Merkle Tree的原理以及其在区块链中的应用

ZHANG RUNLIN / 10.28

01 Merkle Tree原理

HASH HASHLIST TREE

02 Merkle Tree在区块链中的应用

BTC ETH MPT

Merkle Tree的诞生

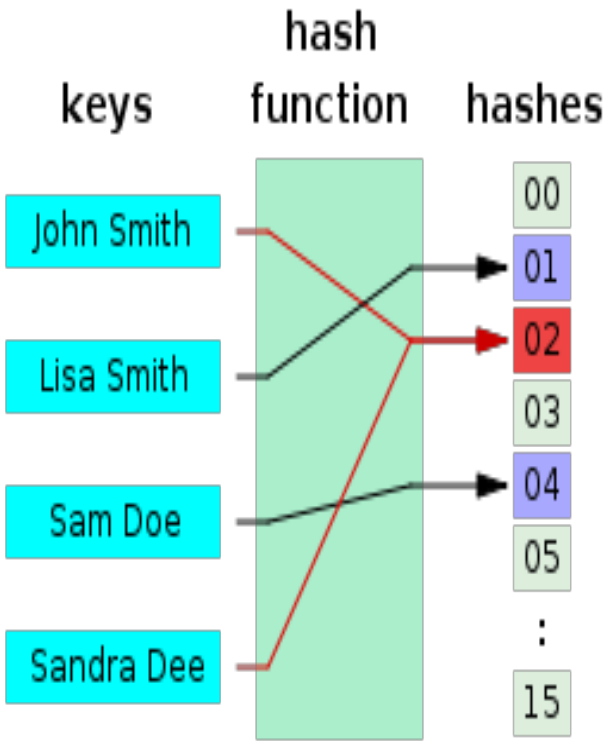
在1979年，Ralph Merkle取得了MerkleTree的专利权（改专利在2002年过期）。其概括的描述为：“该发明包含了一种提供信息验证的数字签名的方法，该方法利用单向的认证树对密码数字进行校验。”

Life is good. More is better.

HASH 是什么

Hash是一个把任意长度的数据映射成固定长度数据的函数。例如，对于数据完整性校验，最简单的方法是对整个数据做Hash运算得到固定长度的Hash值，然后把得到的Hash值公布在网上，这样用户下载到数据之后，对数据再次进行Hash运算，比较运算结果和网上公布的Hash值进行比较，如果两个Hash值相等，说明下载的数据没有损坏。不过是因为输入数据的稍微改变就会引起Hash运算结果的面目全非，而且根据Hash值反推原始输入数据的特征是困难的

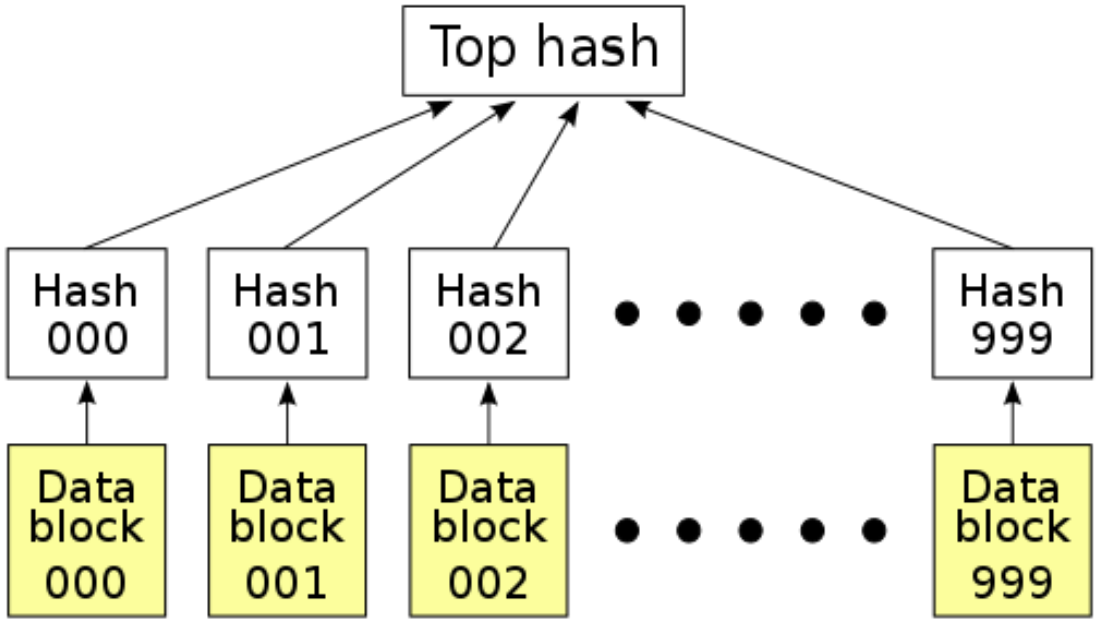
所以一个结论：但如果数据源不稳定，一旦数据损坏，就需要整个数据重新下载，这种下载的效率是很低的。



