区块链与数字货币

浙江大学 杨小虎

2022年10月11日

04 以太坊技术分析

以太坊是什么？

- Ethereum is a technology for building apps and organizations, holding assets, transacting and communicating without being controlled by a central authority. There is no need to hand over all your personal details to use Ethereum - you keep control of your own data and what is being shared. Ethereum has its own cryptocurrency, Ether, which is used to pay for certain activities on the Ethereum network.

以太坊是什么？

Ethereum

The intent of Ethereum is to create an alternative protocol for building decentralized applications, providing a different set of tradeoffs that we believe will be very useful for a large class of decentralized applications, with particular emphasis on situations where rapid development time, security for small and rarely used applications, and the ability of different applications to very efficiently interact, are important. Ethereum does this by building what is essentially the ultimate abstract foundational layer: a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions. A bare-bones version of Namecoin can be written in two lines of code, and other protocols like currencies and reputation systems can be built in under twenty. Smart contracts, cryptographic "boxes" that contain value and only unlock it if certain conditions are met, can also be built on top of the platform, with vastly more power than that offered by Bitcoin scripting because of the added powers of Turing-completeness, value-awareness, blockchain-awareness and state.

以太坊白皮书

以太坊是什么？

ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER

BERLIN VERSION 8fea825 – 2022-08-22

DR. GAVIN WOOD
FOUNDER, ETHEREUM & PARITY
GAVIN@PARITY.IO

ABSTRACT. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, with Bitcoin being one of the most notable ones. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

以太坊黄皮书

Gavin Wood

- 拥有约克大学计算机博士学位以及音乐可视化博士学位，掌握英语、 意大利语、法语、西班牙语、罗马尼亚语等语言。
- 以太坊主要创始人，设计了 C++ 版以太坊客户端原型、编写《以太坊黄皮书》（曾有人指出，“预计全世界能直接看懂这份黄皮书的人加起来不足一百个。”），并为智能合约开发高级语言Solidity。
- 2015年底，Gavin Wood创立Parity Technologies，推出了广为人知的以太坊客户端Parity。 该应用由Rust语言编写，其性能是Geth和C++客户端的数倍，Wood成为了第一个用Rust编写以太坊客户端的程序员。
- 提出Polkadot项目，提出Web3.0概念。

以太坊发展历史

2013年末，初版白皮书发布，项目启动。2014年3月开始陆续发布了3款测试网络。2014年4月发布了以太坊黄皮书。2014年7月24日起，以太坊进行了为期42天的以太币预售。

是以太坊的最初版本，于2015年7月30日发布，不是一个完全可靠和安全的网络。

2016年3月14日，Homestead版本发布，表明以太坊网络已经平稳运行。在此阶段，以太坊提供了图形界面的以太坊钱包。

2017年10月。以太坊将这一阶段分为两个版本，分别为Byzantium和Constantinople。该阶段旨在使以太坊更轻量、快速和安全。

又称以太坊2.0，把共识机制从PoW转换到PoS。第一次升级工作将在2020年12月2日进行，被称为“阶段0”，部署的是信标链，2022年9月15日完成升级。

以太坊的组成

- **P2P 网络**

  以太坊以P2P方式进行网络通信，通过TCP端口30303访问。

- **交易 Transaction**

  以太坊交易是网络消息，包括转账交易、合约交易等。

- **状态机 State Machine**

  以太坊的状态转移由 以太坊虚拟机（EVM） 处理，这是一个执行 bytecode（机器语言指令）基于栈的虚拟机。称为“智能合约”的EVM程序以高级语言（如Solidity）编写，并编译为字节码以便在EVM上执行。

- **区块链账本**

  以太坊的区块链账本存储在每个节点上，该区块链在Merkle Patricia Tree 的序列化哈希数据结构中包含交易和系统状态。

- **共识算法**

  以太坊1.0使用名为Ethash的工作量证明算法，以太坊2.0过渡到称为Casper的权益证明机制（Proof-of-Stake）。

- **客户端**

  以太坊有几个可互操作的客户端软件实现，最著名的是 Go-Ethereum（Geth）和Parity

以太坊技术架构

关键词：

- 存储
- 数据、区块、交易
- P2P网络、协议
- 共识机制
- 智能合约
- 虚拟机
- 分布式应用软件（Dapp）

以太坊基本概念

以太坊由大量的节点组成，节点有账户与之对应，两个账户之间通过发送消息进行一笔交易。交易里携带的信息和实现特定功能的代码叫做智能合约，运行智能合约的环境是以太坊虚拟机（EVM），EVM类似于Java虚拟机JVM，编译后基于字节码运行，开发时则可以使用高级语言实现，编译器会自动转化为字节码。基于以上的智能合约代码和以太坊平台的应用叫做去中心化应用（DApp，Decentralized Application）。

- **节点**　以太坊是由网络上的许多节点组成的，每一个节点都运行着以太坊虚拟机，通过节点可以进行区块链数据的读写。节点之间使用共识机制来确保数据交互的可靠性和正确性。
- **账户**　以太坊中包含两类账户，包括外部账户和合约账户。外部账户由公私钥对控制，合约账户则在区块链上唯一的标识了某个智能合约。
- **交易**　交易包括转账交易和合约交易，通过状态转移来标记。状态由账户之间的转移价值和信息状态转换沟通。

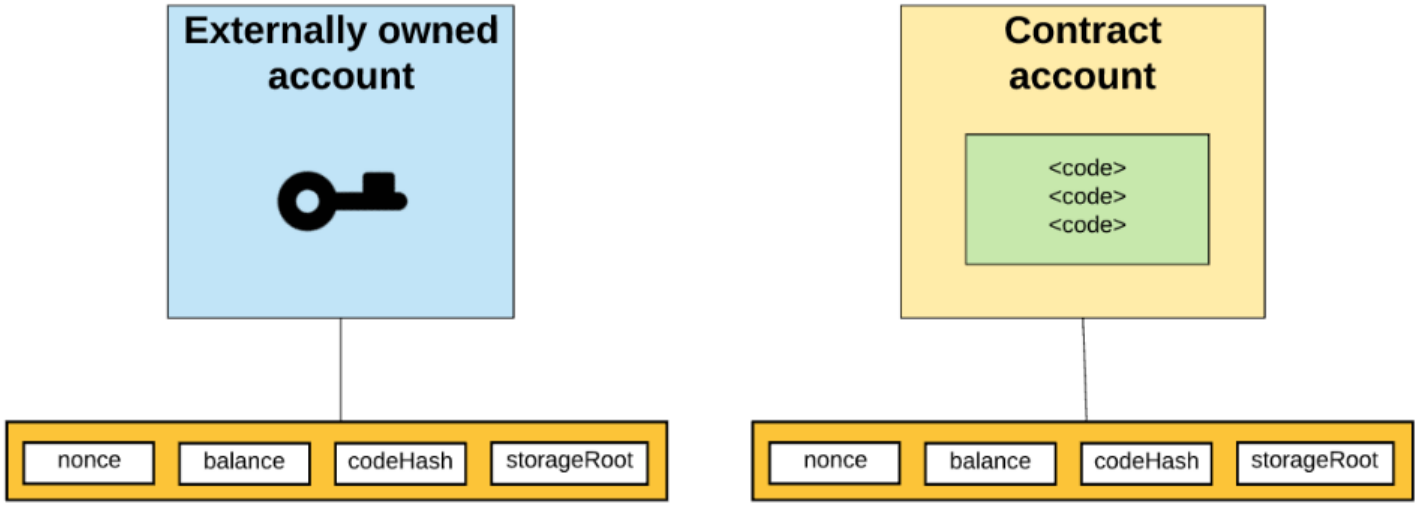
以太坊基本概念（续）

- **智能合约**　合约是代码和数据的集合，存在于以太坊区块链的指定地址。合约方法支持回滚操作，如果在执行某个方法时发生异常（如gas消耗完），则该方法已经执行的操作都会被回滚。但是如果交易一旦执行完毕，是没有办法篡改的
- **EVM**　以太坊虚拟机是以太坊中智能合约的运行环境，并且是一个沙盒，与外界隔离。智能合约代码在EVM内部运行时，是不能进行网络操作、文件I/O或执行其它进程的。智能合约之间也只能进行有限的调用。
- **矿工与挖矿**　矿工是指通过不断重复哈希运算来产生工作量的网络节点。在以太坊中，发行以太币的唯一途径是挖矿。
- **gas**　以太坊上的每一笔交易都有矿工的参与，且都需要支付一定的费用，这个费用在以太坊中称为gas。gas的目的是限制执行交易所需的工作量，同时为执行交易支付费用。

外部账户：由区块链外部主体创建，拥有一对公私钥，账户地址没有合约代码。

合约账户：由合约交易创建，账户拥有合约代码，地址是代码的hash值。

- **nonce**: 对外部账户，代表该账户发出的交易数，对合约账户，表示该账户创建的合约数量。.
- **balance**: 该账户地址的Wei，10^{18} Wei = 1 Ether
- **storageRoot**: Merkle Patricia树的根节点hash值 ， 该树是该账户下存储信息的hash值，缺省为0.
- **codeHash**: 该账户的EVM代码hash值，对于合约账户，是合约代码存储的hash值，对外部账户，是空字符串的hash值。



外部账户和合约账户的区别：

外部账户

•创建外部账户没有成本

•可以启动交易

•两个外部账户间的交易只能是以太币转账交易

•由公私钥对控制

合约账户

•创建合约账户需要成本，因为使用了网络存储

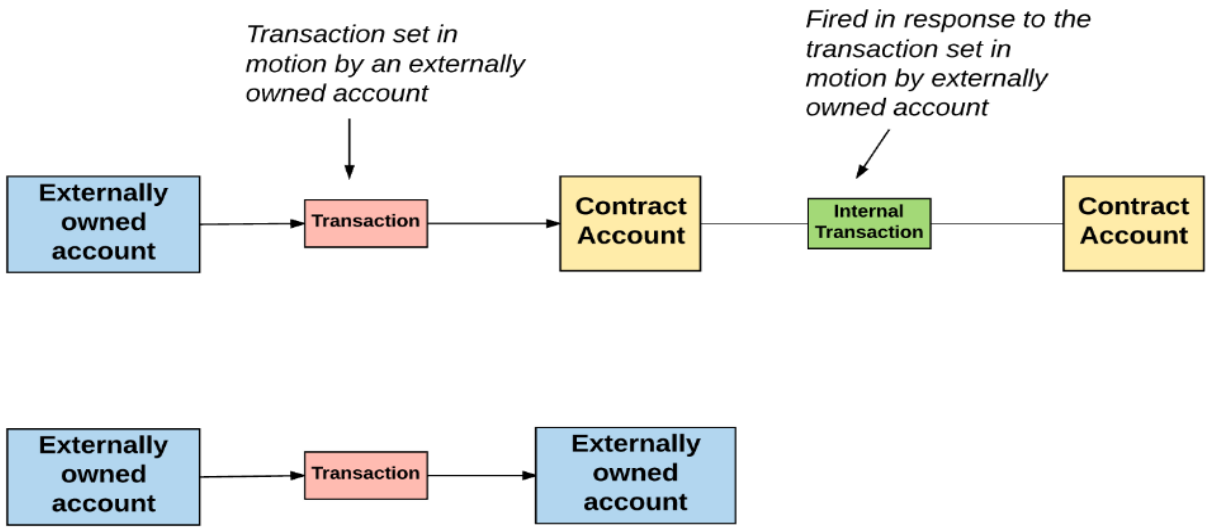
•只能在收到一个交易作出响应而发出一个交易

•从一个外部账户到一个合约账户的交易可以触发合约账户上的代码，执行代码中的各种动作，例如代币转账、创建新合约等

•合约帐户没有私钥，相反，它们由智能合约代码的逻辑控制

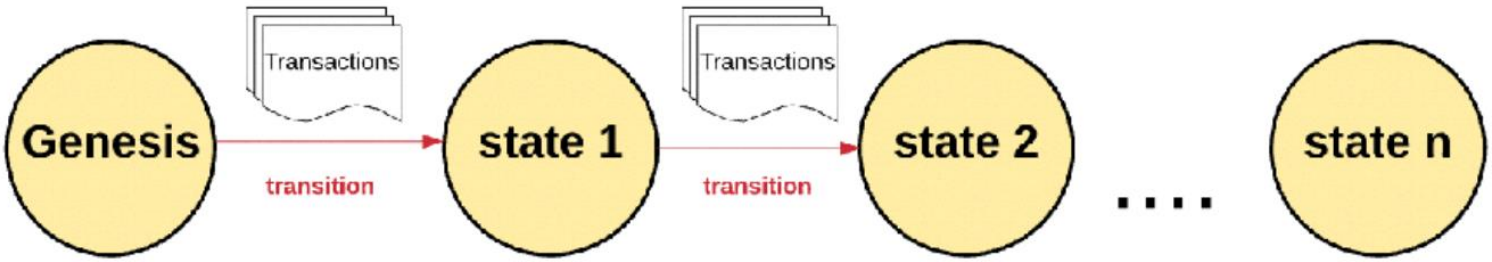
账户间的消息与交易

- 外部账户可以向其他外部账户发送消息，或向其他合约账户发起一个合约交易。
- 两个外部账户间的消息就是一个简单的转账交易。
- 从一个外部账户到一个合约账户的消息可以触发合约账户上的合约代码执行。
- 合约地址自身不能发起一个新的交易，只能在接收到其他外部账户或合约账户发起的交易后作出反应而产生内部交易。



