区块验证服务三级详细设计

# 服务目的

该服务负责对leader发来的区块验证请求进行验证并广播自己的验证结果（即投票结果），然后缓存请求和自己的投票结果。同时接受其他validator的投票结果，结合缓存的请求调用DPOS共识引擎接口，进行DPOS区块验证。

当满足共识时，调用接口向所有矿工广播带签名列表的header；同时向leader身份服务通知区块验证完成、向区块生成服务发送验证结果消息。

# 框架图



按服务目的要求，将服务划分为投票服务、结果收集及验证服务。其中投票服务主要负责区块验证请求的验证。结果收集及验证服务主要负责做DPOS共识的判断。

# 投票服务

## 3.1. 结构定义

type TxVerifier interface {

Verifier(txs []uint32) (error)

}

type voteResult struct {

reqMsg \*BlockVerifyReqMsg

signHash common.Hash

sign []byte

}

type BlockVote struct {

curIdentity int //当前身份

curLeader common.Address //当前leader

curNumber big.Int //当前高度

msgCenter \*event.TypeMux //消息中心，用于订阅消息 及消息发送

dposVerifyReqCh chan \*dposVerifyReq //内部通道，用于向投票收集及验证服务发送dpos请求

bc \*core.BlockChain //链，用于验证header

txVerifier TxVerifier //交易验证接口，用于验证交易列表（CA提供）

quitCh <-chan struct{} //服务停止消息通道，用于停止服务

}

## 3.2. 接口定义

### 3.2.1. 实例创建接口

func NewBlockVote(msgCenter \*event.TypeMux, bc \*core.BlockChain, txVerifier TxVerifier, dposVerifyReqCh chan \*dposVerifyReq, quitCh <-chan struct{}) (\*BlockVote, error)

说明：输入消息通道，区块链和交易验证接口，输出为服务实例或错误信息。

主要负责初始化服务需要的数据。

### 3.2.2. 服务运行、停止接口

func (self \*BlockVote) Start()

func (self \*BlockVote)Stop()