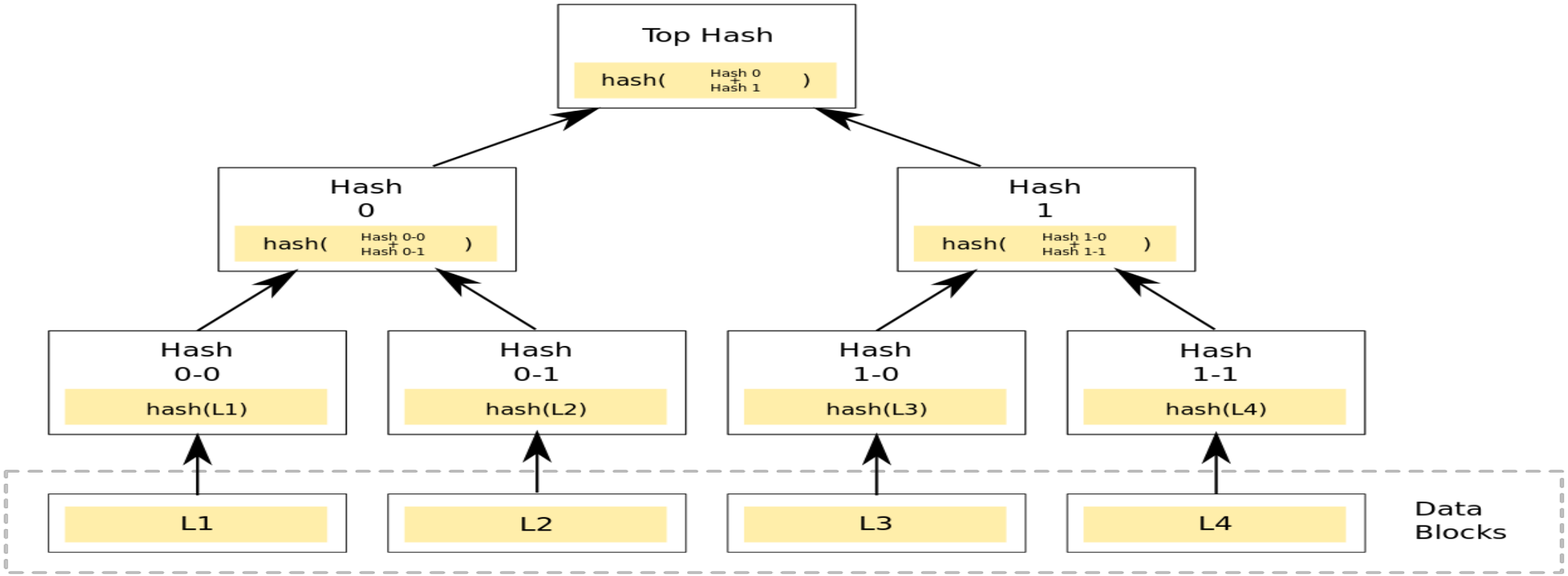
HASH LIST 是文件或文件集中数据块的hash值的列表。

为了校验数据的完整性，更好的办法是把大的文件分割成小的数据块。这样的好处是，如果小块数据在传输过程中损坏了，那么只要重新下载这一快数据就行了，不用重新下载整个文件。

