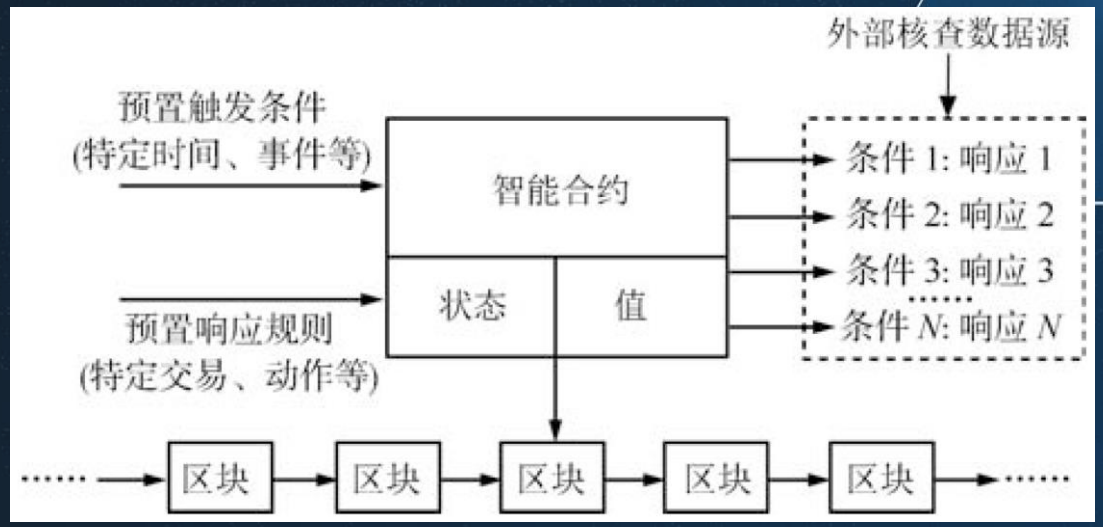
区块链技术与应用

计算机科学与技术学院　李京


07章 以太坊与智能合约

什么智能合约?

- 狭义定义：运行在分布式账本上预置规则、具有状态、条件响应的，可封装、验证、执行分布式节点复杂行为，完成信息交换、价值转移和资产管理的计算机程序。
- 广义定义：无需中介、自我验证、自动执行合约条款的计算机交易协议。按照其设计目的可分为：旨在作为法律的替代和补充的智能法律合约，旨在作为功能型软件的智能软件合约，以及旨在引入新型合约关系的智能替代合约。



特性：自动执行、安全透明、自治自足

智能合约的雏形：比特币脚本

- 基于非图灵完备字节码语言OP-RETURN的比特币脚本是最早应用于区块链的智能合约雏形。
- 计算能力非常有限，不支持循环语句，只能实现基本的算术、逻辑运算及验证加密功能，因此早期的智能合约通常无法具有复杂逻辑。
- 可编程货币。

智能合约的里程碑：以太坊 (Ethereum)

- 以太坊（Ethereum），2015年由俄罗斯天才少年Vitalik Buterin通过众筹融资推出的首个内置图灵完备编程语言，并正式引入智能合约概念的公有链系统和全球共享分布式应用平台。
- 理论上，用户可以按照自己的意愿在以太坊平台上创建并执行任意复杂的操作，从而高效快速地开发出包括加密货币在内的多种去中心化区块链应用（Decentralized Applications, DApps）。

目 录

7.1 以太坊概述

以太坊的产生背景

- 比特币开创了去中心化密码货币的先河。区块链事实上是一套分布式的数据库，如果再在其中加进一个符号（比特币），并规定一套协议使得这个符号可以在数据库上安全地转移，并且无需信任第三方，这些特征的组合完美地构造了一个货币传输体系：比特币网络。
- 然而比特币并不完美，其中协议的扩展性是一项不足，例如比特币网络里只有一种符号（比特币），用户无法自定义另外的符号，这些符号可以是代表公司的股票，或者是债务凭证等，这就限制了一些功能。另外，比特币协议尽管使用了一套基于堆栈的脚本语言，然而却不足以构建更高级的应用，例如去中心化交易所等。以太坊从设计上就是为了解决比特币扩展性不足的问题。

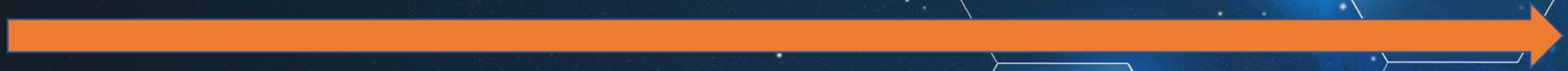
以太坊的产生背景

- 2009年，比特币出现之后，出现各种类似比特币项目，运行在不同节点，每个项目重复、独立创建一个个类似比特币的系统。
- 2013年，Vitalik Buterin提出了以太坊概念，一种能够被重编程用以实现任意复杂计算功能的单一区块链。
- 2014年，以太坊基金会成立，Vitalik、 Gavin Wood和Jeffrey Wilcke创建了以太坊项目，作为下一代区块链系统。 以太坊是一个有智能合约功能的公共区块链平台。以太坊类比操作系统，智能合约类比应用。

什么是以太坊?

- 以太坊(Ethereum)是一个开源的有智能合约功能的公共区块链平台，通过其专用加密货币以太币(Ether)提供去中心化的以太虚拟机(Ethereum Virtual Machine, EVM)来处理点对点合约。
- 以太坊是一个为去中心化应用程序而生的全球开源平台，经常被称为“世界计算机”。从计算机科学的角度来看，以太坊是一个具备确定性但实际上却没有边际的状态机。它有两个特点：具有一个全球范围可访问的单体状态；还有一个执行状态更改的虚拟机。从更加实际的角度来看，以太坊是一个开源的、全球去中心化的计算基础架构，可以执行称为智能合约的程序。它使用区块链同步和保存系统状态，借助以太币这种数字货币来计量并控制程序执行的资源开销。

以太坊的发展历史



2015年7月，边境(Frontier)，第一个版本

2016年3月，家园(Homestead)，稳定性、易用性

2016年6月，The DAO事件，分叉ETH和ETC

2017年10月，拜占庭(Byzantium)，延迟引爆难度、降低区块奖励

2019年3月，君士坦丁堡(Constantinople)，优化性能、承前启后

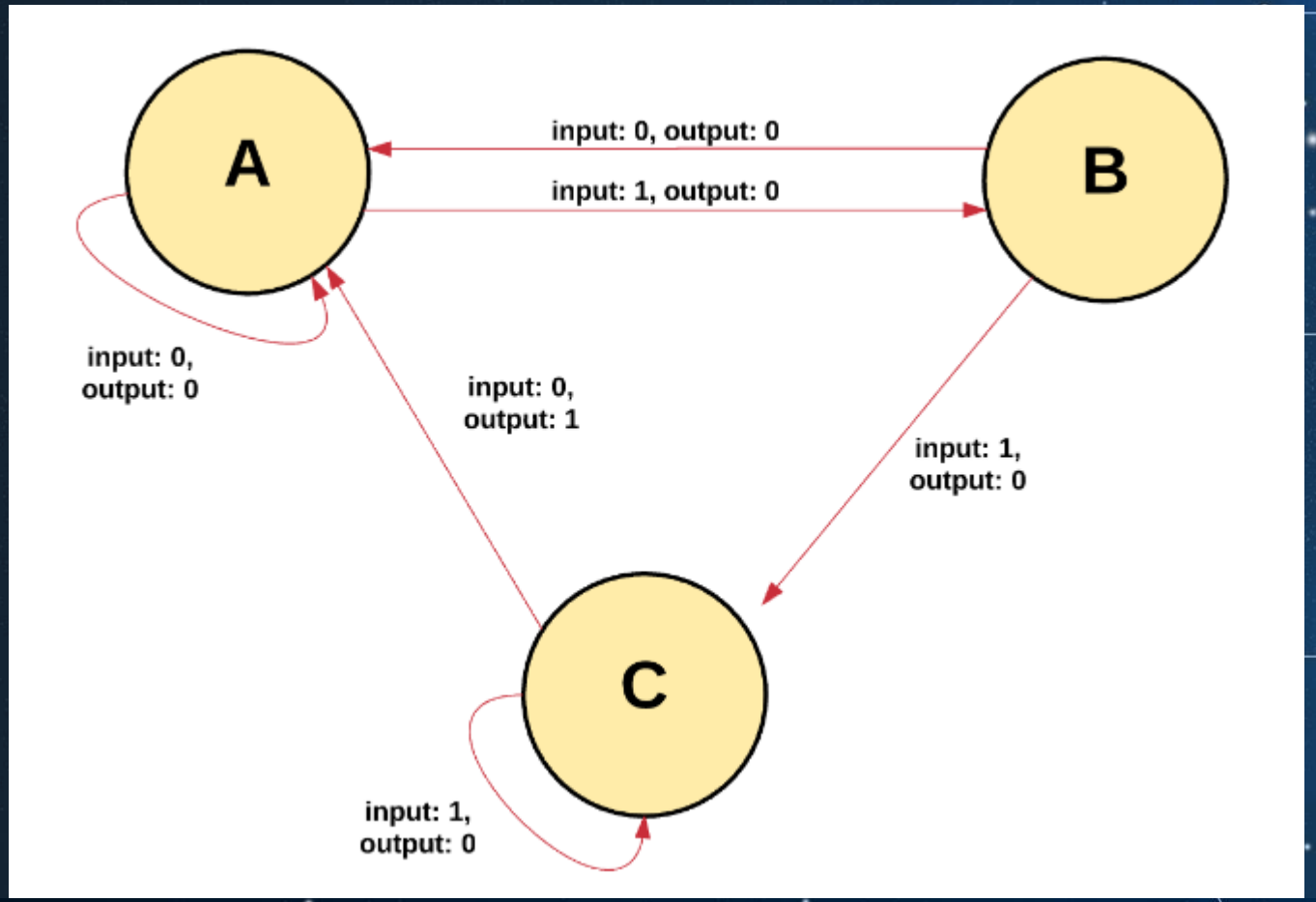
以太坊2.0，宁静(Serenity)

7.2 以太坊基本原理

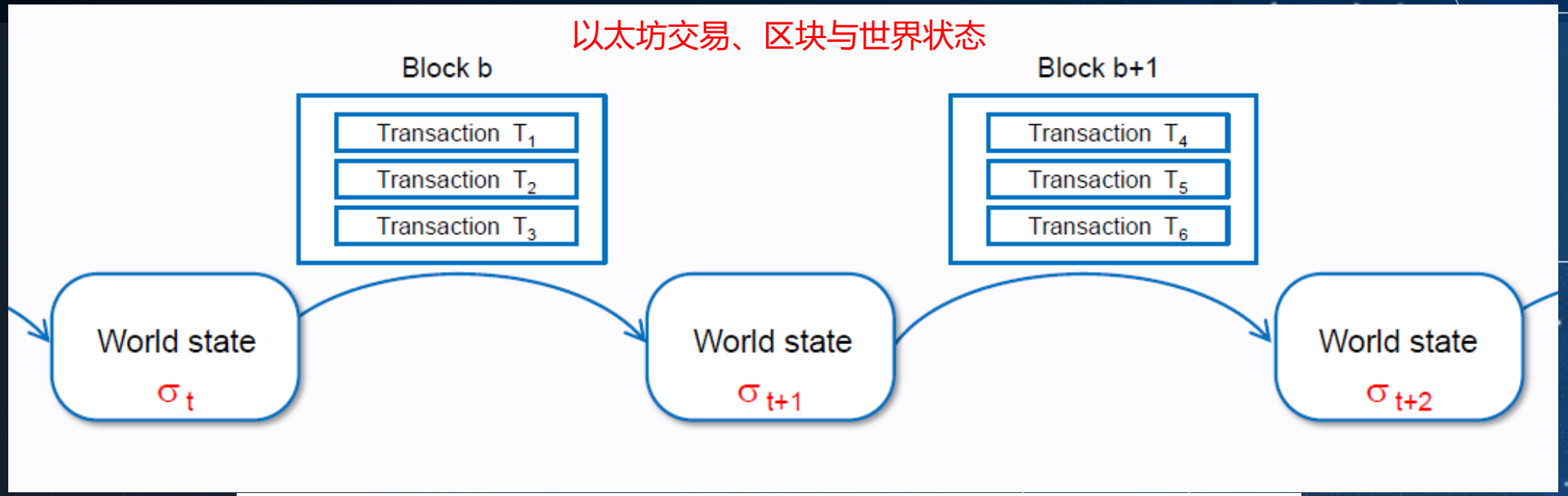
- 以太坊账户
- 以太坊区块链
- 以太坊共识机制
- 以太币

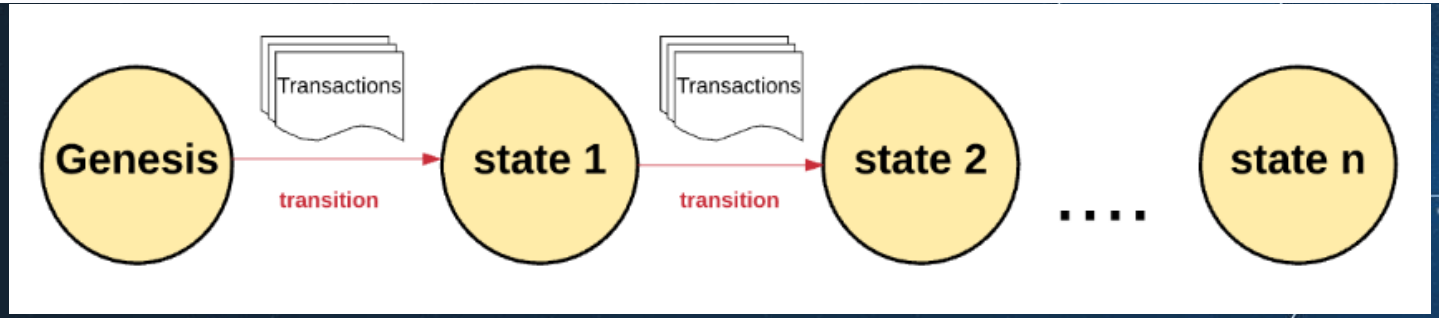
以太坊模型

- 以太坊的本质就是一个基于交易的状态机(transaction-based state machine)。



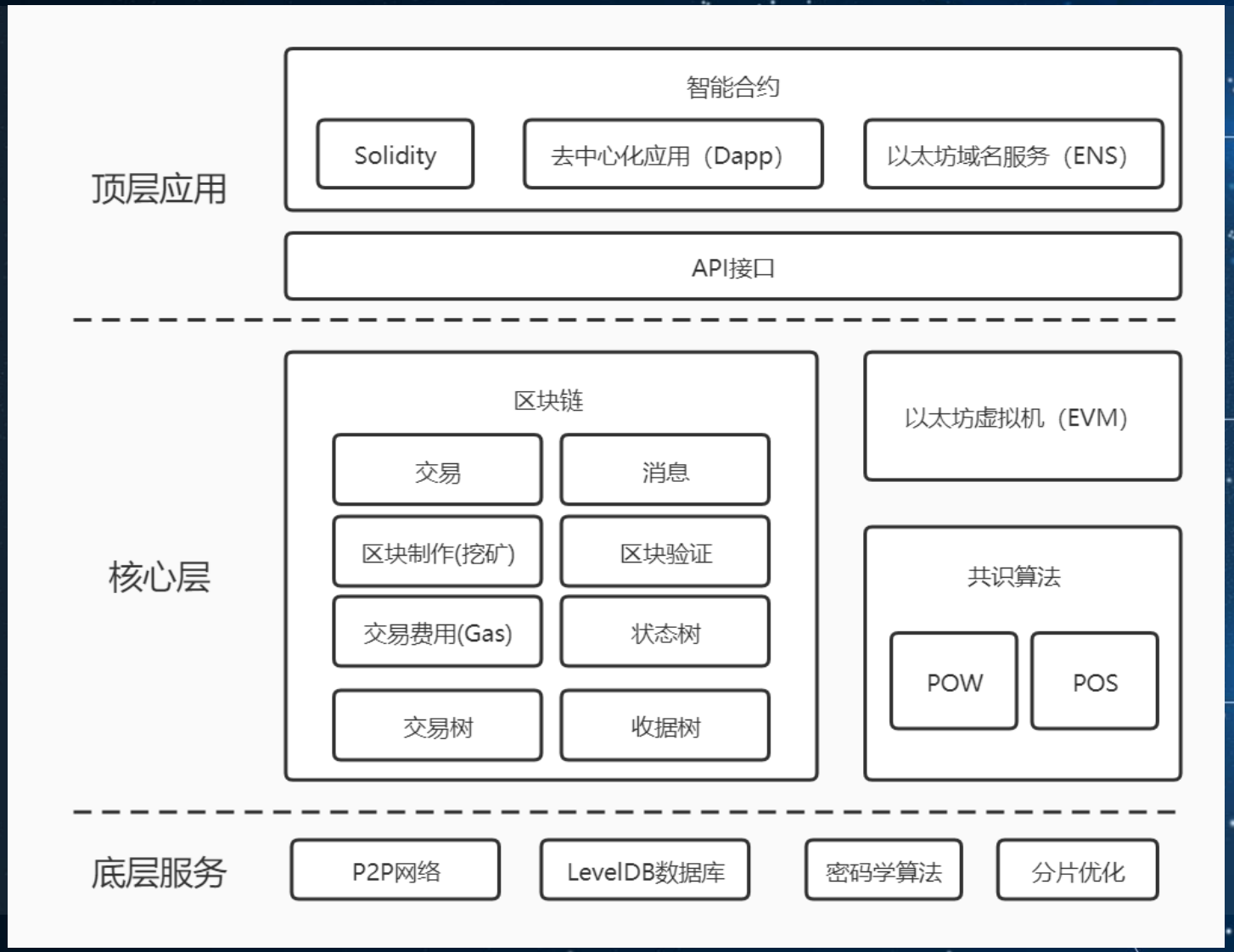
基于交易的状态机





- 以太坊在整体上可看作是一个基于交易的状态机：起始于一个创世（Genesis）状态，然后随着交易的执行，状态逐步改变一直到最终状态，这个最终状态就是以太坊世界的权威版本。

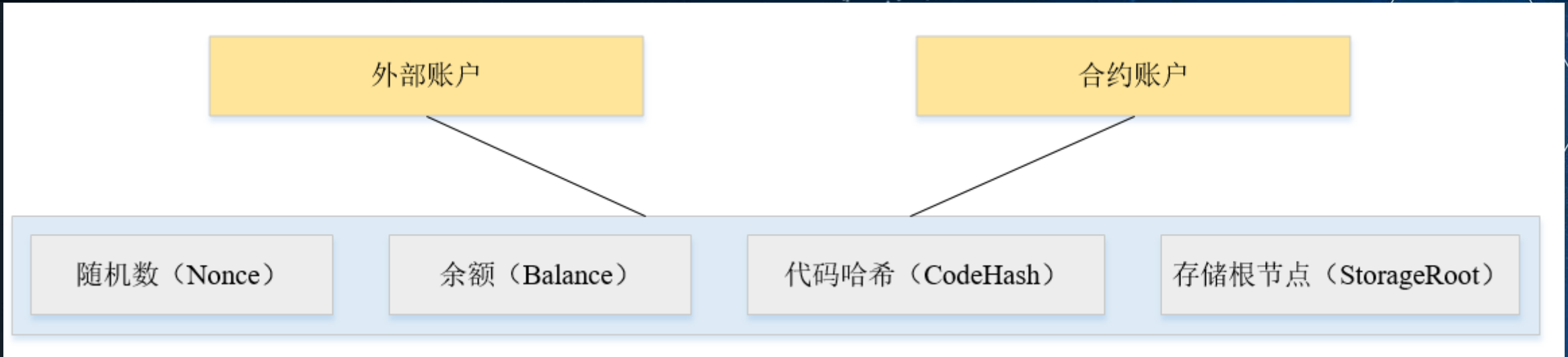
以太坊的系统架构


顶层应用
智能合约
Solidity
去中心化应用（Dapp）
以太坊域名服务（ENS）
API接口
核心层
区块链
交易
消息
区块制作(挖矿)
区块验证
交易费用(Gas)
状态树
交易树
收据树
以太坊虚拟机（EVM）
共识算法
POW
POS
底层服务
P2P网络
LevelDB数据库
密码学算法
分片优化

以太坊的基本组件

- 点对点（P2P）网络：以太坊运行在Ethereum Main Network 上，这是一个通过 TCP 30303端口寻址的网络，网络层运行的协议名为ÐΞVp2p
- 共识规则：以太坊的共识规则，由以太坊黄皮书中的参考标准进行精确定义
- 交易：以太坊交易是一个网络消息，主要包含交易的发送方、接收方、价值和数据载荷。
- 状态机：以太坊的状态转换由以太坊虚拟机(EVM)处理，这是一个基于栈的虚拟机，执行bytecode(字节码指令)。被称为“智能合约”的EVM程序采用高级语言(例如Solidity)编写，并编译为通过EVM执行的字节码。
- 数据结构：以太坊的区块链以数据库(通常采用Google的LevelDB)的方式保存在每一个节点之上，区块链内包含了交易和系统的状态，经过哈希处理的数据保存在Merkle Patricia Tree(MPT)数据结构之内。
- 共识算法：以太坊使用比特币的共识模型Nakamoto Consensus，它使用顺序单一签名块，由PoW加权重要性来确定最长链，从而确定当前状态。但是，有计划在不久的将来转向代号为Casper的PoS加权投票系统。

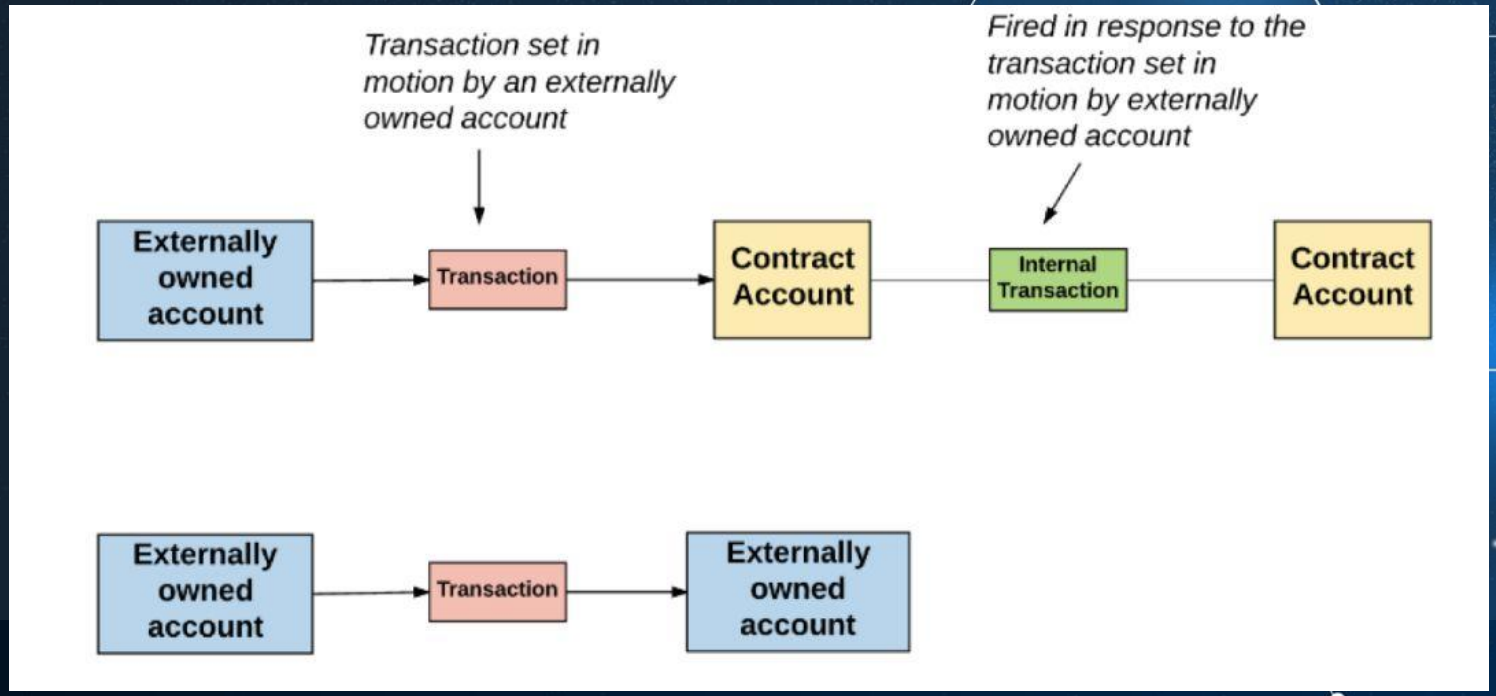
以太坊账户（Account）



- 以太坊引入了账户的概念以取代比特币的UTXO模型。账户以地址为索引，地址由公钥衍生而来。
- 有两种类型的账户：外部账户（通常简称为“账户”）和合约账户。
  - Nonce：如果账户是一个外部账户，代表从此账户地址发送的交易序号。如果账户是一个合约账户，代表此账户创建的合约序号。
  - Balance：余额，即账户拥有的以太币数量，单位为Wei，1Ether=10^{18} Wei。
  - CodeHash：代码哈希，与账户关联的EVM代码的哈希值，外部账户的codeHash为一个空字符串的哈希，创建后不可更改。状态数据库中包含所有代码片段哈希，以便后续使用。
  - StorageRoot：存储根节点，账户内容的Merkle Patricia树根节点的哈希编码。

以太坊账户-外部账户与合约账户

- 外部帐户EOA（externally owned accounts）是由用户实际控制的账户，拥有一对公私钥。可以通过使用其私钥创建和签署交易，将消息发送到其他外部帐户或合约帐户。两个外部账户之间的消息只是一个价值转移。但是从外部账户到合约账户的消息会激活合约账户的代码，允许它执行各种操作（例如转移Token，写入内部存储，创建新的Token，执行一些计算，创建新的合约等）。
- 合约账户（contract accounts）是一个包含合约代码的账户，由合约代码控制。合约账户不能自行发起新的交易，合约帐户只能触发交易以响应其他来自外部帐户或合约帐户的交易。



以太坊账户-外部账户和合约账户对比

账户类型 账户构成	外部账户	合约账户
账户私钥	一般在以太坊的钱包客户端创建通过私钥算法创建生成，私钥由账户所有者自己保管存储	没有对应的账户私钥
账户地址	账户地址基于账户的公钥采用地址生成算法推导得出	账户地址采用智能合约发布者的账户相关信息推导得出
账户链上生成	通过账户私钥签名一笔交易发往以太坊的节点创建生成	通过外部账户向以太坊发布智能合约时创建生成
账户存储	仅仅存储一个账户对应的交易序号和账户的有效余额	除了外部账户的存储信息外，还存储该账户对应的智能合约数据。这些智能合约的数据按照特定的编码方式进行组织和存储
账户代码	无对应的账户代码	保存了智能合约的账户代码内容，该内容即为编写的智能合约代码经过编译器编译后字节码信息

UTXO与账户模型的对比

UTXO模型的好处：

- 可扩展性 - 由于可以同时处理多个UTXO，因此可以实现并行交易并鼓励可扩展性创新。
- 隐私性 - 即使比特币不是一个完全匿名的系统，假设用户每笔交易都使用新地址，那么也能够提供更高级别的隐私。 如果需要更好的隐私性可以考虑更复杂的方案，例如环形签名。

账户/余额模型的好处：

- 简单性 - 以太坊选择了更直观的模型，以使开发人员在开发那些需要状态信息或涉及多方的智能合约时更简单。 例如智能合约，它能够直接跟踪状态并能够执行不同的任务。UTXO的无状态模型会强制交易包含状态信息，这使得合约的设计复杂化。
- 效率 - 除简单性外，帐户/余额模型更有效，因为交易时只需要验证发送帐户是否有足够的余额。
- 账户/余额模型的一个缺点是无法避免双重支付攻击，但通过递增的随机数可以消除这种类型的攻击。在以太坊中，每个帐户都有一个公共可见的随机数，每次进行交易时，随机数增加一。这可以防止同一个交易被多次提交。

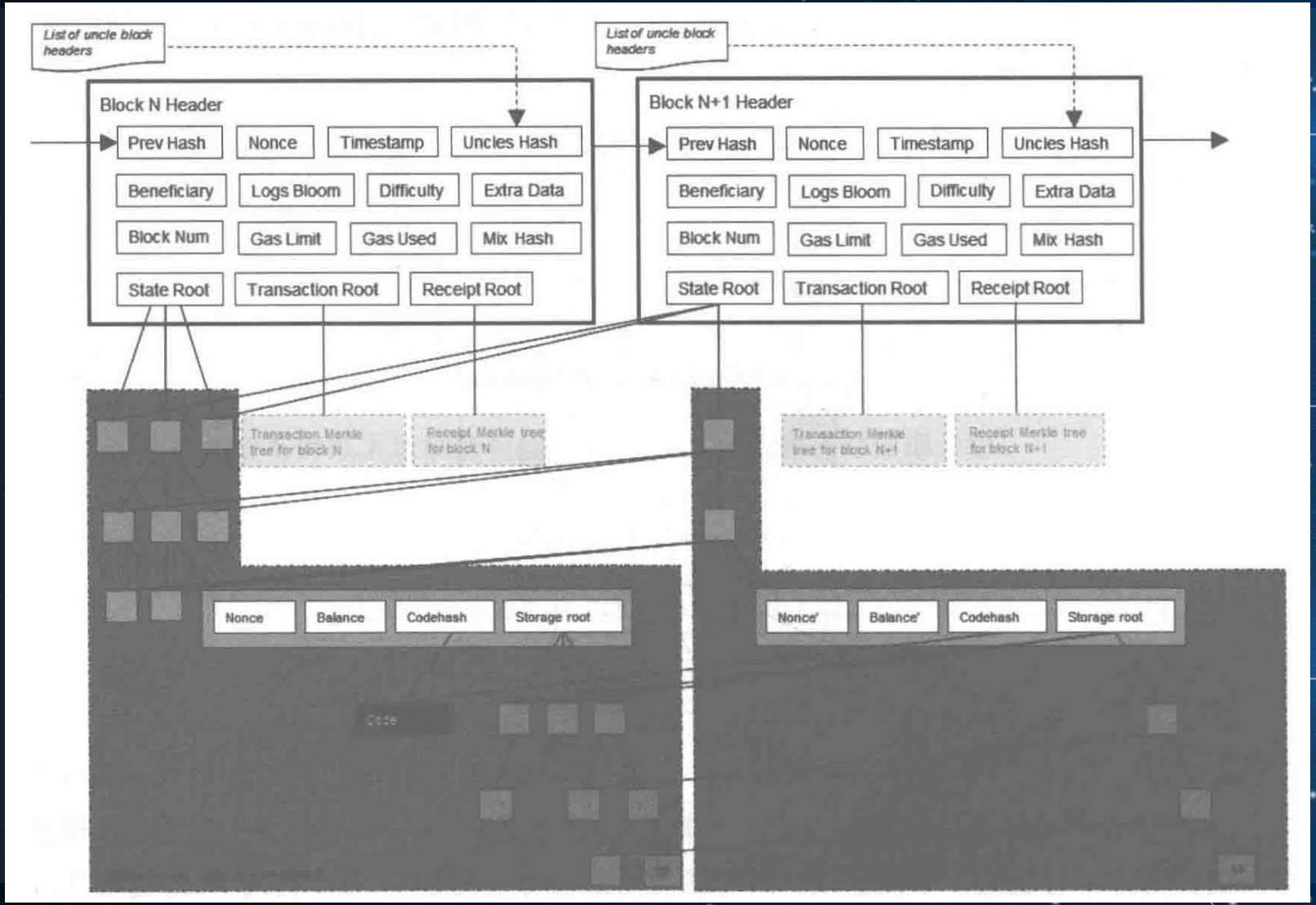
以太坊区块链

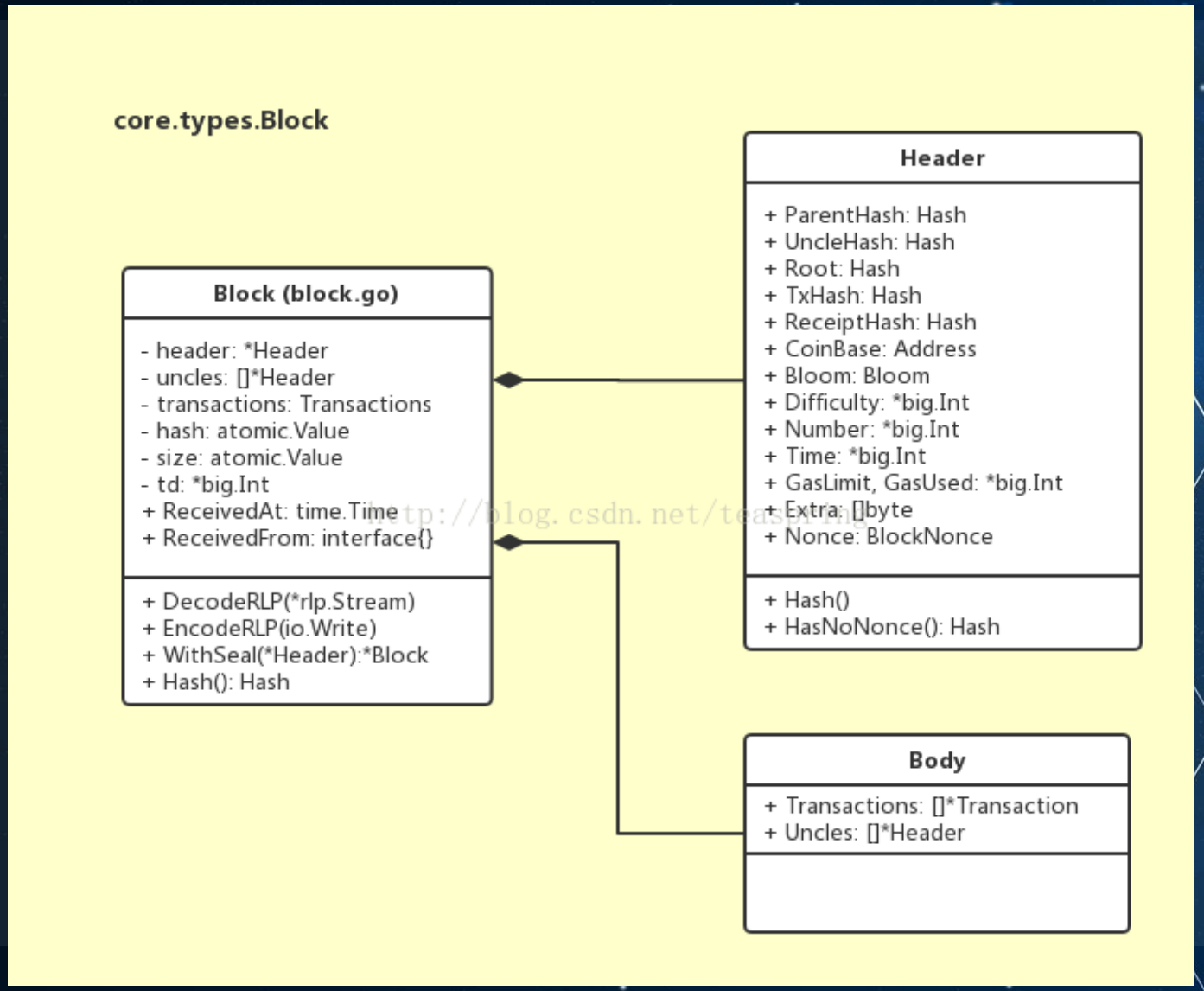
- 区块和区块链
- 交易和消息
- 状态树、交易树和收据树
- LevelDB

以太坊区块链

- 以太坊对比特币区块链技术上做了一些调整，以太坊区块主要由区块头、交易列表和叔区块头三部分组成。
- 区块头包含下列信息：父块的散列值(Prev Hash)、叔区块的散列值(Uncles Hash)、状态树根散列值(stateRoot)、交易树根散列值(Transaction Root)、收据树根散列值(Receipt Root)、时间戳(Timestamp)、随机数(Nonce) 等。以太坊区块链上区块数据结构的一个重大改变就是保存了三棵Merkle树根，分别是状态树、交易树和收据树。存储三棵树可方便账户做更多查询。
- 交易列表是由矿工从交易池中选择打包进区块中的一系列交易。
- 区块链上的第一个区块称为“创世区块”，区块链上除了创世区块以外每个区块都有它的父区块，这些区块连接起来组成一个区块链，出块时间约为15秒。

以太坊区块结构


List of uncle block headers
List of uncle block headers
Block N Header
Prev Hash
Nonce
Timestamp
Uncles Hash
Beneficiary
Logs Bloom
Difficulty
Extra Data
Block Num
Gas Limit
Gas Used
Mix Hash
State Root
Transaction Root
Receipt Root
Block N+1 Header
Prev Hash
Nonce
Timestamp
Uncles Hash
Beneficiary
Logs Bloom
Difficulty
Extra Data
Block Num
Gas Limit
Gas Used
Mix Hash
State Root
Transaction Root
Receipt Root
Transaction Merkle tree for block N
Receipt Merkle tree for block N
Transaction Merkle tree for block N+1
Receipt Merkle tree for block N+1
Nonce
Balance
Codehash
Storage root
Nonce'
Balance'
Codehash
Storage root

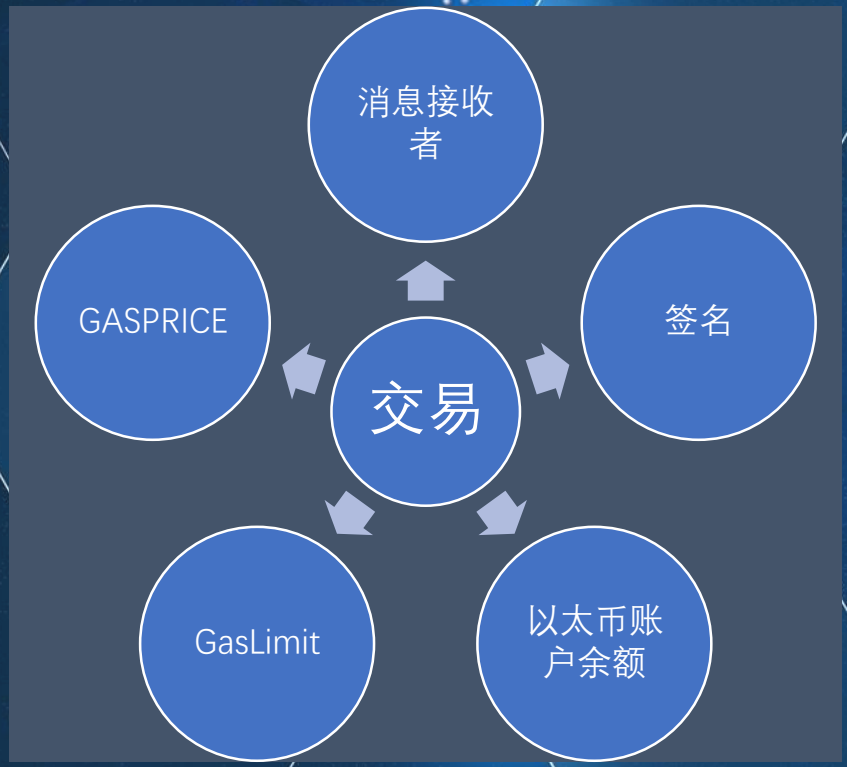

core.types.Block
Block (block.go)
- header: *Header
- uncles: []*Header
- transactions: Transactions
- hash: atomic.Value
- size: atomic.Value
- td: *big.Int
+ ReceivedAt: time.Time
+ ReceivedFrom: interface{}
+ DecodeRLP(*rlp.Stream)
+ EncodeRLP(io.Write)
+ WithSeal(*Header):*Block
+ Hash(): Hash
Header
+ ParentHash: Hash
+ UncleHash: Hash
+ Root: Hash
+ TxHash: Hash
+ ReceiptHash: Hash
+ CoinBase: Address
+ Bloom: Bloom
+ Difficulty: *big.Int
+ Number: *big.Int
+ Time: *big.Int
+ GasLimit, GasUsed: *big.Int
+ Extra: []byte
+ Nonce: BlockNonce
+ Hash()
+ HasNoNonce(): Hash
Body
+ Transactions: []*Transaction
+ Uncles: []*Header

Header成员

- ParentHash：指向父区块(parentBlock)的哈希指针。除了创世块(Genesis Block)外，每个区块有且只有一个父区块。
- Coinbase：挖掘出这个区块的矿工地址。在每次执行交易时系统会给与一定补偿的Ether，这笔金额就是发给这个地址的。
- UncleHash：Block结构体的成员uncles的RLP哈希值。uncles是一个Header数组。
- Root：StateDB中的“state Trie”的根节点的RLP哈希值。Block中，每个账户以stateObject对象表示，账户以Address为唯一标示，其信息在相关交易(Transaction)的执行中被修改。所有账户对象可以逐个插入一个Merkle-Patrica-Trie(MPT)结构里，形成“state Trie”。
- TxHash: Block中 “tx Trie”的根节点的RLP哈希值。Block的成员变量transactions中所有的tx对象，被逐个插入一个MPT结构，形成“tx Trie”。
- ReceiptHash：Block中的 "Receipt Trie”的根节点的RLP哈希值。Block的所有Transaction执行完后会生成一个Receipt数组，这个数组中的所有Receipt被逐个插入一个MPT结构中，形成"Receipt Trie"。
- Bloom：Bloom过滤器(Filter)，用来快速判断一个参数Log对象是否存在于一组已知的Log集合中。
- Difficulty：区块的难度。Block的Difficulty由共识算法基于parentBlock的Time和Difficulty计算得出，它会应用在区块的‘挖掘’阶段。
- Number：区块的序号。Block的Number等于其父区块Number +1。
- Time：区块“应该”被创建的时间。由共识算法确定，一般来说，要么等于parentBlock.Time + 10s，要么等于当前系统时间。
- GasLimit：区块内所有Gas消耗的上限，打包进该区块的交易的GasLimit之和不能大于区块的GasLimit。该数值在区块创建时设置，与父区块有关。具体来说，根据父区块的GasUsed同GasLimit * 2/3的大小关系来计算得出。
- GasUsed：区块内所有Transaction执行时所实际消耗的Gas总和。
- Nonce：一个64bit的哈希数，它被应用在区块的"挖掘"阶段，并且在使用中会被修改。

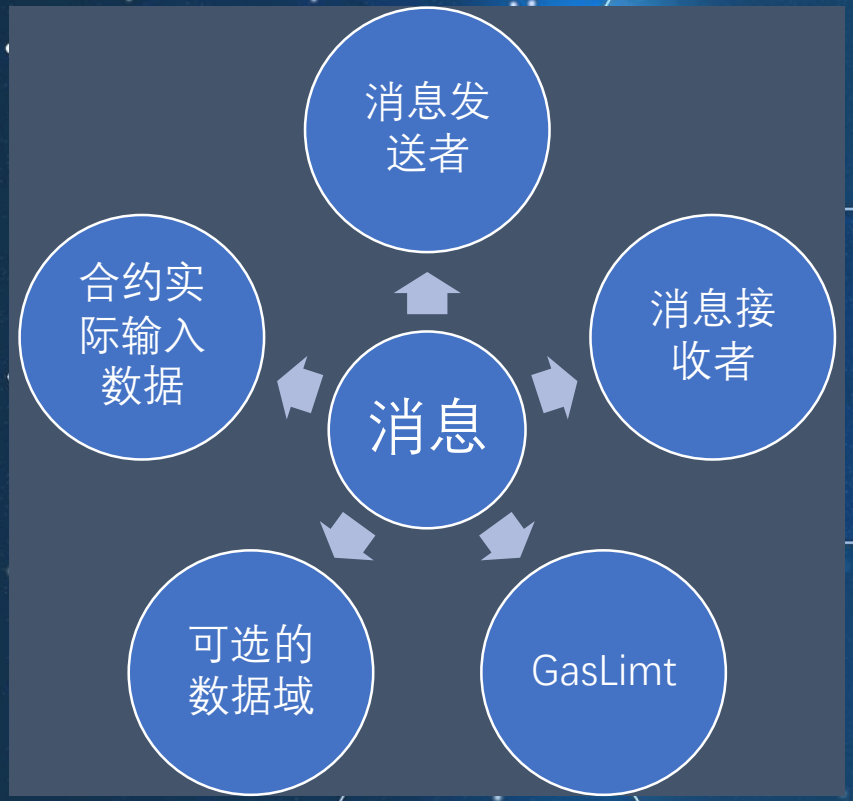
以太坊交易与消息

- 以太坊的“交易”是指一条外部账户发送到区块链上另一账户的消息的签名数据包，包含了发送者的签名、接收者的地址以及发送者转移给接收者的以太币数量等内容。
- 以太坊交易数据包还包括GasLimit和GASPRICE。以太坊的每一笔交易都需要支付一定的费用，用于支付交易执行所需要的计算开销。计算开销的费用并不是以太币直接计算的，而是引入Gas作为执行开销的基本单位，通过Gas Price与以太币进行换算。
- 引入Gas，还可以防止代码的指数型爆炸和无限循环，每笔交易需要对执行代码所引发的计算步骤-包括初始消息和所有执行中引发的消息-做出限制。GasLimit就是限制，GASPRICE是每一计算步骤需要支付矿工的费用。如果执行交易的过程中，“用完了瓦斯”，所有的状态改变恢复原状态，但是已经支付的交易费用不可收回了。如果执行交易中止时还剩余瓦斯，那么这些瓦斯将退还给发送者。
- 创建合约有单独的交易类型和相应的消息类型；合约的地址是基于账号随机数和交易数据的哈希计算出来的。



以太坊交易与消息

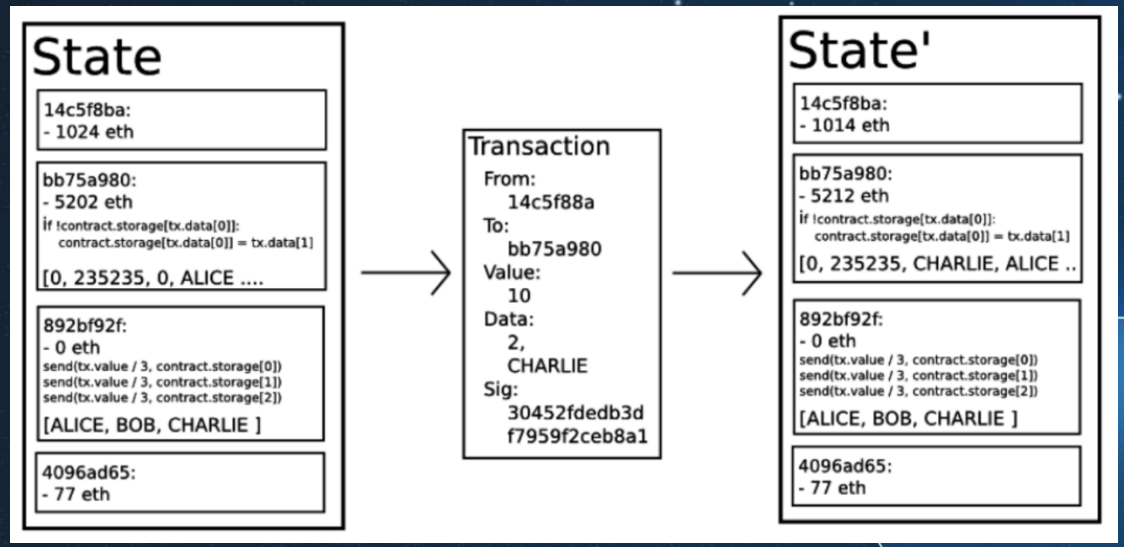
- 以太坊的消息在某种程度上类似于比特币的交易，一个消息就是一个交易
- 消息包括：消息发送者、消息的接收者、可选的数据域、合约实际输入数据、gasLimit，同交易。



交易的类型

- 交易可以分为三种：
  - 转账交易：是最简单的一种交易，从一个账户向另一个账户发送以太币。发送转账交易只需要指定交易的发送者、接收者、转币的数量。
  - 创建合约的交易：将合约部署到区块链上的交易，发送者是合约的创建者，接受者为空，交易数据字段中指定合约的二进制代码。
  - 执行合约的交易：调用合约中的方法，需要将交易的接受者指定为要调用的合约的地址，通过交易数据字段指定要调用的方法以及向该方法传递的参数。一段智能合约是被唯一的（合约）地址所标识，该地址有自己的资金余额（以太币），并且一旦有一笔交易发送至该（合约）地址，以太坊网络节点就会执行合约逻辑。以太坊使用以太坊虚拟机（EVM）来执行智能合约。

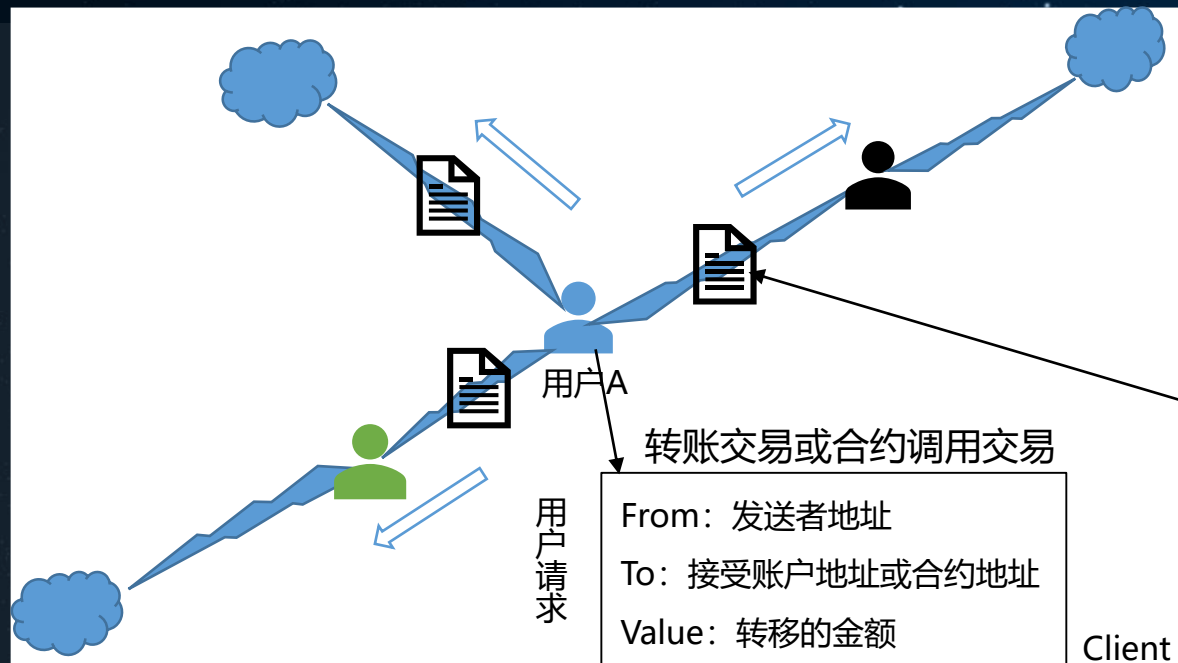
以太坊状态转换函数（交易的执行）



以太坊的状态转换函数：APPLY(S,TX) -> S'，可以定义如下：

- 检查交易的格式是否正确、签名是否有效和随机数是否与发送者账户的随机数匹配。如否，返回错误。
- 计算交易费用：fee=GASLimit * GASPRICE，并从签名中确定发送者的地址。从发送者的账户中减去交易费用和增加发送者的随机数。如果账户余额不足，返回错误。
- 设定初值GAS = GASLimit，并根据交易中的字节数减去一定量的瓦斯值。
- 从发送者的账户转移价值到接收者账户。如果接收账户还不存在，创建此账户。如果接收账户是一个合约，运行合约的代码，直到代码运行结束或者瓦斯用完。
- 如果因为发送者账户没有足够的钱或者代码执行耗尽瓦斯导致价值转移失败，恢复原来的状态，但是还需要支付交易费用，交易费用加至矿工账户。
- 否则，将所有剩余的瓦斯归还给发送者，消耗掉的瓦斯作为交易费用发送给矿工。

以太坊交易流程-转账或合约调用交易



用户请求

转账交易或合约调用交易

From：发送者地址
To：接受账户地址或合约地址
Value：转移的金额
Gas：用来完成交易的Gas数量
GasPrice：交易中愿意付出的Gas的单价
Data：一段要发送给接受者（合约账户）的消息
…

Client交易形式化

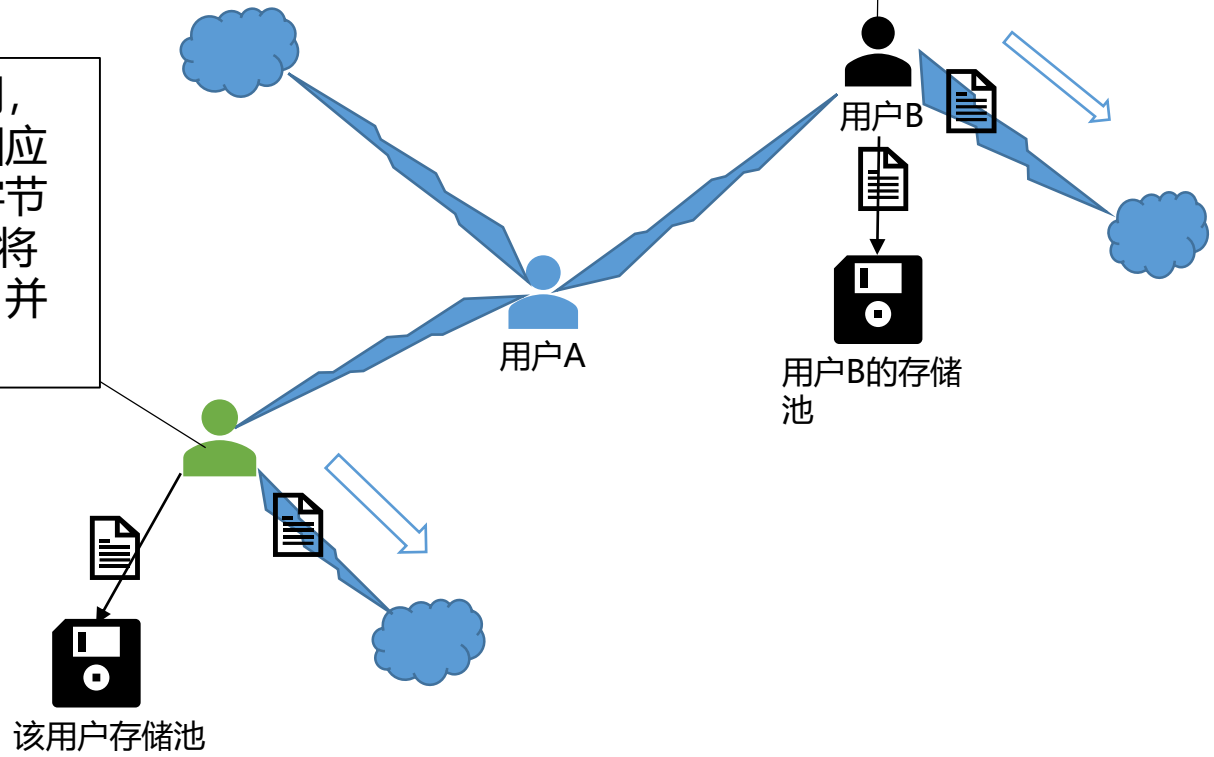
From：发送者地址
To：接受账户地址或合约地址
Value：转移的金额
Gas：用来完成交易（执行合约）的Gas数量
GasPrice：交易中心愿意付出的Gas的单价
Data：一段要发送给接受者（合约账户）的消息
Nonce：整数，用户发起的每笔交易都有唯一的编号，表示该交易是用户发起的第几笔交易
…

发送者发起转账交易或合约调用交易请求

以太坊交易流程

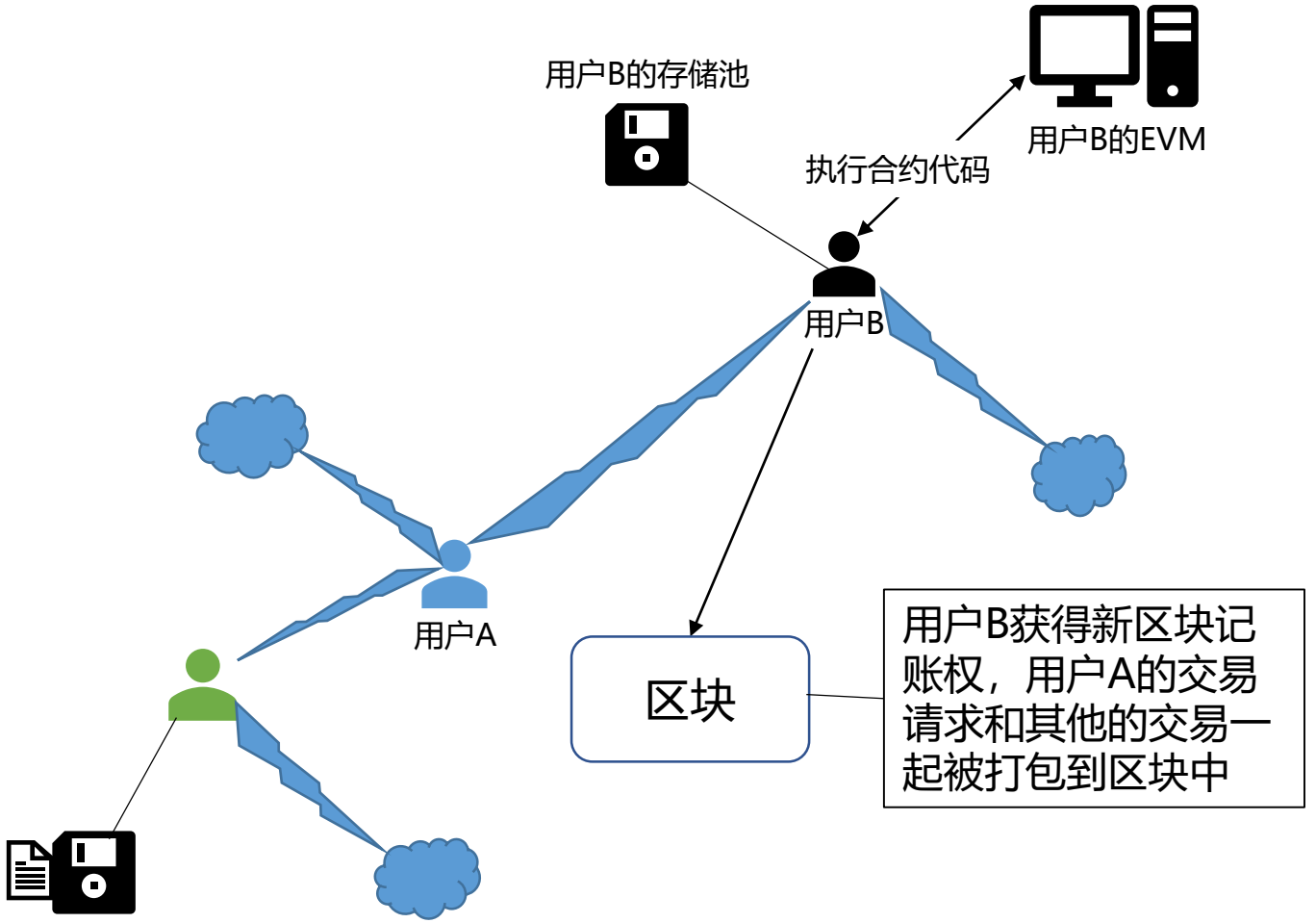
检查交易，计算交易费用，从发送方的账户中减去相应费用，并根据交易中的字节数减去一定量的Gas值。将交易请求放在存储池中，并转发给其他节点

检查交易，计算交易费用，从发送方的账户中减去相应费用，并根据交易中的字节数减去一定量的Gas值。将交易请求放在存储池中，并转发给其他节点



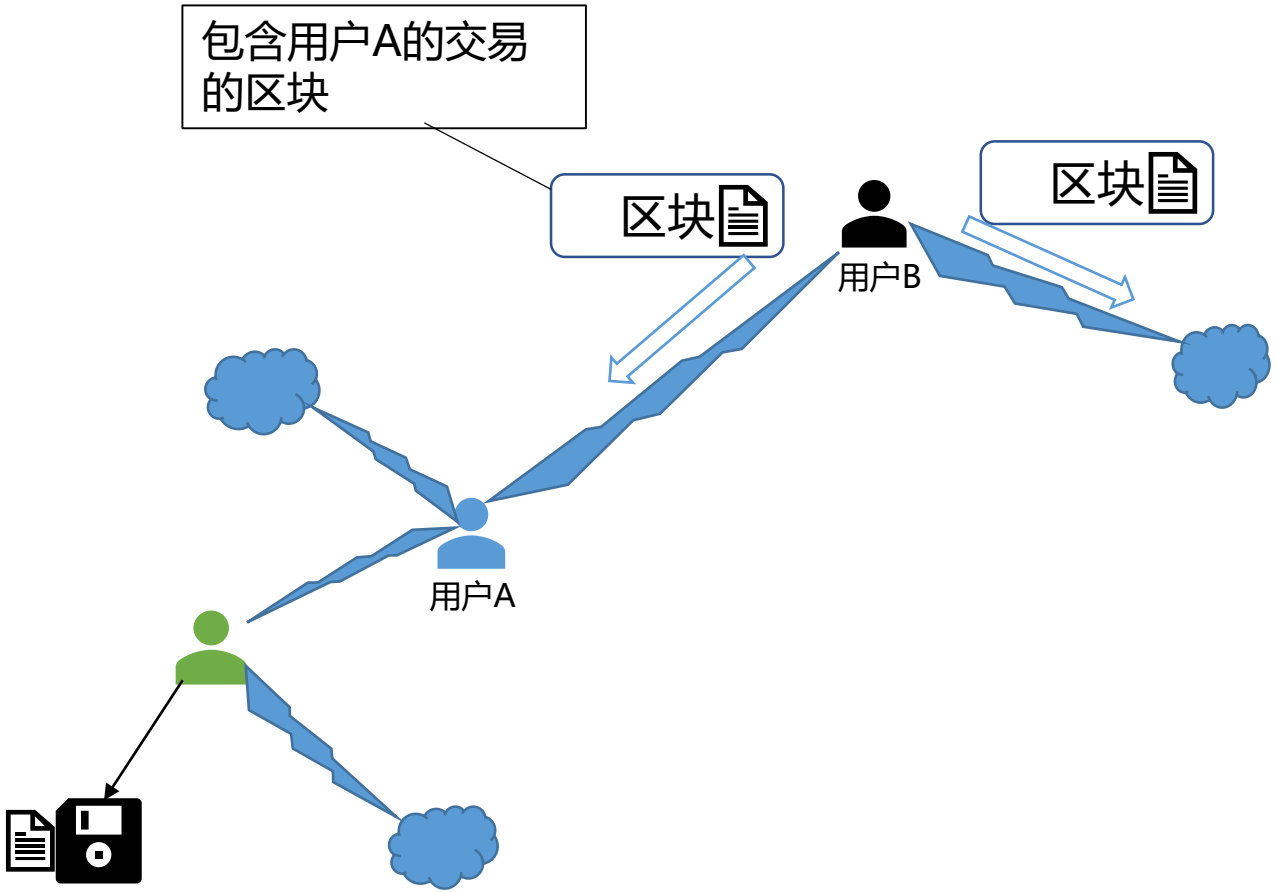
对等节点检验、存储和转发交易

以太坊交易流程



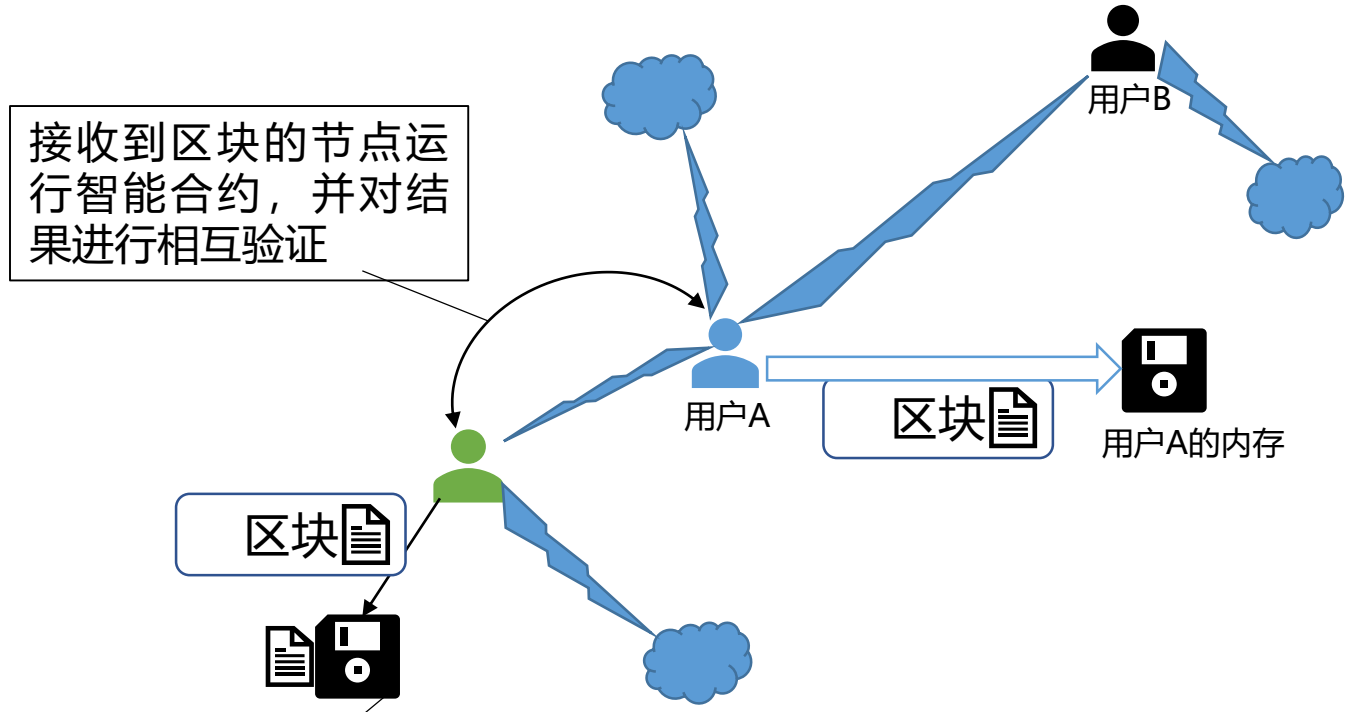
获得记账权的节点打包交易请求并执行合约代码

以太坊交易流程



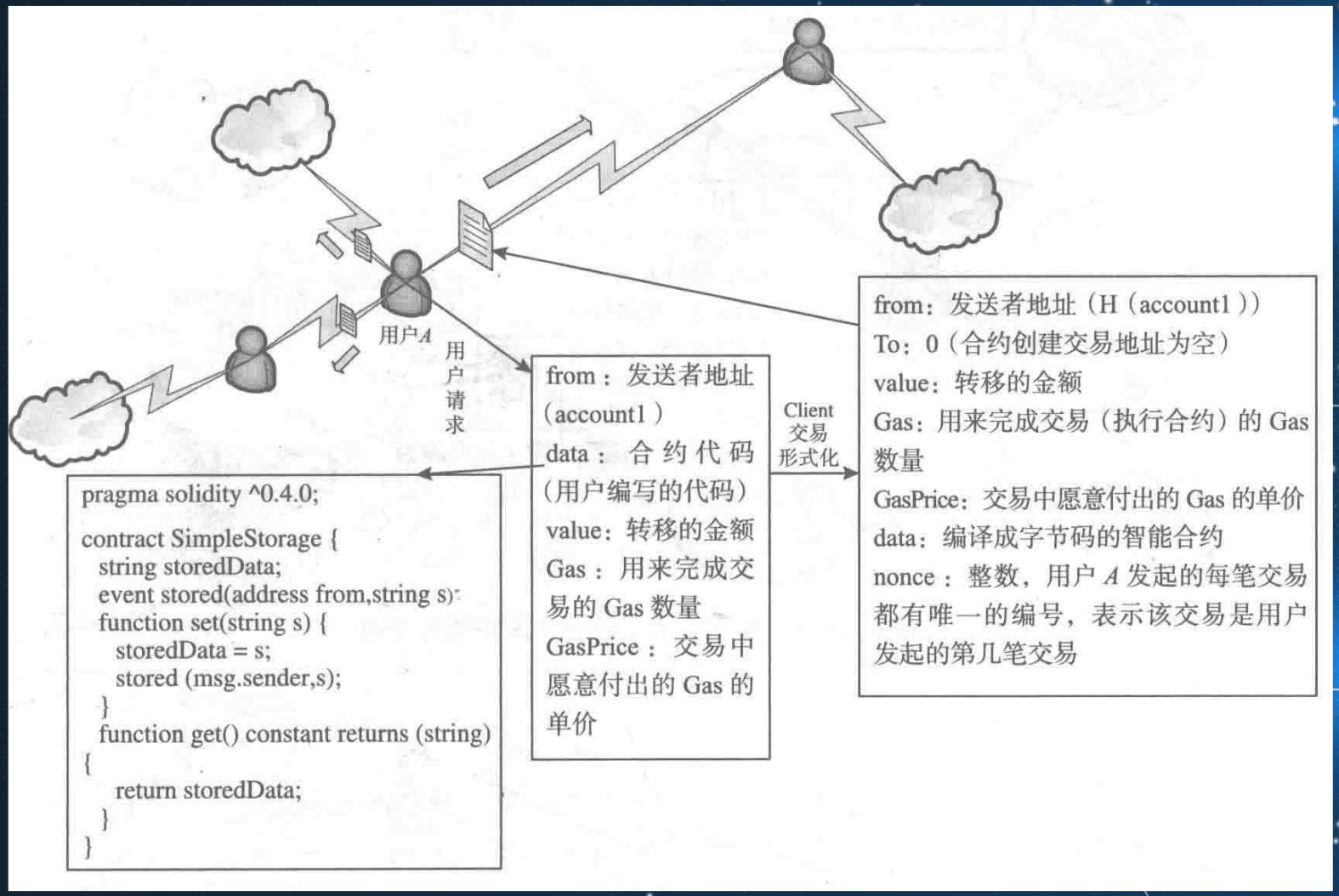
生成区块的节点将区块发送到网络中

以太坊交易流程



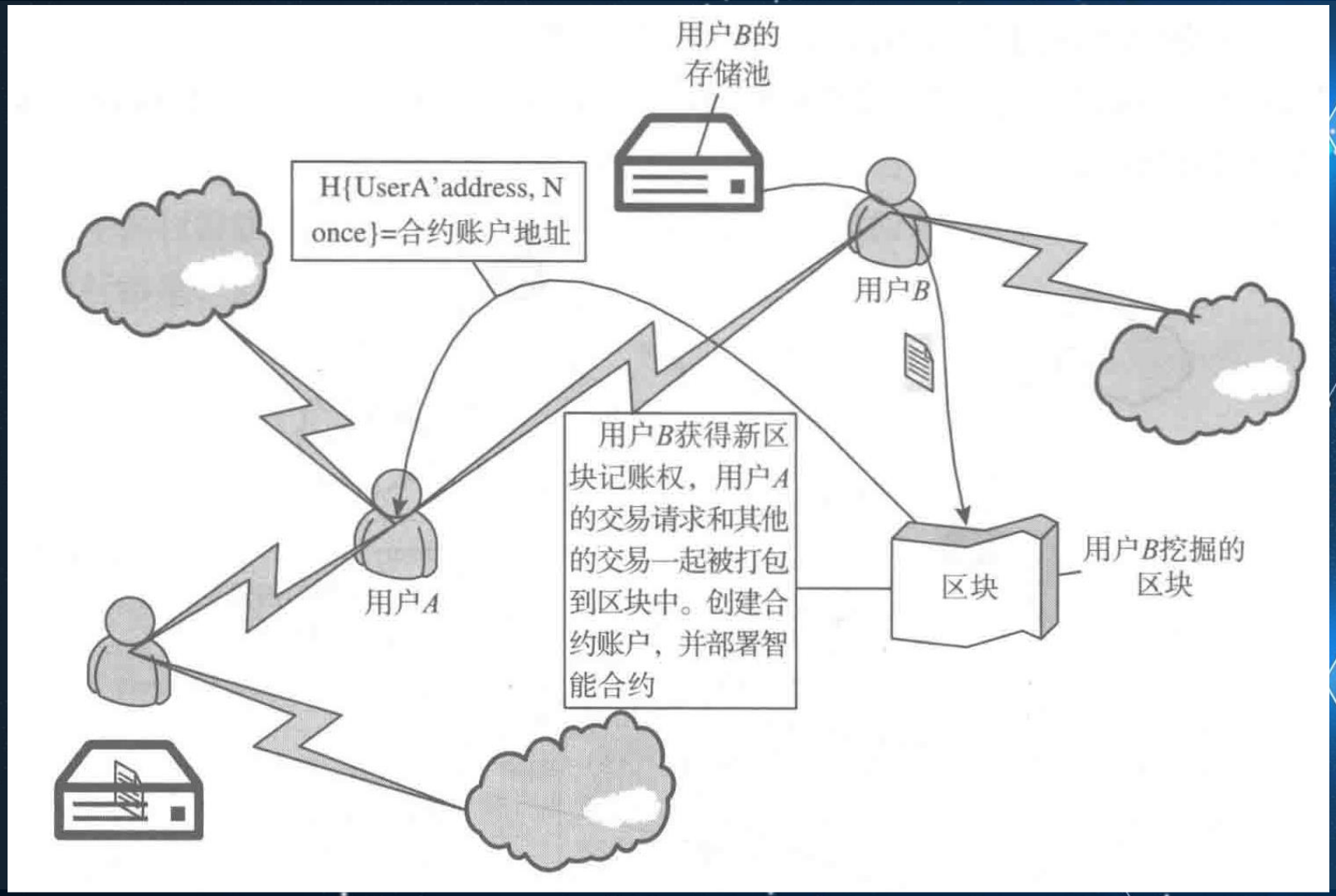
交易被所有节点保存在本地的区块链中

以太坊交易流程-创建合约交易



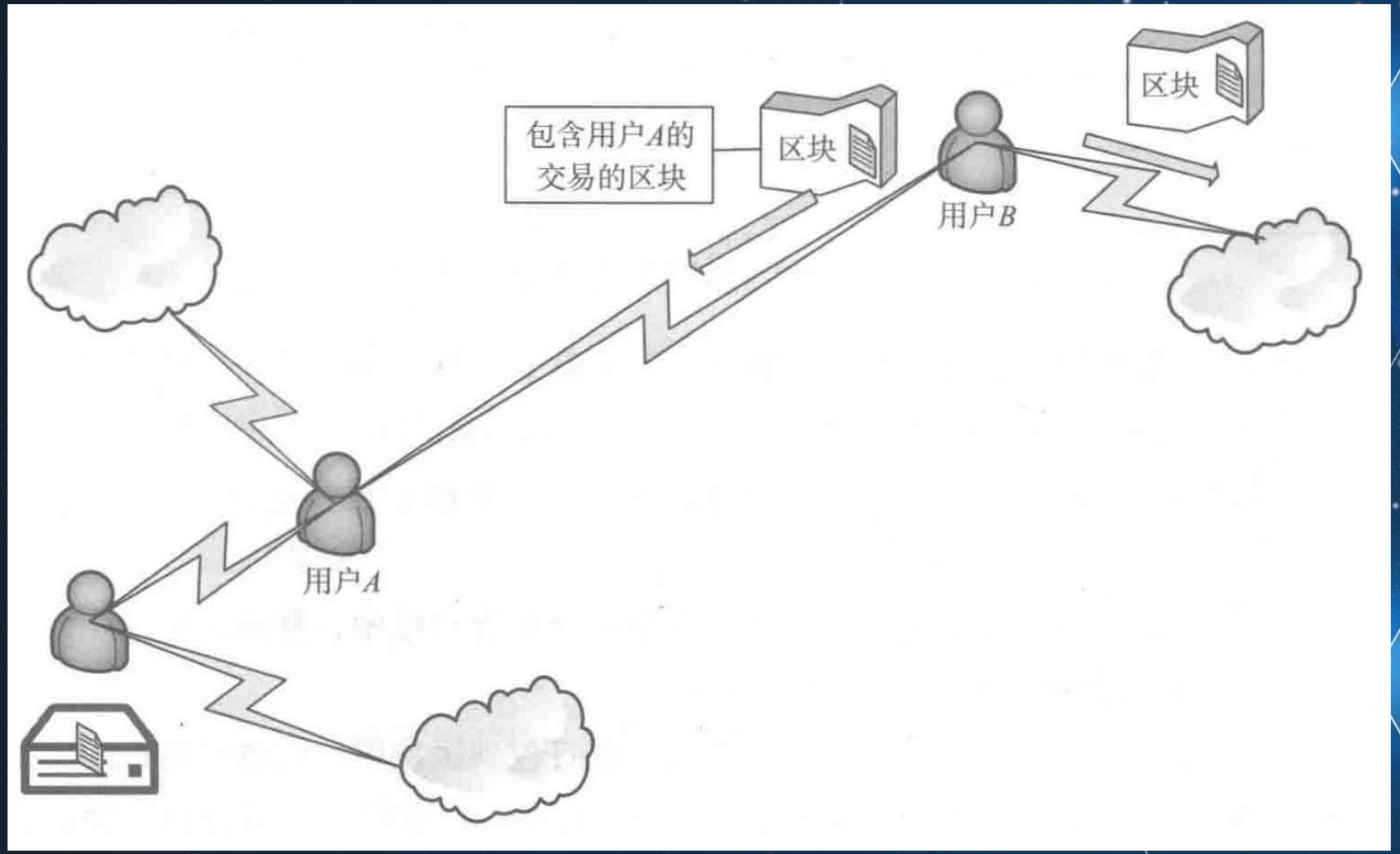
发送者发起创建智能合约的交易请求

以太坊交易流程-创建合约交易



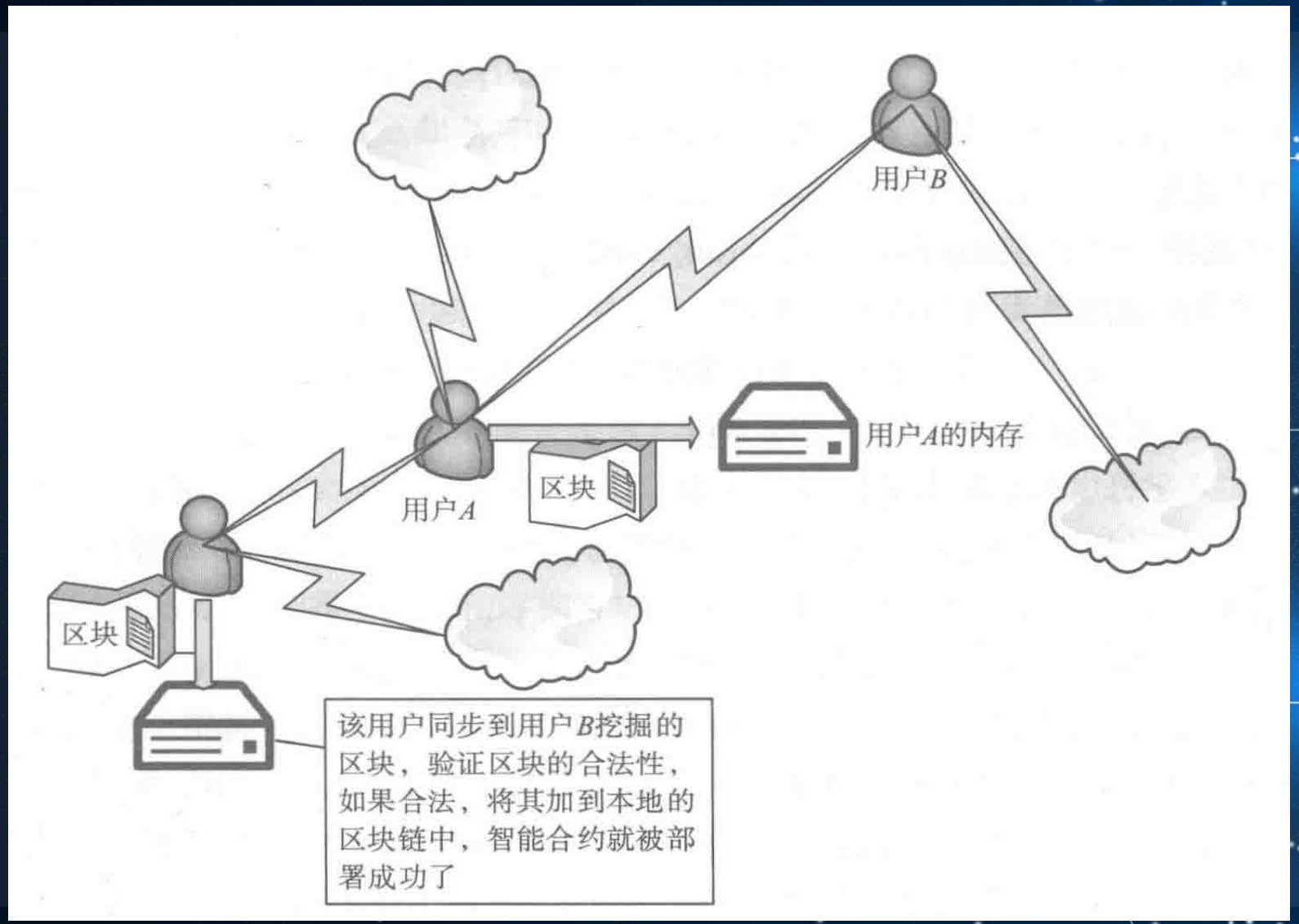
获得记账权的节点打包交易并部署合约

以太坊交易流程-创建合约交易



获得记账权的节点发送新区块至以太坊

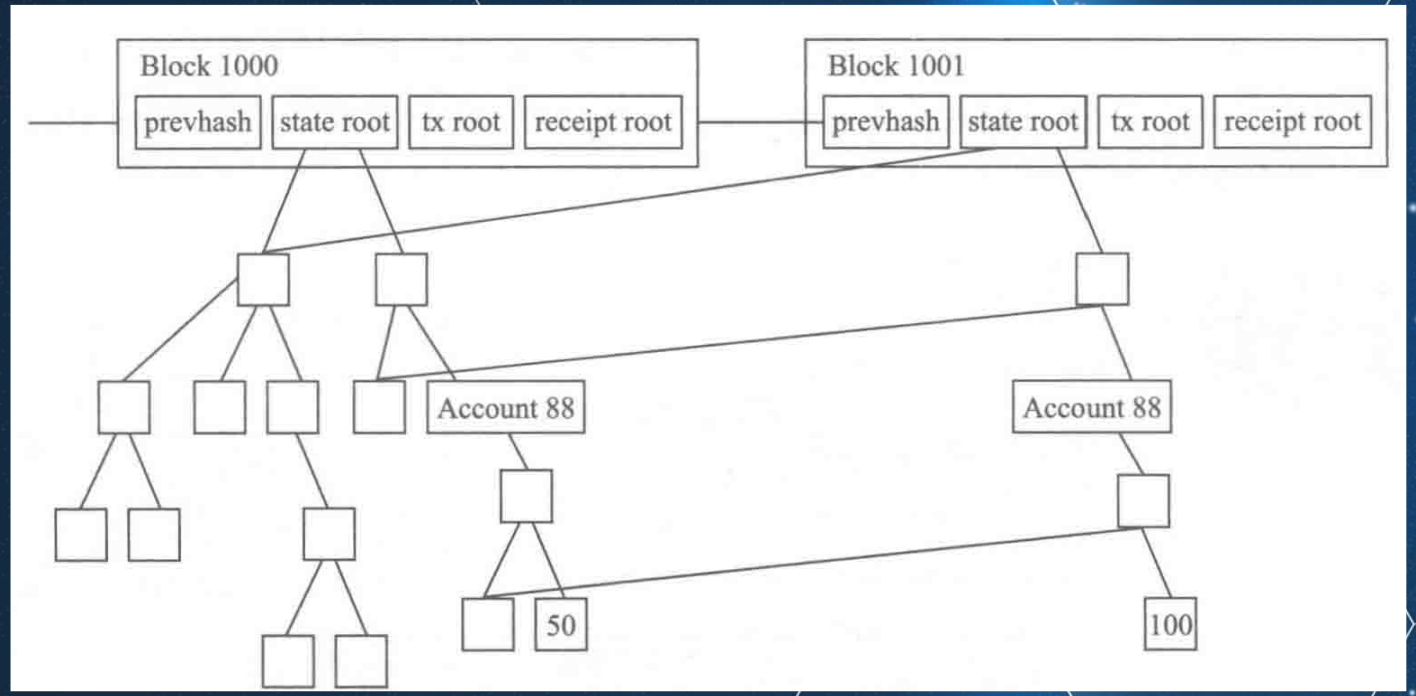
以太坊交易流程-创建合约交易



智能合约被部署在本地区块链上

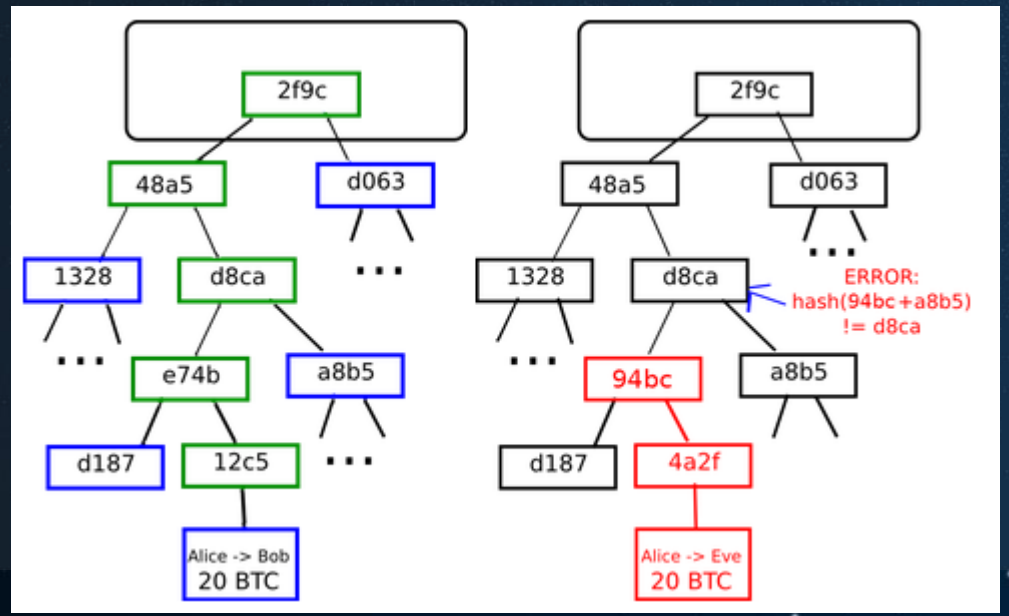
状态树、交易树和收据树

- 每个以太坊区块头保存了三棵树，这三棵树组织成MPT树：
- 状态树（State Tree）
- 交易树（Transaction Tree）
- 收据树（Receipt Tree）



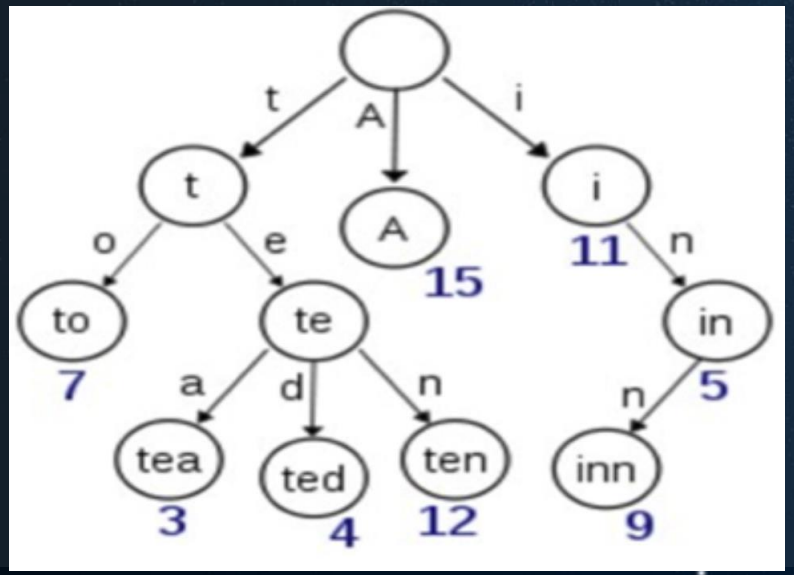
Merkle Patricia树MPT

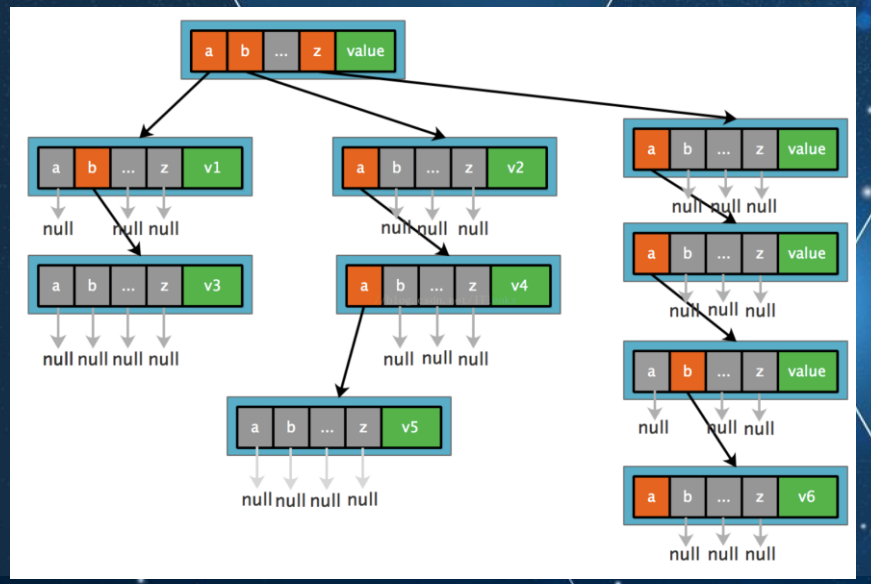
- 以太坊使用了Merkle Patircia树(又称Merkle Patricia Trie，MPT)作为数据组织形式，用来组织管理用户的账户状态、交易信息等重要数据。MPT是一种加密认证的数据结构，它融合了Merkle树和Trie树（前缀树）两种数据类型的优点。
- Merkle树又称哈希树。



Trie树

- Trie树也叫做Radix树，通常用来存储（key，value）对。key代表的是从树根到对应value的一条真实路径。即从根节点开始，key中的每个字符（从前到后）都代表着从根节点出发寻找相应value所要经过的子节点。value存储在叶节点中，是每条路径的终节点。
- 假如key中的每个字符都来自一个容量为N且所包含的字母都互不相同的字母表，那么树中的每个节点最多会有N个孩子，树的最大深度便是key的最大长度。

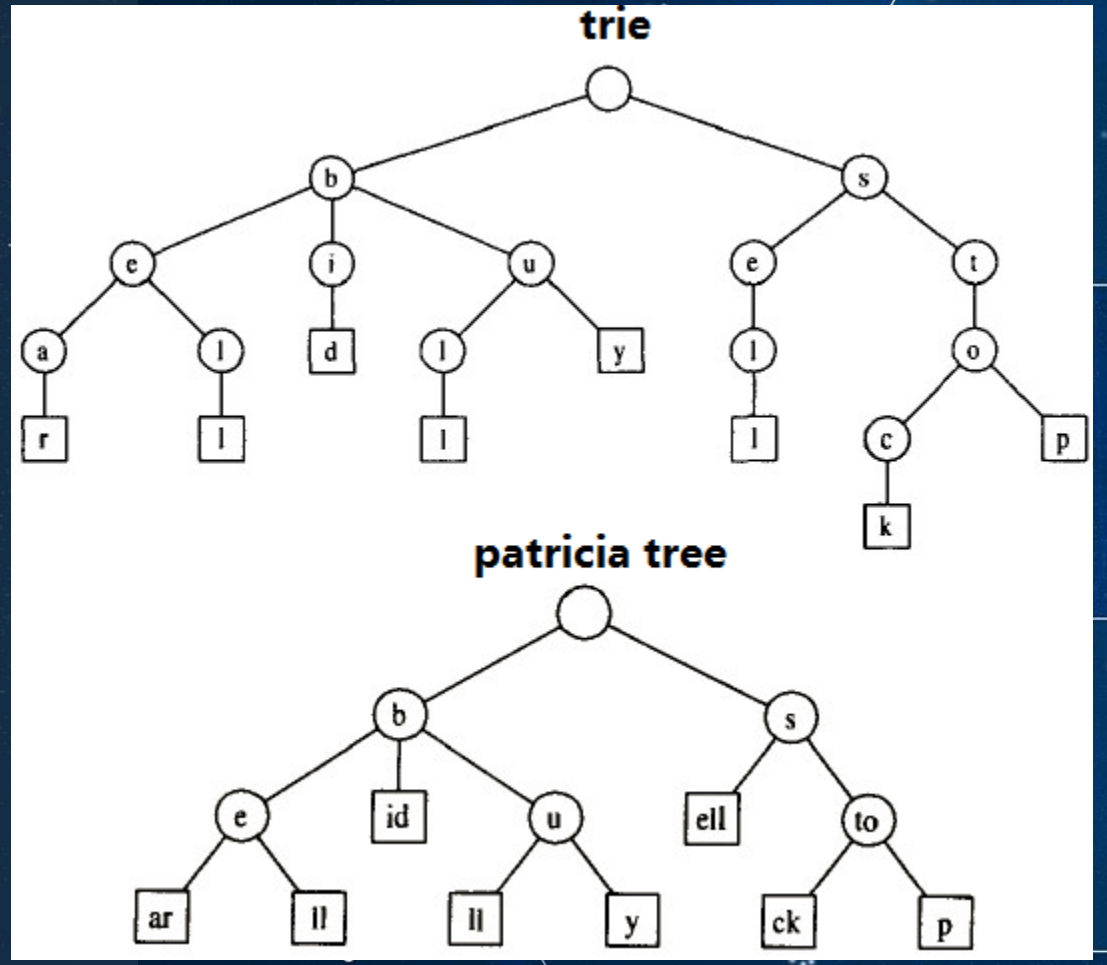




'a'=V1, 'ab' V3, 'b'=V2, 'ba'=V4, 'baa'=V5, 'zaaba'=V6

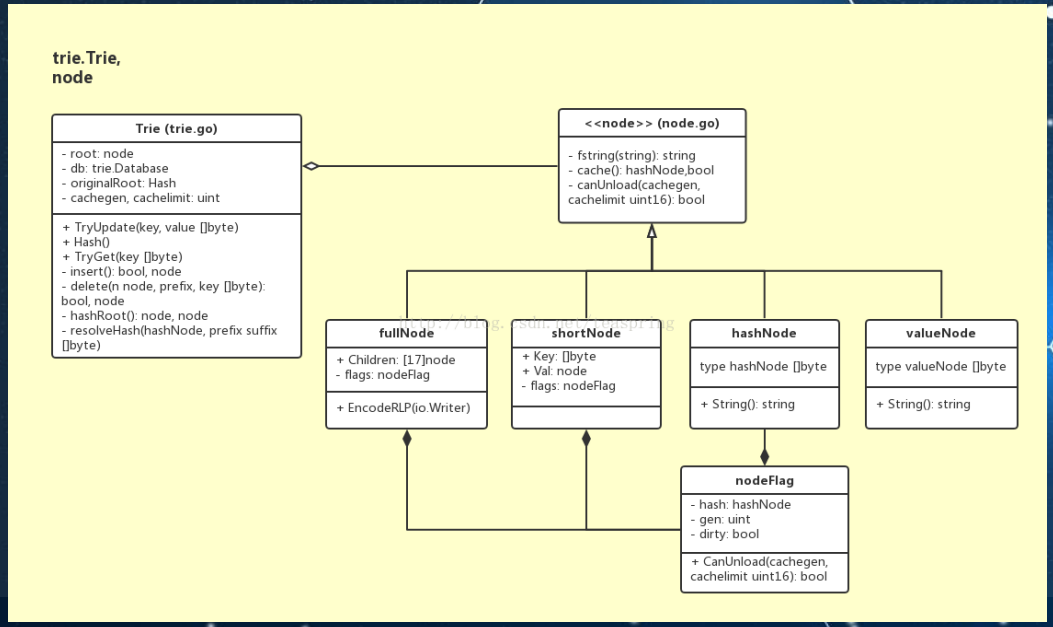
Trie树和Patricia树

- Patricia树是一种压缩的Trie树。
- 非根节点不是只能存储字符，而是可以存储字符串，也就是路径压缩了的Trie，因此节省了在内存空间的开销。

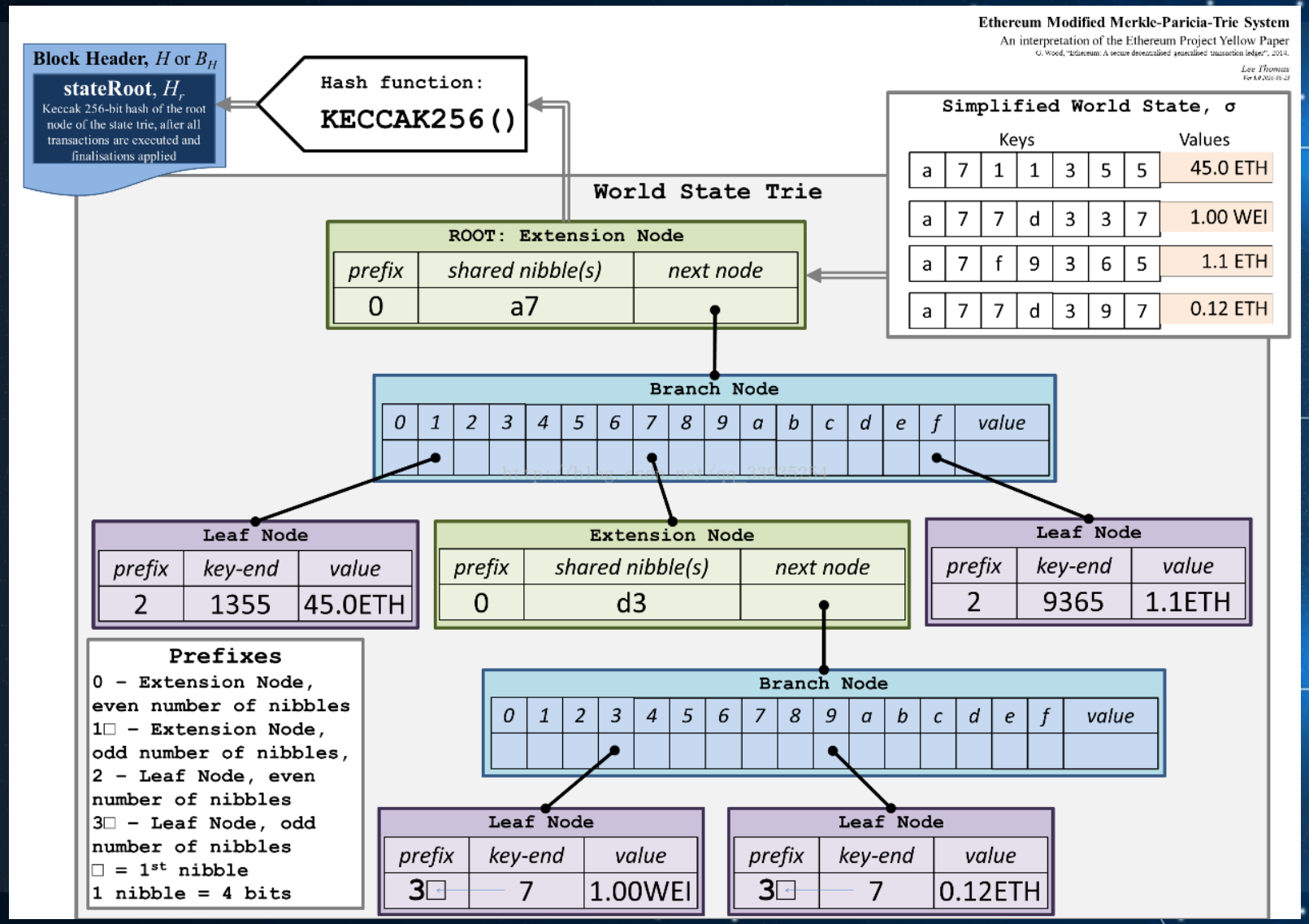


Merkle Patricia树

- MPT树结合了Patricia树和Merkle树的优点，在Patricia树中根节点是空的，而MPT树可以在根节点保存整棵树的哈希校验和，而校验和的生成则是采用了和Merkle树一致的生成方式。以太坊采用MPT树来保存交易、交易的收据以及世界状态。以太坊还使用了一种特殊的十六进制前缀(hex-prefix，HP)编码，使得字母表只有16个字符（一个字符称为一个nibble）。MPT树节点分成了四种类型：
  - 空节点（hashNode）
  - 叶节点（valueNode）
  - 分支节点（fullNode）
  - 扩展节点（shortNode）



以太坊的MPT树


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 45.0 ETH
a 7 7 d 3 3 7 1.00 WEI
a 7 f 9 3 6 5 1.1 ETH
a 7 7 d 3 9 7 0.12 ETH
World State Trie
ROOT: Extension Node
prefix
shared nibble(s)
next node
0
a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix
key-end
value
2
1355
45.0ETH
Extension Node
prefix
shared nibble(s)
next node
0
d3
Leaf Node
prefix
key-end
value
2
9365
1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits
Leaf Node
prefix
key-end
value
3□
7
1.00WEI
Leaf Node
prefix
key-end
value
3□
7
0.12ETH

状态树

- 状态树中的每个节点最多有16个孩子节点，每个叶节点表示一个账户，这些叶节点的父节点由叶节点的散列组成，而这些父节点再组成更高一层的父节点，直至到形成根节点。
- 状态树包含一个键值映射，其中键是账户地址，值是账户内容，主要是{nonce, balance, codeHash, storageRoot} 。
- 状态树代表发布区块后的整个状态。

交易树

- 每个区块都有一棵独立的交易树。区块中交易的顺序主要由“矿工”决定，在这个块被挖出前这些数据都是未知的。不过“矿工”一般会根据交易的GasPrice和nonce对交易进行排序。首先会将交易列表中的交易划分到各个发送账户，每个账户的交易根据这些交易的nonce来排序。每个账户的交易排序完成后，再通过比较每个账户的第一条交易，选出最高价格的交易，这些是通过一个堆(heap) 来实现的。每挖出一个新块，生成一个交易树。
- 在交易树包含的键值对中，其中每个键是交易的编号，值是交易内容。

收据树

- 每个区块都有自己的收据树，收据树不需要更新，收据树代表每笔交易相应的收据。收据树也包含一个键值映射，其中键是索引编号，用来指引这条收据相关交易的位置，值是收据的内容。
- 交易的收据是一个RLP编码的数据结构：[medstate，Gas_used，logbloom，logs]。其中，medstate是交易处理后树根的状态；Gas_used是交易处理后Gas的使用量；logs是表格[address，[topic1，topic2，…]，data]元素的列表，表格由交易执行期间调用的操作码LOG0 … LOG4生成（包含主调用和子调用），address是生成日志的合约地址，topic*n*是最多4个32字节的值，data是任意字节大小的数组； logbloom是交易中所有logs的address和topic组成的布隆过滤器（由Howard Bloom在1970年提出的二进制向量数据结构，它具有很好的空间和时间效率，被用来检测一个元素是不是集合中的一个成员）。区块头中也存在一个布隆过滤器，使用布隆过滤器可以减少查询的工作量，这样的构造使得以太坊协议对轻客户端尽可能的友好。

- 利用存储的三棵MPT 树，以太坊客户端可以轻松地查询以下内容：
  - 某笔交易是否被包含在特定的区块中。
  - 查询某个地址在过去的30天中发出某种类型事件的所有实例（例如，一个众筹合约完成了它的目标）。
  - 目前某个账户的余额。
  - 一个账户是否存在。
  - 假如在某个合约中进行一笔交易，交易的输出是什么。

数据存储LevelDB

- LevelDB是Google 实现的一个非常高效的键值对数据库，其中键值都是二进制的，可支持十亿级别的数据量。
- 以太坊中共有三个LevelDB数据库，分别是BlockDB、StateDB和ExtrasDB。
  - BlockDB保存了块的主体内容，包括块头和交易；
  - StateDB保存了账户的状态数据；
  - ExtrasDB保存了收据信息和其他辅助信息。

- 可以把三个MPT树理解为LevelDB所存储数据的索引，便于以太坊系统查询和处理这些数据。

以太坊的共识机制 PoW+PoS

- 以太坊共有四个阶段，即Frontier（前沿）、Homestead（家园）、Metropolis（大都会）、Serenity（宁静）。以太坊前三个阶段采用的是PoW共识机。第四个阶段将采用自己创建的PoS机制，名为Casper投注共识，这种机制增加了惩罚机制，并基于PoS的思想在记账节点中选取验证人。

PoW-Ethash算法

- Ethash是以太坊的PoW算法，使用了Dagger-Hashimoto算法的演化版本，混合了Vitalik Buterin提出的Dagger算法和Thaddeus Dryja 的Hashimoto 算法。
- 用Ethash算法来代替原有的PoW算法，是为了解决挖矿中心化问题。Ethash算法的特点是挖矿的效率基本与CPU无关，而与内存大小、带宽正相关，目的是去除专用硬件的优势，抵抗ASIC 。该算法的基本流程：
  ① 对于每一个区块，首先通过扫描区块头的方式计算出一个种子，该种子只和当前块的信息有关，然后根据种子生成一个16M的伪随机数据集（Cache）。
  ② 基于Cache生成一个1GB大小的数据集合DAG(有向无环图)，它是一个完整的搜索空间。
  ③ 挖矿的过程就是从DAG中随机选择元素(类似于比特币挖矿中查找合适Nonce)再进行哈希运算，可以从Cache快速计算DAG指定位置的元素，进而哈希验证。挖矿者存储数据集，该数据集周期性更新，每3000个块更新一次，并且规定DAG的大小随着时间推移线性增长，从1G开始，每年大约增长7G左右。
  ④ 验证者能够基于缓存计算得到DAG中自己需要的指定位置的元素，然后验证这些指定元素的散列是不是小于某个散列值，也就是验证”矿工”的工作是否符合要求。

PoS-Casper

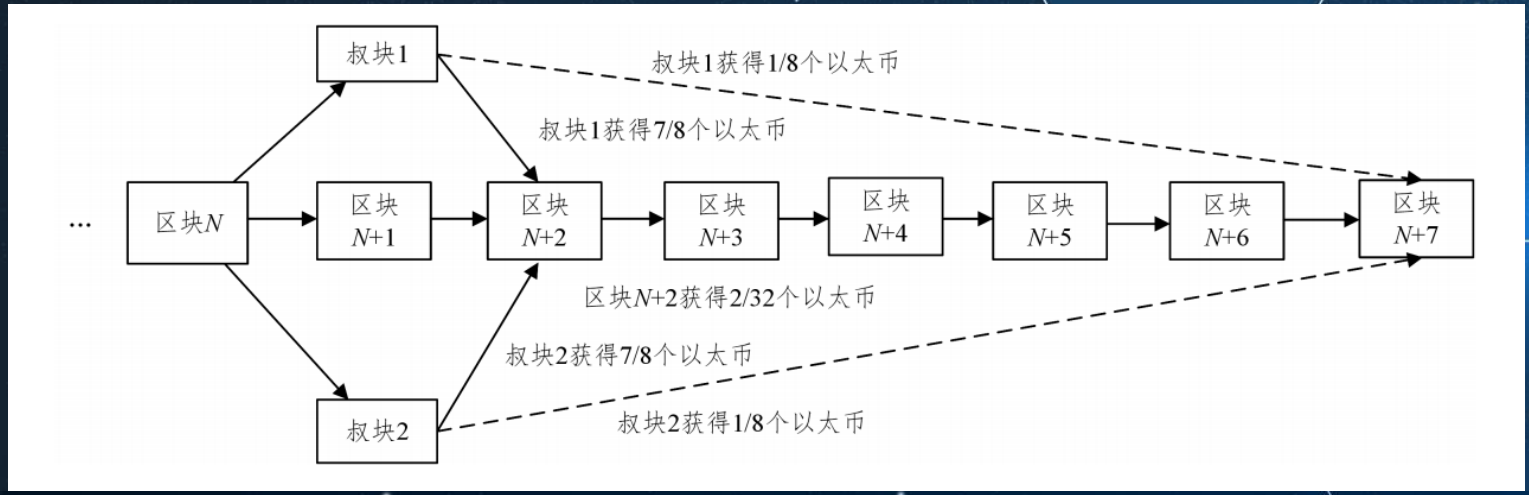
- 以太坊的PoS共识协议称为Casper，采用“权益”（Stake，即以太币）为记账权背书。
- 大致的思路是，将Casper的应用逻辑通过智能合约来实现，在这个合约中，记账权归属于“验证者” 。任何拥有以太币的账户都可以在Casper合约中成为验证者， 前提是必须要在Casper智能合约中抵押一定的以太币（抵押的以太币越多，被选中作为验证者的概率就越高）。之后Casper合约通过一种随机方式，选出一个验证者集合。被选中的验证者集合按照一定顺序依次验证区块（当然也可以选择放弃），如果区块没有问题，就将其添加到区块链中，同时相应的验证者将会获得一笔与他们的抵押成比例的奖励。如果验证者不遵守合约规定的规则，合约就会没收他抵押的以太币作为惩罚。

以太坊Ghost协议和叔块（Uncle Block）

- GHOST（Greedy Heaviest Observed Subtree）是一种主链选择协议（不是侧链选择协议）。GHOST协议是以包含子树数目最多为基本原则。
- GHOST协议，认为孤块也是有价值的，会给发现孤块的矿工以回报，也会给引用孤块的矿工以奖励。在以太坊中，孤块被称为“叔块”(uncle block)，它们可以为主链的安全作出贡献。
- 叔块上的交易不会得到执行。
- 叔块的好处：
  1. 以太坊十几秒的出块间隔，大大增加了孤块的产生，并且降低了安全性。通过鼓励引用叔块，使引用主链获得更多的安全保证(因为孤块本身也是合法的)
  2. 比特币中，采矿中心化成为一个问题。给与叔块报酬，可以一定程度上缓解这个问题。

以太坊中的叔块(uncle block)

- 区块可以引用0-2个叔块。叔块必须是区块的前2层~前7层的祖先的直接的子块，被引用过的叔块不能重复引用。
- 引用叔块的区块，可以获得挖矿报酬的1/32，也就是5*1/32=0.15625Ether。最多获得2*0.15625=0.3125Ether。
- 被引用的叔块，其矿工的报酬和叔块与引用它的区块之间的间隔层数有关系。



以太币ETH

- 以太币(ETH) 是以太坊发行的一种数字货币，是以太坊中一个重要元素，在公有链上发起任何一笔交易都需要支付一定的以太币。
- 以太币的总供给及其发行率是由2014年的预售决定的，以太币来源包括“矿前 + 区块奖励 + 叔块奖励 + 叔块引用奖励 “ 。具体的分配大致如下：
  - 预付款的贡献者总共有6000万个以太币。
  - 挖出一个新区块的矿工奖励为5个以太币。
  - 叔块被引用，每个叔块会为矿工产出大约4.375个以太币，矿工每引用一个叔块，可以得到大约0.15个以太币（最多引用两个叔区块） 。

以太币的货币单位

- 以太坊设定了一套以以太币为标准的货币单位。每个面额都有自己的命名术语，最小的货币单位为wei（维）。

- **Kwei(Babbage) = 10^3 Wei**
- **Mwei(Lovelace) = 10^6 Wei**
- **Gwei(Shannon) = 10^9 Wei**
- **Microether(Szabo) = 10^{12} Wei**
- **Milliether(Finney) = 10^{15} Wei**
- **Ether = 10^{18} Wei**

延伸阅读：https://zhuanlan.zhihu.com/p/28994731


7.3 智能合约

智能合约的定义

- 从用户角度看，“智能合约”是根据事先任意制订的规则来自动转移数字资产的系统。
- 从开发者角度看，“智能合约”就是存储在区块链上的代码，用以实现执行特定的功能。

- 例如，一个人可能有一个存储合约，形式为“A可以每天最多提现X个币，B每天最多Y个，A和B一起可以随意提取，A可以停掉B的提现权”

智能合约使用步骤

谢谢!