说明：负责启动、停止服务。

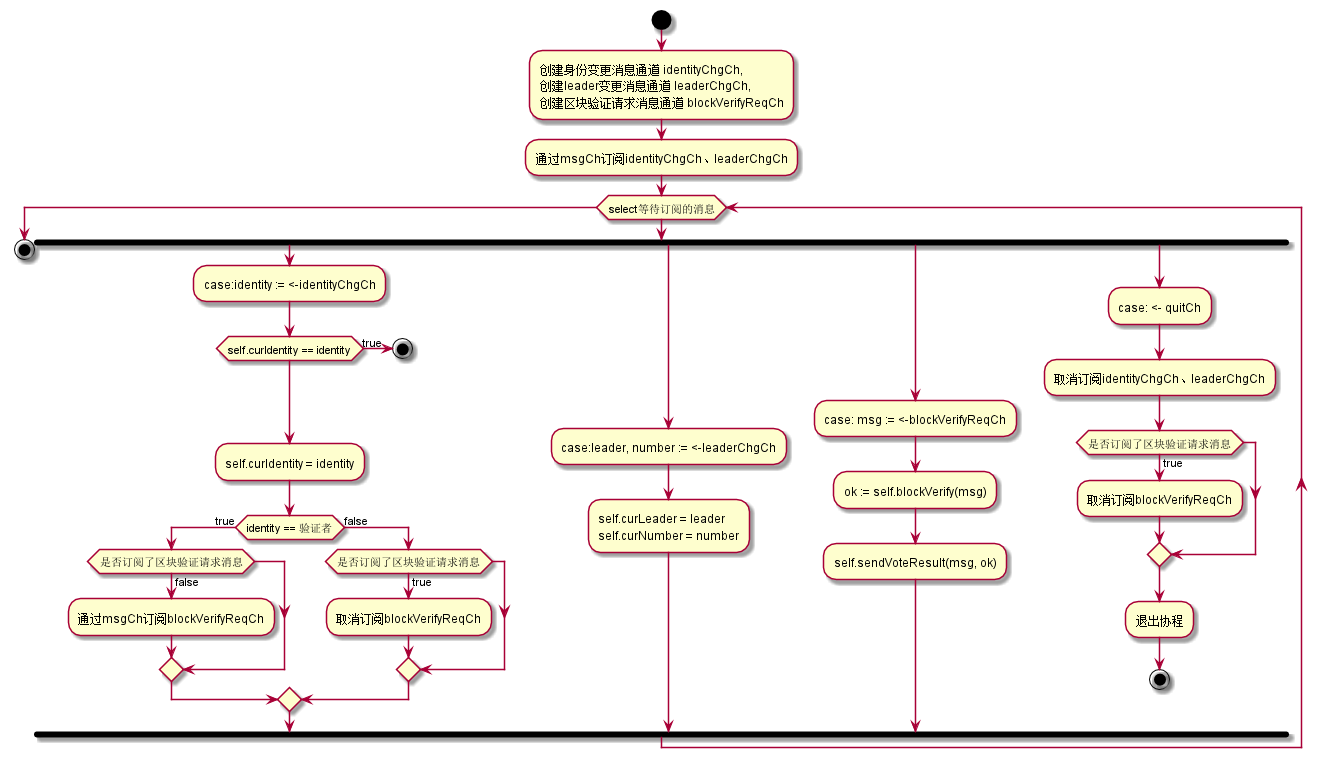
启动接口主要是开启消息监听协程。

停止接口则是通过quitCh，停止消息监听协程。

### 3.2.3. 消息监听协程

func (self \*BlockVote)update()

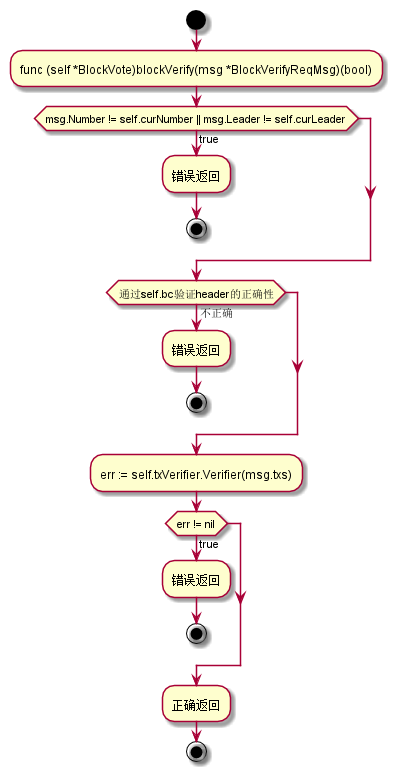
* 流程图:



### 3.2.4. 区块验证接口(内部接口)

func (self \*BlockVote)blockVerify(msg \*BlockVerifyReqMsg)(bool)

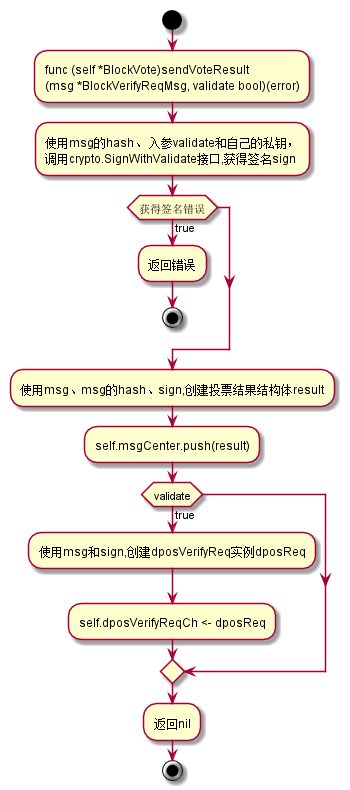
* 输入：区块验证请求消息结构体指针
* 输出：验证结果
* 流程图:



### 3.2.5. 发送投票结果接口(内部接口)

func (self \*BlockVote)sendVoteResult (msg \*BlockVerifyReqMsg, validate bool)(error)

* 输入：区块验证请求消息结构体指针、验证结果
* 输出：错误信息
* 流程图:



# 结果收集及验证服务

## 4.1. 结构定义

type voteInfo struct {

time uint64 // 时间戳，收到的时间

sign []byte // 签名

from common.Address // 来源

signHash common.Hash // 签名对应的msg的hash

}

type votePool struct {

voteMap map[common.Address]map[common.Hash]\*voteInfo // 投票缓存

// 缓存结构为：map <from, map <msgHash, \*data> >

timeIndex list.List // 按投票到来先后的索引，用于删除过期数据

timeoutInterval uint64 // 超时时间

eachFromVoteCountLimit uint32 // 每个用户的投票数量限制

}

type BlockDPOSVerify struct {

curIdentity int //当前身份

votePool \*votePool //票池缓存

curLeader common.Address //当前leader

curNumber big.Int //当前高度

msgCenter \*event.TypeMux //消息中心，用于订阅消息

dposVerifyReqCh <-chan \*dposVerifyReq//内部通道，用于向接受dpos请求

dposData \*dposVerifyReq //当前需要进行DPOS验证的数据

consensus DPOSConsensus //DPOS共识引擎

quitCh <-chan struct{} //退出通道

}

## 4.2. 接口定义

### 4.2.1. 票池接口

#### 创建票池接口

func newVotePool() (\*votePool)