怎么确定小的数据块没有损坏哪？只需要为每个数据块做Hash。比如BT下载的时候，在下载到真正数据之前，我们会先下载一个Hash列表。这个Hash列表是把每个小块数据的Hash值拼到一起，然后对这个长字符串在作一次Hash运算，这样就得到Hash列表的根Hash(Top Hash or Root Hash)。下载数据的时候，首先从可信的数据源得到正确的根Hash，就可以用它来校验Hash列表了，然后通过校验后的Hash列表校验数据块。



Merkle Tree是什么



Merkle Tree可以看做Hash List的泛化产物（Hash List可以看作一种特殊的Merkle Tree，即树高为2的多叉Merkle Tree）

在最底层，和哈希列表一样，我们把数据分成小的数据块，有相应地哈希和它对应。但是往上走，并不是直接去运算根哈希，而是把相邻的两个哈希合并成一个字符串，然后运算这个字符串的哈希，这样每两个哈希就结婚生子，得到了一个”子哈希“。于是往上推，依然是一样的方式，可以得到数目更少的新一级哈希，最终必然形成一棵倒挂的树，到了树根的这个位置，这一代就剩下一个根哈希了，我们把它叫做Merkle Root。

Merkle Tree的操作

1. 创建Merckle Tree

加入最底层有9个数据块。

step1：（红色线）对数据块做hash运算，$Node_{0i} = hash(Data_{0i})$, i=1,2,…,9

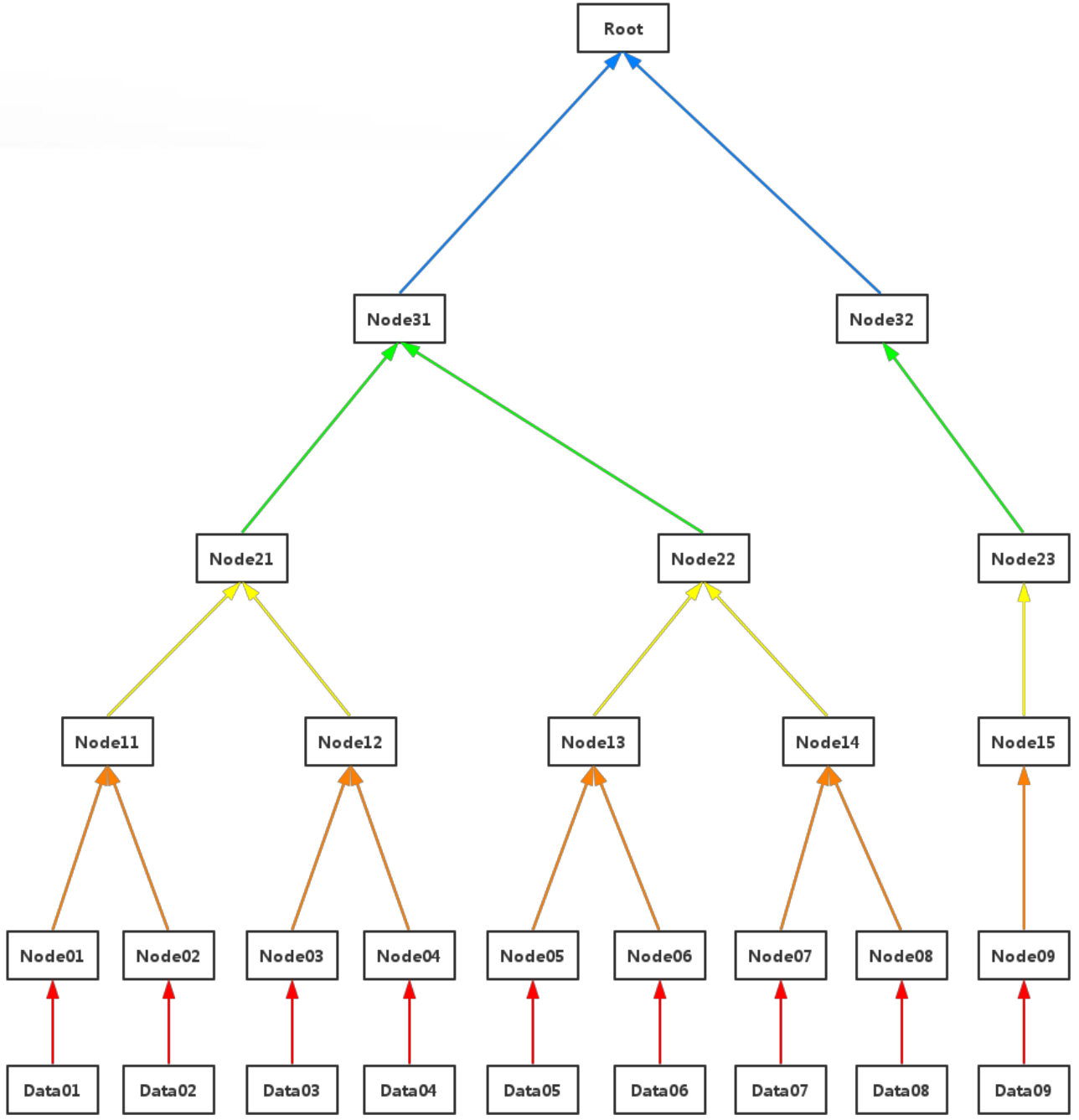
step2:（橙色线）相邻两个hash块串联，然后做hash运算，$Node_{1((i+1)/2)} = hash(Node_{0i}+Node_{0(i+1)})$, i=1,3,5,7;对于i=9, $Node_{1((i+1)/2)} = hash(Node_{0i})$

step3:（黄色线）重复step2

step4：（绿色线）重复step2

step5：（蓝色线）重复step2，生成Merkle Tree Root

创建Merkle Tree是O(n)复杂度(这里指O(n)次hash运算)，n是数据块的大小。得到Merkle Tree的树高是log(n)+1。



Merkle Tree的操作

2.检索数据块

Step1. 首先比较v0是否相同,如果不同，检索其孩子node1和node2.

Step2. v1 相同，v2不同。检索node2的孩子node5 node6;

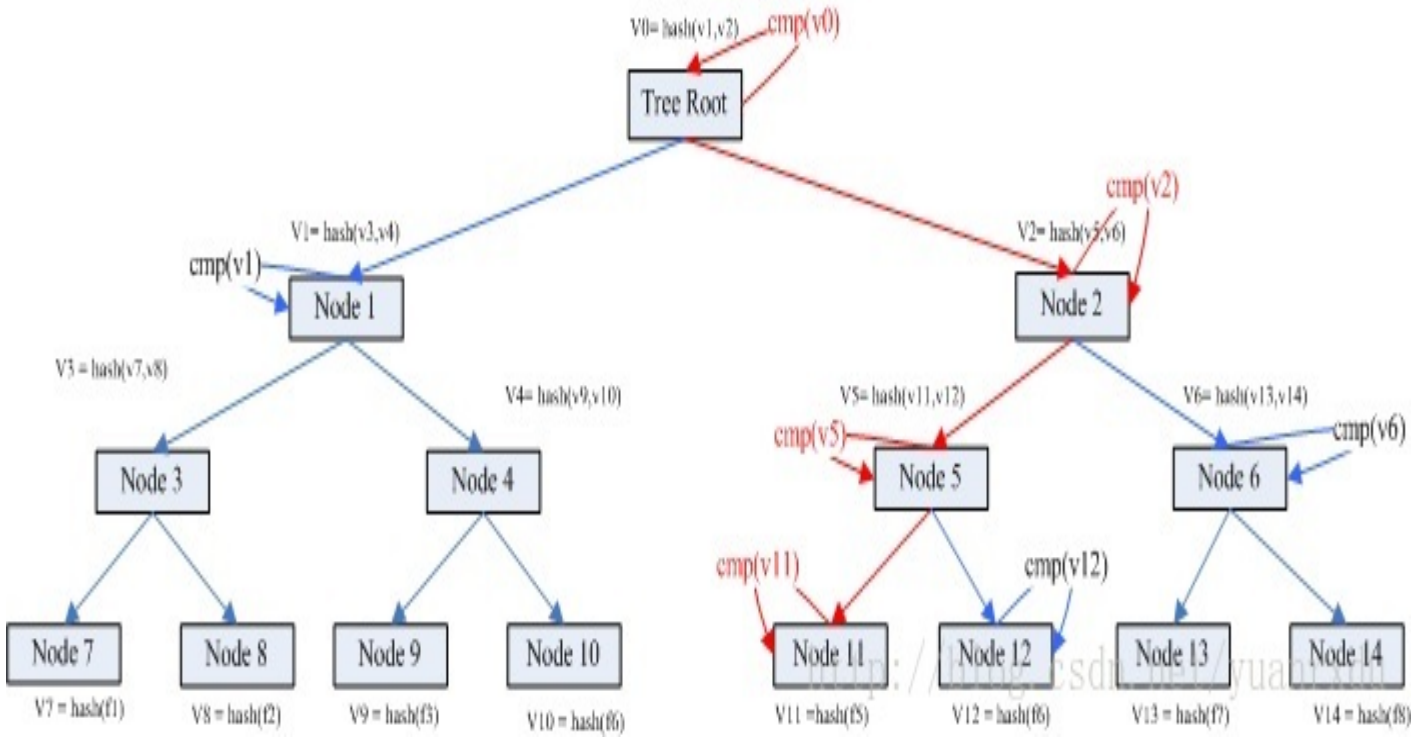
Step3. v5不同，v6相同，检索比较node5的孩子node 11 和node 12

Step4. v11不同，v12相同。node 11为叶子节点，获取其目录信息。

Step5. 检索比较完毕。

以上过程的理论复杂度是Log(N)。

从上图可以得知真个过程可以很快的找到对应的不相同的文件。



Merkle Tree的特点

是一种树，大多数是二叉树，也可以多叉树，无论是几叉树，它都具有树结构的所有特点

叶子节点的value是数据集合的单元数据或者单元数据HASH。

非叶子节点的value是根据它下面所有的叶子节点值，然后按照Hash算法计算而得出的。

为什么要使用Merkle Tree

1.显著减少了要达到证明数据完整性的置信度所需的数据量。

2.显著减少了维护一致性、数据校验以及数据同步所需的网络I/O数据包大小。

3.将数据校验和数据本身分离——Merkle tree可以存在本地，也可以存放与受信任的权威机构上，也可以存在分布式系统上（你只需要维护属于你的树即可。）将“我可以证明这个数据是合法的”和数据本身解耦意味着你可以为Merkle Tree和数据存储提供适当的分离（包括冗余）持久性。

Merkle Tree可以提供什么?

01

一致性验证

你可以用它验证两份日志的版本是否一致:

- 最新的版本包含了之前所有版本的信息。
- 日志中的记录顺序是一致的。
- 所有新的记录都是跟在旧版本记录的后面的。

02

数据校验

任何人都可以用一份日志来请求Merkle审计证明，校验某条凭证记录确实存在于日志当中，审计者会将这些类型的请求发送至日志以便它们检验TLS客户端的证书。如果Merkle审计证明不能生成与Merkle Tree哈希值匹配的根哈希值，则表示证书没有在日志当中。

03

数据同步

Merkle tree在分布式数据存储中的数据同步中发挥着重要的作用，这是因为它允许分布式系统中的每个节点可以迅速高效地识别已经更改的记录而无需将发送所有的数据来进行比对。一旦树中有特定的叶节点的变更被识别，我们只需要将与该特定叶节点相关的数据上传至网络即可

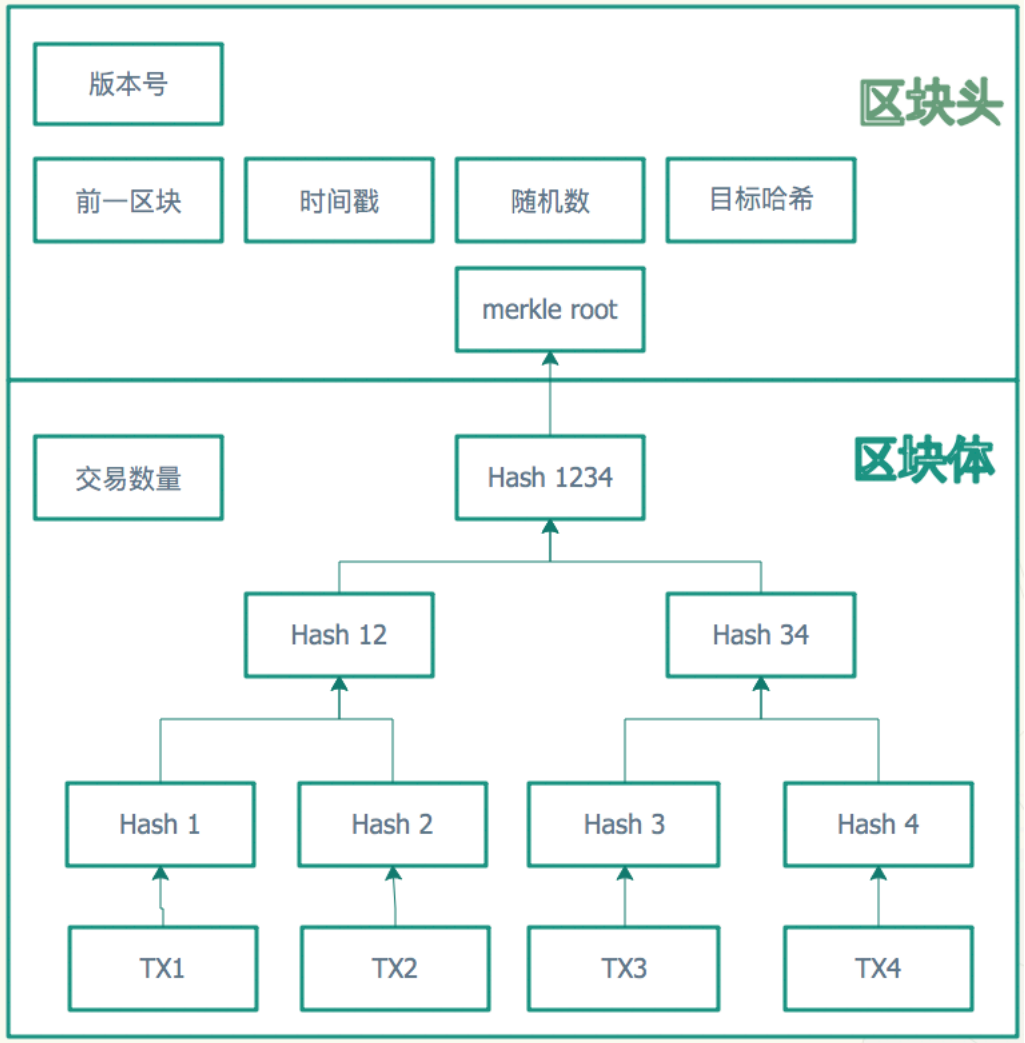
04

证明的重要性

一致性证明和审计证明的重要性在于客户端可以自己进行验证。这意味着当客户端请求服务器来验证一致性时，服务器并不是简单地回复答案“是”或“不是”，即使在“是”的情况下也会向你发送客户端可以验证的相关的证明。

Merkle Tree应用在哪里

Merkle Tree在BTC中的应用



子结构名称	作用说明	大小
版本号	数据区块的版本号	4字节
前一个区块的记录	记录了前一个数据区块的HASH值，当前区块的HASH值一定比它小	32字节
Merkle树的根值	记录了当前区块中所有交易Merkle树的根节点的HASH值	32字节
时间戳	记录了当前区块生成的时间，按照UNIX时间格式	4字节
目标值	当前区块生成所达成目标值的特征，用于矿工的工作量证明	4字节
随机数	当前区块工作量证明的参数	4字节

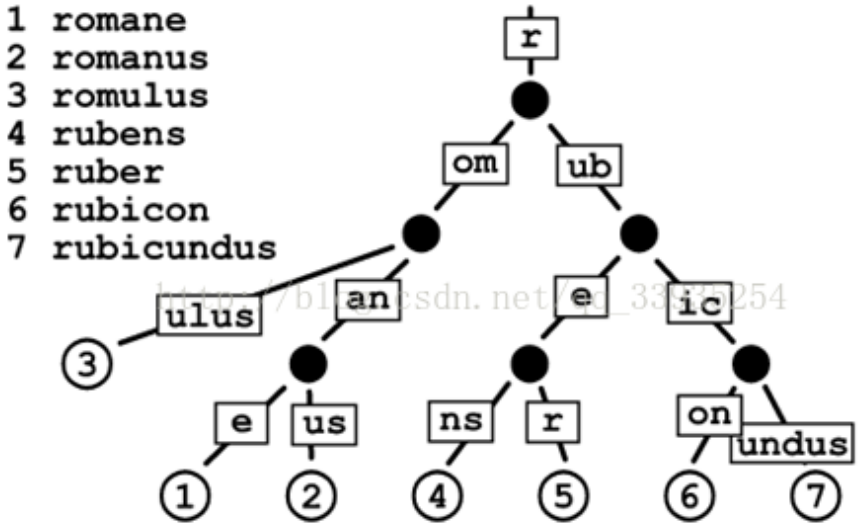
SPV：Merkle Tree + Bloom filter

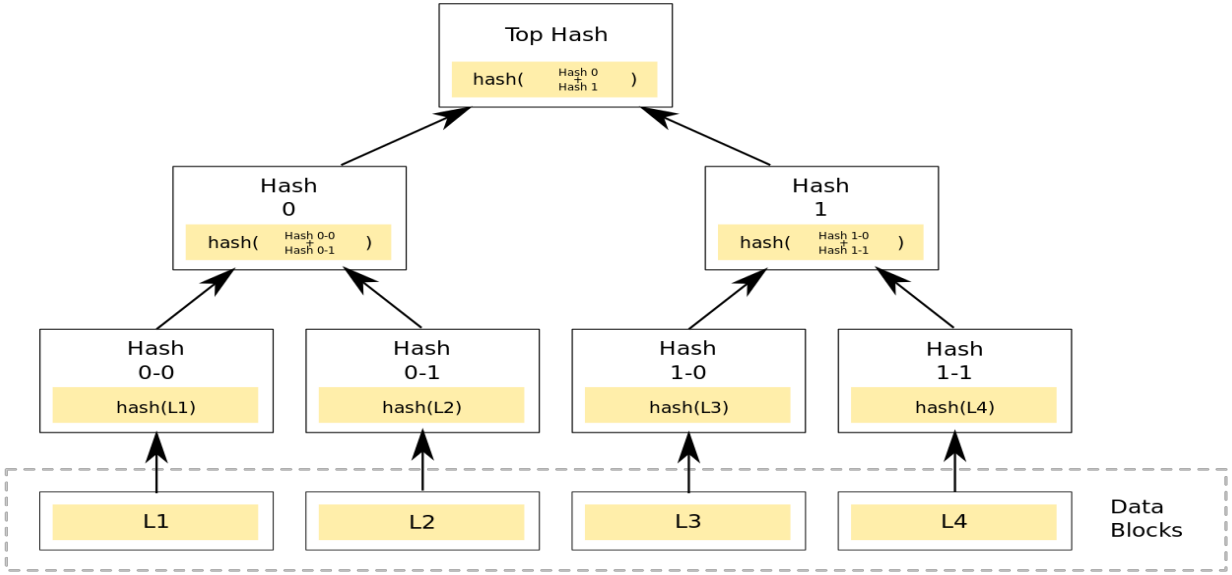
1、SPV充分利用默克树结构，在寻找交易时，只需下载寻找区块头而不是整个区块。

2、区块头只有80字节，每小时6个，一年也就4M大小。

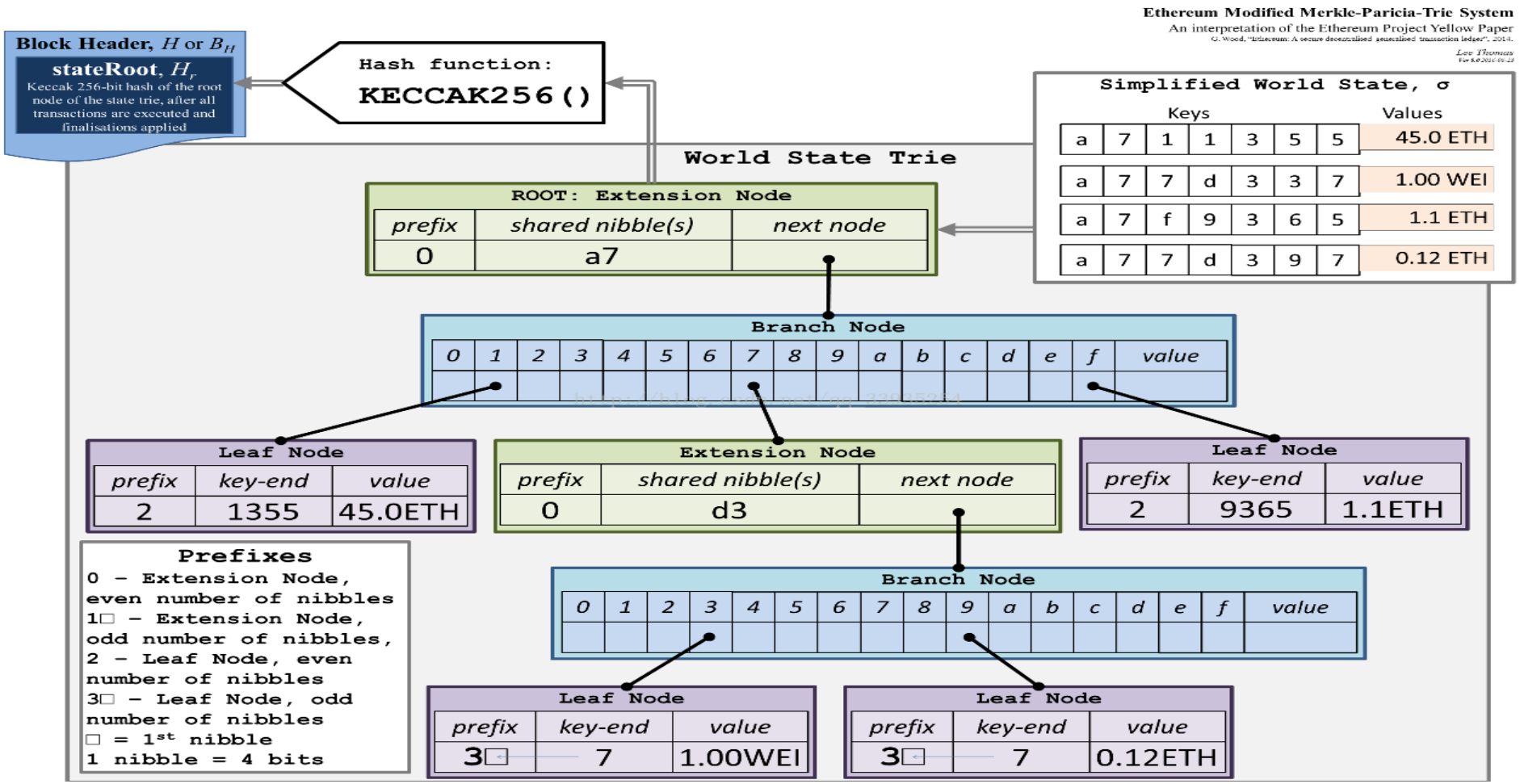
3、如何定位区块：节点会在通信链路上建立BF，限制只接受含有目标地址的交易，从而能过滤掉大量不相关的数据，减少客户端不必要的下载量。

MPT = Patricia Tree + Merkle Tree





Merkle Tree在Ethereum的应用



THANKS