世界状态

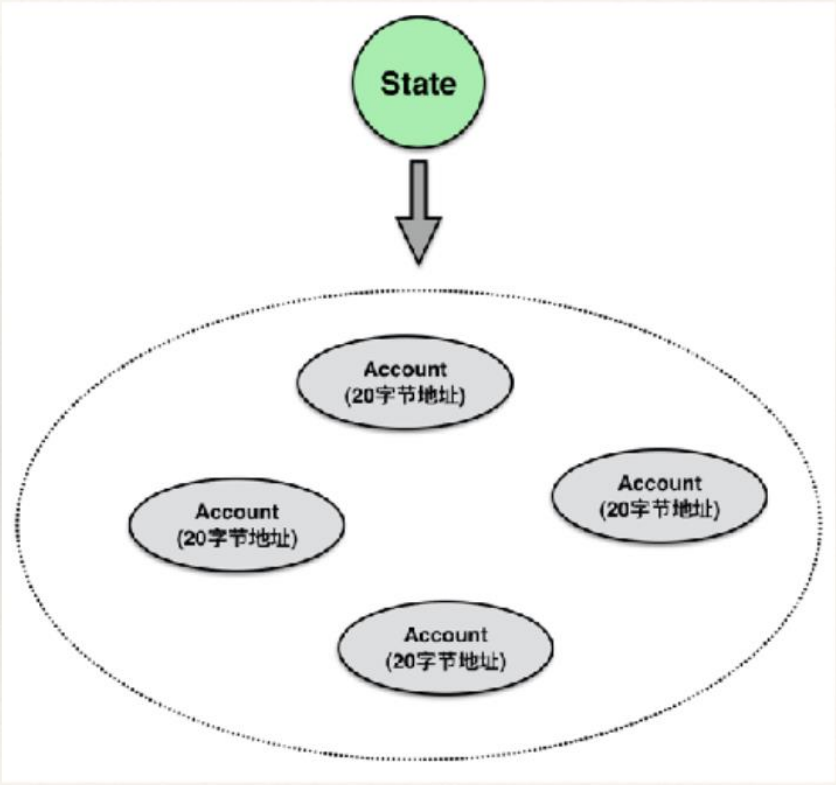
- 世界状态 world state：所有账户（包括外部账户和合约账户）的状态合集。



- 以太坊本质上是一个基于交易的状态机
- 以太坊有一个初始状态我们称为Genesis
- 状态转换的最小单元是交易(原子性、一致性)
- 每次执行一条或者多条交易后发生状态转换

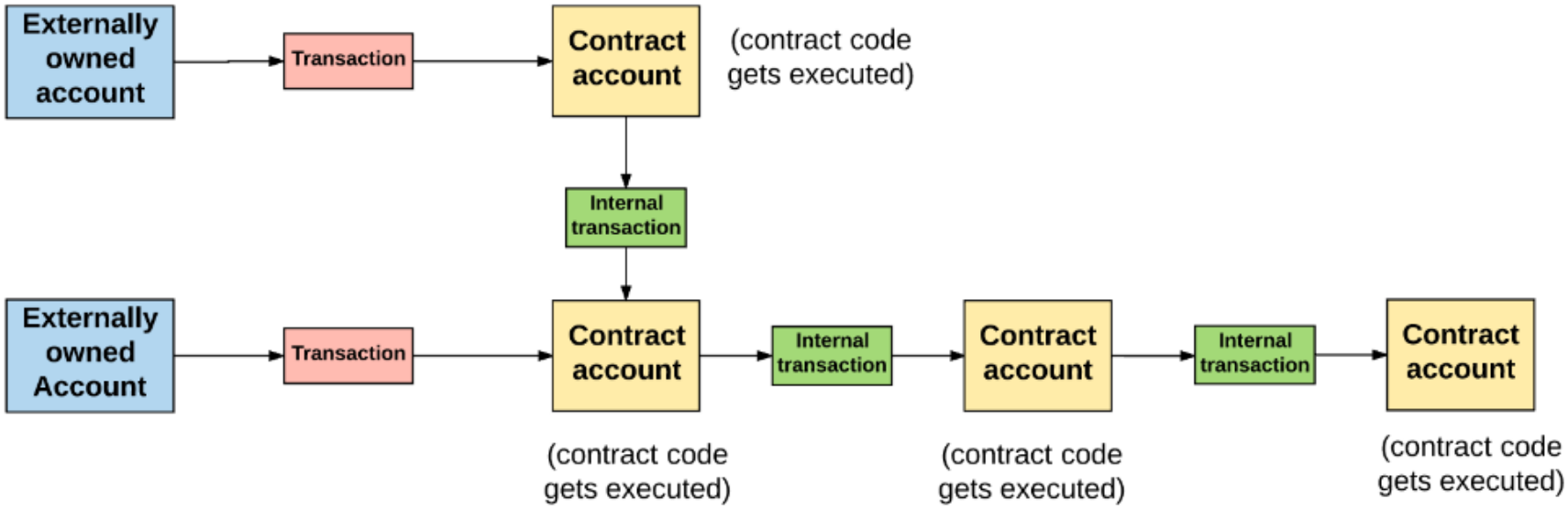
世界状态

- “世界状态”是由一系列“账户”所组成
- 每一个账户都有一个唯一的地址
- 地址的产生规则分为两类
  - 由账户对应的公钥计算所得
  - 由部署者的信息计算所得



以太坊交易举例

- 从A账户转x个以太币到B账户
- 发布智能合约代码
- 用参数Y调用在地址X上的智能合约



智能合约举例

- 外部账户A
  - 发布2个合约
    - Function PlaySong()，参数是调用者网名
    - Function Song1Here()
- 某程序员开发一个DApp，输入账户地址、签名后可播放链上歌曲SongHere，每次听歌费用0.0001ETH
- 用户C运行DApp时，DApp发起一个合约调用交易
  - 由用户C的账户发起，先确定是否有足够余额支付听歌费用和合约运行gas
  - 合约调用交易：
    - Function PlaySong()的codehash，参数（用户网名）
    - 再触发合约调用交易 调用Function Song1Here

交易的类型

- **简单支付交易** - 以太币的账户间转账，从知己账户转到对方账户，不涉及智能合约，无需动用以太坊虚拟机，“耗油”也最少。
- **存证交易** – 在简单支付交易中把支付额设置成0，需要存证的内容写在data字段中，就成了存证交易。当然，也可以为存证机制专门部署一个智能合约，以后要存证时就调用这个合约，这样可以为存证增添一些附加的操作。因为是0支付，对方账户就无关紧要，但不能是0。
- **合约部署交易** - 将对方地址设置成0，就可以将智能合约的程序部署到以太坊网络中，实际上是所有的验证节点上。一旦交易记录进入区块链即部署生效，以太坊网络会在交易“收据”中返回新建合约账户的地址。
- **合约调用交易** - 对智能合约的调用，依具体智能合约的不同，又可以分为以下几种：
  - 复杂支付交易，更确切地说是数字资产转移交易，这是对智能合约中代表着数字资产的Token的转移。如果用Token实现各种虚拟货币，这样的转移就成了采用某种自定义虚拟货币的支付。
  - 查询交易，查询是区块链网络中常用的操作。以太坊网络中提供了若干“系统合约”，即由系统提供、无需用户部署的合约，其中之一就是用于查询。
  - 其它（智能合约调用）交易，如在电子商务，电子政务等领域的应用。

交易的数据结构

class **Transaction** {}

] byte[] hash; //注意这是对RLP编码之后的交易请求Tx的Hash值。

] byte[] nonce; //实质上是Tx的序号，用以防止对同一交易请求的重复处理。

] byte[] value; //支付的币值，Wei是以太币的最小单元。

] byte[] receiveAddress; //对方地址，这地址可以是账户地址或合约地址。

] byte[] gasPrice; //油价

] byte[] gasLimit; //可以付出的最大油量

] byte[] data; //见下页解释

] Integer chainId; //子网/子链ID

] ECDSASignature signature; //签名

] byte[] sendAddress; //发出方地址

交易的数据结构：data字段作用

- 在支付交易中，可以用这个字段作付款说明。
- 如果把支付额设置成0（收款方地址不能为0），就可以用来存证。
- 在合约部署交易中（以0作为对方地址），这个字段的内容就是所要部署到以太坊网络中的合约。
- 在合约调用交易中，用这个字段传递调用函数名和参数。

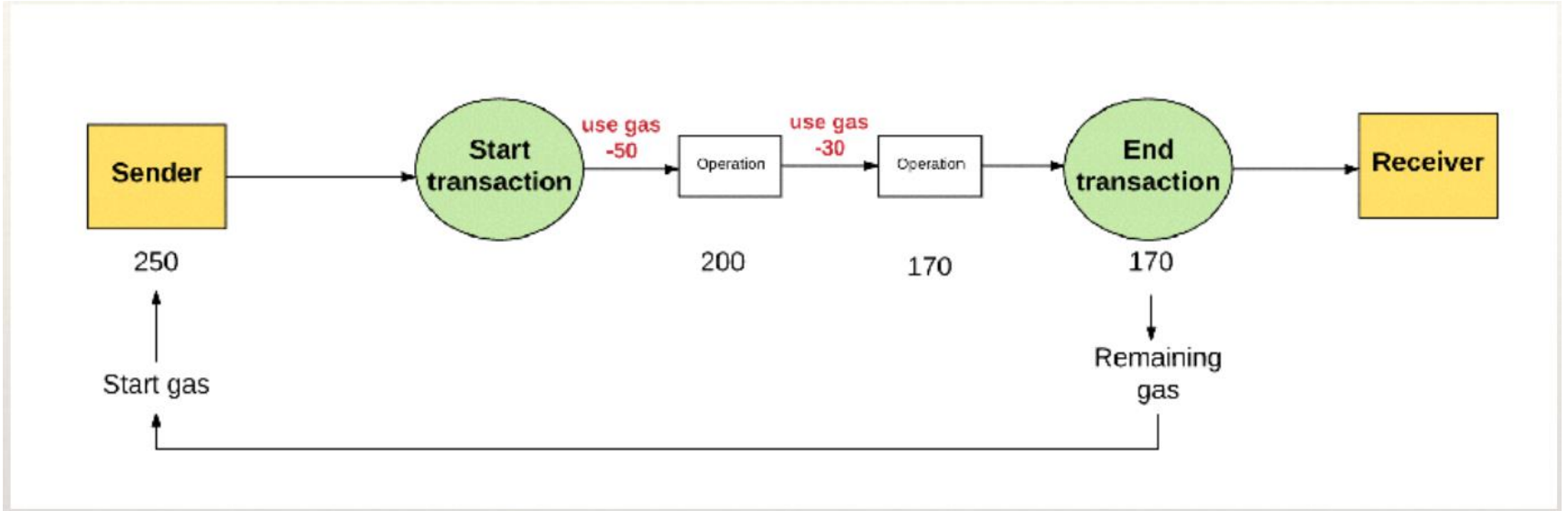
Gas

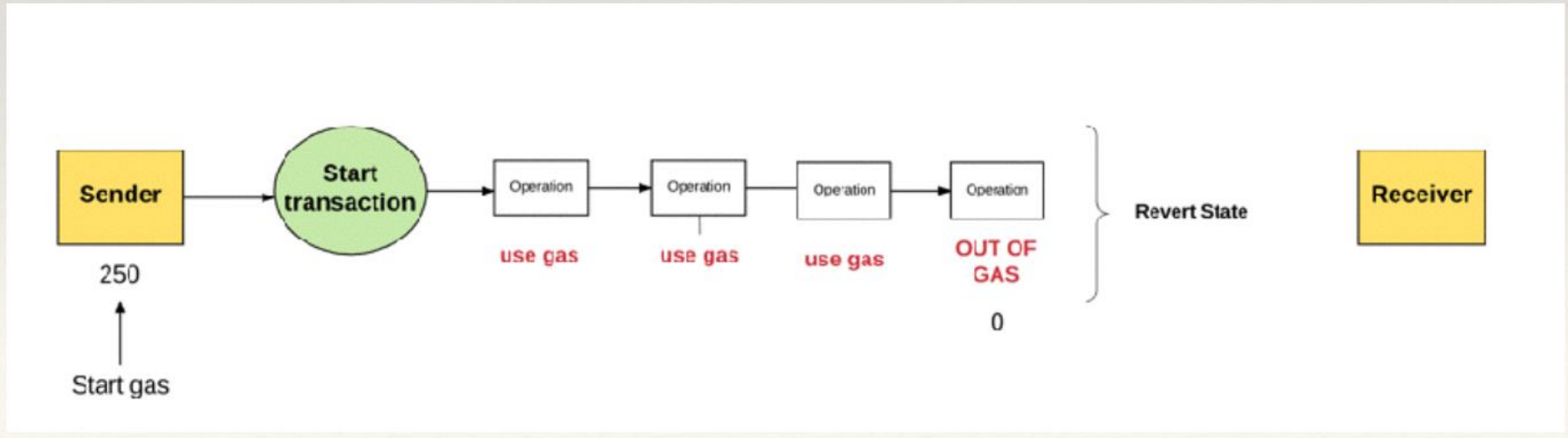
- 在以太坊上，任何引起状态转移的操作都是需要收费的
  - 数学运算
  - 状态存储
- Gas是用来计量以太坊系统资源使用情况的最小计量单位
  - 状态转移中所有的动作都有一个复杂度的衡量公式
  - Add操作花费3个Gas，SStore操作花费20000个Gas
- 一次交易执行过程，累积消耗Gas超过发送者预付的总量，交易执行失败

Gas

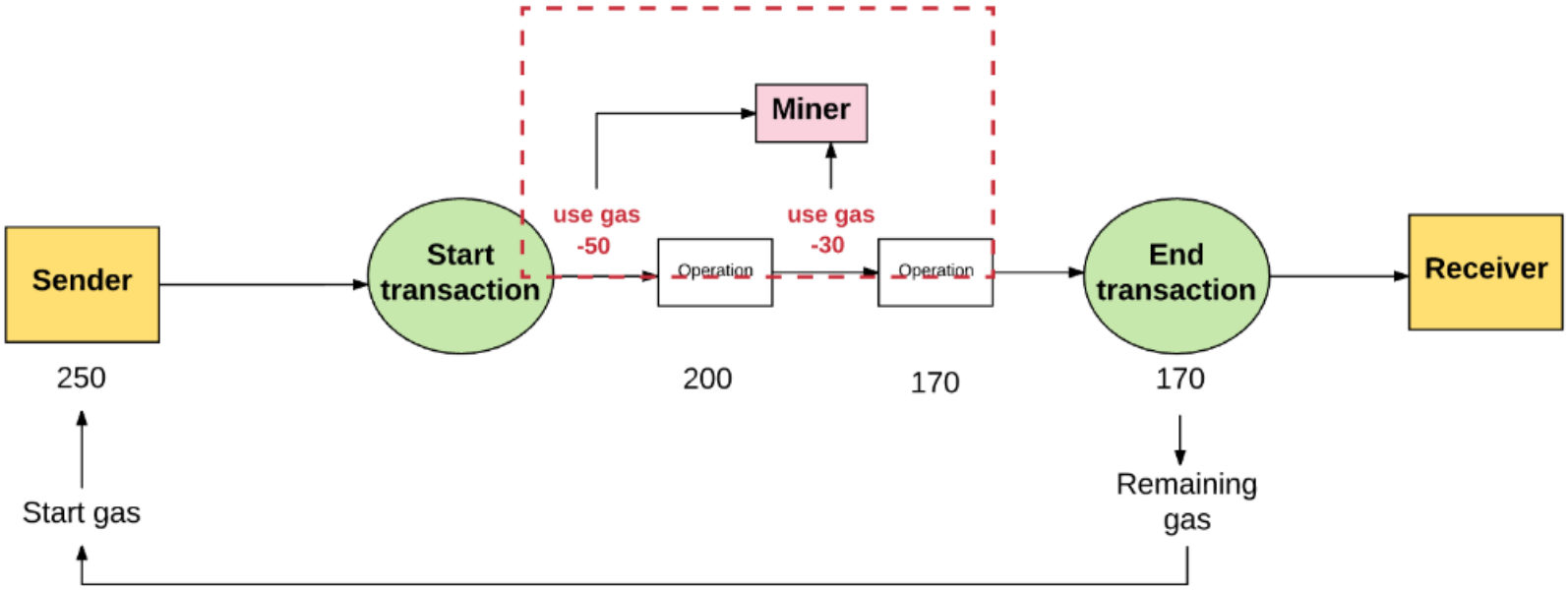
指令	Gas Used	Notes
STOP	0	停止执行
ADD	3	两个数相加
MUL	5	两个数相乘
SUB	3	两个数相减
DIV	5	两个数相除

Gas


Sender
Start transaction
use gas -50
Operation
use gas -30
Operation
End transaction
Receiver
250
200
170
170
Start gas
Remaining gas


Sender
Start transaction
Operation
Operation
Operation
Operation
Revert State
Receiver
250
use gas
use gas
use gas
OUT OF GAS
0
Start gas

Gas


Miner
use gas
-50
use gas
-30
Sender
Start
transaction
Operation
Operation
End
transaction
Receiver
250
200
170
170
Start gas
Remaining
gas

Gas

- GasPrice表示发送者预付的Gas价格
- Fee = Gas * GasPrice
- 发送者必须有足够多Ether余额来支付交易费用
- 交易所产生的手续费作为Block Producer的经济激励

Gas

不仅仅计算需要付费，存储也需要付费

存储费用是32字节的最小倍数。

因为以太坊状态数据库在每个节点均有存储，为鼓励较少的存储数据量，如果一个交易清除存储中的一项，这项操作的执行费用被免除，并有奖励费用。

交易的数据组成（最新）

- recipient – 接收者地址 (如果是外部账户，就转币值，如果是合约账户，就执行该地址上的合约)
- signature – 发送者签名
- nonce - 该账户发起的交易计数器
- value – 转账的币值(in WEI)
- data – 可放置任意数据
- gasLimit – 本交易可消耗的gas上限
- maxPriorityFeePerGas - 给验证节点的最大小费单价
- maxFeePerGas - 支付交易的最大费用单价（包括 baseFeePerGas 和 maxPriorityFeePerGas）

base fee, priority fee, max fee

- base fee：该区块内所有交易的最小gas单价，由以太坊软件根据前一区块的gas费用计算调整

Block Number	Included Gas	Fee Increase	Current Base Fee
1	15M	0%	100 gwei
2	30M	0%	100 gwei
3	30M	12.5%	112.5 gwei
4	30M	12.5%	126.6 gwei
5	30M	12.5%	142.4 gwei
6	30M	12.5%	160.2 gwei
7	30M	12.5%	180.2 gwei
8	30M	12.5%	202.7 gwei

- priority fee：交易发起者愿意支付的小费gas单价
- max fee：发起者愿意支付该交易的最大gas单价
  - max fee >= base fee + priority fee

转账交易举例

```
{
  from:
      "0xEA674fdDe714fd979de3EdF0F56AA9716B898ec8",
  to: "0xac03bb73b6a9e108530aff4df5077c2b3d481e5a",
  gasLimit: "21000",
  maxFeePerGas: "300",
  maxPriorityFeePerGas: "10",
  nonce: "0",
  value: "10000000000"
}
```

转账交易举例

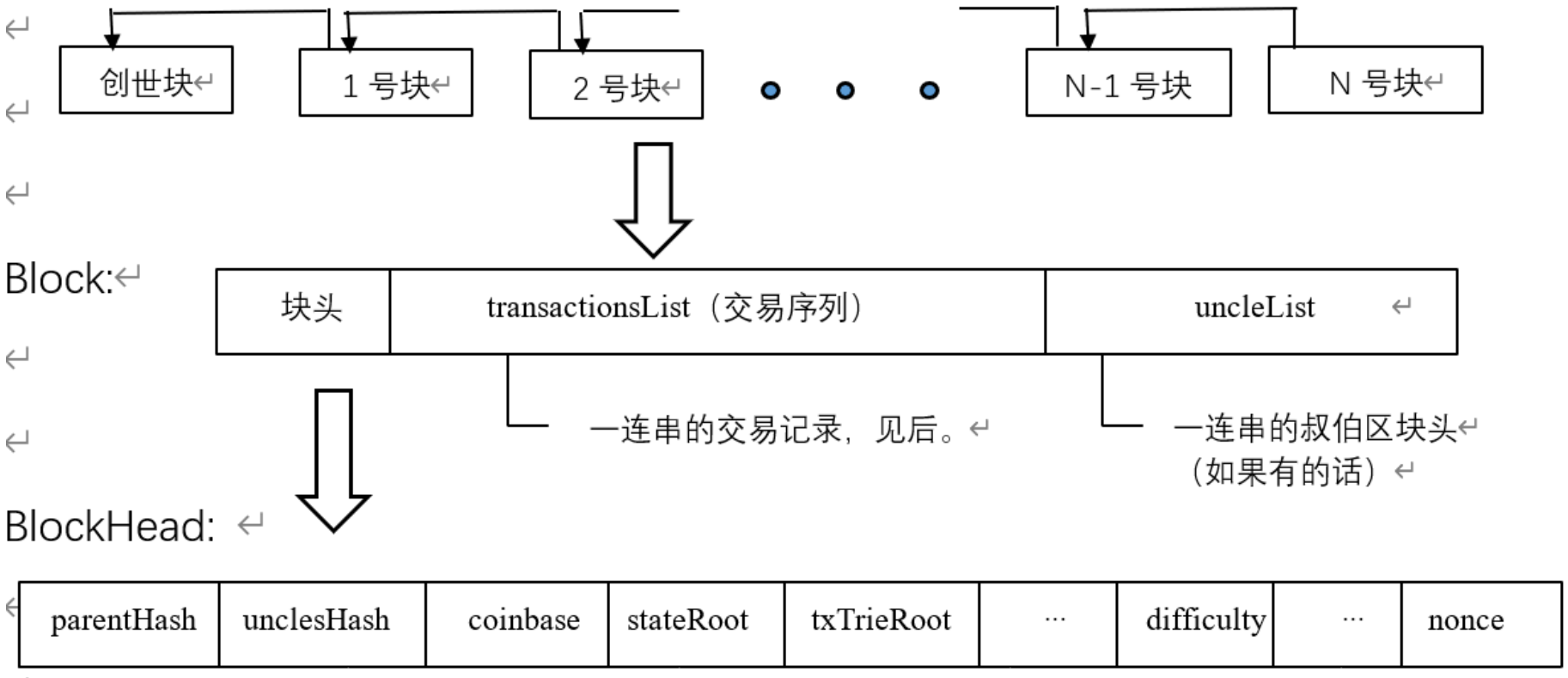
```
(190 + 10) * 21000 = 4,200,000 gwei
--or--
0.0042 ETH
```

发送者账户减去1.0042 ETH

接收者账户收到 1 ETH

Gas费用 0.0042 ETH，其中基本gas 0.00399ETH，小费0.00021ETH

以太坊区块链结构


创世块
1 号块
2 号块
N-1 号块
N 号块
Block:
块头
transactionsList（交易序列）
uncleList
一连串的交易记录，见后。
一连串的叔伯区块头
（如果有的话）
BlockHead:
parentHash
unclesHash
coinbase
stateRoot
txTrieRoot
…
difficulty
…
nonce

区块数据结构（PoW共识版本）

class **Block** {}　　//Block类是一个数据结构，内含下面这些结构成分：
] BlockHeader **header**;　　//块头
] List<Transaction> **transactionsList**　//交易记录序列
] List<BlockHeader> uncleList　　//叔伯块的块头序列

class **BlockHeader** {}　　//最长可达800字节
] byte[] parentHash;　　//前导块（父块）的块头Hash值
] byte[] unclesHash;　　//块身中uncleList的Hash值
] byte[] coinbase;　　//表示本区块的Coinbase和手续费应该给谁，其160位地址。
] byte[] stateRoot;　　//执行完本块所含全部交易后的状态树Hash值。
] byte[] txTrieRoot;　//块身中所有交易记录的树根Hash值。
] byte[] receiptTrieRoot;　//各个交易收据所构成树根的Hash值。
] byte[] difficulty;　//挖矿难度，用以调整发块周期长度
] long timestamp;　//时戳
] long number;　　//本块在区块链中的高度，即区块链中处于本块之前的区块个数。
] byte[] gasLimit;　//本块所含所有交易所提供“汽油”量即手续费的上限总和。
] long gasUsed;　　//本块所含所有交易实际消耗“汽油”量的总和。
] byte[] extraData;　//有关本区块的任意额外数据，不超过32字节。
] byte[] nonce;　　//

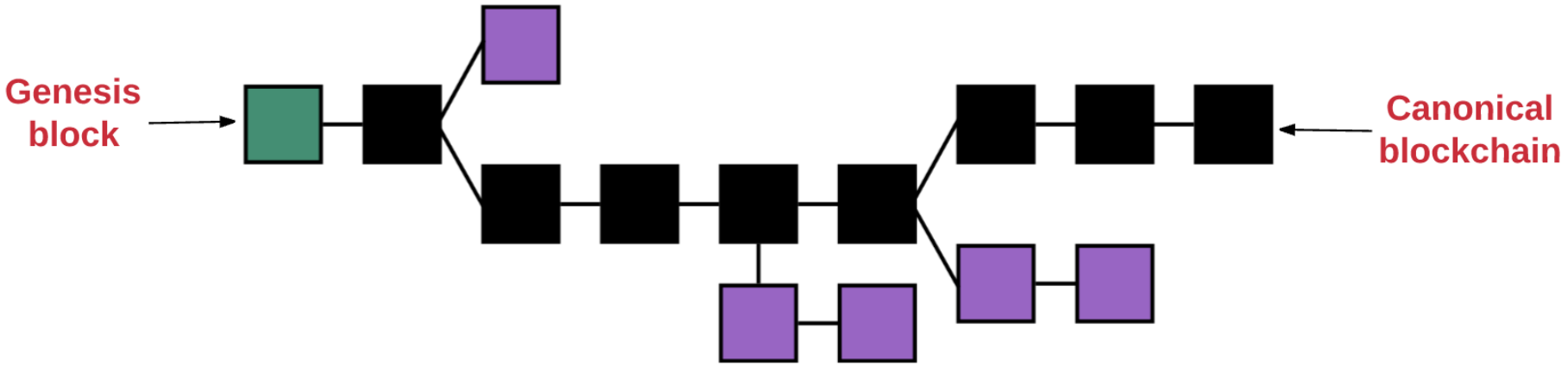
区块大小和出块速度

- 区块大小可变，每个区块的目标大小是容纳至少消费15M gas的交易，但区块的大小将根据网络需求增加或减少，上限是消费30M gas的交易（目标区块大小的2倍）。

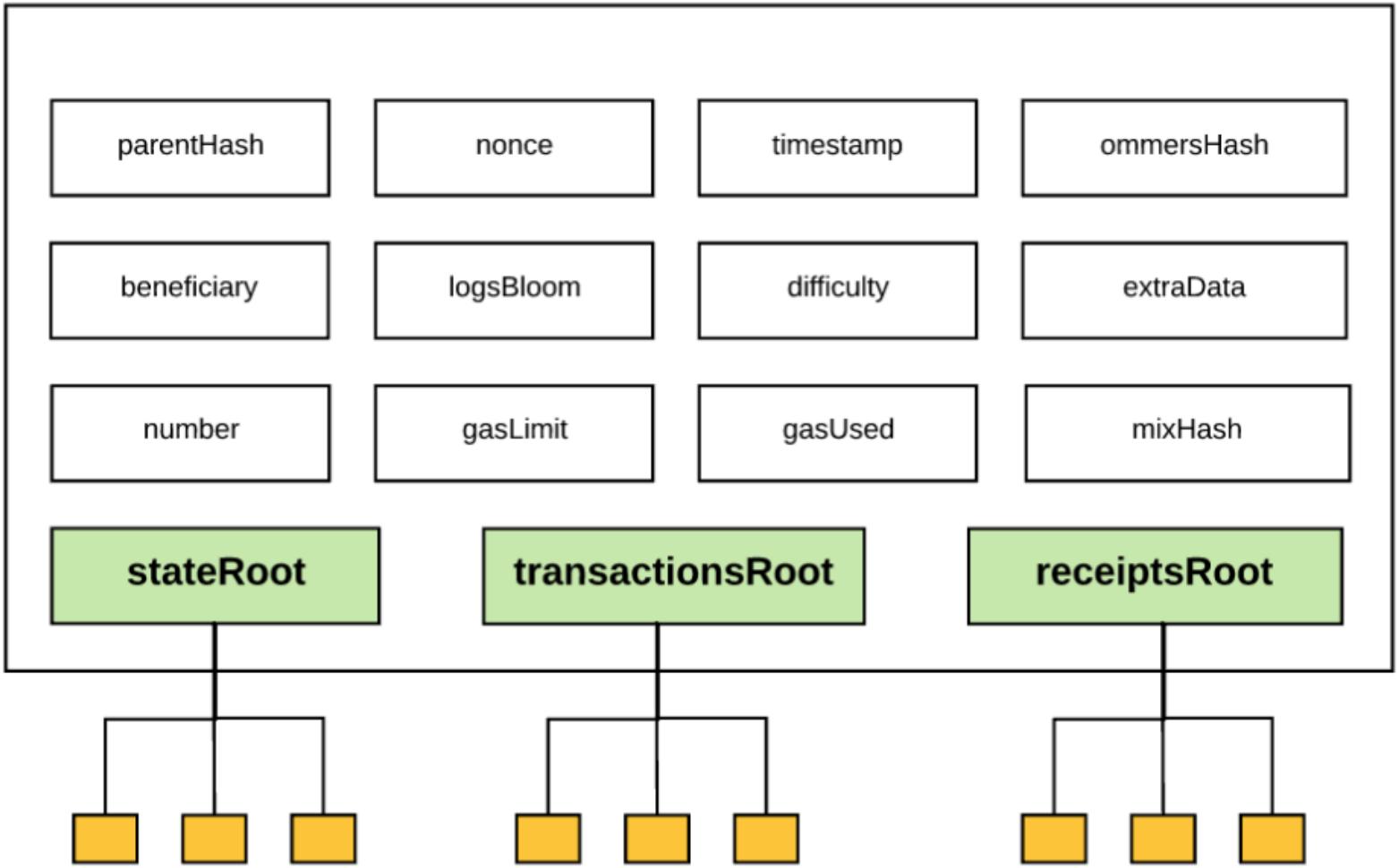
- PoW共识时的出块时间平均为15秒
  - Hash函数会产生较多的碰撞，产生较多的区块链分叉

以太坊区块链分叉应对措施

- GHOST — Greedy Heaviest Observed Subtree
  - 挑选计算量最大的那条路径



Block header



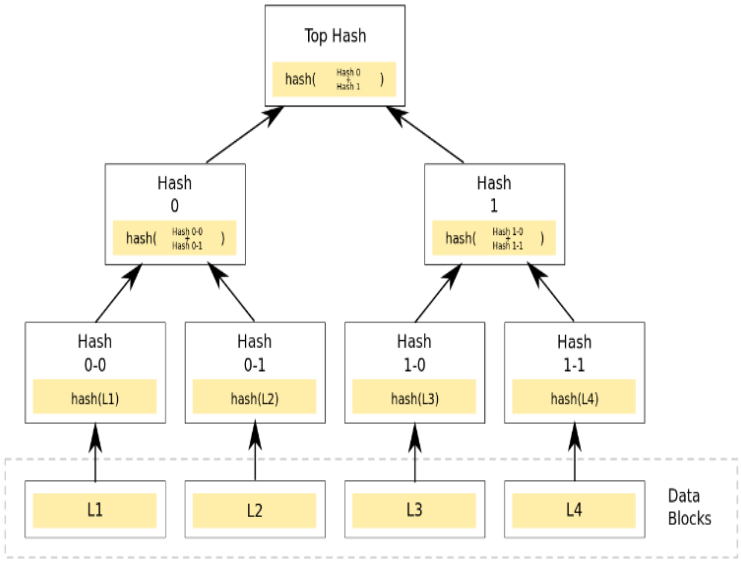
回执

- **回执包含的信息:**
  - 区块编号
  - 区块哈希
  - 交易哈希
  - 当前交易用掉的gas
  - 当前交易执行后当前区块中使用的累计gas
  - 执行当前交易时创建的日志

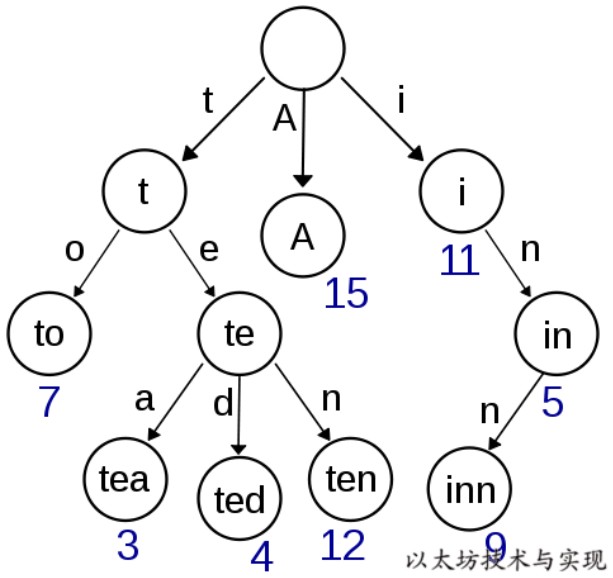
日志

- 对于以太坊上部署的智能合约来说，外部拥有账户所发起的交易，是链下世界对链上世界的输入
- 智能合约必须也需要某种途径把链上世界的信息传递出去 - 日志
- 合约编码者可以在智能合约中定义Event
- 当智能合约运行过程中执行该语句，便会产生一个虚拟机日志，并且将其存储在回执中

以太坊的数据存储：Merkle Patricia Trie

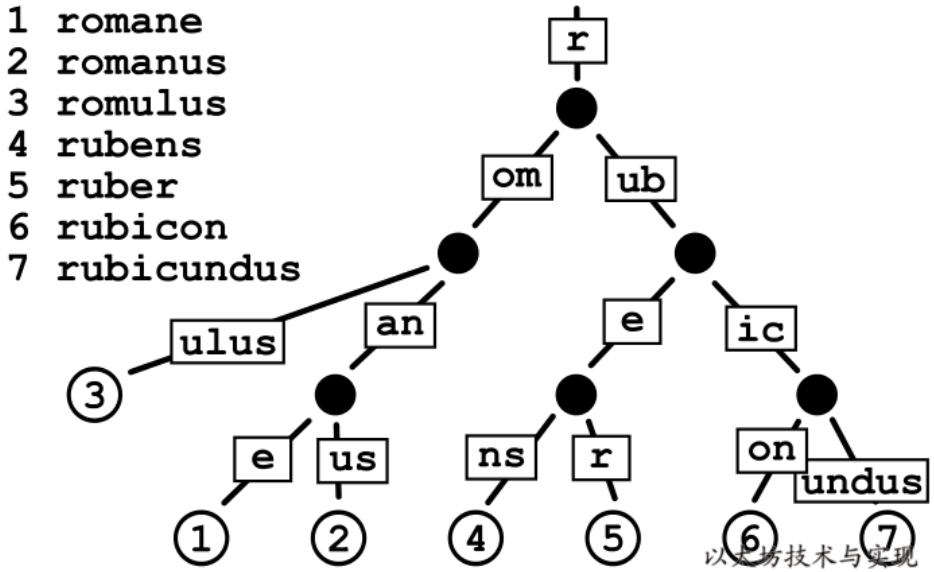


Merkle树



前缀树

以太坊的数据存储：Merkle Patricia Trie



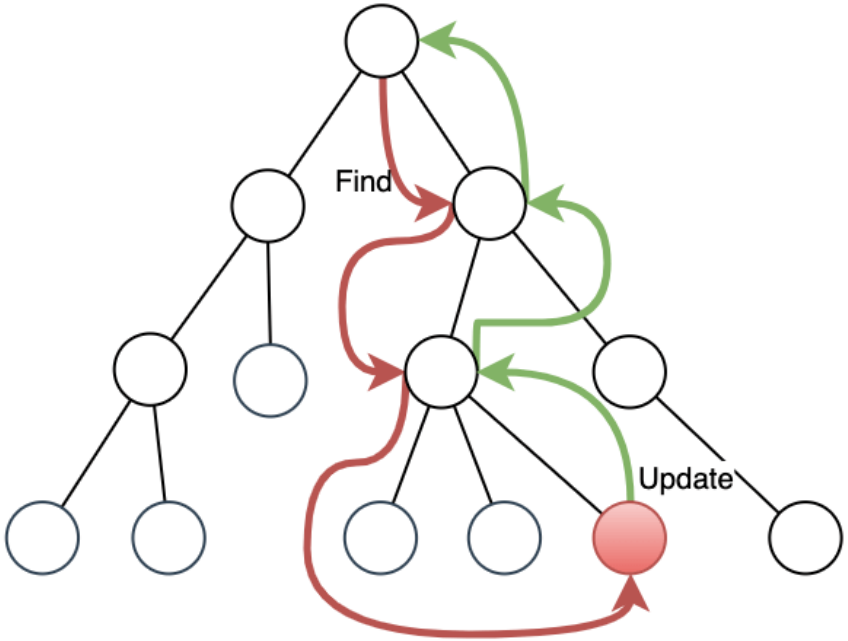
Patricia树

以太坊的数据存储：Merkle Patricia Trie

MPT树更新

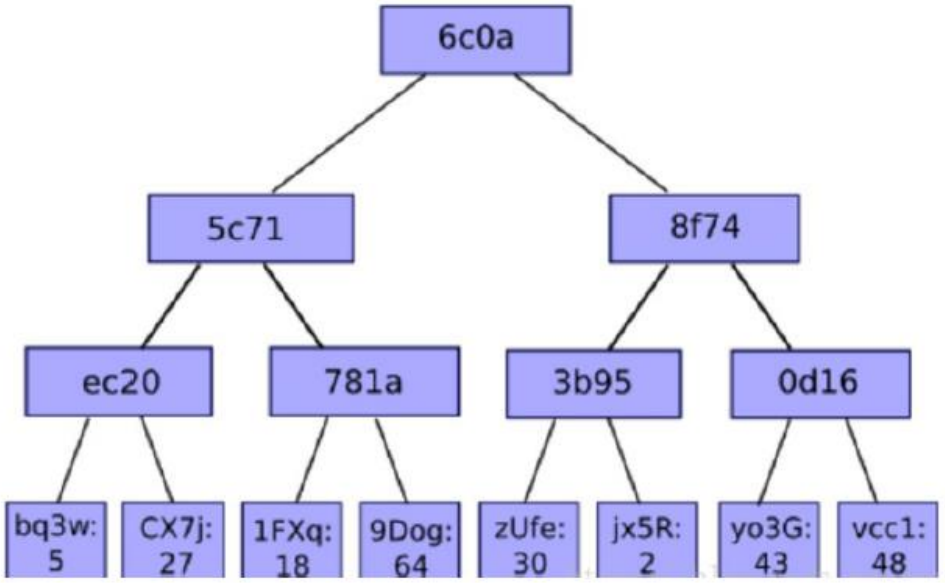
（1）将根节点传入作为当前处理节点，传入目标节点的Key作为路径path；

（2）传入新的Value值，若Value值为空，则找到该节点并删除，反之，创建一个新节点替换旧节点。

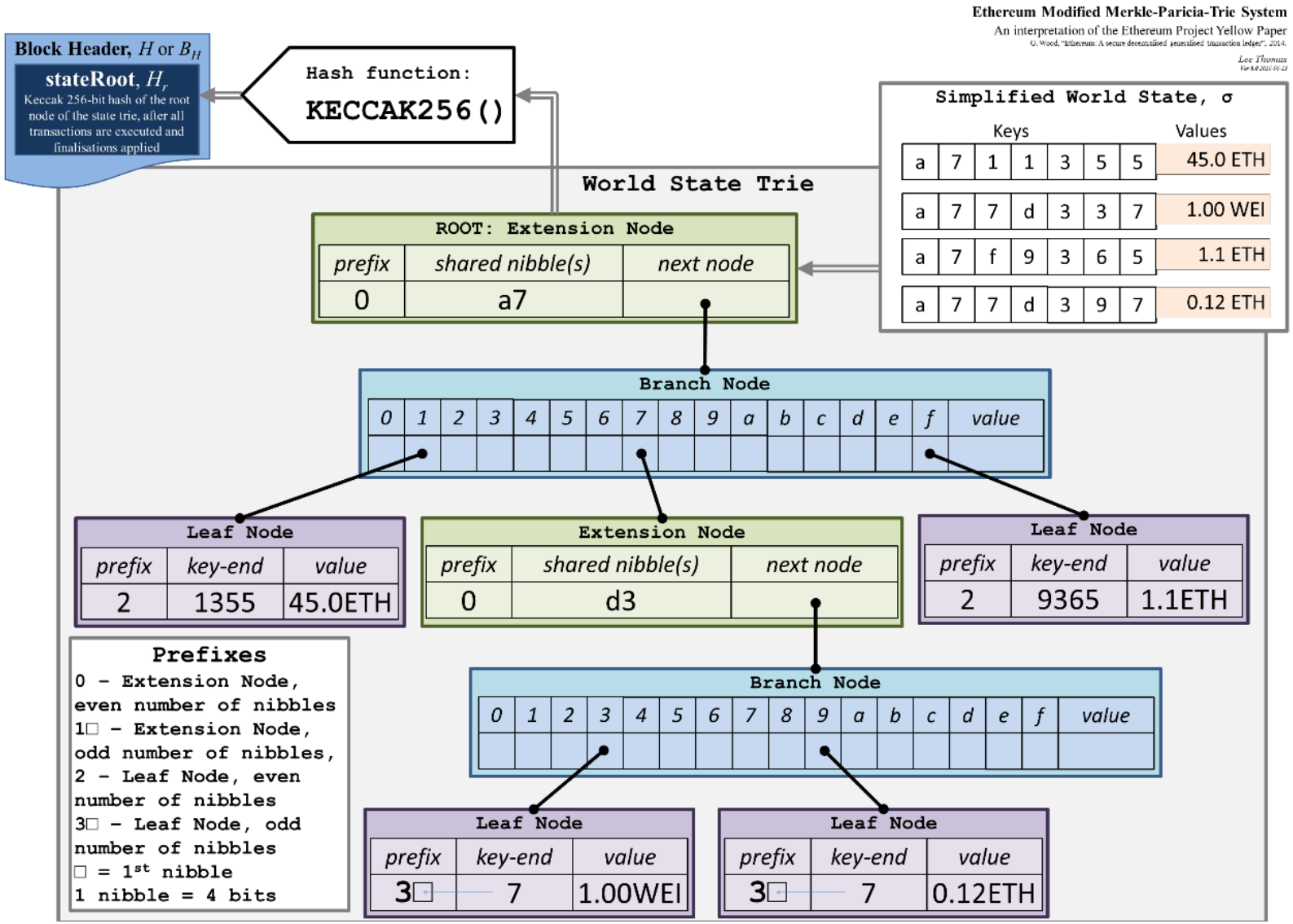


以太坊的数据存储：Merkle Patricia Trie

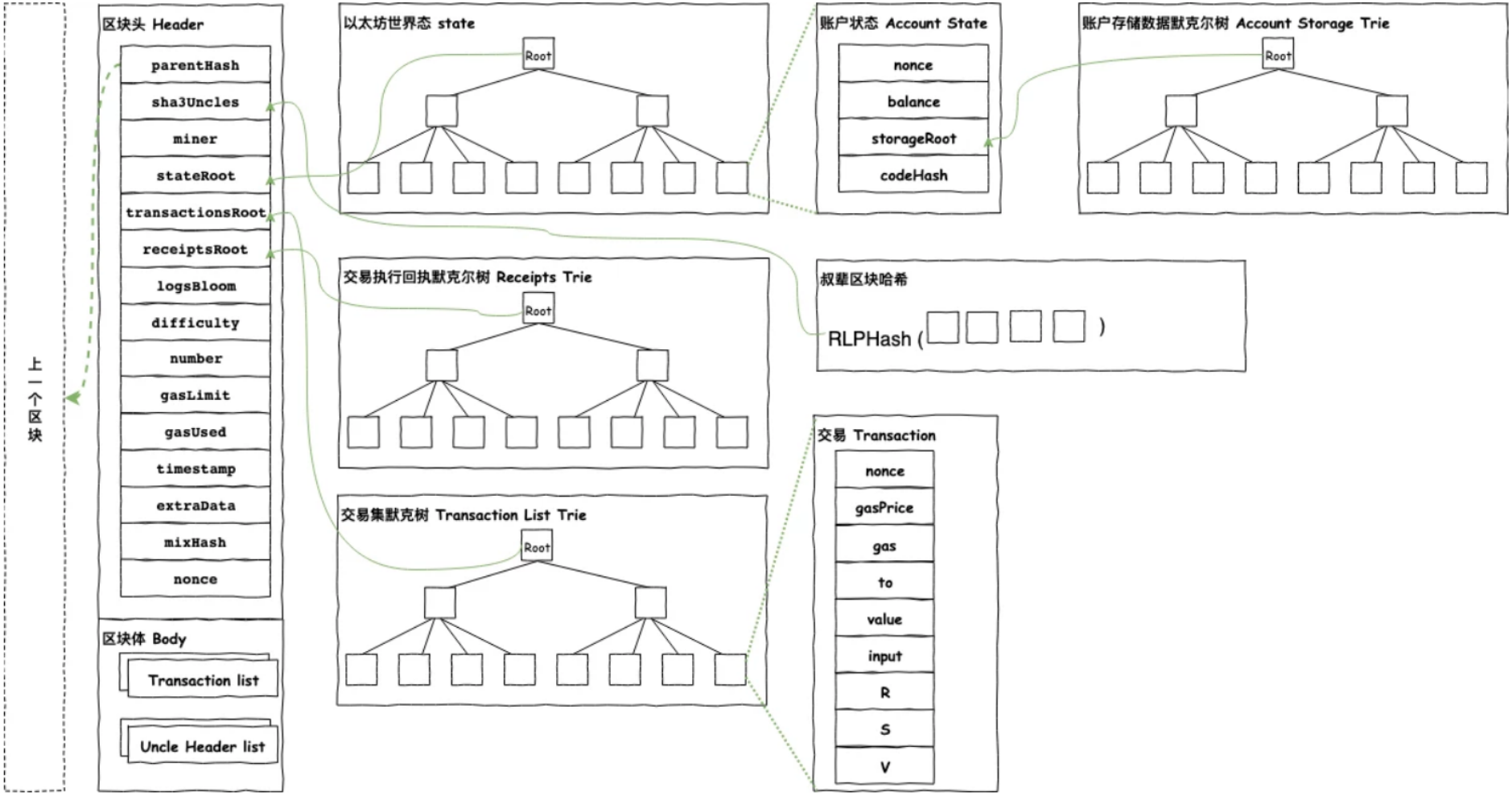
```
1  6c0a5c71ec20bq3w => 5
2  6c0a5c71ec20CX7j => 27
3  6c0a5c71781a1FXq => 18
4  6c0a5c71781a9Dog => 64
5  6c0a8f743b95zUfe => 30
6  6c0a8f743b95jx5R => 2
7  6c0a8f740d16y03G => 43
8  6c0a8f740d16vcc1 => 48
```



以太坊的数据存储：Merkle Patricia Trie



以太坊的数据存储


上一个区块
区块头 Header
parentHash
sha3Uncles
miner
stateRoot
transactionsRoot
receiptsRoot
logsBloom
difficulty
number
gasLimit
gasUsed
timestamp
extraData
mixHash
nonce
区块体 Body
Transaction list
Uncle Header list
以太坊世界态 state
Root
账户状态 Account State
nonce
balance
storageRoot
codeHash
账户存储数据默克尔树 Account Storage Trie
Root
交易执行回执默克尔树 Receipts Trie
Root
叔辈区块哈希
RLPHash ( )
交易集默克树 Transaction List Trie
Root
交易 Transaction
nonce
gasPrice
gas
to
value
input
R
S
V

智能合约举例

- 外部账户A
  - 发布2个合约
    - Function PlaySong()，参数是调用者网名
    - Function Song1Here()
- 某程序员开发一个DApp，输入账户地址、签名后可播放链上歌曲SongHere，每次听歌费用0.0001ETH
- 用户C运行DApp时，DApp发起一个合约调用交易
  - 由用户C的账户发起，先确定是否有足够余额支付听歌费用和合约运行gas
  - 合约调用交易：
    - Function PlaySong()的codehash，参数（用户网名）
    - 再触发合约调用交易 调用Function Song1Here

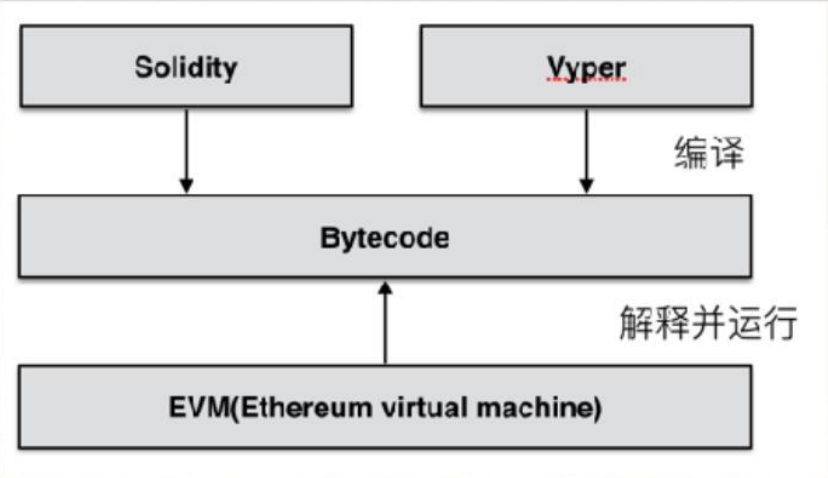
以太坊的网络

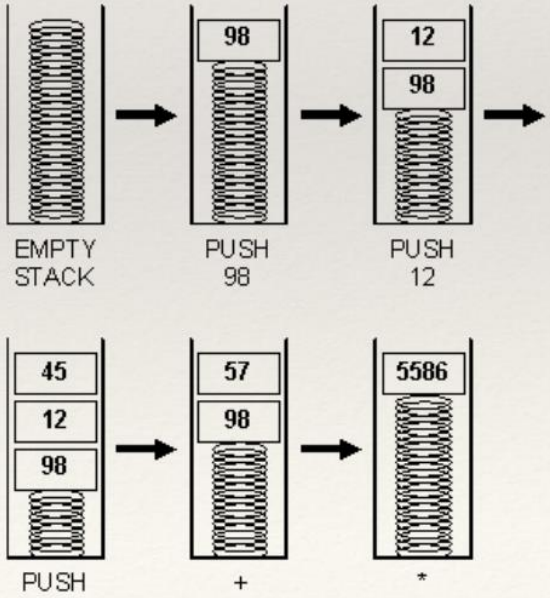
- 常见的两个
  - 主网 main net
  - 测试网 test net
- 其他
  - 以太坊经典主网
  - 以太坊经典测试网
  - ……

```
public class ChainId {              /* Ethereum chain ids. */
  public static final byte NONE = -1;
  public static final byte MAIN_NET = 1;
  public static final byte TEST_NET = 3;
}
```

以太坊虚拟机EVM

- EVM(Ethereum Virtual Machine) 是指用来解释跟执行合约账户字节码的解释器
- EVM基于栈的解释器
- EE(Execution environment) 是EVM的执行环境，包含环境参数以及存储空间读写函数
- EVM是拥有完全隔离的执行环境，合约无法访问宿主机的“网络”，“文件系统”等系统资源





以太坊虚拟机EVM

- EVM有一个固定的指令集
- 指令集包含：算数运算，比特运算，逻辑运算，跳转指令，状态读取、存储指令等
- 所有指令的运算必须是确定性的（例如高精度的浮点运算是不支持的）

以太坊虚拟机：图灵完备

- 图灵完备（Turing Completeness ）是针对一套数据操作规则而言的概念。数据操作规则可以是一门编程语言，也可以是计算机里具体实现的指令集。当这套规则可以实现图灵机模型里的全部功能时，就称它具有图灵完备。
- 高级语言具备了if/else语句、循环语句等，可以实现图灵完备。
- EVM是图灵完备的，但为了防止出现死循环，设置gas机制。

以太坊虚拟机（EVM）

运行智能合约的环境，运行在每一个节点上，类似于一个独立的沙盒，严格控制了访问权限，合约代码在EVM中运行时，是不能接触网络、文件或者其他进程的。

EVM模块主要分为以下三大模块：

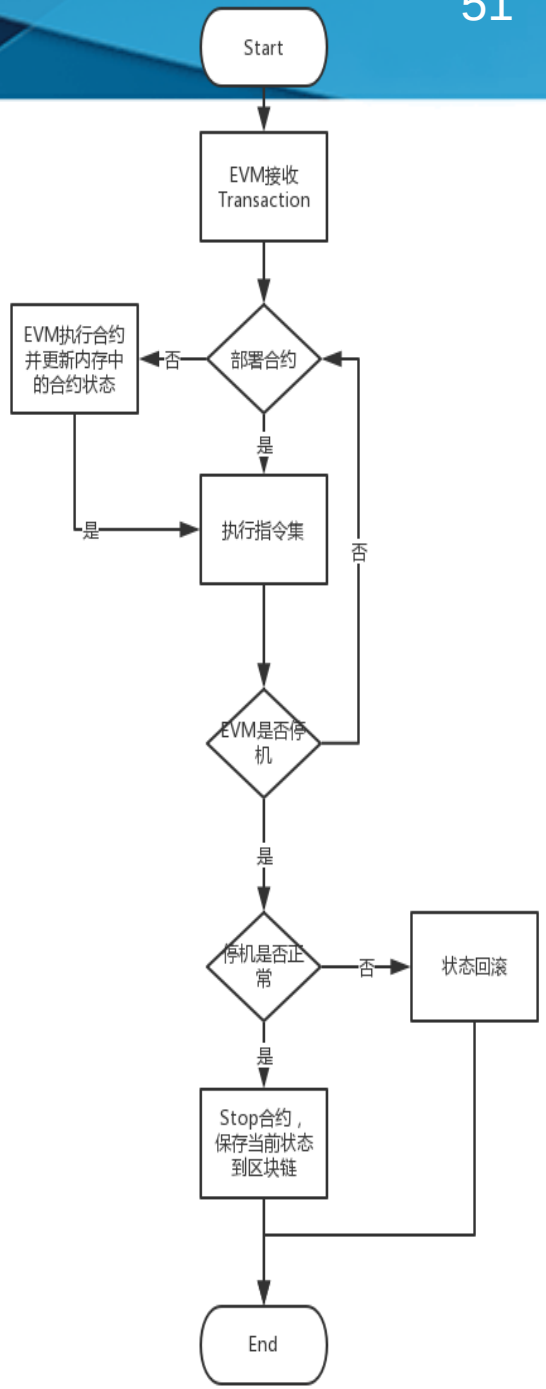
- **编译合约模块**： 主要是对底层Solc编译器进行一层封装，提供RPC接口给外部服务，对用Solidity编写的智能合约进行编译。编译后将会返回二进制码和相应的合约abi，abi可以理解为合约的手册，通过ABI可以知道合约的方法名、参数、返回值等信息。
- **Ledger模块**：主要是对区块链账户系统进行修改和更新，账户一共分为两种，分别是普通账户和智能合约账户，调用方如果知道合约账户地址则可以调用该合约，账户的每一次修改都会被持久化到区块链中。
- **EVM执行模块**（核心模块）：主要功能是对交易中的智能合约代码进行解析和执行，一般分为创建合约和调用合约两部分。同时为了提高效率，EVM执行模块除了支持普通的字节码执行外还支持JIT模式的指令执行，普通的字节码执行主要是对编译后的二进制码直接执行其指令，而JIT模式会对执行过程中的指令进行优化，如把连续的push指令打包成一个切片，方便程序高效执行。

EVM执行模块的大概流程

1. EVM接收到Transaction信息，然后判断Transaction类型是部署合约还是执行合约，如果是部署合约，则新建一个账户来存储合约地址和编译后的代码；如果是执行合约或是调用合约，则使用EVM来执行输入指令集。

2. 执行上一条指令集之后，判断EVM是否停机，如果停机则判断是否正常停机，正常停机则更新合约状态到区块链，否则回滚合约状态。如果不停机则继续执行下一条指令集，重复2；

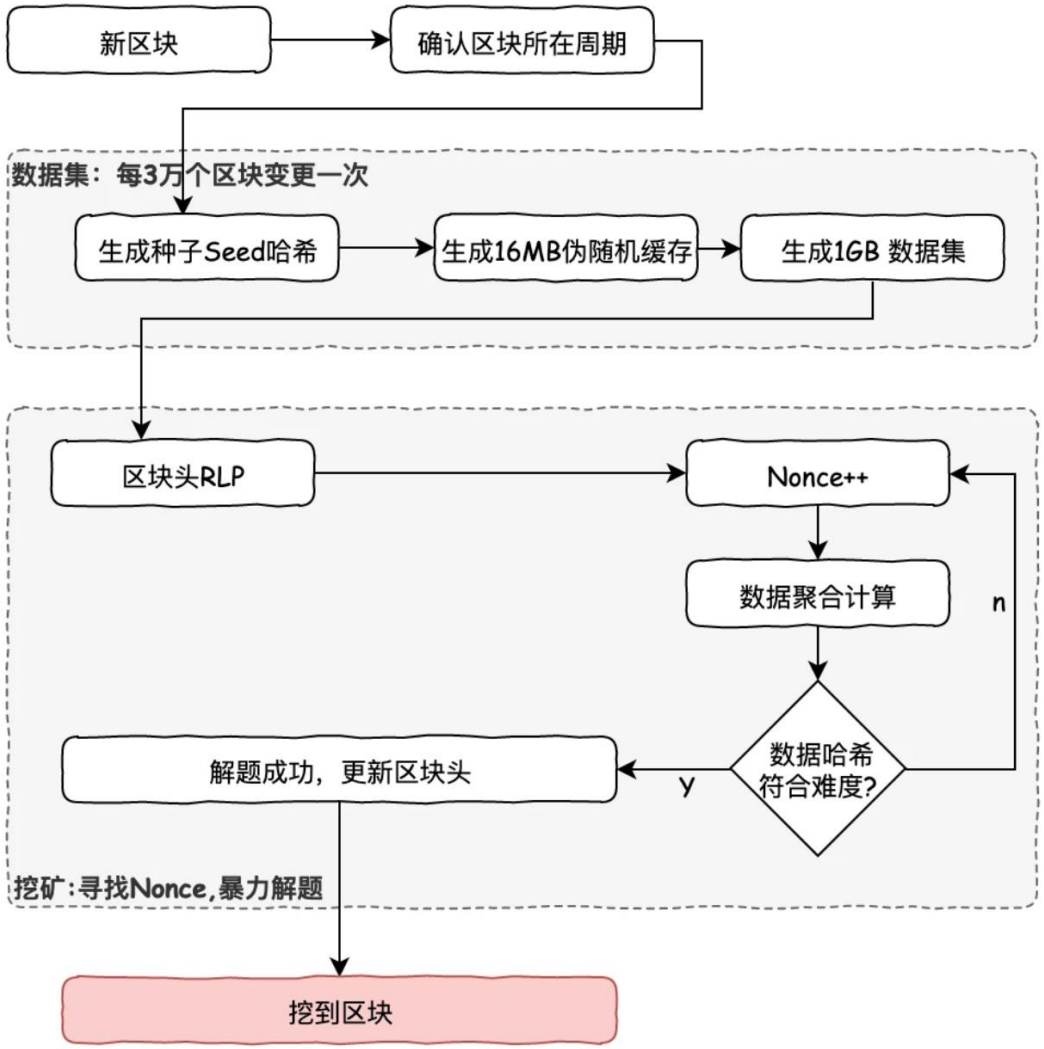
3. 执行完的合约会返回一个执行结果，EVM会将结果存储在Receipt回执中，调用者可以通过Transaction的hash来查询其结果。



以太坊的共识机制

- 以太坊1.0
  - PoW
- 以太坊2.0
  - PoS Vitalik Buterin 2013年就提出设想
  - 2020年12月1日，信标链 beacon chain上线
  - 信标链上没有交易，也没有 Token 或 DeFi 应用。它是一条“空链”，仅仅是为了成为一条运行 PoS 共识机制的区块链。
  - 2022年9月15日，完成以太坊主链与信标链的合并，转为PoS共识

以太坊PoW算法 ehash



以太坊中采用的hash算法是Keccak256，Keccak256后来被NIST采用，改造为SHA3,SHA3不是SHA2的替代品，而是一种有别于SHA2的全新哈希设计方案，2015年8月被NIST正式批准。

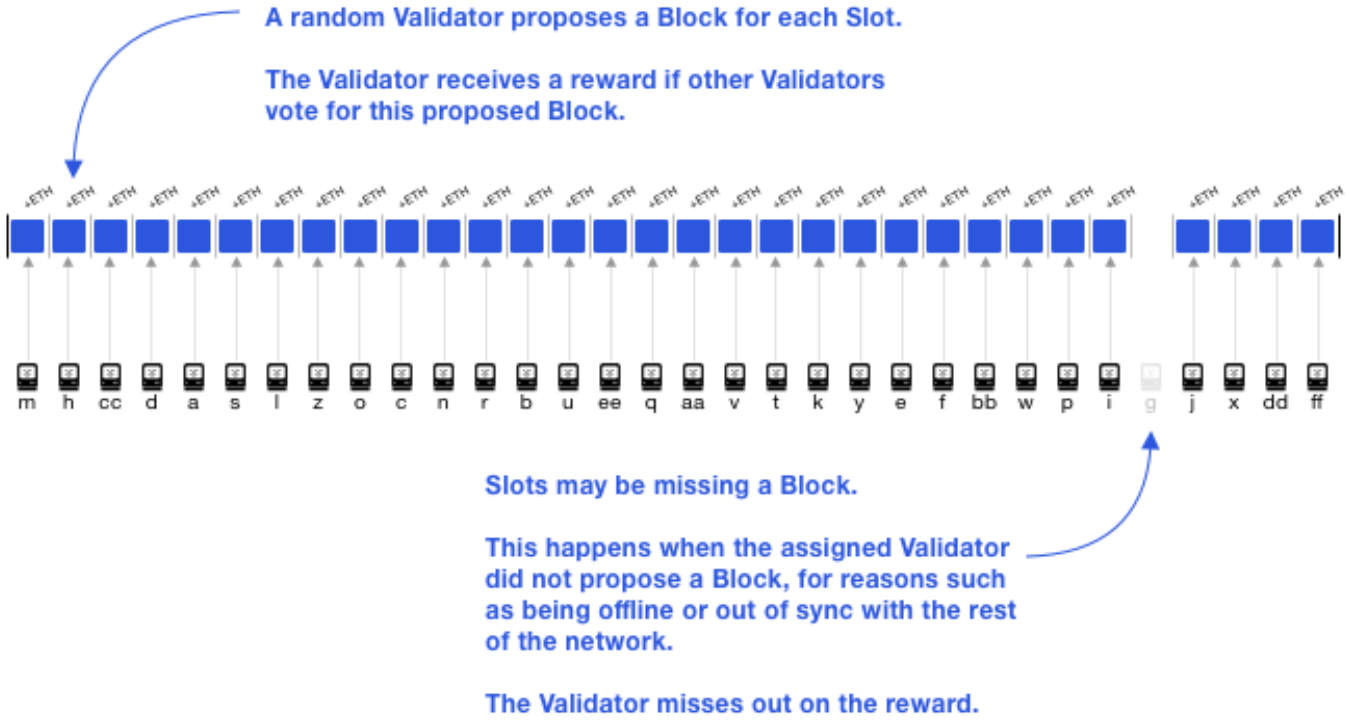
以太坊PoS共识

- 验证者（validator）以ETH的形式将资本抵押给以太坊上的一个智能合约。抵押的ETH充当抵押品，如果验证者行为不诚实或懒惰，可被销毁。然后，验证者负责检查通过网络传播的新块是否有效，并偶尔创建和传播新块。
- 要成为validator，用户必须将32 ETH存入存款合约并运行三个软件：执行客户端，共识客户端和验证器。在存入ETH时，用户加入一个激活队列。一个用户可以发起多个验证者，目前全球验证者数量40万个。
- 一旦激活，验证者就会从以太坊网络上的对等方接收新的区块。重新执行区块中交付的交易，并检查区块签名以确保区块有效。然后验证者在网络上发送支持该块的投票（称为证明 attestation）。

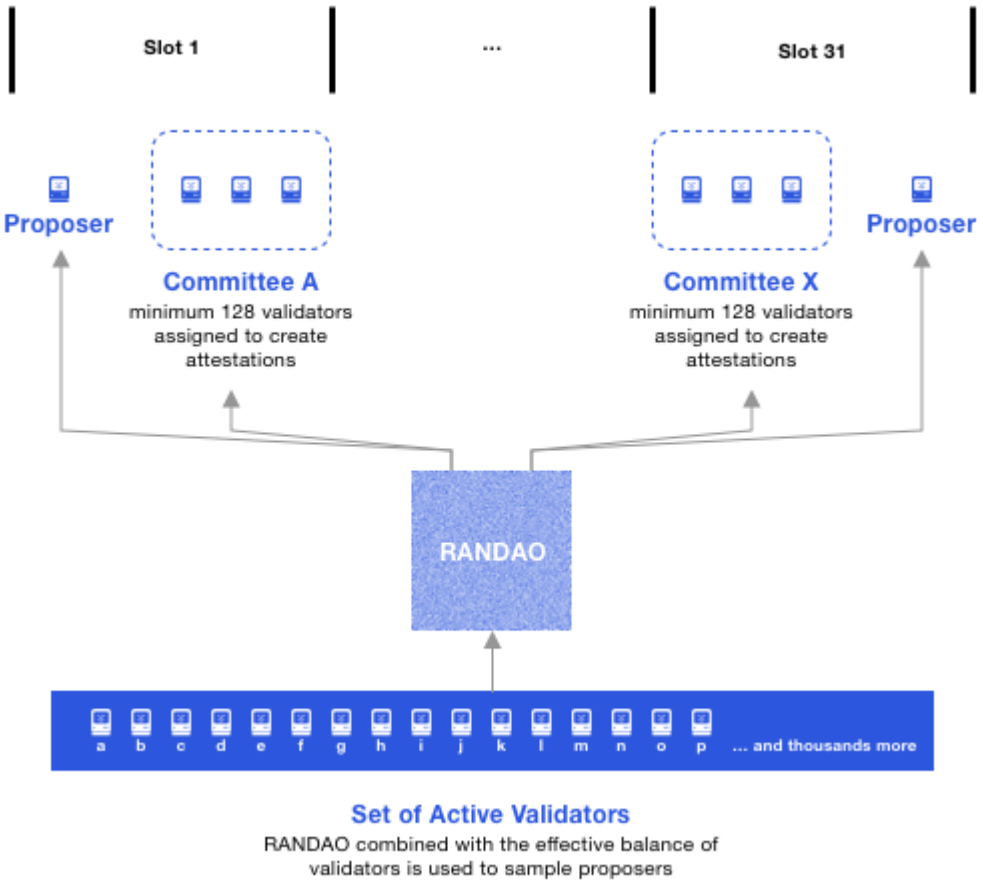
以太坊信标链 Beacon Chain

- 每12秒为一个时段slot
- 每32个slot为一个纪元 epoch（6.4分钟）
- 每个slot产生一个新区块，但有可能没有
- 信标链的Genesis区块在slot 0

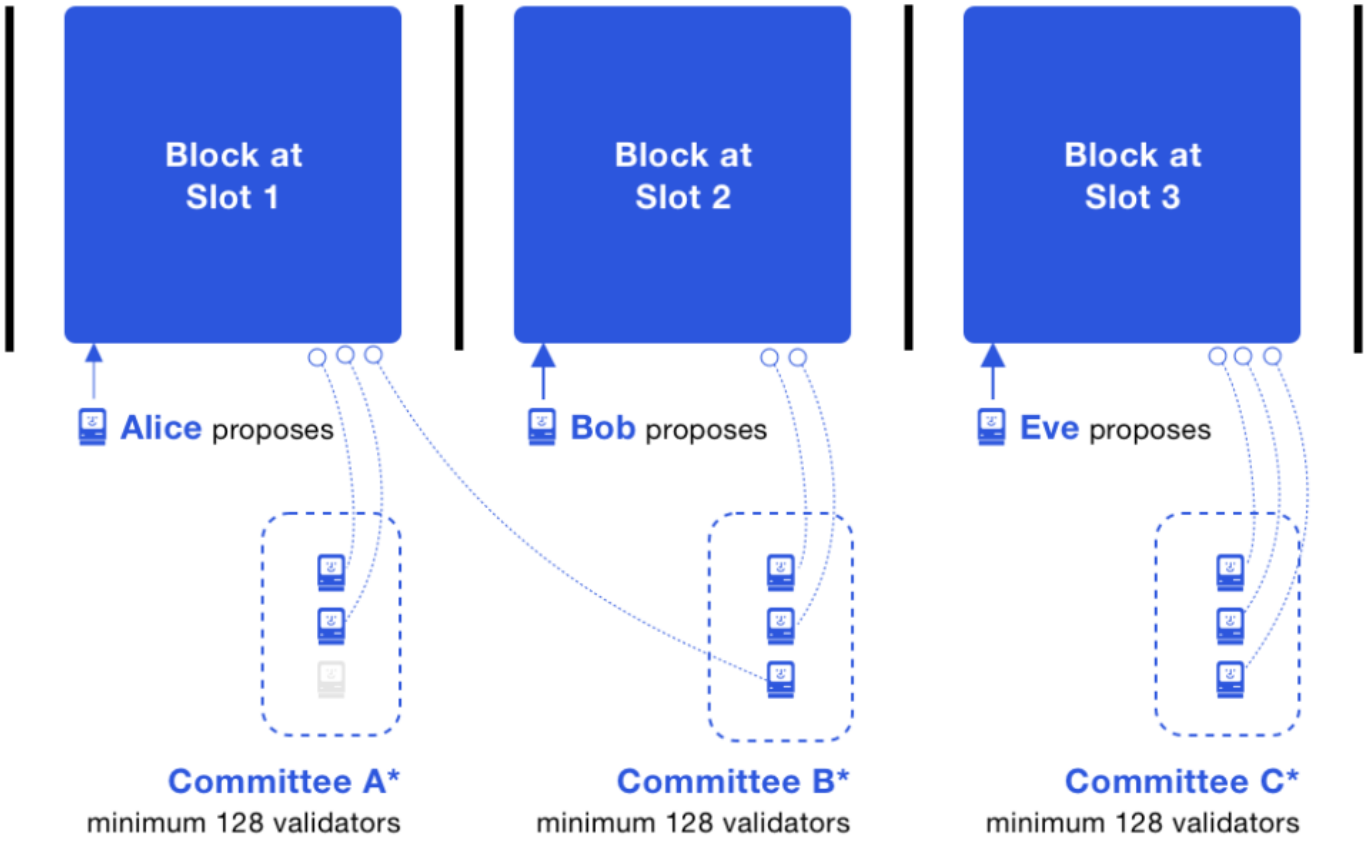
- 每个时段slot有一组验证者组成委员会
- 其中一个验证者被随机选中成为区块的proposer，发出区块
- 委员会的其他成员投票attestation，如2/3支持则该区块发布



- 每个slot至少有一个委员会，成员至少128个validators
- 活跃validator集被随机选择成员组成每个slot的委员会和proposer
- 一个validator可能被同时选为proposer和委员会成员



- 每个validator在每个epoch只能属于一个委员会。通常有超过8192个验证者：这意味着每个slot都有一个以上的委员会。所有委员会的规模相同，至少有128名验证者。当验证者少于 4096 个时，安全性会降低，因为委员会成员将少于 128 个。
- 每个epoch，验证者被平均分配到各个slot中，然后再细分为适当规模的委员会。洗牌算法会增加或减少每个slot的委员会数量，以使每个委员会至少有128个成员。
-

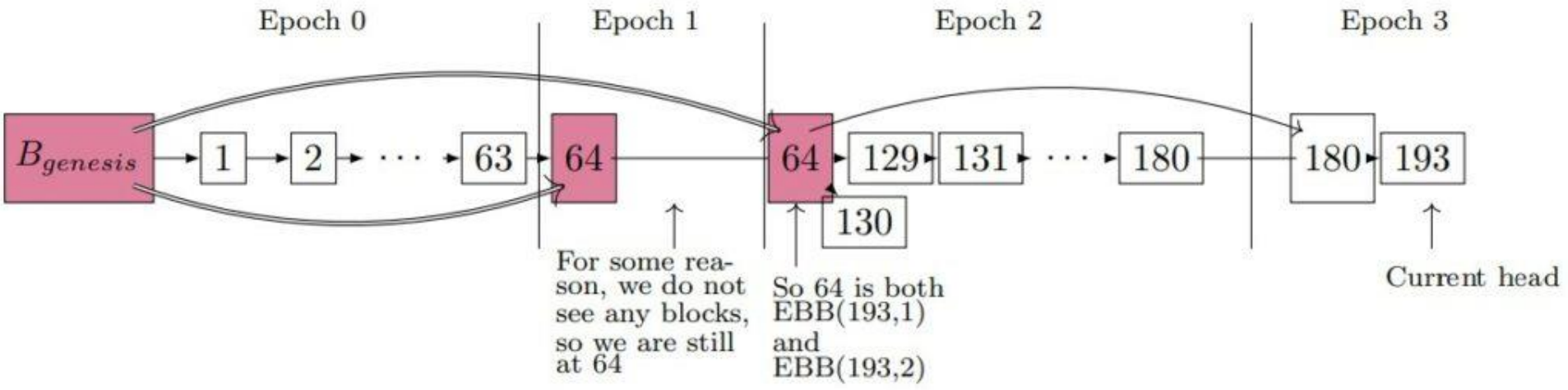


Validators in the committees are supposed to attest to what they believe the head of the blockchain is

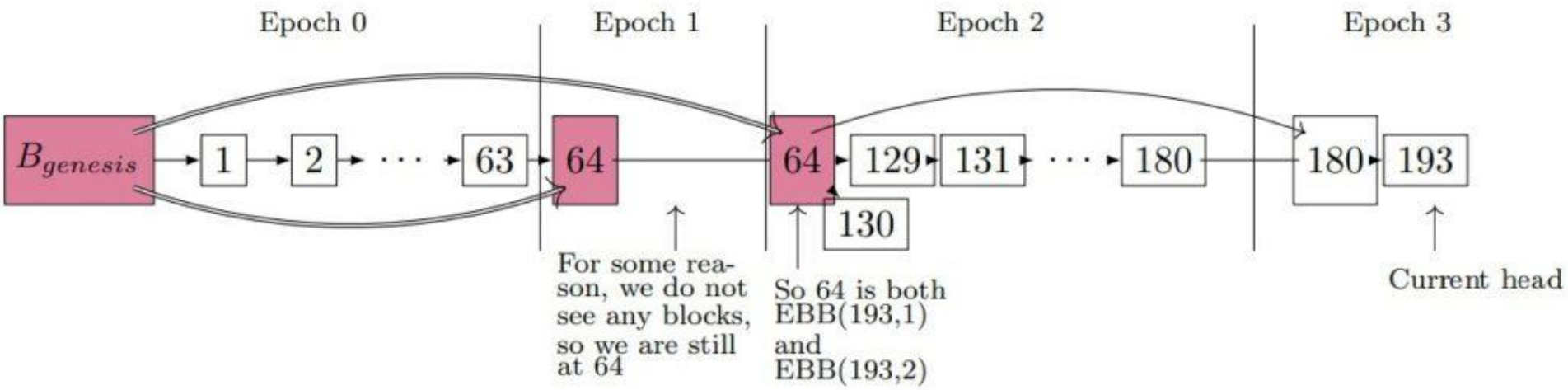
*Note there can be more than one committee per slot.

检查点 checkpoint

- 检查点是epoch的第一个slot中的一个区块。 如果没有这样的区块，则检查点是前面最近的区块。 每个epoch始终有一个检查点区块。一个区块可以是多个纪元的检查点。

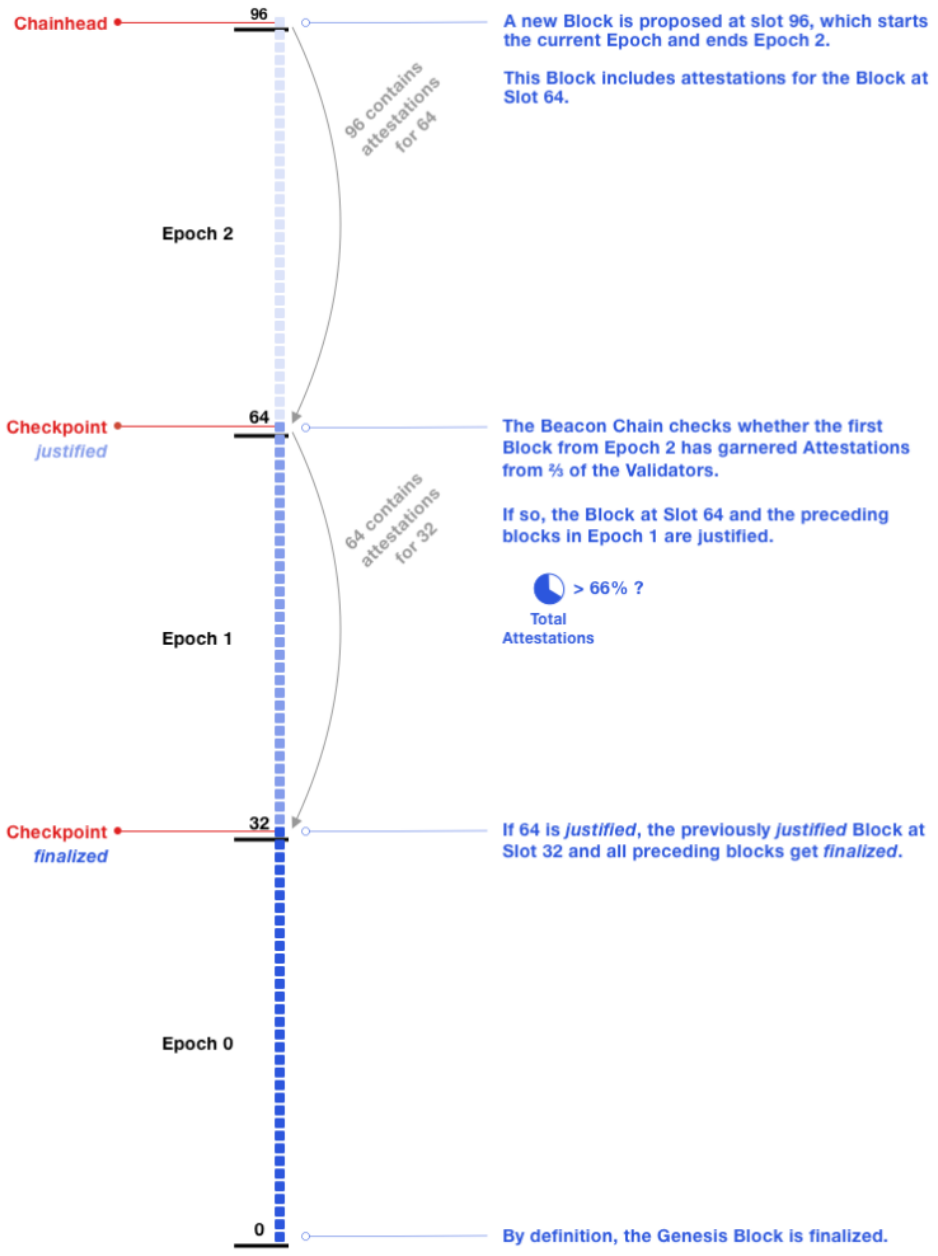


- 当验证者提交LMD GHOST 投票时, 他投票当前epoch的checkpoint, 称为 *target*.
- 这个投票称为Casper **FFG 投票**, 还需要包含前一个checkpoint, 称为 *source*.
- 图中Epoch 1 的验证者投票的source checkpoint 是genesis 区块, target checkpoint 在Slot 64的区块. Epoch 3中, 验证者投票的source checkpoint 是slot 64区块， target checkpoint是180区块



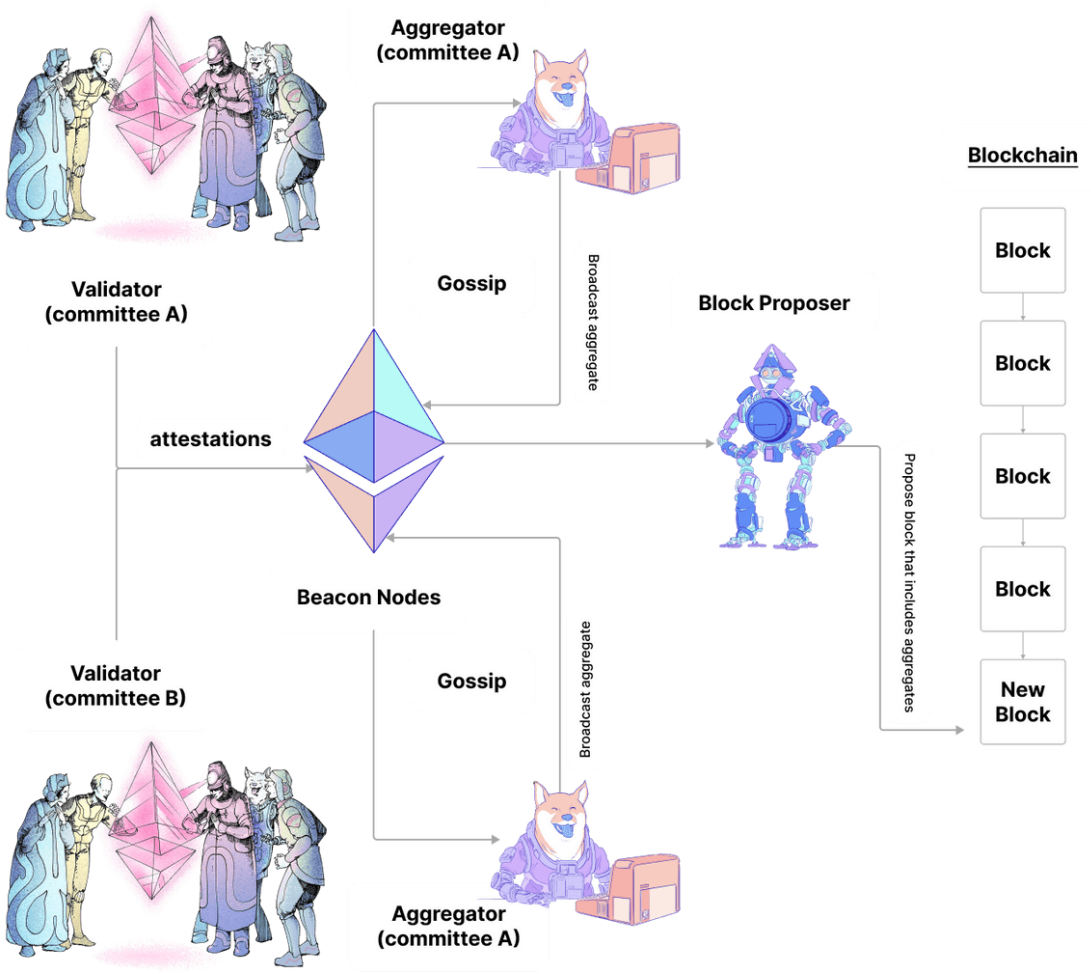
Checkpoint的justified和finalized：

- 一个epoch结束时，如果它的checkpoint获2/3验证者投票支持，其状态改为justified
- 一个justified的checkpoint如果其后续的checkpoint状态改为justified，则它的状态升级为finalized
- 一般情况下一个checkpoint经过2个epoch后（12.8分钟）变为finalized



Attestation 投票接纳过程：

- 生成 generation
- 传播 propagation
- 聚合 aggregation
- 传播 propagation
- 接纳 inclusion



The DAO事件

2016年4月30日开始，一个名为“The DAO”的初创团队，在以太坊上通过智能合约进行ICO众筹。 28天时间，筹得1.5亿美元，成为历史上最大的众筹项目。

THE DAO创始人之一Stephan TualTual在6月12日宣布，他们发现了软件中存在的“递归调用漏洞”问题。 不幸的是，在程序员修复这一漏洞及其他问题的期间，一个不知名的黑客开始利用这一途径收集THE DAO代币销售中所得的以太币。6月18日，黑客成功挖到超过360万个以太币，并投入到一个DAO子组织中，这个组织和THE DAO有着同样的结构。

THE DAO持有近15%的以太币总数，因此THE DAO这次的问题对以太坊网络及其加密币都产生了负面影响。

The DAO事件

针对The DAO攻击事件，当时有两种提议：

- 软分叉：进行一次软分叉，不会有回滚，不会有任何交易或者区块被撤销。软分叉将从块高度1760000开始把任何与The DAO和child DAO相关的交易认做无效交易，以此阻止攻击者在27天之后提走被盗的以太币。
- 硬分叉：要求矿工彻底解除盗窃并且归还The DAO所有以太币，这样就能自动归还给代币持有人，从而结束The DAO项目。

2016年7月20日，在块1,920,000 以太坊实施了DAO硬分叉，因此创建了两个以太坊：

- 以太坊 Ethereum
- 以太坊经典 Ethereum Classic

以太坊是什么？

- In the Ethereum universe, there is a single, canonical computer (called the Ethereum Virtual Machine, or EVM) whose state everyone on the Ethereum network agrees on. Everyone who participates in the Ethereum network (every Ethereum node) keeps a copy of the state of this computer. Additionally, any participant can broadcast a request for this computer to perform arbitrary computation. Whenever such a request is broadcast, other participants on the network verify, validate, and carry out ("execute") the computation. This causes a state change in the EVM, which is committed and propagated throughout the entire network.

以太坊是什么？

- Requests for computation are called transaction requests; the record of all transactions as well as the EVM's present state is stored in the blockchain, which in turn is stored and agreed upon by all nodes.
- Cryptographic mechanisms ensure that once transactions are verified as valid and added to the blockchain, they can't be tampered with later; the same mechanisms also ensure that all transactions are signed and executed with appropriate "permissions".

深入阅读

- 以太坊白皮书
- 以太坊技术文档

谢谢！