说明：创建票池，其中超时时间和每个用户的投票数量限制使用params中的默认值，定义为3秒和5个投票限制。

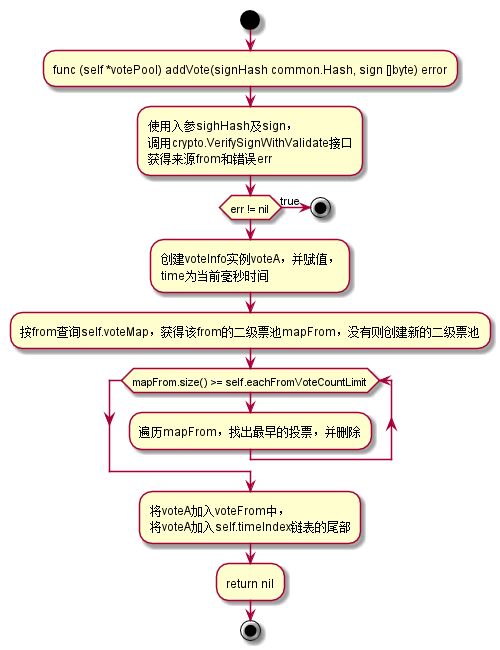
#### 增加投票接口

func (self \*votePool) addVote(signHash common.Hash, sign []byte) error

输入：签名对应msg的hash及签名数据

输出：错误信息

流程图：

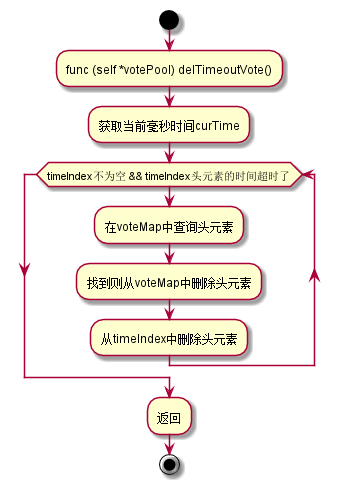


#### 删除过期投票接口

func (self \*votePool) delTimeoutVote()

说明：根据timeIndex的链表，删除过期的投票

流程图:



#### 获取投票列表接口

func (self \*votePool)getVotes(signHash common.Hash)[][]byte

输入：投票目标的signHash

输出：签名列表

说明：遍历voteMap，查询所有对signHash的sign

#### 删除投票接口

func (self \*votePool)delVotes(signHash common.Hash)

输入：投票目标的signHash

输出：无

说明：遍历voteMap，删除所有对signHash的sign

#### 重置缓存接口

func (self \*votePool) clear()

说明：清空voteMap及timeIndex

### 4.2.2. 服务接口

#### 实例创建接口

func NewBlockDPOSVerify (msgCenter \*event.TypeMux, dposVerifyReqCh <-chan \*dposVerifyReq, consensus DPOSConsensus, quitCh <-chan struct{}) (\*BlockDPOSVerify, error) {

说明：负责创建实例，并初始化

#### 服务运行、停止接口

func (self \* BlockDPOSVerify) Start()

func (self \* BlockDPOSVerify) Stop()

