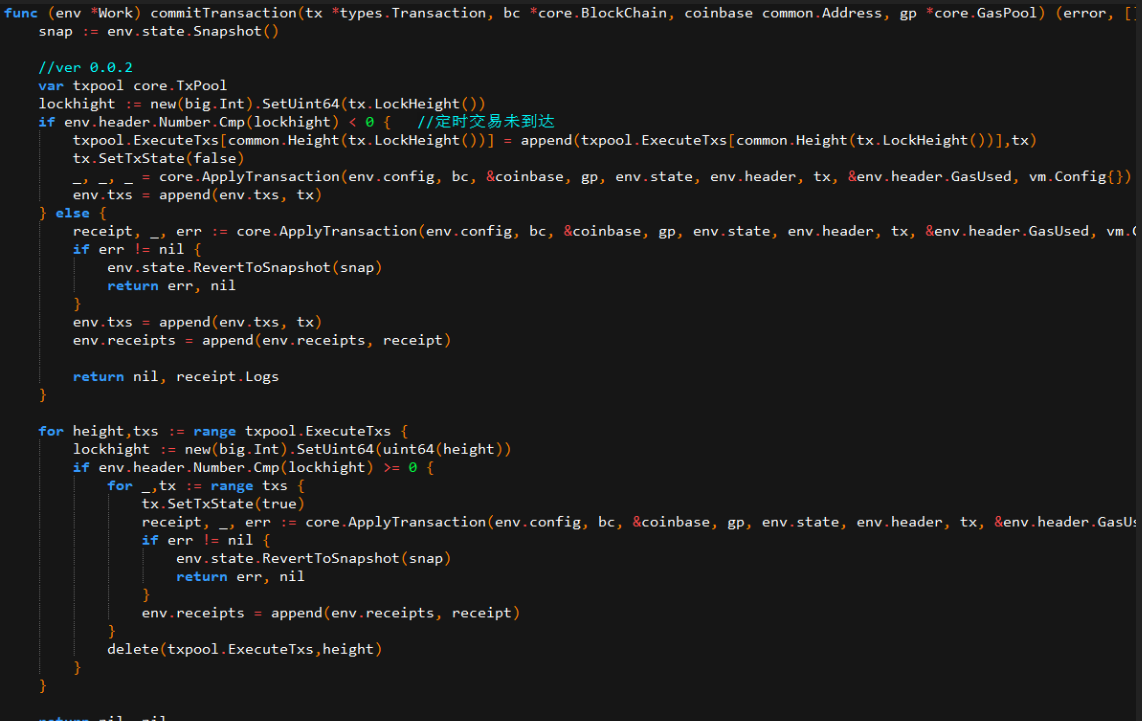
在交易池结构体中增加定时交易状态机，区块高度作为key值，value为当前高度的交易集合。



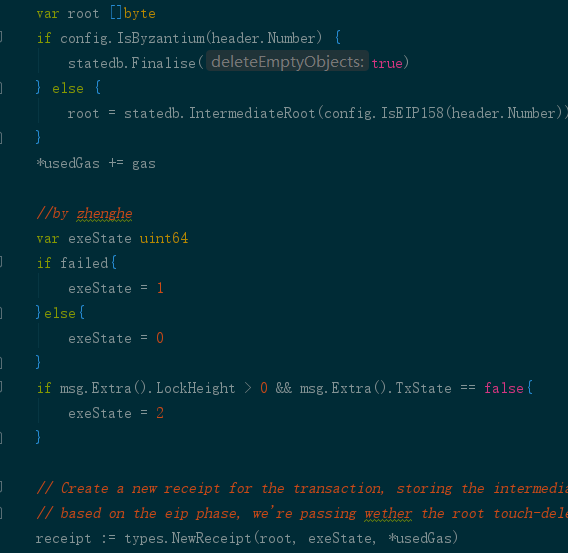
在交易验证时，对定时交易只做简单验证即可入池（ValidateTx方法）。

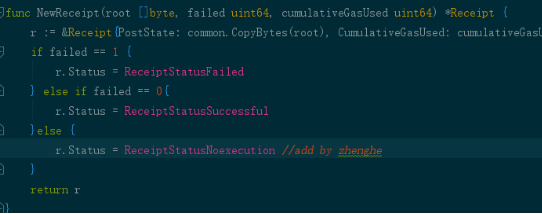


在交易提交时，如果是普通交易或者定时交易高度小于等于当前区块高度的交易则先执行，如果是定时未到达的交易，则添加到定时交易状态机里，遍历状态机里的交易，如果到达执行高度，则执行，执行完毕后从状态机里删除（commitTransaction方法）。

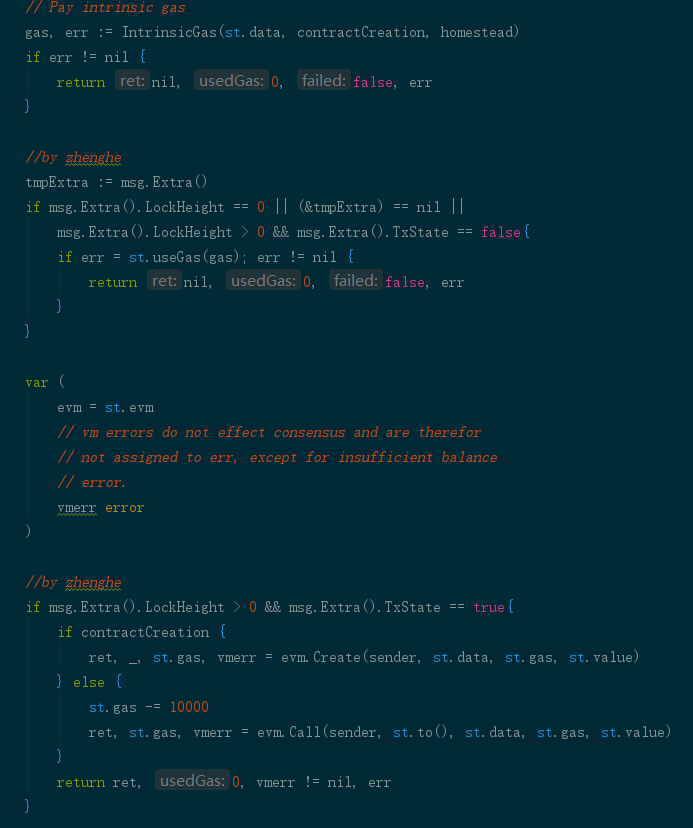


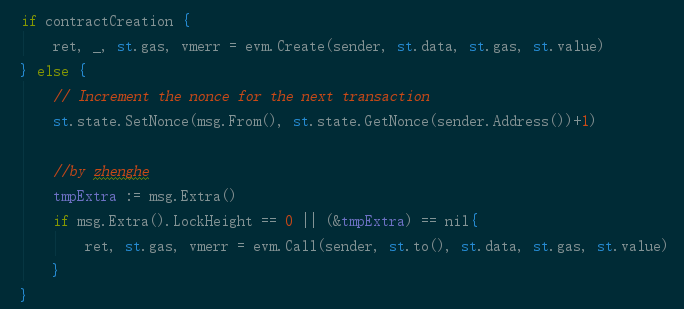
在接收者交易状态中增加未执行状态（原来是执行成功和失败两种状态NewReceipt方法）。



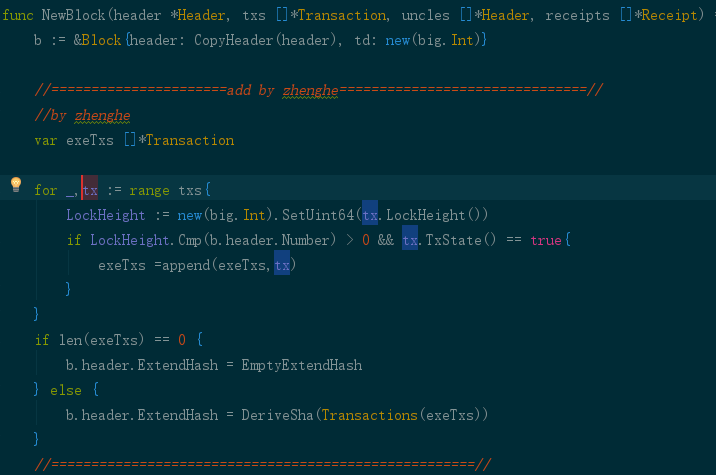


在交易执行时先扣除gas费用不扣除账户余额，等到定时交易可执行时（先扣除10000gas）不再重复扣除gas，只扣除账户金额。





在区块头新增定时交易Mekel trie，当某区块有定时交易执行时，newBlock时改变这棵树的状态



### 流程图



### 涉及的接口

1. type TxPool struct{}
2. func NewTxPool()
3. func (pool \*TxPool) addTx
4. func (pool \*TxPool) validateTx
5. func (self \*worker) wait()
6. func (env \*Work) commitTransaction()
7. func ApplyTransaction()
8. func NewReceipt()
9. func (st \*StateTransition) TransitionDb()
10. func NewBlock()

### 相关逻辑

见流程图

### 参数配置

type Tx\_to struct{  
 Recipient \*common.Address `json:"to" rlp:"nil"` *// nil means contract creation* Amount \*big.Int `json:"value" gencodec:"required"`  
 Payload []byte `json:"input" gencodec:"required"`  
}  
type Matrix\_Extra struct{  
 TxType byte `json:"txType" gencodec:"required"`  
 LockHeight uint64 `json:"lockHeight" gencodec:"required"`  
 TxState bool `json:"State" gencodec:"required"`  
 ExtraTo []Tx\_to `json:"extra\_to" gencodec:"required"`  
}  
  
type txdata struct {  
 AccountNonce uint64 `json:"nonce" gencodec:"required"`  
 Price \*big.Int `json:"gasPrice" gencodec:"required"`  
 GasLimit uint64 `json:"gas" gencodec:"required"`  
 Recipient \*common.Address `json:"to" rlp:"nil"` *// nil means contract creation* Amount \*big.Int `json:"value" gencodec:"required"`  
 Payload []byte `json:"input" gencodec:"required"`  
  
 *// Signature values* V \*big.Int `json:"v" gencodec:"required"`  
 R \*big.Int `json:"r" gencodec:"required"`  
 S \*big.Int `json:"s" gencodec:"required"`  
  
 *// This is only used when marshaling to JSON.* Hash \*common.Hash `json:"hash" rlp:"-"`  
 Extra []Matrix\_Extra ` rlp:"tail"` *//by zhenghe*}

# 测试用例

## 用例名称（定时交易状态测试\_01）

| **用例编号** | **定时交易起始状态\_01** | **用例类型** | | **功能测试** | |
| --- | --- | --- | --- | --- | --- |
| 用例描述 | 定时交易起始状态测试，要求定时交易（正确的交易）发送后即可在新区块里查到此交易 | | | | |
| 设计人 | 郑贺 | 设计日期 | |  | |
| 执行人 |  | 执行日期 | |  | |
| 关联的需求名称 | 定时交易 | 涉及UI | |  | | |
| 测试条件 | 3台docker节点 | | | | | |
| 输入数据 |  | | | | |
| 测试过程  测试步骤：   1. 启动3个挖矿节点开始挖矿。 2. 一台节点执行eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1],value: web3.toWei(0.01),txType:1,lockHeight:250)命令。   3.挖出新快时，可以在区块上查到该笔定时交易。  预期结果：   1. 可以通过区块链浏览器查到交易 2. 查看账户余额除了gas费用外没有扣除转账金额 | | | | | |
| 期望输出 |  | | 实际输出 | |  |
| 测试结果分析和建议 |  | | | | |

表格 1 [用例名称]

## 用例名称（定时交易状态测试\_02）

| **用例编号** | **定时交易执行状态\_02** | **用例类型** | | **功能测试** | |
| --- | --- | --- | --- | --- | --- |
| 用例描述 | 定时交易执行状态测试，要求定时交易在指定区块到达后转账金额扣除 | | | | |
| 设计人 | 郑贺 | 设计日期 | |  | |
| 执行人 |  | 执行日期 | |  | |
| 关联的需求名称 | 定时交易 | 涉及UI | |  | | |
| 测试条件 | 3台docker节点 | | | | | |
| 输入数据 |  | | | | |
| 测试过程  测试步骤：   1. 启动3个挖矿节点开始挖矿。   2.一台节点执行eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1],value: web3.toWei(0.01),txType:1,lockHeight:250)命令。  预期结果：  当前区块高度大于250时，查看from账户余额转账金额已扣除，to账户金额对应增加。 | | | | | |
| 期望输出 |  | | 实际输出 | |  |

表格 2 [用例名称]

## 用例名称（定时交易参数测试\_01）

| **用例编号** | **定时交易参数测试\_01** | **用例类型** | | **功能测试** | |
| --- | --- | --- | --- | --- | --- |
| 用例描述 | 定时交易参数测试，如果输入的定时执行高度小于当前区块链高度则该交易可执行 | | | | |
| 设计人 | 郑贺 | 设计日期 | |  | |
| 执行人 |  | 执行日期 | |  | |
| 关联的需求名称 | 定时交易 | 涉及UI | |  | | |
| 测试条件 | 3台docker节点 | | | | | |
| 输入数据 |  | | | | |
| 测试过程  测试步骤：   1. 启动3个挖矿节点开始挖矿。   2.一台节点执行eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1],value: web3.toWei(0.01),txType:1,lockHeight:10)命令，要求当前区块高度大于10。  预期结果：  当有新区块生成时可以查看到该交易，并且to账户金额增加。 | | | | | |
| 期望输出 |  | | 实际输出 | |  |

## 用例名称（定时交易参数测试\_02）

| **用例编号** | **定时交易参数测试\_02** | **用例类型** | | **功能测试** | |
| --- | --- | --- | --- | --- | --- |
| 用例描述 | 定时交易参数测试，如果输入的定时执行高度小于当前区块链高度则该交易可执行 | | | | |
| 设计人 | 郑贺 | 设计日期 | |  | |
| 执行人 |  | 执行日期 | |  | |
| 关联的需求名称 | 定时交易 | 涉及UI | |  | | |
| 测试条件 | 3台docker节点 | | | | | |
| 输入数据 |  | | | | |
| 测试过程  测试步骤：  1.启动3个挖矿节点开始挖矿。  2.一台节点执行eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1],value: web3.toWei(0.01),txType:1,lockHeight:10)命令，要求当前区块等于10。  预期结果：  当有新区块生成时可以查看到该交易，并且to账户金额增加。 | | | | | |
| 期望输出 |  | | 实际输出 | |  |

## 用例名称（定时交易容量测试）

| **用例编号** | **定时交易容量测试\_01** | **用例类型** | | **功能测试** | |
| --- | --- | --- | --- | --- | --- |
| 用例描述 | 定时交易容量测试，如果没执行的定时交易数量超过2000，则定时交易不可发送 | | | | |
| 设计人 | 郑贺 | 设计日期 | |  | |
| 执行人 |  | 执行日期 | |  | |
| 关联的需求名称 | 定时交易 | 涉及UI | |  | | |
| 测试条件 | 3台docker节点 | | | | | |
| 输入数据 |  | | | | |
| 测试过程  测试步骤：   1. 启动3个挖矿节点不启动挖矿。   2.一台节点执行eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[1],value: web3.toWei(0.01),txType:1,lockHeight:100)命令，要求当前区块小于100。  3. 连续执行2000条该执行  预期结果：  当定时交易超过2000后，交易会出错 | | | | | |
| 期望输出 |  | | 实际输出 | |  |