说明：负责启动、停止服务。

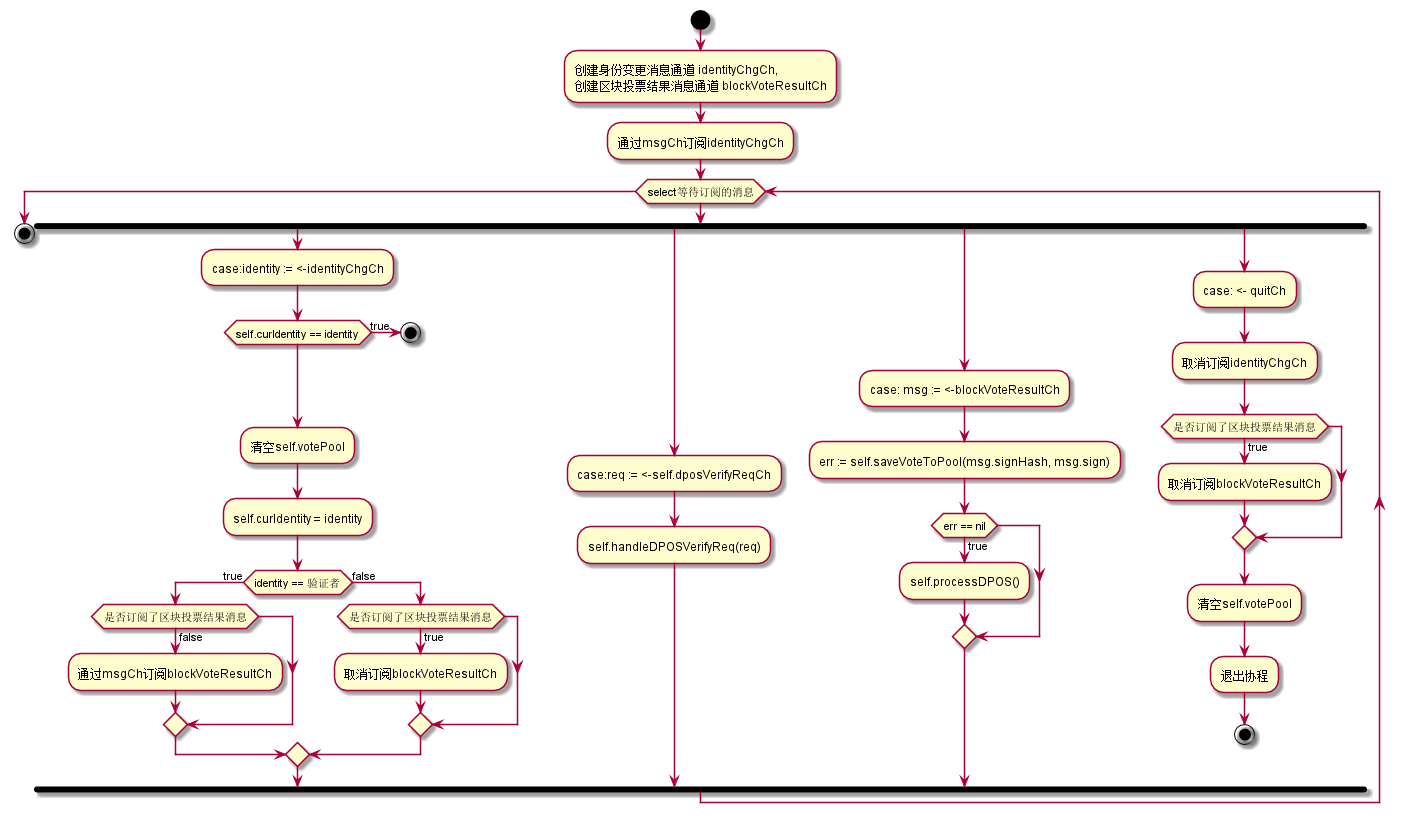
启动接口主要是开启消息监听协程。

停止接口则是通过quitCh，停止消息监听协程。

#### 消息监听协程

func (self \* BlockDPOSVerify)update()

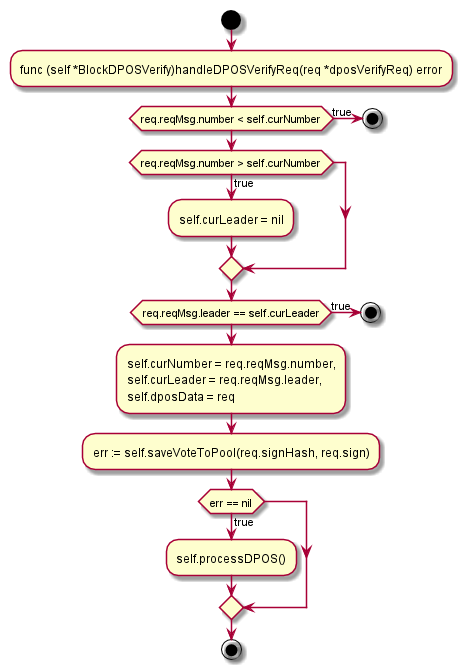
流程图:



#### DPOS共识请求消息处理接口(内部接口)

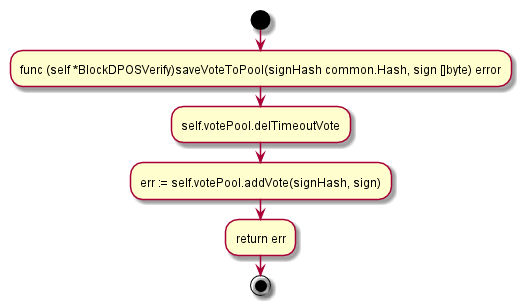
func (self \*BlockDPOSVerify)handleDPOSVerifyReq(req \*dposVerifyReq) error

流程图：



#### 投票结果存入票池接口(内部接口)

func (self \*BlockDPOSVerify)saveVoteToPool(signHash common.Hash, sign []byte) error



#### 进行DPOS验证接口(内部接口)

func (self \*BlockDPOSVerify)processDPOS() error

流程图:

