学校代码：10255

学　号：2141510

基于区块链的去中心化交易关键技术研究及应用

RESEARCH AND APPLICATIONS ON THE KEY TECHNIQUES OF DECENTRALIZED TRANSACTION BASED ON BLOCKCHAIN

学科专业：计算机科学与技术

论文作者：安庆文

指导老师：黄秋波

答辩日期：2017.01.05

东华大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

东华大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权东华大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密 □，在 ____ 年解密后适用本版权书。

本学位论文属于

不保密 □。

学位论文作者签名：　　指导教师签名：

日期：　　年　月　日　　日期：　　年　月

东华大学
计算机科学与技术博/硕士学位论文答辩委员会成员名单

姓名	职称	职务	工作单位	备注
韩耀军	教授	答辩委员会主席	上海外国语大学	
史有群	教授	答辩委员会委员	东华大学	
徐光伟	副教授	答辩委员会委员	东华大学	
燕彩蓉	副教授	答辩委员会委员	东华大学	
朱明	副教授	答辩委员会委员	东华大学	
陶然	高级实验师	答辩委员会委员	东华大学	

基于区块链的去中心化交易关键技术研究及应用

摘 要

随着 2009 年比特币的问世，区块链技术也随之产生。区块链技术由于具有去中心化、透明可信任的特点，因此这项技术也迅速引起了广大专家学者的关注。以太坊作为区块链技术的一个实现，提供了一个去中心化的应用平台，去中心化的应用也随之产生，区块链技术的应用也逐渐从最初的金融行业发展到各行各业。近些年来，随着一些行业联盟成立并尝试应用区块链技术解决联盟内去中心化的问题并初见成效，联盟链的形式越来越得到企业的青睐。因此以联盟链应用为背景对区块链技术进行研究有着极大的未来价值和现实意义。

本文以联盟链的形式展开对区块链技术的研究，针对以太坊在联盟链应用的不足提出相关改进，并实现了去中心化任务发布系统，其中系统中任务发布者与接受者之间的赏金交易通过区块链中智能合约实现。本文的主要工作有：

（1）以以太坊为例，深入学习区块链相关技术与概念，并对区块链中的数据层和共识层深入探讨，并引出智能合约相关概念。

（2）针对以太坊区块链基于工作量证明的共识机制在联盟链场景下表现出的算力浪费的问题，本文提出采用 PBFT 共识算法，并针对该共识机制进行改进，通过设计区块生成协议并结合超时机制的方法，减少共识过程中的通信开销。

（3）根据联盟链的特点，提出对以太坊中账户存储优化方案，

结合 Merkle Patricia 树的数据结构特点，对每笔交易涉及到的两个账户，找到账户地址的公共前缀后同时修改两个账户的内容，并采用 K-means 聚类算法将关联账户聚类，提高了交易执行过程中账户修改的效率。

（4）对改进后的以太坊进行联盟链形式的区块链搭建，并开发实现了该平台下的去中心化任务发布系统，该系统中交易相关的操作由区块链完成。

关键词：以太坊；PBFT ；Merkle Patricia 树；K-means

RESEARCH AND APPLICIATIONS ON THE KEY TECHNIQUES OF DECENTRALIZED TRANSACTION BASED ON BLOCKCHAIN

ABSTRACT

The Blockchain technology was invented along with the creation of Bitcoin in 2009. As a public ledger of all Bitcoin transactions that have ever been executed, the Blockchain technology is transparent and reliable and it has an ability of decentralization. Therefore, this innovated technology has drawn a lot of attention from specialists and scholars dramatically. Ethereum is an extended invention of the Blockchain technology, which was designed as a decentralized platform for decentralized applications. Ethereum enables the users to expand the use of the Blockchain technology from financial industry to other industries. Recently, members of consortium from different industries have successfully solved problems relate to decentralization with the Blockchain technology, therefore, the Consortium Blockchain is now accepted by more and more corporations. The objective of this essay is to describe the analysis and study of the Blockchain technology using Consortium blockchain in order to improve any weakness that may exist in Ethereum on applying Consortium Blockchain and the conduction of decentralized platform based on the Blockchain technology is introduced.

This essay is to describe the analysis and study of the Blockchain technology using Consortium blockchain in order to improve any weakness that may exist in Ethereum on applying Consortium Blockchain. Furthermore, it also introduces the conduction of decentralized platform based on the Blockchain technology. The main objectives of this essay are described as follow:

(1)Using Ethereum as an example to demonstrate the related concepts of the Blockchain then further analyzing the data layer and the consensus layer to bring out the related concepts of Smart Contract.

(2)In terms of the issue regarding to the waste of hash rate in consortium blockchain based on the proof-of-work, which is a consensus mechanism, the essay demonstrates an improvement of reducing transmission consumption during consensus process by using PBFT consensus algorithm to design a new block produce rule to apply with timeout mechanism.

(3)Demonstrating an accounts storage optimization plan based on the characteristics of

consensus blockchain, which includes modifying two accounts from a single transaction simultaneously by using Merkle Patricia tree's data structure and improving account modification efficiency by using K-means clustering algorithms to cluster relevant accounts.

(4)Building new consensus blockchain on the improved Ethereum, then apply it to the conduction of a new decentralized platform.

Qingwen An (Computer science and technology)

Supervised by Qiubo Huang

KEY WORDS: Ethereum; PBFT; Merkle Patricia Tree; k-means

目录

第 1 章 绪论

近些年来，互联网贸易越来越普及，然而却始终摆脱不了所有的交易系统都基于信用模式的特点，即需要一个可信任的第三方去保证交易的执行。2009 年中本聪在《比特币：一种点对点电子现金系统》中，首次提出了区块链这一概念[1][2]。比特币的产生真正意义的实现了去中心化可信任交易平台。区块链技术的提出也引起了计算机相关技术领域的学者的关注并进行研究，为区块链的发展产生了巨大的动力。

本章首先介绍了本文的研究背景，对区块链技术的发展进行了简要的概述，引出本文对区块链上的应用研究的意义与价值；其次详细地介绍了区块链的研究现状和以太坊中相关技术的研究成果；最后给出本文的研究内容和论文的组织结构。

1.1 课题研究背景和意义

比特币的出现引出了区块链这项技术，比特币区块链一种基于电子虚拟货币的去中心化可信任的交易系统。比特币的出现在金融领域与计算机领域内引起了广泛的关注，专家和学者们也展开了对区块链技术的进一步研究。近些年来，随着区块链技术的不断创新进步，在国内外引起了研究区块链的浪潮。基于区块链的应用也从最初的电子货币向着智能合约的方向发展，通过编写智能合约实现不同的应用，并部署到区块链中，实现应用的去中心化执行。区块链从本质上看是一套分布式系统，通过共识机制和激励机制实现了系统的去中心化与安全可信。早期的区块链技术应用就是利用虚拟电子货币实现去中心化的交易。采用数字加密技术实现点对点的交易的基础上，将交易与时间戳一并写入到一个基于随机哈希工作量证明 （Proof of Work，PoW）的区块中，将区块添加到一个无限延伸的链式结构中，并以激励机制鼓励全网共同维持该链正确的延伸下去[3][4][5][6]。区块链技术的产生打破了人们对传统交易的认识束缚，随着以太坊[7]开发出一套可以执行图灵完备的脚本语言的虚拟机，并引入智能合约的概念，在以太坊区块链中可以通过编写智能合约实现各种去中心化应用（Decentralized Application，DApp），这也使得区块链的应用已经不仅仅局限于电子货币的交易，基于区块链的去中心化应用也推动着区块链技术应用于各行各业。

而另一方面，人们也逐渐发现区块链技术所带来的机遇与创新，人们也不断的去开发区块链技术的商业价值。美国纳斯达克在 2015 年 12 月首次推出基于区块链技术的证券交易平台 Linq，这也是金融证券市场去中心化趋势的重要里程碑[8]；2016 年 9 月，区块链支付公司 Cricle 与苹果的 iMessage 联合起来，允许用户通过 iMessage 短信平台使用美元、欧元、英镑和比特币付款。随着区块链

应用商业化，区块链技术也从公链形式向联盟链、私链形式发展。各个区块链联盟也在近几年大批涌现，国外 R3 和 HyperLedger[9]项目比较出名，致力于全球化账本的研究与实验，实现高效低成本的交易。其中 R3 联盟最初是由全球 30 多家银行共同组织的机构，目的是利用区块链技术解决目前银行之间清算中心化的问题。Ripple 网络更是在 2016 年 10 月与 R3 联盟进行了一次去中心化境外转账实验，从而解决现有的境外交易清算慢并且需要集中化清算的问题，R3 联盟成员通过区块链技术实现联盟内去中心化，完成去中心化的境外交易处理。联盟链可以说是区块链技术商业化应用的最重要的一个方向，也是现阶段区块链技术研究机构的主要研究方向。

现有几家企业各自有着本企业下用户的账户信息，企业想要通过相互之间联合账户信息搭建跨企业用户的任务发布平台，并根据任务完成情况进行转账交易，采用传统的中心化运行该任务发布平台会造成企业之间的不信任，需要第三方进行监管，这样做势必会造成一定的开销浪费。本文基于以上背景，提出采用区块链技术实现去中心化可信的任务发布交易平台，并以联盟链为应用场景，实现适用于联盟链的区块链应用平台，通过智能合约控制任务状态并实现赏金去中心化转账的功能，完成去中心化任务发布平台。

1.2 国内外研究现状

随着区块链技术的关注度不断提高，该技术也在近些年取得了快速的发展，区块链的应用场景也从最初的电子虚拟货币的发放交易，延伸到金融领域，到如今已经发展到各行各业的应用当中。随着区块链技术的提高，区块链 1.0、区块链 2.0、区块链 3.0 并行的高速发展，实现真正意义的将去中心化可信系统全行业覆盖[10]。

比特币的出现第一次引出了区块链的概念，区块链作为一种分布式系统，在决策权高度分散的去中心化系统中，实现了高效的共识机制，这也是区块链技术实现去中心化的核心要素。比特币采用基于工作量证明的共识机制达成系统的共识，通过算力竞争保证整个系统的一致性和安全性。在比特币中各个节点（矿工）共同竞争解决一个求解过程复杂但却很容易验证的一个 SHA256 数学难题[11]，这个过程称为挖矿。PoW 机制结合比特币的发行、交易和验证，可以抵御节点的恶意破坏，通过算力竞争来保证系统去中心化可信任的运行[12][13]。然而 PoW 共识机制需要节点间进行算力竞争，因此会造成大量算力资源和电力资源的浪费[14]。而随着区块链的应用场景的拓宽，人们对共识机制也在不断的研究。King S[15]等首次提出了基于权益证明(Proof of Stake, PoS)的共识机制，该机制根据每个节点所占权益等比例降低挖矿难度系数，从而加快挖矿速度。然而该机制并没有解决区块链需要挖矿的问题，仍然会造成大量的算力浪费，依然不适用于商业应用。Kwon J[16]提出了基于授权股权证明（Delegated Proof of Stake，DPoS）的共

识机制，该机制采用类似于董事会投票，由持币多者投票选出节点并由这些节点进行一致性验证，从而大幅提高达成共识所需要的时间，并且不需要挖矿过程。但是该机制需要依赖于代币才可以实现，同时股权的分配不均容易造成权力集中从而影响可信性。Ripple Lab[17][18]提出了 RPCA（Ripple Protocol Consensus Algorithm）共识机制，该机制结合拜占庭将军问题，摆脱了通过挖矿达成共识的限制，该共识机制的提出也推动了联盟链的发展，使区块链技术走向商业化，但 Ripple 注重于解决跨地域交易的问题，注重账本管理，重心并不在去中心化应用中。Linux 基金会推出的超级账本项目[19][20][21]项目借鉴了 Ripple 提出的共识机制，采用了 PBFT（Practical Byzantine Fault Tolerance）机制达成一致性，然而超级账本项目仍然处于开发阶段，并没有投入到生产应用中去。以太坊提出了下一代共识机制 Casper[22]，系统的一致性需要依靠验证人的投注完成，这个机制也实现了共识过程的高效性，但这个机制如今只是一个想法，并没有实现。这些共识机制的提出也拓宽了区块链的应用场景，针对不同的场景选取合适的共识机制显得尤为重要。

与此同时，区块链的应用从最初的电子虚拟货币交易系统向更多的领域的应用发展。国内外近些年建立了诸多区块链联盟，将区块链技术应用于现实的生产环境中去，这也使得各行各业看到区块链技术将会带来的商机，尤其引起了金融相关行业的关注。其中全球几大银行所建立的 R3 联盟便是为了致力于应用区块链技术，以去中心化的方式提供更优质的服务，并提出了基于联盟链形式的分布式私人账本 Corda[23]，实现联盟内的去中心化，并应于金融行业，但该项目并没有开源，并仅服务于联盟内成员。但是 R3 联盟所取得的成功，引起了区块链联盟的热潮，不论是由 Linux 基金会创建的超级账本联盟还是国内建立的中关村产业联盟都受到了广泛的关注，而这些联盟更加注重对区块链技术的应用研究，挖掘区块链技术所支持的去中心化应用场景，并都取得一定的成功，但这些项目仍然处于开发阶段。区块链的联盟链发展是推动区块链技术不断进步的重要因素，也是未来几年内区块链应用的一个大的方向。但是由于联盟链刚刚起步，针对于联盟链场景下的区块链研究也比较少，如何将比较成熟的基于公链的区块链技术应用于联盟链场景中还没有很好地解决方案。因此本文着眼于联盟链，设计研究基于联盟链的应用实现具有极大的现实意义和未来价值。

以太坊是继比特币之后又一个基于公链的区块链应用。以太坊从本质上讲和比特币类似，定义了一个全球化账本，并且该账本是通过各个不可信节点通过 P2P（peer-to-peer）网络共同维护，各个节点通过求解一个数学难题进行算力竞争来保证其去中心化可信任的功能。以太坊最大的突破是可以执行图灵完备的脚本语言，这也使得以太坊不单单是一个去中心化的货币发行和交易系统，更是一个可以去中心化执行合约的系统，智能合约的出现也拓宽了区块链的使用场景，推动着区块链 2.0、区块链 3.0 的发展，实现应用的去中心化，这也是以太坊的

最大优势。与此同时，以太坊结合近些年的区块链研究成果进行整合。以太坊对于区块结构进行了修改，以 Merkle Patricia Tree（MPT）[24]的数据结构在区块中存储了账户树，解决了查询账户余额需要对交易回溯的过程，高效的存储与校验也是其一大优势，但是在用户量大的情况下，Merkle Patricia 树深度较深，往往会引起交易执行耗时较长的问题，影响交易执行效率。以太坊从设计开始就定位应用于公链中，因此仍然采用了 PoW 共识机制。从矿工角度看，挖矿过程中往往有大型矿池通过叠加 GPU 实现高算力实现很高的挖矿效率，从而影响了以太币发放的公正性。以太坊为了避免这种情况提出了基于 Ethash[25]的挖矿算法，使挖矿过程和内存相关，在一定程度上解决了上述问题。而从整个系统角度，挖矿过程存在着不确定性，因此期望出块时间过快往往会造成支链过多需要花更多时间来进行最长链的校验，这也是比特币选择 10min 出块的依据。而 Yonatan Sompolinsky 等[26][27]提出的“幽灵协议”（GHOST，Greedy Heaviest Observed Subtree）,对区块链验证采用了树的结构验证节点，而不是对最长链进行校验，这样做保证区块链的安全性的同时提高了基于 PoW 共识机制的出块率，解决了传统 PoW 共识机制校验慢的问题，该方案被以太坊采用实施，并且结合奖励机制，对正确的支链也进行以太币的发放，弥补由于网络延迟对个别节点的不公现象。

虽然该协议一定程度的提高了 PoW 共识机制的达成共识的效率，但这并没有本质的解决以太坊基于 PoW 共识机制不适用于联盟链的问题[28]。PoW 机制不可避免会造成算力浪费，同时在联盟中节点少的情况下，单台节点很容易将算力提升超过全网的 50%，从而掌控整个联盟链。与此同时，将区块链技术作为商业应用，不可避免的会对效率有着更高的要求，以目前 Merkle Patricia 树修改账户信息的方式，当账户多的情况下效率较低，提高账户修改效率也是将以太坊应用于联盟链中需要解决的关键问题。但是以太坊的成熟的以太坊虚拟机（Ethereum Virtum Machine，EVM）可以高效的处理图灵完备的脚本语言，为实现去中心化应用提供了前提条件，也是本文基于以太坊提出实现联盟链的主要原因。因此本文提出采用改进 PBFT 作为以太坊共识机制并加以实现，同时对以太坊中账户存储结构进行优化，并针对该系统实现去中心化的任务发布系统，不同联盟体账户之间通过任务发布接收实现赏金的流通，完成账户间交易。

1.3 研究的主要内容与创新点

本文以企业间实现联合账户信息搭建跨企业用户的任务发布系统为背景，并以账户积分作为赏金流通于任务发布系统中，设计并实现基于改进 PBFT 以太坊共识机制的任务发布平台。本文首先对区块链技术进行了深入学习，针对区块链技术中共识机制的研究现状进行了分析对比，并针对联盟链场景提出采用 PBFT 算法应用于以太坊共识机制中。结合以太坊现有区块结构实现基于 PBFT 算法的

以太坊共识机制，同时针对 PBFT 算法中的检查点机制和视图切换协议进行改进，与以太坊区块生成机制结合采用超时机制完成检查点协议和视图切换协议的相关功能，从而减少了系统在这两个过程中的信息传输量。接着本文针对以太坊中的状态树效率低的问题提出了账户存储优化改进方案，结合 Merkle Patricia 树结构提出了针对一笔交易中的两个账户同时更改账户信息的算法，减少账户存储过程中的查找和哈希运算次数，提高交易在以太坊中的执行效率，并结合该算法采用 K-means 聚类算法对关联账户进行聚类，并针对 K-means 算法中初始值选择提出了基于关联度的选择算法，并进行实验验证。最后将改进的以太坊进行联盟链搭建，通过 Solidity 语言编写智能合约，并在合约中实现任务发布接受等功能，并部署到联盟链中，实现基于联盟链中的去中心化的任务发布系统平台，由区块链完成账户之间的交易去中间化实现。

本论文的主要创新点有：

（1）在去中心化任务发布平台中，需要进行转账交易，实现赏金的功能。针对这类交易（类似于银行的转账交易），本文提出采用区块链技术搭建联盟链进行解决，并通过智能合约的方式代替第三方监管实现系统的联盟内去中心化。

（2）由于以太坊中采用的 PoW 共识机制造成大量算力浪费，同时在联盟链节点少的情况下可能由于某节点计算能力强可以对区块链系统进行攻击，使系统不可信。因此本文认为 PoW 共识机制不适用于联盟链中，因此本文提出基于 PBFT 以太坊共识机制，并修改区块校验方法，解决 PoW 共识机制在联盟链中的弊端。

（3）本文结合以太坊区块结构改进 PBFT 算法，提出将协商过程与执行过程进行分离，减少校验次数提高校验效率，同时结合以太坊区块生成协议，对 PBFT 算法中的检查点机制和视图切换协议进行更改，减少该算法在执行检查点检查和视图切换协议时的通信开销。

（4）本文针对以太坊状态树存储结构，提出改进账户存储方式，结合 Merkle Patricia 树对交易时账户修改过程提出交易账户同时修改算法，减少索引和哈希次数实现优化。根据上述算法提出关联账户的想法，通过聚类算法筛选出关联账户并放到同一父节点下，对账户的操作采用以交易的方式提交，同时修改交易双方账户，从而达到优化目的。

1.4 论文的章节安排

本文一共分为六个章节，各个章节的内容安排如下：

第一章为绪论，简要介绍了课题的研究背景与意义、国内外对区块链的研究现状和存在的问题、论文研究的主要内容与创新点以及章节安排。

第二章介绍了本文所需要用到的相关技术原理。介绍区块链中区块的数据结构和实现去中心化的原理，对现阶段常见的共识机制进行阐述与对比，分析每种

共识机制的优势和不足，最后介绍了智能合约的相关概念。

第三章对于改进 PBFT 机制的设计以及基于该共识机制的以太坊实现。详细的介绍了改进 PBFT 共识机制的基本执行原理，并从区块的生成、消息的通讯以及区块的校验给出详细的设计，并对 PBFT 中检查点机制和视图切换协议进行改进，减少通信开销，实现改进 PBFT 的以太坊共识机制，并给出相关实验证明。

在第四章对于以太坊中存储账户信息的 Merkle Patricia 树进行存储优化，提出多账户数据批量修改算法，并通过 K-means 聚类算法获得关联账户，结合批量修改算法实现对账户操作的优化过程，并结合样本数据进行实验。

第五章对本文第三、四两章提出的改进进行应用的实现，通过搭建改进后的以太坊联盟链，实现联盟内的去中心化平台，并设计基于该平台的任务发布系统，其中该系统中赏金流通采用以太坊智能合约实现去中心化。

第六章对全文内容做了总结，并对未来的工作了做了展望。

第 2 章 区块链相关技术

本章首先介绍了以太坊的相关概念，并对以太坊中相关技术进行着重介绍。接着针对以太坊中PoW共识机制的不足，详细地介绍了其他共识机制，并对近些年来常用的共识机制进行分析，分析每种共识机制的优劣。最后介绍下区块链中的智能合约的相关概念。

2.1 以太坊的相关概念

以太坊是一个基于公链的分布式计算平台，并提供了一个去中心化虚拟机，该虚拟机可以执行图灵完备的脚本语言。以太坊的系统架构如图 2-1 所示。以太坊是由智能合约层、激励层、共识层、网络层和数据层构成，其中数据层中包含了以太坊中最基本的数据结构和账户加密算法，这也是以太坊的基础组成部分；网络层中主要包含以太坊中各节点的数据传输校验机制；共识层中以太坊采用基于工作量证明的共识机制；激励层则将奖励机制包含进来，主要用来激励节点自主挖矿，维持以太坊运行。数据层、网络层、共识层、激励层也构成了基本的区块链结构。智能合约层可以说是以太坊特有的，智能合约层封装了可以执行图灵完备的脚本语言的虚拟机，可以通过编写脚本语言作为智能合约部署到以太坊区块链中实现应用的去中心化。

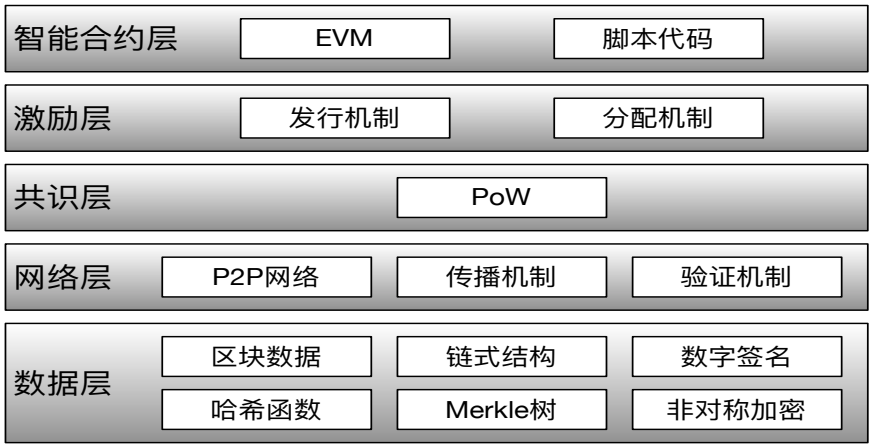


图 2-1 以太坊架构模型

2.1.1 数据层

数据层封装了区块的数据结构和数据加密相关的内容。其中区块结构如图 2-2 所示。一个区块中分为区块头和区块体两个部分。在区块体中包括了所有的交易相关信息，而区块头中则包含了前一区块的哈希值、时间戳、随机数、难度系数和该区块中 Merkle 树[29][30]根哈希值，其中时间戳是区块的生成时间，随机数是数学难题的解，难度系数可以动态调整该数学难题的难度，Merkle 树则是对该区块体中交易的唯一的标识。这些组成元素通过一定的哈希算法形成该区块

的哈希值，区块之间通过区块中的 parenthash 和区块中的前一区块哈希值对应，并进行首尾相连，形成链式结构，并将从创世块开始所形成的最长的链称为主链。在向区块链中逐个添加区块的过程中，往往会因为网络延迟造成多个新的区块同时添加到主链产生分支的情况，这里就需要对难度系数的进行比较，以整条链中累计工作量最大的支链作为主链，继续进行区块的添加过程。区块头中的时间戳则保障了区块的有序性，对于篡改或伪造区块数据有一定的防范作用。难度系数是用于动态调整挖矿难度，维持出块时间在期望时间上下浮动，同时难度系数是工作量证明的验证依据。

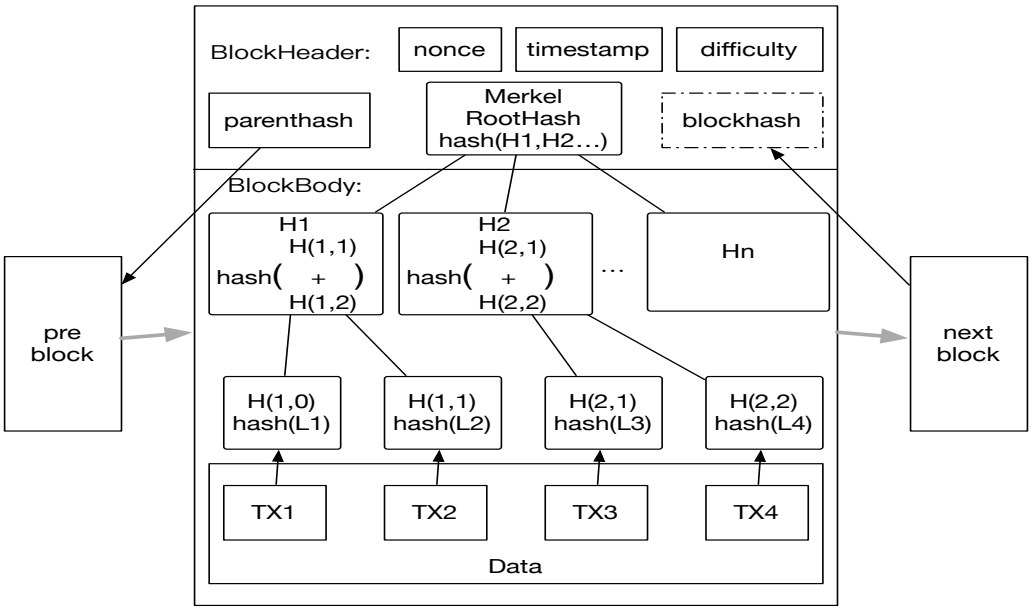


图 2-2 区块结构

Merkle 树作为区块链中至关重要的数据结构，实现了对交易的快速校验。在以太坊中区块中总共包含三个 Merkle 树，分别记录交易树、账户状态树和收据树。其中交易树如图 2-2 所示，Merkle 树通过将交易进行分组哈希，将哈希值插入到 Merkle 树中并递归哈希，最终形成 Merkle 树根哈希值，在 Merkle 树中任何节点的数据发生改变，都会造成通往该节点的路径中的所有节点的哈希值发生改变，因此将 Merkle 树根哈希值封装到区块链头中，只需要校验根哈希值便可以校验区块所有交易的正确性。在以太坊中交易树与收据树采用的是 Merkle 树，而针对于账户状态存储，以太坊设计 Merkle Patricia 树存储账户状态树。每当执行交易或者操作账户时，都需要对状态树进行查询或者修改，频繁地增删改查的操作，需要状态树具有快速查询和快速计算根哈希的能力，将 Merkle 树和字典树结合，提高了查找效率并且能够快速计算根哈希值。通过这样的数据结构，可以在一次插入或修改操作后快速生成 RootHash 而不需要重新生成一颗树。同时树的深度是有限制的，即使考虑攻击者会故意地制造一些交易，也不会使得该树无限延伸，否则攻击者可以通过操纵树的深度，执行拒绝服务攻击（Denial-of Service Attack, DoS）[31]，使得更新变得极其缓慢。并且树的根哈希值只和叶子

节点的数据有关，和数据的更新顺序无关。换个顺序进行更新不会改变根的哈希值。

在区块链中的挖矿过程和区块存储等过程中都需要进行哈希计算。以太坊中采用的哈希函数是 Ethash 和 SHA3 哈希算法，其中 Ethash 主要用于挖矿过程进行求解随机数中使用，而在存储区块过程中采用了 SHA3 哈希函数，该哈希函数和 SHA256 哈希算法类似，都是安全可靠地加密哈希算法。将任意长度的数据经过编码后进行 SHA3 哈希，形成 256 位二进制数字来进行存储。一方面 SHA3 哈希算法的抗碰撞性很高，很难发生哈希碰撞[32]；另一方面经过哈希之后数据长度一致，并且哈希的单向性可以很好的保护原始数据。

以太坊从一定程度上可以看作对全球公开的账本，而账户的安全性则依靠数字签名和非对称加密算法实现[33]。以太坊通过公钥和私钥对数据进行加密和解密。一个账户首先通过私钥加密后将数据发送到公网中，只有对应的公钥可以解密从而确保信息的安全性。以太坊正是以这种方式来构建节点间的相互信任。账户的签名验证过程如图 2-3 所示。其他用户当收到 B 发来的消息 2 后，首先通过 B 的地址找到 B 的公钥，然后用 B 的公钥对消息 2 进行解密，然后将消息 1 的内容与 C 的公钥相连经过哈希计算与解密结果比较，如果相同则认为消息是由 B 发来的。以太坊正是利用这项技术实现了对消息的加密过程。

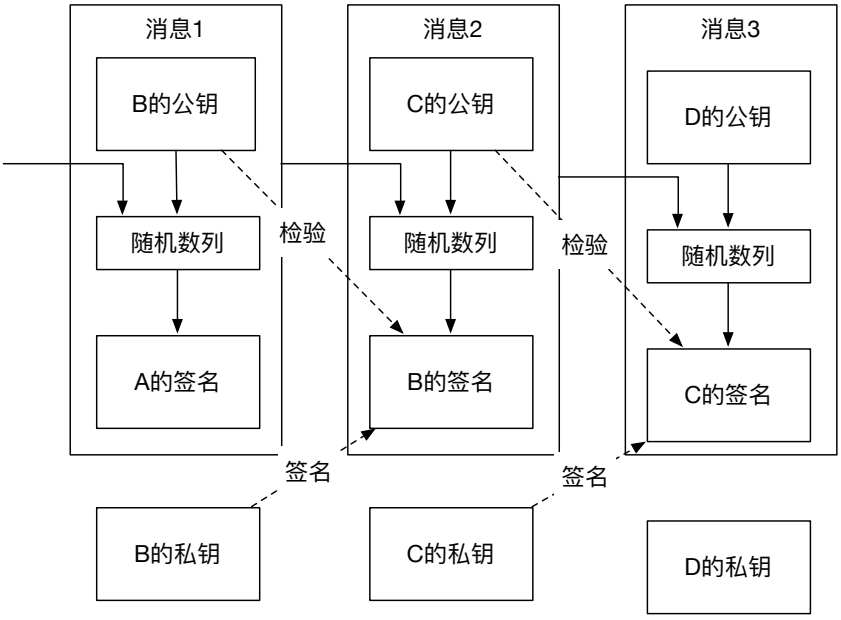


图 2-3 账户的签名验证过程

2.1.2 网络层

网络层中包括了 P2P 网络[34][35][36]的实现、消息传播机制和数据验证这三方面。网络层是实现以太坊节点间交互的网络保障。以太坊采用 P2P 网络来解决以太坊中去中心化等特点，其网络模式如图 2-4 所示。P2P 网络采用扁平式拓扑结构[37]，因此各个节点的地位相同，没有中心化服务器，每个节点都等同的参与以太坊系统中的路由、校验、广播等功能。P2P 网络架构是以太坊实现去中心化的

网络基础。

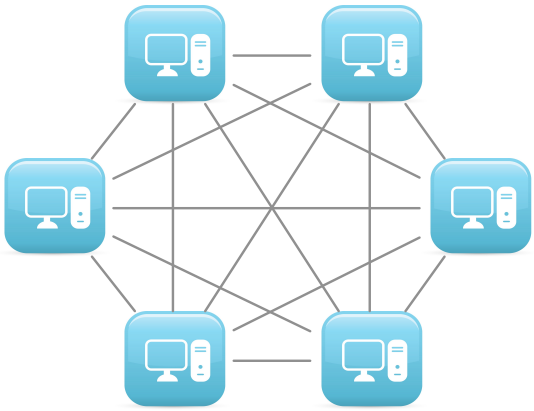


图 2-4 P2P 网络模式

以太坊中数据传播分为主动发送和索要数据两种方式。当一个节点接收到交易或是生成的区块时，会以广播的形式向全网其他节点发送消息。由于采用的是 P2P 网络架构，因此在消息广播的时候并不能保证所有节点都接收到该消息，这时候节点可以向邻近节点发送数据索取请求，并由邻近节点发送数据进行同步。同时网络层中还包括数据校验机制，当节点接收到邻近节点传来的信息时，会对数据的有效性进行校验，对于有效信息会将该条信息向邻近节点进行广播，实现全网消息同步，而对于无效信息，节点不做数据转发的操作，防止造成以太坊网络中恶意攻击，同时会对传送无效信息节点在一段时间内断开连接。

以太坊的网络层运用了分布式网络技术，利用 P2P 网络协议实现了去中心化，整个网络具有高可用性。

2.1.3 共识层和激励层

在一个高度自治的去中心化分布式系统中，如何使各节点达成共识是以太坊的核心内容，也是分布式系统所面临的不可避免的问题。以太坊采用了基于工作量证明的共识机制，让一个决策权分散的系统达成一致。各个节点通过算力竞争解决一个数学难题，并对第一个解决出来的节点进行奖励，从而促进整个网络实现算力竞争。

在以太坊中使用的工作量证明哈希函数是 Ethash，该哈希算法已经被证明没有有效的办法进行逆向运算，只能通过尝试求解。每一次的挖矿过程首先根据区块头中的难度系数计算出一个目标值，然后对区块头中的信息和一个随机数进行哈希，得到的哈希值小于目标值则认为挖矿成功，该随机数则是这次运算的解。因为哈希算法的特点，所以每次输入即使只有一位不同也会造成哈希值差异很大，因此通过哈希算法的不可逆性使该求解过程只与计算机算力有关。该求解过程虽然会消耗大量算力，但是对解的验证却十分简单，只需要进行一次哈希运算后与目标值进行比较便可校验。而攻击者倘若想要伪造区块信息并添加到区块链中，则需要联合全网 51%的算力来同时进行伪造，因此通过工作量证明的共识机制实现了以太坊一致性的问题。

基于 PoW 共识机制首先需要网络中节点的总算力要足够大，这样才不会出现算力集中的情况，因此需要依靠激励机制来鼓励全网挖矿，这样才能维持全网算力总和足够大，保证以太坊的去中心化是安全可信的。与比特币相似，以太坊为了激励矿工挖矿向成功挖矿的矿工发放虚拟货币——以太币，以太币是流通于以太坊中的一种电子虚拟货币，矿工通过成功挖矿赚取以太币。

2.1.4 合约层

合约层主要包括脚本语言和执行脚本语言的虚拟机[38]。以太坊中可以运行图灵完备的脚本语言，因此可以通过编写复杂的智能合约实现去中心化应用。合约层是区块链实现去中心化应用的必要条件。其中智能合约是一种编码，在合约中有着可以自动运行的业务逻辑，并依靠以太坊虚拟机进行执行。

以太坊中智能合约的代码是基于堆栈的字节码低级语言所写的，因此代码是由一些字节组成，每一种字节代表着不同的操作。通常来讲，代码的执行过程是无限循环的，程序每执行一次，计数器就会增加 1，直到代码执行完成或是执行过程中发生错误，遇到中断指令。代码的执行操作中分为三种存储数据空间：

（1）堆栈：先进先出的数据结构

（2）内存：一个无限的字节数组

（3）合约的长期存储：存放了键值对，不同于堆栈和内存，合约内容可以长期保存

以太坊虚拟机是执行交易代码的引擎，这也是以太坊与其他区块链最大的不同之处。它不是沙盒而是完全独立的，这也意味着代码在 EVM 中运行的时候不需要网络、文件系统甚至与其他进程独立。以太坊中有两种账户一种是本文之前提到的需要依靠公私钥管理的账户，另外一种则是合约账户，合约账户的地址取决于合约的建立时间。EVM 的执行过程非常简单，当以太坊虚拟机运行的时候，它的计算状态可以通过元组（block_state, transaction, message, code, memory, stack, pc, gas）定义，其中 block_state 是包含所有账户信息的全局状态。程序执行时，根据 pc 找到当前指令，并执行该指令并对元组数据进行修改，从而实现智能合约的执行过程。

2.2 共识机制

分布式系统中，最重要的就是节点间的共识问题。以太坊采用了基于工作量证明的共识机制，该机制提供了一种解决去中心化系统中各节点如何达成一致的问题。然而该机制需要依靠大量的算力来保证共识，造成大量的算力资源浪费，人们正在寻找更好的替代机制实现区块链各节点达成一致问题。

2.2.1 PoW 共识机制

作为区块链技术的开创者比特币使用的共识机制就是基于工作量证明的共识机制，并且这一机制在其他公链环境下的区块链系统中也得到广泛应用。在

PoW 机制中，每一个参与算力竞争的节点叫做矿工，求解随机数的过程叫做挖矿。每一个矿工当接收到交易或者有新的区块成功添加到区块链上，会进行新一轮的挖矿。挖矿过程是求解一个数学难题的过程，将该矿工正在生成区块的区块头信息进行哈希，由于区块头中包含唯一标示该区块的信息，因此矿工的每次挖矿中区块的哈希值一定各不相同。在每次挖矿前，会计算出一个目标值，该目标值与难度系数相关，矿工的每次挖矿过程就是生成一个随机数，并将这个随机数与唯一标识区块的哈希值进行双 SHA3 哈希，并将生成的结果与目标值进行比较，如果小于目标值则认为挖矿成功，否则将重新生成随机数继续运算直到小于目标值为止。整个过程由于双 SHA3 是不可逆哈希过程，对于求解小于目标值的过程只能通过不断计算获得，整个过程只与计算能力相关，因此可以通过求解的速度来判断算力。挖矿过程虽然耗时，但校验过程却异常简单，只需要将随机值与区块头哈希重新进行一次双 SHA3 哈希并判断是否小于目标值即可。通过 PoW 共识机制，全网共同竞争求解随机数，倘若某些矿工想使区块链发生错误只有汇集全网超过 50%的算力进行挖矿才可以办到，而这对于公链来说是近乎不可能完成的，因此 PoW 共识机制可以维持全网节点的一致性。

中本聪将 PoW 共识机制与奖励机制结合，促使全网共同挖矿，这一新的理念可以说是真正意义上实现了去中心化。但是通过算力竞争实现分布式系统的一致性代价很大，以比特为例，在 2016 年比特币全网的算力要比全球前 500 的超级计算机算力总和还要多。同时该机制也逐渐表现出其不可靠性，近年来越来越多组织通过集结全球最大的矿池获取更高的算力，当某一组织可以获得全网的 50%的算力，整个区块链将不再可靠。

2.2.2 PoS 共识机制

中本聪在 2010 年提出的币龄概念，币龄的定义是货币所持有的时间段，即将货币的金额与所持有的时间进行数学运算后得出的结果。而 PoS 机制[15]正是基于币龄这个概念提出的，PoS 共识机制也可以保证区块链去中心化可信的运行。

PoS 共识机制的目的是提供一种在分布式系统中达成共识的方法。不同于工作量证明的共识机制，该机制中节点生成一个新的区块时会提供一种证明，证明该区块在被网络接受之前获得过一定数量的货币。生成一个区块需要输入一定数量的货币，这样可以证明该节点对货币的所有权。而这些货币将影响到生成区块时挖矿过程的难度系数。

在基于权益证明的区块中定义了一种新的交易称之为利息币交易。利息币的交易与传统的交易不同，利息币交易过程中会消耗交易者的币龄从而获取在网络中生成区块的权利，同时也获得对 PoS 机制下的造币的权利。利息币中首先要输入一定的核心（Kernel），接下来的过程与 PoW 机制中挖矿过程类似，同样是通过求解随机数，使其经过哈希计算满足目标值。但是 PoS 共识机制中，哈希计算对随机数求解时，会降低求解难度，这样会极大的缩小寻找随机数的空间，从而

减少能源的消耗。因此 Kernel 消耗的越多，哈希计算空间则会越小，随机数的计算则越容易。与 PoW 机制中所有节点的目标值一样不同，PoS 机制中各节点的目标值不同，因此权益大的节点要更加容易挖矿[15]。

在校验方面，该机制中对主链的判断不再是通过难度系数判断，而是通过对币龄的消耗进行判断。因为在区块中的每笔交易都会消耗币龄，最终区块链选择币龄消耗最多的链作为主链。这样的设计也降低了基于 PoW 共识机制所提出的51%的算力攻击问题，因为在 PoS 共识机制中，节点首先要控制众多的货币达到足够的币龄后才可以伪造区块进行攻击，而这个过程所要消耗的成本要远高于集中全网 51%的算力的成本。同时攻击者在攻击主链时会消耗币龄，对于攻击者来说也是一种损失。

PoS 共识机制中，挖矿的过程不仅和节点的算力有关，也和节点的权益相关。权益大的节点的挖矿过程会比权益小的节点更加容易，从而提高了挖矿效率，减少算力浪费情况，PoS 机制的出现也说明人们逐渐意识到 PoW 机制所带来的算力浪费的问题。然而 PoS 机制从本质上并没有摆脱挖矿的束缚，节点仍需要进行挖矿，因此还是会造成较大的算力浪费。

2.2.3 DPoS 共识机制

DPoS（Delegated Proof of Stake）机制[16]是在可信的加密货币网络中，提供事务处理和去中心化的共识协议的一种方法，目的是为了减少基于 PoW 共识机制中的算力浪费和资源开销。

DPoS 共识机制和董事会投票表决有些类似。在一个去中心化系统中，将决策权力分发给所有股东，而当股东投票超过 51%时，则认为该决定被通过，并且该决定不可逆。在该机制中有一个重要角色叫做代表，代表是生成区块的节点，想要成为代表首先要支付一定的保障金来保证代表的可信性。而用户则拥有选举代表的权利。每个用户可以投票选举一个值得信任的代表，在全网中获票最多的前 n 个代表则有生产区块的权利，这 n 个代表持有的票数相当于该节点持有的股数。这 n 个代表将按时间表进行轮流生产区块，生成的区块通过的股票数超过51%则认为区块生成成功，代表每生成一个区块将从区块中交易的手续费中获得收益。而这些收益也将是代表维持在线参与的奖励机制。

DPoS 共识机制中，代表的可靠性显得格外重要，在股东选取代表时，股东可以看到代表出块的错误率，从而股东可以正确的选择代表。另一方面，代表出块是按时间轮流产生，因此当一个代表错误的产生一个区块时，只要不得到 51%的认可，那么该区块将会在下一个时间段由其他的代表生成。

可以说 DPoS 共识机制相比与 PoS 共识机制更加有效。DPoS 共识机制是真正意义上摆脱挖矿的共识机制，然而该机制依赖于所有参与者的投票，当参与度不够的时候，那么代表往往会集中在全网中持有大量选票的持有人手中，从而失去了去中心化的特点。

2.2.4 拜占庭共识机制

区块链可以被看作是一个去中心化的共享数据库，并且可以抵御任何节点的攻击和恶意行为。通常来说，拜占庭容错（Byzantine Fault Tolerance，BFT）共识协议[39][40][41]是可以容忍恶意行为的共识协议。该共识机制叫做拜占庭的原因是这种容忍恶意行为的问题和历史中拜占庭将军尝试通过使者来通知其他将军一同攻打罗马，然而其他将军中可能存在叛徒的场景非常相似，因此采用了拜占庭共识机制这个名字。

当一个进程停掉了，该进程只是无法继续工作。但是拜占庭错误，程序错误可以是任意形式的。系统崩溃的处理过程很简单，因为在这个过程中进程不会对其他进程传送假消息。系统只需要容忍这个错误并通过半数以上的进程同意，便可以达成共识，因此在这种情况下系统可以容忍一半的程序停掉。如果发生错误的进程是f，那么要求该系统至少有 $2f$+1 个进程。但是拜占庭错误更加复杂，假如一个系统中有 $2f$+1 个进程，其中f个进程发生拜占庭错误，那么他们可以协商并发送相同的信息给另外f+1 个没有错误的进程，在这种情况下，系统仍会发生错误。因此一个拜占庭系统中可以容忍错误进程的数量要小于不是拜占庭系统的进程数量。实际上，这个系统f的限制条件是$f < N/3$，其中N是系统中所有的进程数。该机制便是拜占庭机制。

在 1999 年，Castro 和 Liskov[42]提出了实用容错拜占庭（Practical Byzantine Fault Tolerance，PBFT）机制，首次提出了实用的拜占庭容错机制，该机制下 TPS 可以达到 10 万以上，因此该机制在工业系统中得到了广泛的应用。而 Hyperledger 区块链项目也正是使用了该机制，但该项目仍处于开发阶段，同时该项目更加注重的全球化账本的管理工作。一方面，PBFT 共识机制可以很好的解决分布式系统中的共识问题，尤其在联盟链中对容错率不太高的情况下。然而另一方面该机制会在发生拜占庭错误时进行大量消息传递，这对于网络开销是昂贵的。

2.3 智能合约相关技术

2.3.1 智能合约

区块链技术通过其网络架构和其实现的虚拟电子货币简化了价值转移过程。在区块链中，一笔交易分为以下三步：

（1）A 向区块链网络中发送一条消息，并在消息中定义了一笔交易

（2）B 通过广播接受该交易表明 B 接受该交易

（3）区块链网络中的参与者验证这笔交易的合法性并完成交易

可以肯定的是这种交易模式定义了基于区块链中某种货币的所有者发生了改变的过程。但是当发送或者接受交易时，也可以传递消息内的价值，就像电子信封一样，在电子信封中或许会包含电子货币并通过网络传递，但也可以传递一些其他的信息，实现附加的效用。

一个典型的交易可以看作一个简单“脚本”，这个脚本相当于一组指令集，这些指令集让区块链中节点执行交易。然而当脚本发生改变，并包含交易以外的信息，那么用户就可以利用这个脚本实现复杂的“交易”。智能合约是一个方便、可验证、可执行的商业协议条款的计算机协议，这个概念并不是新的概念，也不是区块链所提出来的。区块链实现的是智能合约的去中心化，换句话说智能合约是基于一个安全公开账本并没有中心化监管执行的合约。参与者可以在没有第三方监管的前提下，自动履行合约内容。智能合约也是区块链中实现去中心化应用的基础。

2.3.2 以太坊中智能合约

在以太坊中，智能合约[43][44]是由二进制字节码组成，可以通以太坊中推荐的 Solidity 语言进行编写后通过编译生成二进制字节码存储于区块中。Solidity 是一种合约语言，类似于 JavaScript 的高级语言，在以太坊中可以调用 Solidity 语言编译器编译合约。

在以太坊中智能合约是代码和数据的集合，并以智能合约账户地址存储于以太坊区块链中。前文提到过，智能合约以消息的形式在网络中传递，以太坊中智能合约存放于交易之中。EVM 通过一笔交易是否含有代码来判断是否是合约类型的交易，如果一笔交易中的信息是代码则会执行该合约。执行过程中发现合约账户是 NULL 则会为该合约创建合约地址，并将合约写入到该账户下。EVM 通过执行合约中的二进制字节码实现合约的执行。在以太坊智能合约定义了应用二进制接口（Application Binary Interface，ABI）是强类型的，ABI 将在智能合约编译过程中形成并持久化。在这里 ABI 类似于现实合约中的条例，是每次合约调用时必须获得的。一个函数调用数据的前四个字节指定了要调用的函数。账户通过智能合约 ABI 进行合约中的方法调用。

2.4 本章小结

本章主要介绍了以太坊的基本概念，以以太坊为例介绍区块链的相关技术。主要介绍了以太坊的系统架构，该架构主要分为数据层、网络层、共识层、激励层和合约层。其中，主要对以太坊中数据层进行了详细介绍，数据层是实现区块链校验达成共识等过程中最重要组成部分。同时，针对当前区块链技术使用到的共识机制进行了详细介绍，并对每个共识机制进行优劣分析，为本文提出的改进 PBFT 共识机制提供基础。最后对智能合约进行概念与技术的描述。本章主要针对本文所设计的相关技术概念进行分析，为后文的共识机制以及以太坊账户存储优化的提出做准备。

第 3 章 改进 PBFT 的以太坊共识机制及实现

由于以太坊采用的是 PoW 共识机制，需要进行大量的哈希运算进行算力竞争，以确保达成全网的共识。PoW 共识机制会造成算力的浪费，并且在联盟链中该机制并不可信。本文针对于该问题，提出了使用 PBFT 共识机制作为联盟链场景中的以太坊共识机制，解决了算力浪费的问题，同时结合以太坊区块特征改进 PBFT 的视图切换机制和检查点机制，减少了通讯开销，本文将改进 PBFT 算法结合以太坊给出实现过程，并进行实验验证。

3.1 问题描述

本文设计的共识机制是基于联盟链场景。联盟链往往是由一些商业联盟出于共同的利益并由这些商业联盟体共同进行区块链系统的搭建。联盟链不同于公链，联盟链中网络不对外公开，从这点看联盟链更趋向于私有链，因为区块链只由商业联盟管理，但是从另一方面，这些商业联盟之间并没有完全相互信任，因此作为每一个联盟成员都有义务去对区块链中其他成员进行监督，维持区块链系统正确的运行，这一方面联盟链和公链又非常接近，因此联盟链往往定义于公链与私链之间。联盟链中由于联盟之间存在利益进行合作，因此在联盟链中即使没有激励机制，各联盟成员也会去维护区块链运行。

以太坊中可以执行图灵完备的脚本语言，这也是区块链技术的一大进步，通过脚本语言的编写实现去中心化的应用是以太坊的一大特色，这也是区块链技术得到了商业市场的关注的一个原因。由于以太坊在最初设计时采用 PoW 共识机制，虽然 PoW 共识机制可以有效的保证区块链的可信性，并且很好的实现了去中心化，但是该机制并不适用于联盟链中，主要原因如下：

（1）PoW 机制中需要进行大量的计算来证明工作量，算力竞争会造成大量的算力的浪费和资源消耗。

（2）联盟链中往往没办法提供公链中相同数量级的总算力，因此采用 PoW 共识机制很容易被联盟链中的个别节点通过算力堆积达到全网算力的 51%，因此区块链的可信性也面临着挑战。

（3）联盟链中会因为业务不同对于交易有着很高的 TPS 要求，在以太坊的共识机制中，当 TPS 高时容易产生支链，那么对于一笔交易的确认时间就会延长，这对于要求快速响应的场景是无法忍受的。

针对以上问题，本文提出改进 PBFT 机制（Advances Practical Fault Tolerance，APBFT） 作为以太坊共识机制的研究与实现，将 PBFT 机制结合以太坊现有结构并提出改进，并将该共识机制应用于区块的一致性校验中。本章基于以太坊 JAVA 开源项目对共识机制的设计与实现，保留了以太坊中智能合约执行的相关

逻辑，对以太坊实现区块链技术的共识层和网络层进行重新设计与修改，因此本文设计目标是使系统更加适用于联盟链中，同时又保留了以太坊中智能合约的优势。

3.2 问题假设及定义

3.2.1 问题假设

本文提出机制是基于联盟链场景下，并有如下假设：

（1）有N个联盟节点，分别为$\{N_1, N_2, \dots, N_n\}$，各节点相互独立。节点之间去中心化。在系统运行过程中，节点不存在动态添加和退出的情况。

（2）节点之间建立连接采用以太坊提供的校验方式，认为该校验是安全的，并且恶意节点不能冒充其他节点连接到网络中。

（3）当节点断开连接时，系统会尝试重新连接该节点，但超过超时时间仍没连接成功，本文认为该节点发生错误，即为错误节点。

3.2.2 算法定义

在 PBFT 机制中有如下定义：

定义 1. Quorum 是由系统节点集合组成的，并且任意两个 Quorum 的交集不为空。假设系统节点集合为U，$\partial = \{Q_1, Q_2, Q_3 \dots, Q_n\}$，且$Q_i \subseteq U$，满足公式 3-1 则称$Q$为一个 Quorum。

$$\forall Q_i，\ Q_j \in \partial 且 Q_i \cap Q_j \neq \emptyset \tag{3-1}$$

Quorum 有如下性质：

（1）任意两个 Quorum 至少有一个共同的并且正确的节点。

（2）一定存在着没有错误的 Quorum。

其中本文定义 $2f$+1 个节点为一个 Quorum，这样可以保证在一个 Quorum 至少有f+1 个节点没有发生错误，其中f是该系统中出现允许最大错误节点数。

定义 2. PBFT 机制将执行过程划分为视图（View），对于每一个视图进行编号记作v，在一个共有N个节点的视图中有一个主节点，主节点的编号记为p，其余节点为备份节点称为 replica，每一个节点用整数表示依次为$\{0,1, \dots, N-1\}$，满足

$$p = v \ mod \ N \tag{3-2}$$

其中v是视图的编号，当群中主节点发生故障，则下一个编号的节点成为主节点，视图也随之发生切换，视图编号增加 1。

定义 3. 证书（certificate），本文定义系统达成共识过程中会进行消息传输，该消息称作为证书。本文采用$\{REPLY, b, v, i\}$消息格式作为证书，其中$REPLY$为消息类型，根据证书类型不同b可能为区块或区块哈希，v代表视图 View 编号，i代表节点编号。

3.3 系统设计

3.3.1 共识机制描述

在有N个节点的系统中，主节点p负责生成区块，而所有节点对该区块进行协商和验证两个过程后，对该区块达成共识。为了可以容忍拜占庭错误，本文设计系统中至少要有四个节点。各节点在初始化时采用相同的顺序。每个节点根据配置文件中 IP 地址建立节点间通讯，建立通讯后会向其他节点发送对应的公钥，其他节点通过私钥签名完成验证，验证通过说明节点间建立有效连接。在协商的过程中，对于一条消息节点有三个状态分别是 Pre-prepare、Prepare 和 Commit，分别由$PR_i^{\{v,m,n\}}$、$P_i^{\{v,m,n\}}$、$C_i^{\{v,m,n\}}$表示，其中v代表视图 View 的编号，m为证书消息，n为发送证书消息节点编号，i为当前节点的编号，共识算法如下所示。

Algorithm：Consensus()
Input：证书消息m
Output：是否达成共识
1: repeat
2: if（$m \neq \emptyset$） then
3: if（state=$PR_i^{\{v,m,n\}}$）
4: if（$i = v \ mod \ N$） then
5: send pre-prepare certificate to other
6: state← $P_i^{\{v,m,n\}}$
7: else
8: state← $P_i^{\{v,m,n\}}$
9: if（state=$P_i^{\{v,m,n\}}$） then
10: send prepare certificate to other
11: if Num(m)>$\frac{2}{3}N$ then
12: state← $C_i^{\{v,m,n\}}$
13: if（state=$C_i^{\{v,m,n\}}$） then
14: send commit certificate to other then
15: if Num(m)> $\frac{2}{3}N$ then
16: verify certificate m
17: return true
18: until v发生改变
19: return false

其中，当一条证书产生首先判断该节点所处的状态，并会执行当前状态的方法，同时进入下一个状态，其中 Pre-Prepare 状态中需要对主节点判断，只有是主节点才可以执行 Pre-Prepare 的方法函数。当一条证书经过了三个状态的条件判断则认为该证书内容在节点间达成共识。

3.3.2 区块生成协议

在区块链中，交易以区块的形式提交并永久记录到区块链中。本文采用以太坊交易校验协议，自身节点先校验交易的有效性后，将交易批量写入区块中，并发送区块，该区块中交易经过全网其他节点校验通过，并添加到区块链中则认为交易被执行。当交易到来时，由主节点将交易写入到区块中并广播该区块，当全网节点对该区块达成一致后，尝试将该区块添加到区块链中，整个过程是异步的，节点间通过区块号和区块记录的父区块哈希值保证区块的有序性。当交易列表为空时，节点会监听区块链中最优区块的时间戳与系统时间间隔，当时间超过t会生成一个空的区块并添加到区块链中，这里考虑到在信息传输过程会造成网络延迟，区块从生成到达成共识并添加到区块链的最长时间为Δt，其中t需要满足$t > \Delta t$，这样可以保证在生成空区块时，所有交易已经在全网达成一致。当区块链中添加空区块后，主节点停止生成区块，并等待交易到来时再重新触发生成区块。

区块生成部分的流程如图 3-1 所示，系统第一次启动的时候会对同步情况进行监听，当发现系统不接受同步或者同步完成才开始生成区块。当系统完成启动阶段后会循环监听执行状态改变事件，当发现当前正在生成区块为空，或者当前区块正要添加到区块链中的区块号不大于区块链中最优块的区块号再或者发现有新提交的交易，满足这三个条件，系统进行重新生成区块的过程。在生成区块的方法中，首先会判断距离当前区块链 BestBlock 的时间是否超过了t，不满足结束方法并等待下一次监听进入该方法。当满足区块生成条件后会结束当前任务，这个过程会用到 JAVA 锁机制，防止多线程之间调用出现不一致的情况，接着创建新的添加生成区块的任务并执行该任务，等待新区块被校验并添加到区块链中。当成功添加到区块链中会返回监听器添加成功的信息，并触发新的线程将当前正在生成区块置为空，从而监听器会被触发进行新的区块生成的过程，完成区块链的添加过程。

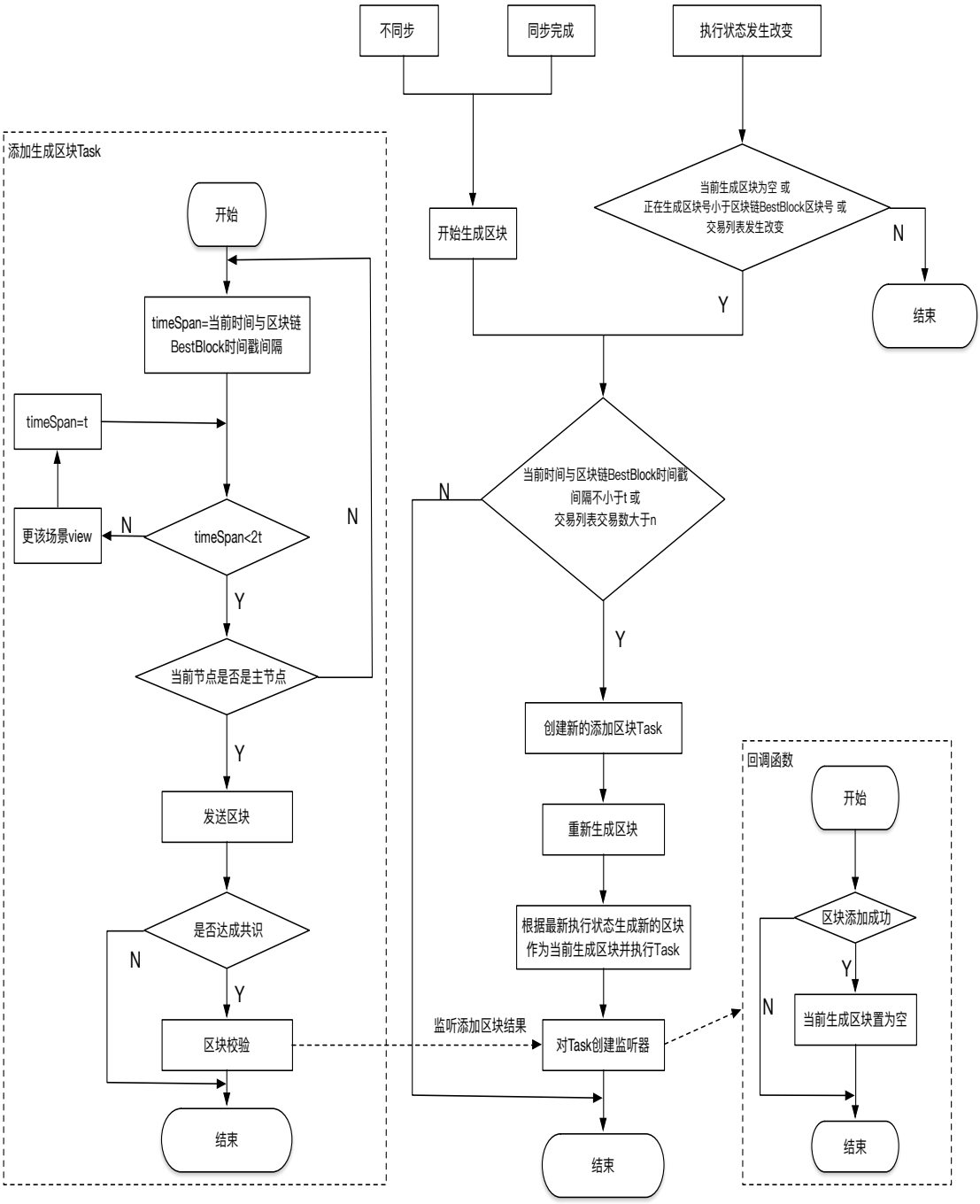


图 3-1 生成区块流程图

3.3.3 一致性协议

由拜占庭算法可知，一个系统可以容忍不超过全部节点的三分之一发生拜占庭错误，因此这里我们假设在区块链系统中共有 $3f+1$ 个节点，其中f是该系统中出现允许最大错误节点数，一个 Quorum 至少包含 $2f+1$ 个节点。在一个可以容忍拜占庭错误的系统中f的最小取值为 1，因此系统至少由 4 个节点组成。图 3-2 是由 4 个节点组成的系统进行一次正常请求的执行过程。由图 3-2 可知，一个信息达成共识并执行需要经过三阶段执行协商后达成一致执行，三阶段协商过程如下：

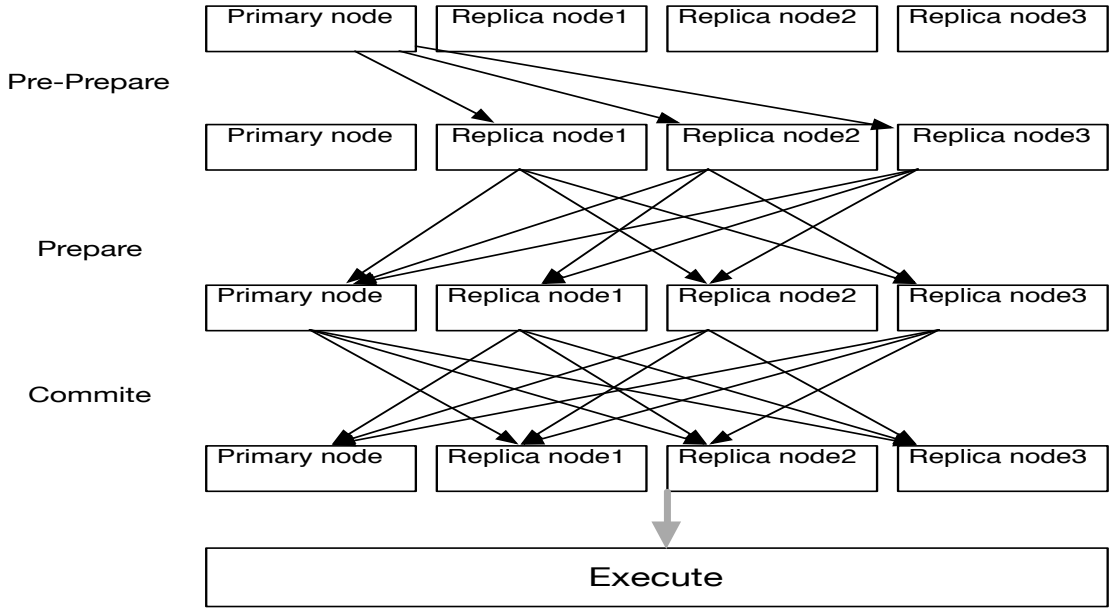


图 3-2 正常请求执行过程

（1）当主节点中满足生成区块条件时生成一个新区块时，主节点生成 Pre-Prepare 证书，将 Pre-Prepare 证书发送给其他节点后，主节点进入 Prepare 状态。

（2）replica 收到 Pre-Prepare 证书时就接收到了新生成区块的信息，同时该节点进入到 Prepare 状态，当该节点发现消息来自于主节点时并且第一次接收时，会将 Prepare 证书发送给其他节点，该节点将证书信息记录，当发现该信息通过了 $2f$ 个节点反馈并同意某一条证书时，即该区块信息通过了一个 Quorum 的同意，那么对于这条证书该节点进入 Commit 的状态，并向其他节点发送 Commit 消息。

（3）replica 会接收到来自其他节点的 Commit 的证书，当发现该信息得到了 $2f+1$（包括自身）个节点同意，则认为该 Block 信息在系统中达成共识，并尝试将该区块添加到区块链中。

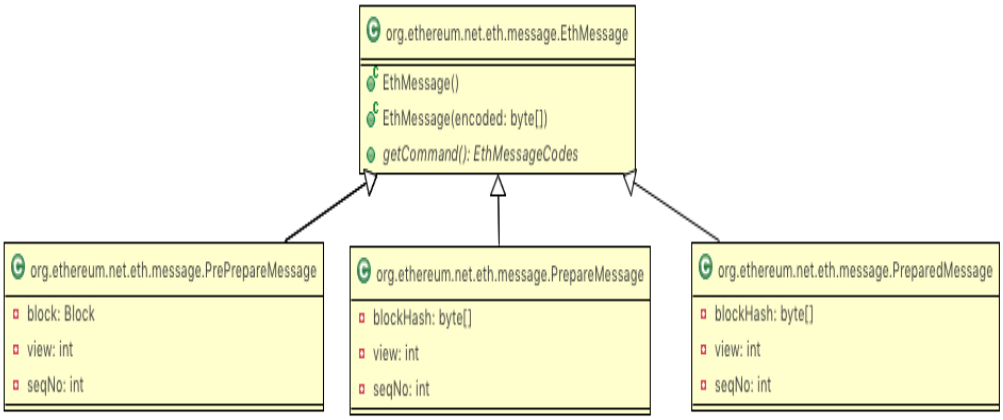


图 3-3 证书消息

其中证书消息如图 3-3 所示，Pre-PrepareMessage 中信息包含完整区块， 而 PrepareMessage 和 CommitMessage 则只包含区块的哈希值，这样做是为了减少通讯开销，因为当一个区块进入了 Prepare 状态后，本地一定会保存了一个完整区块，因此在 Prepare、Commit 证书传输时，只需要传输区块哈希值，因为区块哈希值可以唯一标识一个区块，这样可以减少传输证书中的通讯开销。view 代表这条证书所处的视图编号。seqNo 代表证书编号，本文以区块号作为证书编号。当接收到证书会对证书进行校验，这里主要对证书中的区块的时间戳、区块号、parenthash 等校验，对于不合法的证书从证书列表中删除，并添加到不合法证书列表中，并且该节点拒绝接收这类证书。

通过三阶段提交方式，实现了发送的区块信息达成一致，以图 3-2 为例考虑到当一个 replica 节点发生拜占庭错误，由于另外两个 replica 是乖节点，因此仍可以满足 $2f$ +1 个节点通过验证，乖节点之间可以保证区块的一致性；当主节点发生拜占庭错误时，通过在 replica 节点中重新选出主节点，并由主节点生成区块并发送消息， 将该区块添加到区块链，正确的区块会成功添加到区块链，并触发下一个区块的生成，循环执行该过程。

本文将协商过程与区块的校验过程分开执行，这样设计的目的是加快系统达成一致的效率。由于主节点会发送区块内容，在区块校验过程中最耗时的是交易的校验，如果区块中包含n条交易信息，一条交易（TX）信息的校验时间函数为$\lambda(TX)$，则该区块交易校验则需要耗时$\sum_{i=1}^{n}\lambda(TX_i)$，采用协商校验一起执行的方法，则交易校验就至少需要耗时$(2f+1)\sum_{i=1}^{n}\lambda(TX_i)$，在这里还没有考虑到某些错误节点会进行错误攻击，发送大量交易的区块的情况，当区块中交易较多，区块校验将是很大一笔时间开销。而采用协商和校验分离的机制，在达成一致阶段不需要进行校验，从而缩短了了达成一致性的时间，因此本文采用该策略来缩短达成共识的时间。

使用 PBFT 共识机制结合以太坊校验机制，使区块链系统形成由主节点广播，备份节点负责投票的机制，保证区块链中即使出现坏节点，仍能保证区块链系统去中心化可信的运行。

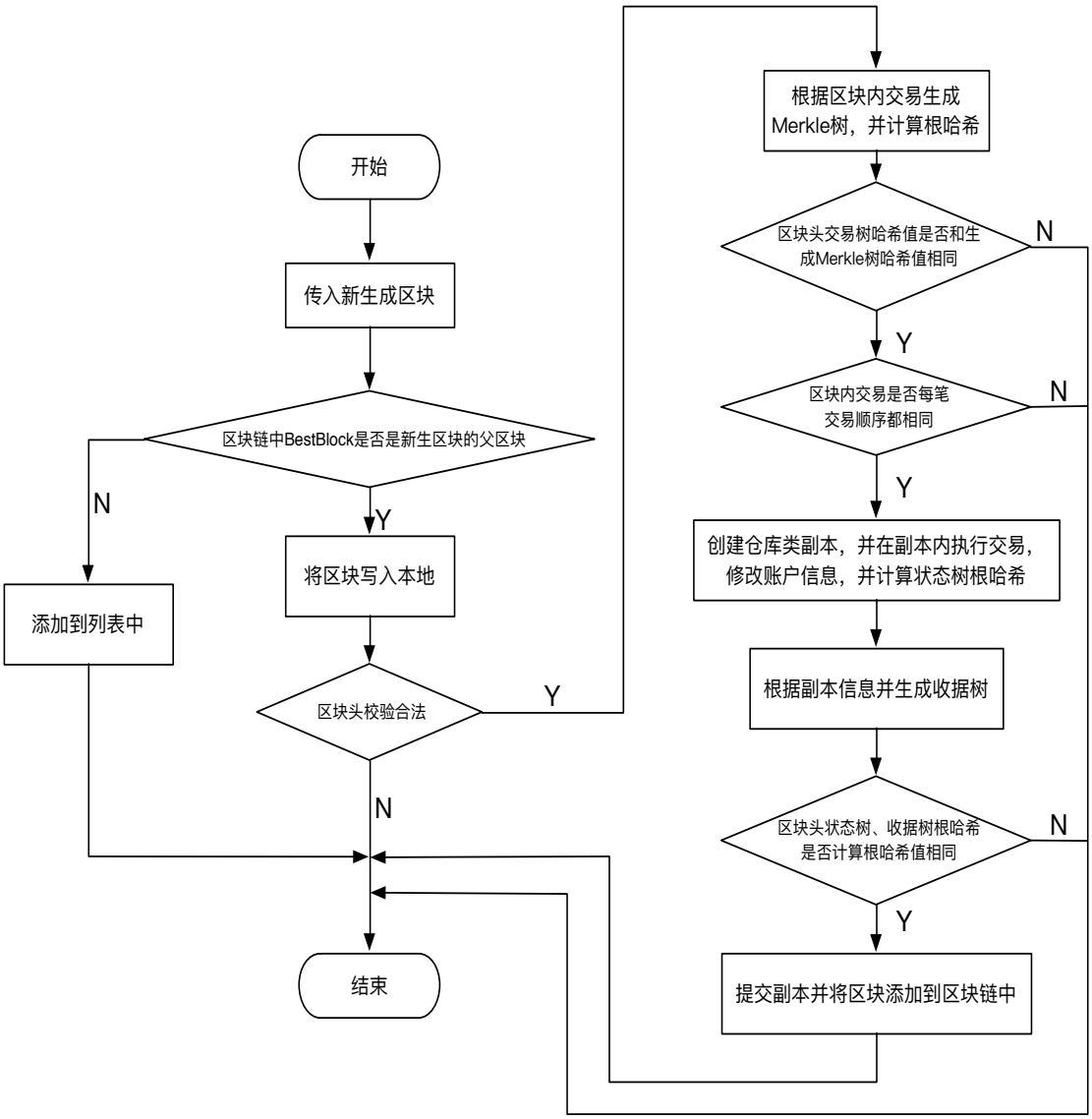


图 3-4 区块校验流程图

3.3.4 区块校验

区块的校验过程主要针对区块头信息进行校验，本文结合以太坊中区块结构进行区块校验，校验流程图如图 3-4 所示。首先要对 ParentHash 校验，判断新生成区块是否指向本节点中区块链的 BestBlock，当该区块没有指向当前区块链 BestBlock，会将该区块首先加到列表中，并由另外一个线程循环判断该列表中是否有指向当前区块链 BestBlock 的区块，有的话执行区块校验流程，并尝试添加到区块链中。当区块指向当前区块链的 BestBlock 时首先会对区块中交易进行校验，区块中的交易按交易执行的时间顺序排序，根据这个顺序可以生成 Merkle 树，并对 Merkle 树的 rootHash 与新区块的交易树根哈希进行校验，校验通过会对每笔交易进行校验，然后生成 Repository 对象的副本，其中 Repository 包含了该节点的本地库数据，接下来操作在副本中进行，因为对状态树校验需要对 Repository 状态进行修改，在副本中操作可以进行回滚，当对区块中交易数、状态树和收据树 roothash 校验都通过，再提交副本，并将该区块添加到区块链中。

交易校验是整个区块链校验的核心内容，这里我们定义状态交易函数为γ，交易为T，交易的执行会更改本地状态数据库，这里定义数据库状态为S，则有：

$$S' = \gamma(S, T) \tag{3-3}$$

其中S'为更改后的状态，一笔交易有效判定需要满足以下条件：

$$\mathrm{s(T)} \neq \emptyset \wedge \mathrm{S[s(T)]} \neq \emptyset \wedge T_n = S[s(T)]_{nonce} \wedge v_0 \leq S[s(T)]_{balance} \tag{3-4}$$

其中s(T)是一笔交易的发起者，T_n、v_0分别为这笔交易中的 nonce 和交易金额， nonce 是记录每个账户的交易次数，用来保证交易的有序性，并记录在该账户信息中。当一笔交易失败的时候，本地仓库要实现回滚的功能，这里我们定义状态S_0为回滚状态，则有：

$$S_0 = S \quad except:\ S_0[s(T)]_{balance} = S[s(T)]_{balance} - c,\ S_0[s(T)]_{nonce} = S[s(T)]_{nonce} + 1 \tag{3-5}$$

其中c为执行这笔交易的花费。接下来会执行该区块添加到区块链中的操作，并且成功添加后会将正在生成区块状态置为空，这两个状态的改变都将会触发下一次的区块生成过程。

3.3.5 检查点协议

该协议主要目的是为了维护节点状态规模，主要是对证书信息的回收，降低节点的内存开销。M Castro[42]定义的 PBFT 机制中定义检查点协议，是通过节点间定时协商，对失效的记录达成共识后一并删除。因此清除过期证书也是一个全网达成一致的过程，这势必又要进行 3 阶段提交过程，造成通信浪费。本文根据区块链中最优区块的时间戳进行证书清除。区块链是以链表的形式按照区块的生成时间相连而成，当一个区块添加到区块链中，则说明该区块时间戳之前的证书都已经被校验过，即该节点中这些证书相关的状态已经广播完毕，并可以进行清除，而证书的信息则以区块的形式永远保存在该节点中。因此对添加区块事件进行监听，每当有区块添加到区块链中，将该节点中该区块时间戳之前的证书进行清除。整个过程不需要节点间相互通信，同时也可以保证证书及时的清理，使检查点协议中的通信量变为零。

以太坊中包含了节点间同步区块的解决方案，当一个节点发现自身维护的区块链与其可信任的节点维护的区块链的最优区块的区块号相差一定大小时，该节点会向其信任的节点（一般为开发者维护的节点）索要区块并将区块添加到区块链中。而在联盟链中并不存在完全可信的节点，因此该方案并不可行。本文提出当某节点区块链状态与其他节点不一致时，向该视图中的 2f+1 个节点请求该区块链需要添加的区块的区块哈希，区块哈希是可以唯一标识区块的 256 位字节数组，当有不少于f+1 个节点返回的区块哈希一致，则认为该区块哈希对应的区块在全网达成共识。该节点首先会在 Pre-Prepare 证书中查找是否存在该区块哈希的证书，不存在会向其中一个节点索要该区块哈希对应的区块，并将该区块添加到区块链中，实现同步。该过程中数据传输量对于网络开销可以忽略不计。

3.3.6 视图切换协议

该机制是针对主节点发生错误，更换主节点的协议。M Castro[42]定义的 PBFT 机制详细地阐述了视图切换的过程，因为视图的切换不当会使得整个系统没有主节点，因此这个过程不可避免的也需要进行节点间的相互通信。本文采用超时机制触发视图切换。算法如下：

```
Algorithm：ViewChange（）
Input：该节点维护的区块链和交易列表
Output：是否进行视图切换
1：time←CurrentTime
2：Repeat
3：    if  交易列表不为空  then
4：       if  BestBlock 的交易信息为空  then
5：          if  CurrentTime-time 大于 t  then
6：             ChangeView
7：       else
8：          time←BestBlock 的时间戳
9：          if  CurrentTime-time 大于 t  then
10：            ChangeView
11：    else
12：       if  BestBlock 的交易信息为空  then
13：          if  下一个区块交易信息为空
14：             ViewChange
15：          else
16：             Thread.Sleep(t)
17：             time←CurrentTime
18：       else
19：          time←BestBlock 的时间戳
20：          if  CurrentTime-time 大于 t  then
21：             ChangeView
22：Until ViewChange
```

当节点中交易列表不为空时，因为一个区块从生成到添加到区块链的时间小于t，当某一个节点中区块链最优区块时间戳与当前时间间隔大于t，则认为主节点出现异常，此时触发视图切换。因为t是系统最长达成一致的时间，因此在进行视图切换时认为区块链已经完成对正确区块的一致性校验并添加到区块链的过程。当节点的交易列表为空时，会判断当前区块链最优区块是否为空。当区块不为空时，如果t时间内没有区块添加到区块中，则认为主节点发生异常，并进行视图切换。视图切换过程中将证书列表进行清除，并由新的主节点完成提交交

易的操作，并继续维持系统的稳定。通过上述算法，当主节点发生错误时，系统会在t时间内完成视图的切换。该算法需要更长的超时时间保证在主节点发生错误时，全网已经达成一致。这对于一般的分布式系统可能会造成服务中断等问题，而对于联盟链环境下以太坊来说，服务并不会停止。其他正确节点仍然可以将交易存入交易列表中，并由其他节点各自维护的本地数据提供服务。整个视图切换过程根据区块链中最优区块时间戳执行，在区块可容忍的延迟的范围，完成主节点的切换，不需要节点间相互通信，减少了通信消耗。

3.4 系统实现

本文针对以太坊中 PoW 共识机制不适用于联盟链场景，提出使用 PBFT 共识机制进行解决。PBFT 机制中，以太坊节点不需要进行挖矿，产生区块的过程采用时间和交易两种触发方式，因此节点不必要进行大量的运算，并通过算力竞争来达到共识，解决了 PoW 共识机制所造成的资源浪费的问题。在 PoW 共识机制中可以容忍全网 50%的算力发生错误，而 PBFT 共识机制中只能容忍不足 1/3 的节点发生错误，如果节点的算力相同，从容错率看 PBFT 机制没有 PoW 机制高，但 PBFT 机制解决了以太坊中产生支链的问题，由于 PoW 需要计算随机数，即使通过难度系数的调整也很难让出块时间稳定，当出块时间过快通信延迟就显得影响很大，在联盟链中往往由于节点少，因为通信延迟造成分支情况，而本文采用的共识机制考虑到通信延迟，并且有较为稳定的出块时间，更加利于系统的维护。

本文对于 PBFT 共识机制进行改进，采用了协商和校验分开执行的方式，减少对区块的校验次数，提高系统达成共识的效率。同时设计了以太坊的出块规则，并以超时机制对证书回收和视图切换进行控制，根据区块链中最优块的出块时间控制证书回收和视图切换，很大程度减少了系统的通信开销，同时各节点数据的一致性得到了保证。

3.5 实验数据

本文实验硬件采用 4 核 8 线程的 Intel(R) Core(TM) i7-4870HQ 处理器，16G 内存的硬件平台。虚拟机 8 台，1GB 内存，CPU 共享主机的一个处理器核心，网络连接 100Mbps LAN。

首先对基于改进 PBFT 以太坊共识机制的功能进行试验，本文采用 8 台虚拟机分别搭建了基于 PoW 共识机制和基于改进 PBFT 共识机制的以太坊联盟链，这里认为 8 台机器的算力是相同的，并使节点发生拜占庭错误进行实验，其实验结果如表 3-1 所示。从实验结果可以看到，本文设计的改进 PBFT 以太坊共识算法可以实现系统的去中心化运行，在容忍系统节点错误率的情况中，PoW 共识机制最多可以容忍 3 个节点发生任何形式的错误，而在改进 PBFT 共识机制中，

当主节点发生错误时，可以完成视图切换以及主节点的重新选择，但是该系统最多可以容忍 2 个节点发生错误。该实验证明了该机制可以满足本文所提的要求，但在容错率中，不如 PoW 共识机制的容错率高。

表 3-1 联盟链节点一致性分析

名称	系统是否正常运行			
坏节点个数	1		2	3
节点类型	主节点	非主节点		
PoW	是		是	是
改进 PBFT	是	是	是	否

接下来对 PoW 共识机制和改进 PBFT 共识机制的算力使用情况做了相关实验和分析，本文首先进行单节点测试，在相同的机器分别运行基于 PoW 共识机制和改进 PBFT 共识机制的以太坊，其中基于 PoW 共识机制采用满线程运行。由于 PoW 共识机制中 Ethash 需要提前生成 1G 的内存空间，为了只对比稳定运行的 CPU 使用率，因此提前生成好内存哈希值。本文实验 CPU 是 8 线程因此以太坊设置为 8 线程挖矿方式，CPU 使用率结果如图 3-5 所示。从结果可以看出基于 PoW 共识机制的 CPU 使用率接近 100%，而基于改进 PBFT（APBFT）共识机制的 CPU 稳定使用率仅为 16%左右，因此改进 PBFT 共识机制很大程度上减少了算力，解决了 PoW 共识机制算力浪费的问题。

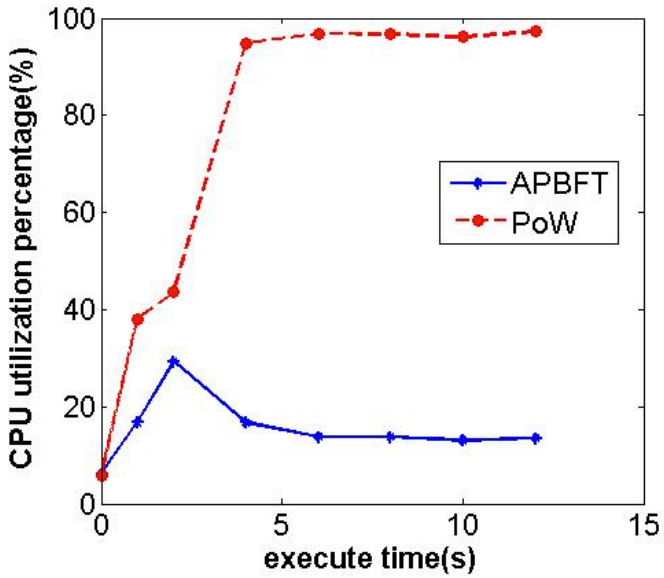


图 3-5 CPU 使用率

最后对当节点出现错误时改进 PBFT 机制可以减少数据的传输进行实验验证。本文分别采用 PBFT 机制和改进 PBFT 机制进行测试， 其中在一次完整的删除证书过程中，证书的传递网络开销测试结果如图 3-6 所示。其中图中横坐标是一次检查点需要清除区块的证书的个数，图中用 block count 表示，纵坐标代表传输消耗，即每次检查点执行需要传输区块哈希的个数。从图中可以看到，采

用 PBFT 共识机制的以太坊证书传递开销会随着证书中包含的区块个数成正比增加，这是因为每一个清理请求都需要传递区块的哈希值，因此区块越多，传递的哈希值越多，证书大小就越大，因此造成一定网络开销，而采用改进 PBFT 共识机制，没有证书传递，因此为零。图 3-7 中显示了当主节点发生错误的时候，在一个完整的视图切换过程中的网络开销。其中图中横坐标代表系统内节点个数，纵坐标代表每次视图切换达成共识 Quorum 的数量，视图切换过程中，证书大小相同，因此通讯开销和 Quorum 有关。从图中可以看出 PBFT 共识机制中，会随着最大容错节点个数增多，开销增大，而改进 PBFT 共识机制中，该部分开销为零。由此也可以说明该改进 PBFT 共识机制在网络开销上较 PBFT 共识机制有一定的提高。

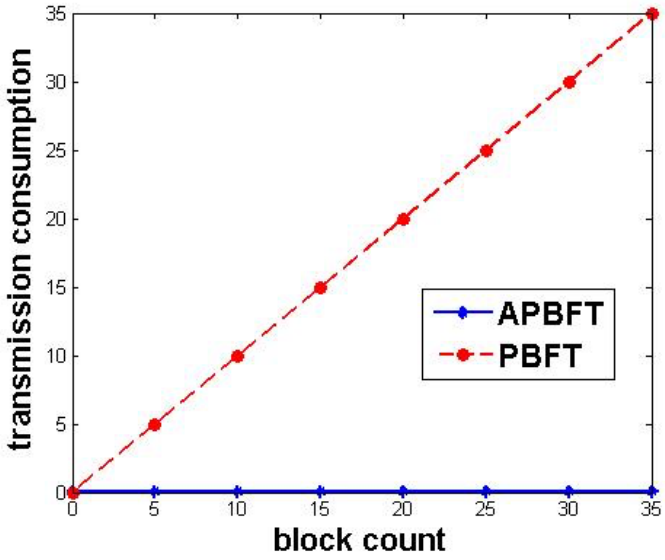


图 3-6 证书传输开销

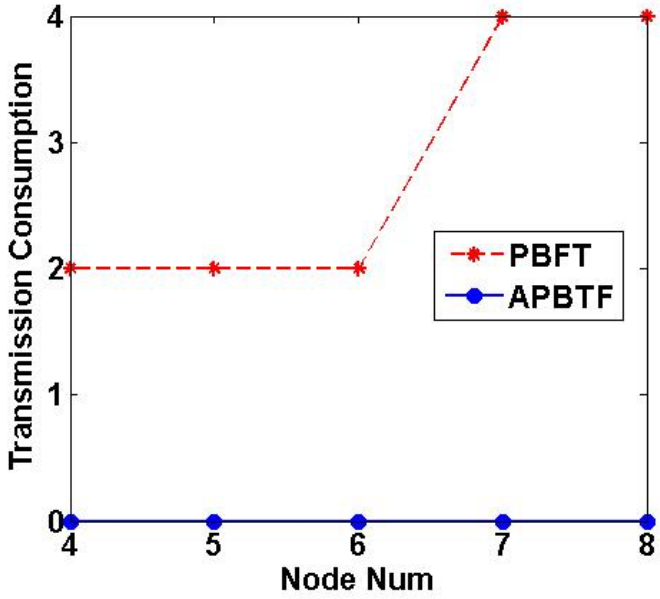


图 3-7 视图切换传输消耗

3.6 本章小结

本章首先分析了现阶段以太坊应用在联盟链中所遇到的问题，问题的核心是基于工作量证明的共识机制不适用于联盟链中，因此针对该问题提出了一种基于改进 PBFT 的共识机制的解决办法，将 PBFT 机制与以太坊区块结构结合，并详细的阐述了该机制的执行过程。首先针对改进 PBFT 机制进行了描述，并说明该机制用于联盟链以太坊的正确性。接着针对以太坊的环境中如何实现改进 PBFT 共识机制进行详细的阐述，将一个区块从产生、传输和校验流程进行原理和实现上的说明。最后针对于以太坊区块结构，提出了对证书回收机制和视图切换机制的改进，采用超时机制实现检查点协议和视图切换协议，在一定程度上减少了网络的数据传输开销。

第 4 章 Merkle Patricia 树中账户存储优化

本章结合联盟链中，账户状态已知并有账户的交易历史信息的前提下，结合 Merkle Patricia 树结构特点，提出一笔交易执行过程中同时对两个账户进行修改，并基于该算法提出采用 K-means 聚类算法将账户聚类，并针对于 K-means 算法的初始值选择提出了基于关联度选择方案。

4.1 以太坊账户存储结构

以太坊中，为了方便数据的校验，大多数数据都以 Merkle 树的形式存储。Merkle 树从一般意义上讲，是哈希大量聚集数据的一种方式。Merkle 树具备树的一般结构特征如图 4-1 所示，Merkle 树可以是二叉树也可以是多叉树，叶子节点 Value 存放的是数据，非叶子节点的 Value 是该节点下的所有节点组合的哈希值。如图 4-1 中 $H_{1,2}$ 的值需要根据叶子节点 H_1，H_2 的 Value 值进行哈希计算。Merkle 树这种数据结构为提供了一种校验机制，我们称之为 Merkle 证明（Merkle Proof），即当验证了某一个节点哈希的正确性，该节点下的所有分支都是正确的。Merkle 证明可以通过验证少量数据验证大量聚集数据的正确性。区块链正是利用这项技术实现对一个区块中交易的快速校验的过程。

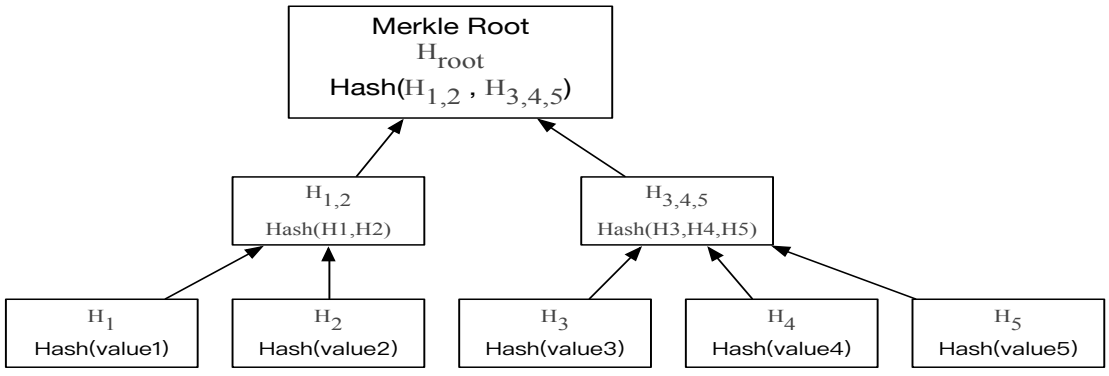


图 4-1 Merkle 树结构

Merkle 树是区块链实现快速校验的重要的数据结构，Merkle 树对于验证“清单”格式的信息而言，是非常好的数据结构，本质上来讲，它就是一系列前后相连的数据块。而对于交易树来说也是可行的，因为一旦树已经建立，花多少时间来编辑这颗树并不重要，对于交易树只需要生成一次并保存到区块中即可。而状态信息则是以键值对的形式存储，以账户的地址作为键值，而该账户下的账户声明、余额、随机数等数据则作为值存入。每当执行交易或者操作账户时，都需要对状态树进行查询修改，频繁的增删改查的操作，需要状态树具有快速查询和快速计算根哈希的能力。针对这种情况，以太坊提出 MPT（Merkle Patricia Tree）数据结构，将 Merkle 树和字典树结合，提高了查找效率并且能够快速计算根哈希值。该树分为 4 种类型节点，分别是空节点、叶子节点、扩展节点与分支节点。

空节点是空的，当执行 insertNode 时，直接插入，此时插入的数据只可能是键值对类型的。叶子节点往往是空节点插入数据后的节点，类型也是键值对类型的。扩展节点也是键值对类型的，该节点值指向其他节点的 hash 值。分支节点是 17 位的数据类型。为了更好的解释 MPT 数据结构，将用一个向空树插入账户信息例子来阐述下 MPT 数据结构，MPT 插入过程是动态调整树的过程，如图 4-1 所示：

（1）插入数据 node1［key＝123，value＝abc］此时生成一个空节点，并将 node1 插入并记为 N1，返回 N1 节点的 hash 值作为 roothash。

（2）插入数据 node2［key＝456，value＝abc］此时树结构生成新的一层 N2，N2 是这棵树的叶子节点也是分支节点，此时 node1、node2 的 key 值第一位作为 N1 到 N2 层的索引，比如 node1 第一位是 1，对应的存储到 N2 中的第一个叶子节点中记作 N2［#1］，因此 node1 的数据变为［key＝23，value＝abc］存放在 N2［#1］中，同理 node2 变为［key＝56，value＝abc］存在 N2［#4］中，这时对 N2 中叶子节点的值进行哈希并将哈希值存放到 N2 的值中，并作为 roothash，而 N2 的地址则作为 N1 的值存入。

（3）插入 node3［key＝156，value＝abc］此时通过索引尝试将 node3 变为［key＝45，value＝abc］并尝试放入 N2［#1］，但是由于 N2［#1］非空，因此树生成新的一层 N3，类似于执行第二步将 node1 变为［key＝3，value＝abc］存到 N3［#2］，node1 变为［key＝6，value＝abc］存到 N3［#5］，此时 N3 的地址存储在 N2［#1］的值中，N3 的哈希存储在 N3 的值中。这时计算 N2 的哈希对于 N2［#1］取得是 N3 的哈希值，而不是 N2［#1］值中的内容。N2 的哈希作为 roothash 存储在 N2 的值中。

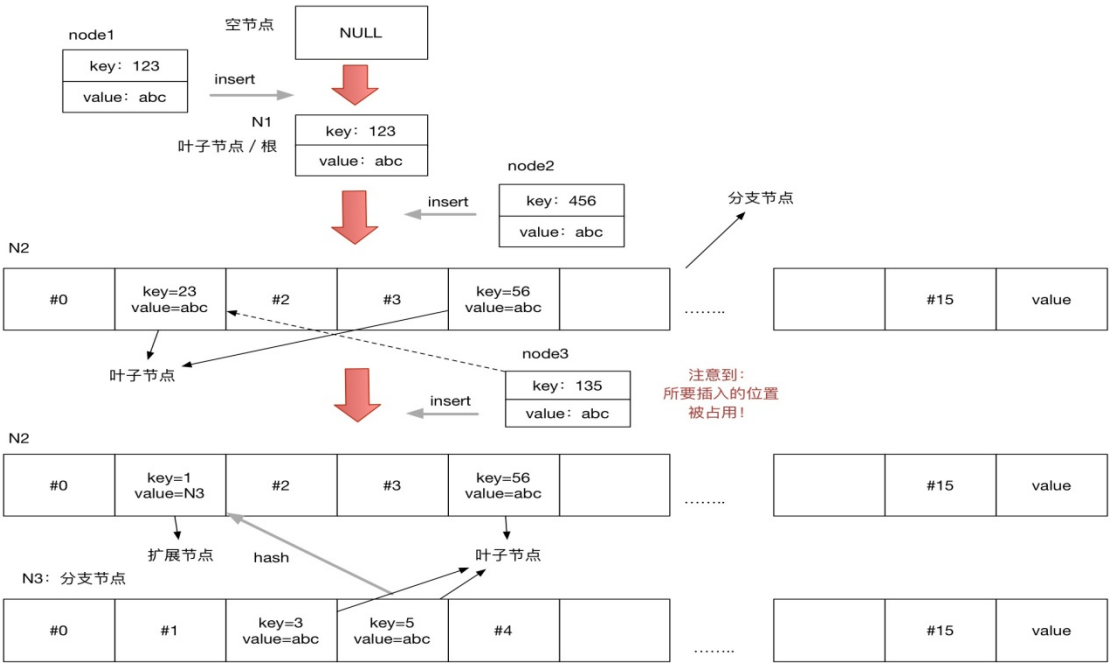


图 4-2 Merkle Patricia 树数据插入过程

修改的操作与插入类似，而查询过程则是与字典树相同，因此不再赘述。通过这样的数据结构，可以在一次插入（修改）操作后快速生成 RootHash，与此同时，树的深度是有限制的，不然攻击者可以通过操纵树的深度，执行拒绝服务攻击（DOS Attack），使得更新变得极其缓慢。

以太坊在最初的 Merkle Patricia Tree 的设计过程中，因为考虑到以太坊需要在公链中运行，因此设计了 16^{16} 的存储空间进行账户信息存储，秉持对账户不可见性采用了对账户地址（公钥）采取 RLP 编码，编码成长度相等的字节数组，编码后账户在树中位置是随机的，一方面树的深度较深在计算根哈希的时候，需要进行多层的哈希，而哈希过程也是时间开销最大的。另一方面，大量的分支节点的空间浪费，对于系统来说也是一笔开销。在联盟链中，由于账户信息分别由各联盟体掌握，因此不需要对账户进行随机编码，因此可以根据交易的先验数据与对 Merkle Patricia 树进行存储设计，提高 Merkle Patricia 树的效率。本文结合 Merkle Patricia 树的数据结构以及交易模型，提出了对交易账户同时进行数据修改，通过对先验数据进行聚类实现优化。

4.2 Merkle Patricia 树账户存储优化

在现实生活中，对于一笔交易来说，是发生于两个账户中。同时，作为熟人之间的相互转账要比与陌生人之间的多，因此基于以上两点，将账户在 Merkle Patricia 树存储位置修改，并对以太坊账户修改过程进行改进。以太坊中每次的账户操作都是对于一笔交易分解成两个账户分别加减钱。一个账户的经常转账对象往往是经常有交易往来的，本文中称这样的账户为关联账户，那么如果两个账户所在 Merkle Patricia 树的叶子结点在一个父节点的话，那么一笔交易只需要定位一次父节点的位置，并对该节点下的叶子结点进行修改即可，而改变一个数据和改变多个数据对于每一层的哈希运算的开销是相同的，因此这个过程在计算根哈希运算中会减少一次父节点到根结点的哈希过程，对于处于同一父节点下的的两个账户来说，一次交易的计算根哈希开销减少了一半。以图 4-3 为例，账户 A1 和 A2 为关联账户，A1、A2 经常发生转帐交易，那么一笔交易发生时只需要根据 A1、A2 地址公共前缀找到 N1 后，对 N1 的叶子结点进行 Update 操作，那么找到 A1、A2 共用进行 3 次查找就可以完成，哈希过程也只需要进行两次，而不采用关联账户认为 A2 在 A2*的位置，那么找到 A1、A2*要用 4 次，哈希过程需要进行 4 次。假设关联账户账户地址为 Add1、Add2，对应更新内容为 V1、V2，本文对账户进行更新操作改进的伪代码如下：

Function(): InsertAccount（）
Input:Add1,Add2,V1,V2,rootNode Output:rootHash 1: m= Add1、Add2 公共前缀长度 2:If m≠ 0 3: parentNode= get(CopyOfRange(Add1,0, m))) 4: parentHash=insert(parentNode, CopyOfRange(Add1, m ,Add1.length), CopyOfRange(Add1,m,Add2.length),V1,V2) 5: Return insert(rootNode,parentNode,parentHash) 6:else 7: rootHash=insert(rootNode,Add1,V1) then 8:Return insert(rootNode,Add2,V2)

其中 insert 函数是递归函数，输入参数是当前所在结点的 key，要插入节点的 key 和 value，返回值是插入节点后当前结点的哈希值，matchLength 是求两个账户公共前缀的长度，该函数输入的是两个账户地址。该算法首先根据两个账户的最长公共前缀找到公共路径，并索引到该节点上，以该节点作为 Merkle Patricia 树的子树进行两个账户数据的修改。当修改完成计算根的哈希过程中，首先计算子树的根哈希值，即根据最长公共前缀索引到的节点的哈希值，然后计算改变该节点后的根哈希值。如果公共路径长度为零，则将两次账户信息修改分开执行，返回根哈希值。

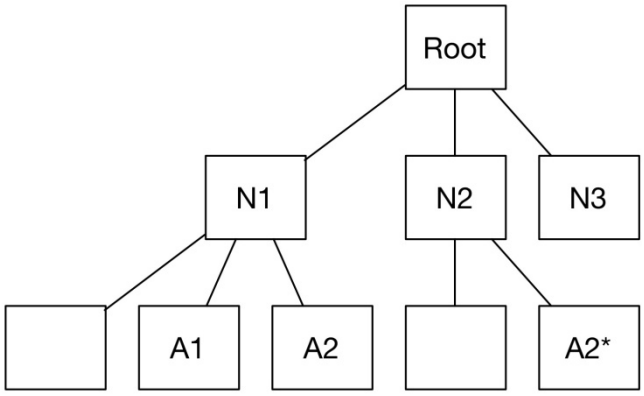


图 4-3 关联账户 Merkle Patricia 树

4.3 关联账户筛选算法

4.3.1 模型建立

针对上一小节中提到的关联账户，本小节给出关联账户的划分算法。本文采用图论中的相关概念描述该模型。所谓图（无向图）[45]是指给了一个集合$V(G)$以及$V(G)$中不同元素的无序对的集合$E(G)$，如图 4-4 所示，$V(G)$、$E(G)$也分别称为图G的点和边。本文认为账户就是一个图中的点，而边则认为是交易，而边的权重则是交易笔数，并以此建立模型。

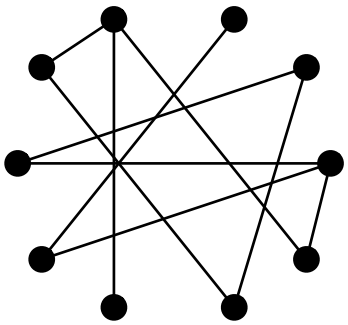


图 4-4 无向图

在一个交易网中，按上述方法构成的无向图往往不是封闭的图，如图 4-5 左图表示的是一个 5 个账户之间的交易关系图。本文将每一条边作为一个属性，进行关联矩阵的构建，构建结果如图 4-5 右图所示， 因为左图所示共有 4 条边，因此矩阵共有 4 列，由于是无向图，如果边与点相连，则该点对应该边的值即为边的权重值。通过如上变换，将交易账户代表一个 n 维空间中的点，其中 n 为边数。

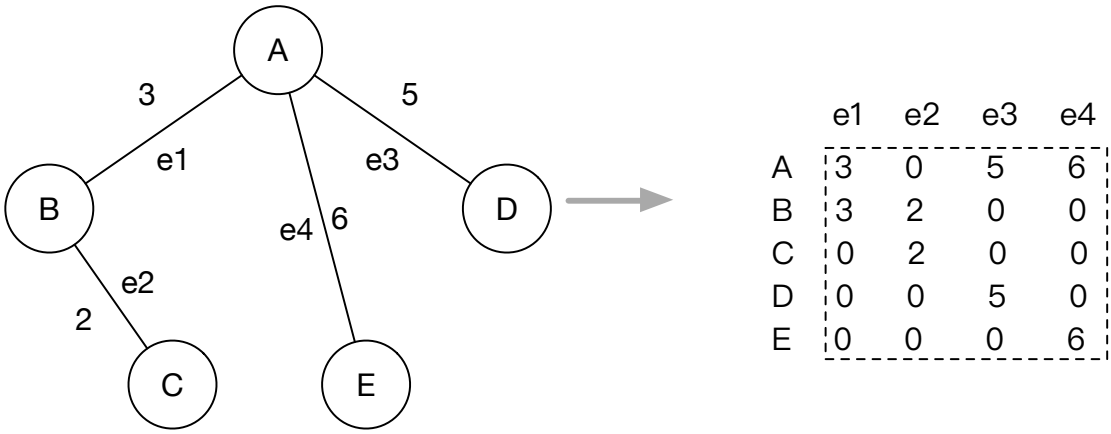


图 4-5 图与关联矩阵的变换

4.3.2 基于 K-means 账户关联筛选算法

MacQueen[46]首次提出了 K-means 聚类算法，并在许多实践应用中取得了很好的效果。K-means 算法是输出数据集中k个聚类的方法[47][48]。数据集由n个对象组成，初始化时，根据输入参数k从数据对象$\{l_1, l_2, \dots, l_n\}$中随机找出k个$\{w_1, w_2, \dots, w_k\}$，其中$l_i = w_j, j \in \{1,2, \dots, k\}, i \in \{1,2, \dots, n\}$作为簇的初始值或者中心，对剩余的对象根据其与各个簇均值的距离，将该对象指派到最相似的簇中，然后计算簇的新的均值，直到最小误差$E = \sum_{j=1}^{k} \sum_{l_i \in C_j} |l_i - w_j|^2$收敛，其中$C_j$表示第$j$个聚类。K-means 算法如下：

Algorithm: K-means()
输入：包含n个对象的数据集合簇的数目k及对象 输出：k个簇的集合 1:初始化k个簇中心$\{w_1, w_2, \dots, w_k\}$，其中$l_i = w_j, j \in \{1,2, \dots, k\}, i \in \{1,2, \dots, n\}$ 2: 使每一个聚类C_j与簇中心w_j相对应 3: repeat 4: for 每一个输入向量l_i，其中$i \in \{1,2, \dots, n\}$ do 5: 将l_i分配给最近的簇中心w_j^*所属的聚类C_j^*，即$\left\|l_i - w_j^*\right\| \le \left\|l_i - w_j\right\|$，$j \in \{1,2, \dots, k\}$ 6: for 每一个聚类C_j，其中$j \in \{1,2, \dots, k\}$ do 7: 将簇中心更新为当前C_j中所有样本的中心点，即$w_j = \sum_{l_i \in C_j} l_i / \lvert C_j \rvert$ 8: 计算最小误差E 9: Until E不再明显的改变或者聚类成员不再发生变化

K-means 聚类算法虽然可以较好的实现分类，但 K-means 算法需要输入k值和初始聚类中心，在本文中对于k值的选取是根据 Merkle Patricia 树结构和账户数量相关，并可以根据人群分类等方式选取，因此对于初始聚类中心的筛选就格外重要。对于初始聚类中心的选择现在主要采用最大最小距离法，该算法首先选取相邻距离最远的样本作为初始的两个聚点，其他点根据递推选取，假设已经生成$n(n < k-1)$个聚点，那么第$n+1$个聚点则是：

$$min\{d(x_{i_{n+1}}, x_{i_r})\} = \max\{\min[d(x_j, x_{i_r})]\}(r = 1,2, \dots n), (j \neq i_1, i_2, \dots, i_r) \quad (4\text{-}1)$$

其中x代表聚点，该算法中认为相邻最远的的点一定是不同类的，而本文结合图的概念提出了基于关联权重的最大最小选择方案。根据交易网构成的无向图中，点的度表示与该账户交易账户的个数，边的权重则表示两个账户交易次数。这里我们定义账户关联度为：

$$C_i = (\sum_{e_{ij} \in E} e_{ij}) / d(v_i) \quad (4\text{-}2)$$

其中v_i代表该点，$d(v_i)$则是该点度，e_{ij}是以该点为端点的边。这里关联度越大则说明该账户与其他账户之间交易密集，对应到图中则说明该点附近的点多。因此本文将关联度的概念和最大最小选择方案结合，将每个点的关联度进行计算并排序，选取关联度最高的前n/k个点中距离最大的两个点作为初始点进行 k-means 聚类，这样做的原因是期望聚类的结果中每一个簇的元素个数接近，这样有利于进行树的插入。对初始点的筛选算法具体算法如下：

Algorithm: FindInitial()
输入：数据集对象$\{l_1, l_2, \dots, l_n\}$及分类k 输出：个簇初始化中心$W = \{w_1, w_2, \dots, w_k\}$ 1:对数据集对象根据式 4-2 分别计算每个点的关联度，并按降序排列形成集合D 2:计算D中前n/k元素路径最远的两个元素，从D移除这两个元素并将其加入到W中 3:while(W的个数小于k) 4: int N=W中元素个数 5: 计算D中前$N * (\frac{n}{k})$元素中距离W集合最远的点，从D移除该元素并将其加入到W中

通过该算法结合了最大最小算法的选取方式，同时对初始点选取结合了空间位置，提高对初始点筛选的准确性，在 k-means 聚类算法执行时，会减少迭代次数。

4.3.3 账户关联算法实现

结合以太坊中 Merkle Patricia 树是 16 叉树的特点，因此期望分类结果中每个簇中元素不多于 16，因此类数目k可以账户总数和 Merkle Patricia 树结构确定。在分类过程中往往不会满足样本较均匀的分布到每一个簇中，因此在进行分类过程中会对分类结果进行去噪处理，当分类的结果簇中元素小于阈值则认为该簇为噪声，在总样本中去除噪声并重新进行分类进行迭代。本文对噪声同样进行处理，形成噪声的簇不为 1 说明这个簇内的元素之间交易频繁，因此将迭代过程中产元素不为 1 的噪音同样进行记录。最后将聚类相同的账户存放在同一个父节点下。分类过程中的关键伪代码如下：

Function: Classify()

输入：数据集对象$L = \{l_1, l_2, \dots, l_n\}$及初始化中心$W = \{w_1, w_2, \dots, w_k\}$
输出：分类结果$R = \{r_1, r_2, \dots, r_k\}$
1:while(E(W)>threshold) {
2: 移除L中噪声 remove(l_1 ∈Noise)// 移除L中噪声
3: 重新计算W initial()//重新计算W
4: for（$l_i \in L$）{
5: 计算l_i到W各簇的距离 dis=calc(l_i)
6: 将l_i归类到距离最小的簇中，并将l_i从L中移除
7: }
8:}
9:$R = W$

通过对阈值有效的选取，可以使聚类结果簇中元素个数比较相近，并按照聚

类结果对 Merkle Patricia 树进行构造。

4.4 实验验证

由于真实交易具有保密性，因此本文根据某企业提供的 400 个用户之间 30 天内进行的积分交易样本作为测试数据并进行关联分类。实验环境采用 Intel(R) Core(TM) i7-4870HQ 处理器，16G 内存的硬件平台，使用 JAVA 实现所有的代码。

对于改进 k-means 聚类初始值选择算法，本文采用对样本数据进行聚类过程中迭代次数体现，本文对样本采用随机采样选取初始中心和本文提出的选取方法进行比较，其中由于随机采样的随机性，因此本文做了 50 次实验并取平均值。实验结果如表 4-1 所示，可以看出改进 K-means 通过对初始簇的选择明显减少迭代次数。

表 4-1 迭代次数比较

算法	迭代次数		
随机选取初始值 K-means 聚类算法	最高	最低	平均
	15	7	10.5
关联度选取初始值 K-means 聚类算法	6 次		

由于样本没有准确的评价标准，因此不能采用和标准库相似的评价方式。因此本文通过样本分类结果，将关联账户信息存放到 Merkle Patricia 树的拥有相同父节点的叶子结点中，通过根据样本数据交易频率进行交易，以执行相同交易笔数的交易的时间判断聚类的相似度。 同时本文样本数据较小，如果仅将样本数据直接插入到 Merkle Patricia 树中，树的深度比较小，而真实情况下账户量比较大，因此本文模拟添加了 50 万账户信息到 Merkle Patricia 树中，然后将 400 个样本账户也添加到 Merkle Patricia 树中进行测试。本文分别对以太坊现有的 RLP 编码方式存储 400 个账户和采用聚类后的账户分类添加到 Merkle Patricia 树中进行测试，本文按样本中的交易信息进行批量交易执行，重复执行实验 50 次并取平均值，实验结果如图 4-6 所示。其中图中 CA1 是采用关联度选取初始值 K-means 聚类算法，CA2 是采用随机采样选取初始值的 K-means 聚类算法，从图中可知按两种聚类结果进行批量交易执行的时间相近，因此认为两种聚类结果相似度较高。而对于 RLP 编码插入的账户，采用以太坊中一笔交易进行两次账户修改的算法，时间消耗明显要高于经过关联账户处理的执行时间，经过关联账户处理的执行时间为 RLP 编码处理的 70%左右。因此通过对账户存储结构的优化，提高了以太坊中状态树的性能。

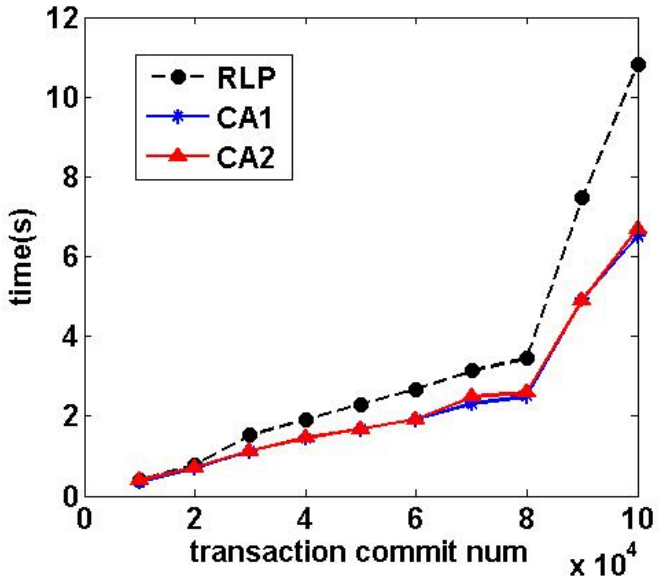


图 4-6 执行交易时间图

4.5 本章小结

本章主要针对以太坊中 Merkle Patricia 树中账户存储的方式进行改进，其中对于交易执行修改账户算法，提出一次对交易中两个账户同时修改的方式，索引到两个账户的公共前缀对应的节点中，并以该节点作为跟节点的子树对两个账户进行修改，从而减少索引和哈希次数，提高效率。根据提出的关联账户的想法，采用 K-means 聚类算法对交易账户进行聚类，根据交易笔数将交易频繁的账户聚类，其中本文根据图的相关概念提出了一种基于边的权重和点的度的初始化数据选择算法。最后根据样本进行性能测试。

第 5 章 去中心化任务发布系统的实现

本章将使用改进 PBFT 作为共识机制的以太坊搭建联盟链，并以任务发布平台作为应用场景，其中任务发布平台中的账户之间的交易通过区块链中智能合约执行，并由区块链维护账户信息，实现去中心化。

5.1 系统架构及需求分析

5.1.1 系统架构

本文要实现去中心化的任务发布平台，其中赏金流动采用智能合约控制，实现去中心化。该平台的系统架构如图 5-1 所示。任务发布平台采用 B/S 架构，每一个联盟体的用户与其 Web 服务器进行交互，而以太坊节点服务器提供 WebService 接口，以太坊节点服务器以联盟链形式链接，实现联盟间去中心化。各个联盟通过协商确定每个联盟体允许运行的以太坊节点个数，因为节点个数不同会对区块决策时的权重有影响，通过联盟各自维护属于自己的以太坊节点，参与区块链的维护过程，实现整个以太坊网络的去中心化，进而实现区块链平台的去中心化应用。其中对于 Web 服务器来说，以太坊节点服务器与传统数据库的功能类似，提供 Web 服务器所需要的数据，不同的是，以太坊中账户信息是去中心化存储的，并没有一个管理者可以管理所有的账户，而是由联盟中所有节点共同管理。

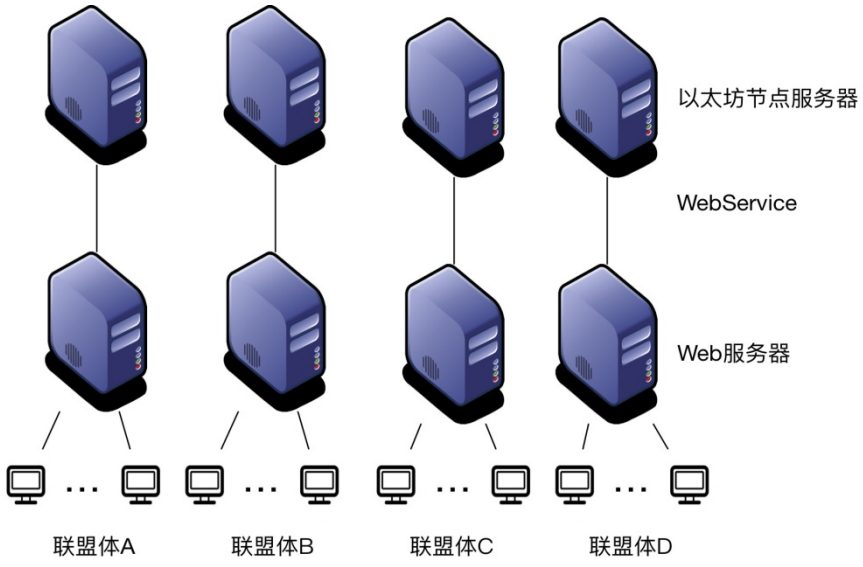


图 5-1 系统框架

5.1.2 需求分析

该系统主要实现任务发布系统的去中心化执行，因此联盟中各群体之间不需要建立相互信任，且不需要第三方进行监管执行。该系统主要功能实现任务发布、接受和删除任务的去中心化执行，并根据任务的执行情况，对任务赏金实现流通，

其用例如图 5-2 所示。

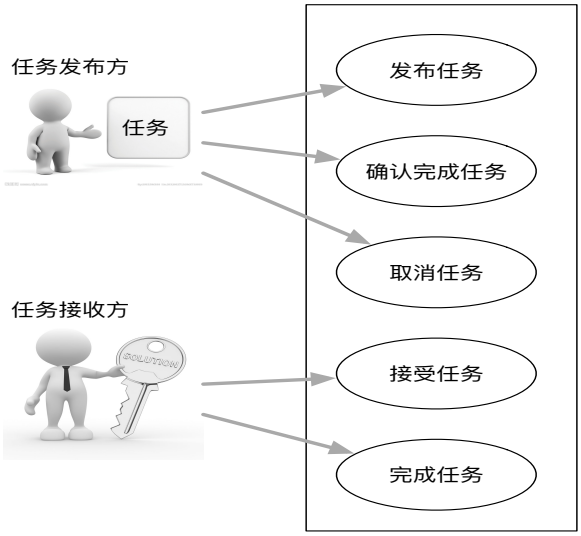


图 5-2 任务发布功能用例图

任务发布方将任务发布后，任务接收方可以看到任务的相关内容与赏金，任务接收方可以选择是否接受任务，当任务接收方完成任务会发出完成任务的通知，并传递给任务发布方，任务发布方会确认任务完成，并反馈给系统任务是否完成，从而实现一次任务发布。当发布交易的时候会从任务发布者账户中扣除赏金，当任务完成时将赏金转移到任务接受者账户中，当任务发布者取消任务，赏金会转回发布者账户中。同时对于超时的任务采取自动删除，并在区块链中同步该消息。整个过程中赏金的流动都在区块链中各个账户之间，每一笔任务发布都可以看做是账户之间交易的过程。

同时作为联盟中每一个联盟体，都需要对其维护的以太坊节点进行检测，因此需要以太坊提供检测指标相关的接口，其中包括对该节点区块链中的区块查询、账户信息查询以及当前区块链中的活跃节点信息，同时在系统初始化时需要进行数据迁移的相关工作，因此需要提供账户批量导入的功能。

5.2 智能合约开发介绍

5.2.1 智能合约开发

以太坊中可以运行图灵完备的脚本语言，这是以太坊实现去中心化应用的核心技术。在以太坊中，开发智能合约的语言有四种，分别是 Serpent、Solidity、Mutan 和 LLL，其中 Serpent 受 Python 语言启发，Solidity 受 JavaScript 语言启发，Mutan 受 Go 语言启发，而 LLL 则受 Liso 语言启发，这些语言都是为了面向合约编程而从底层开始设计的语言。

其中，Solidity 是以太坊智能合约开发的首选语言，Solidity 内置了 Serpent 语言的所有特性，但语法规则却类似于 JavaScript，因此上手容易，逐渐的成为以太坊中智能合约开发使用量最大的一种语言。Solidity 是一种面向对象静态类

型的语言，其语法和一般面向对象语言类似。Solidity 的语言特性比较少，这也是为了更好的开发出安全的智能合约所做出的牺牲。

通常提到的区块链上运行的去中心化应用程序称为 DApp，一个 DApp 是由智能合约和后端代码构成的。从架构角度看，DApp 非常类似于传统的 Web 应用，但是传统的 Web 应用中，客户端持有 JavaScript 代码，并由用户在自己的浏览器执行 JavaScript 代码，服务器端则是中心化的执行数据交互的相关工作。而在 DApp 中所有的服务与逻辑都运行于区块链中，DApp 不仅需要设计前端的应用程序，还需要开发基于以太坊的智能合约代码。这些智能合约代码以 JsonRPC 的方式提供给应用程序进行调用。具体结构如图 5-3 所示。其中前端可以通过 Json-RPC、WebService 和 Restful-API 方式来调用编写的智能合约 ABI。智能合约一旦被调用，智能合约就像以太坊中转账操作一样，被广播到所有的节点上，通知这些节点运行被调用的智能合约 ABI，然后这些被调用的 ABI 会各自运行在该节点的 EVM 中，最后通过区块的生成将运行过程和结果打包进区块中，并通过区块的同步实现全网的一致性。

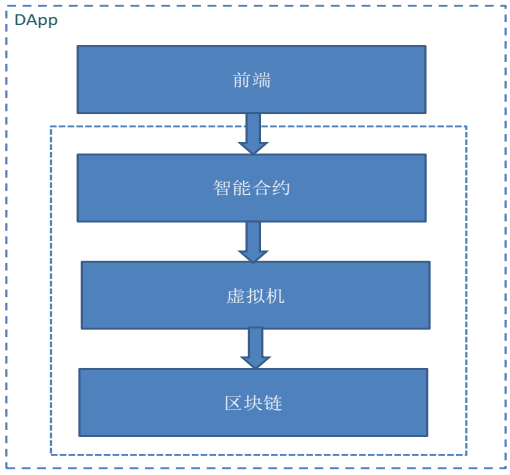


图 5-3 DApp 架构

5.2.2 智能合约开发工具

编写安全的智能合约是实现 DApp 的关键，而开发智能合约在开源社区中提供了一系列的开发工具，方便 DApp 开发人员简便地开发出安全的智能合约，目前比较流行的是 browser-solidity 工具。

该工具是基于浏览器的，用户可以在线编写智能合约代码，并且可以编译成二进制字节码部署到自己所属的区块链网络中。除此之外，编写智能合约时可以在模拟的环境中进行调试。 开发界面如图 5-4 所示，主要分为智能合约代码编写区、编译环境和部署。在编写区中可以编写智能合约，其中第一行需要选择 solidity 版本号，目前最新版本是 0.4.4。第二行是编写智能合约的界面，在编写过程中会在浏览器右半部分显示出语法错误。编译环境区域中，可以选择 Solidity 版本号。部署区域中可以将编写好的智能合约代码部署到所选择的区块链环境中，

并可以显示对应的字节码和 ABI 接口信息等，方便对智能合约的调用。

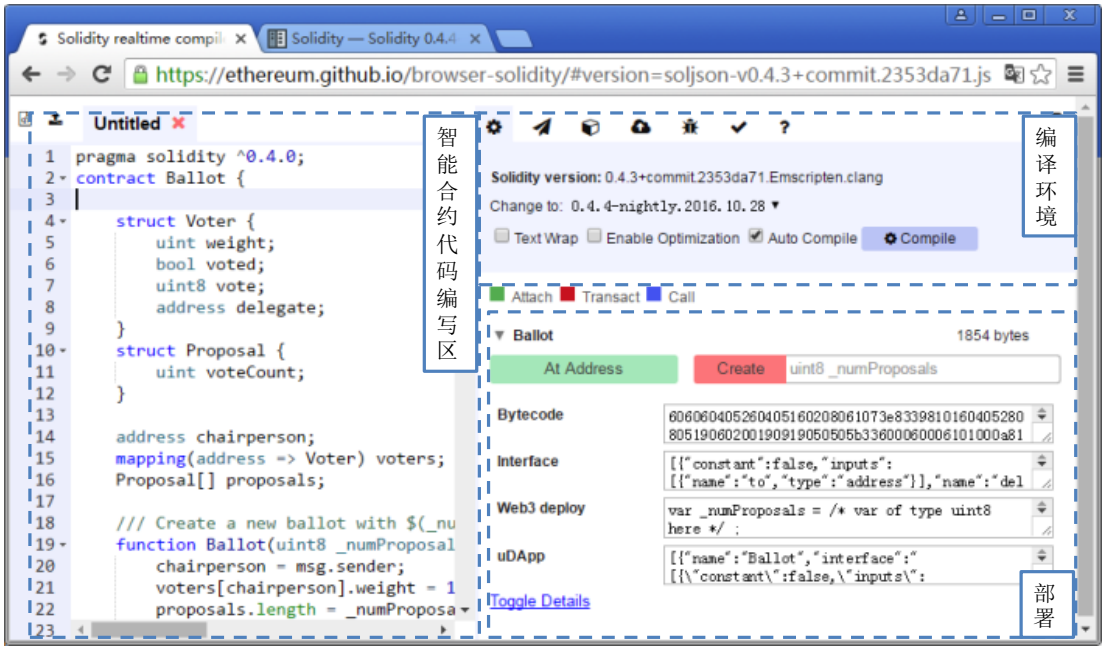


图 5-4 browser-solidity 开发界面

5.3 去中心化任务发布平台实现

本小节根据改进以太坊搭建 8 节点联盟链，模拟 8 个联盟体组成的公共账本，并实现基于该账本的任务发布平台。其中以太坊中接口采用 WebService 接口，Web 服务端则采用 Http 请求，并结合智能合约共同实现去中心化任务发布平台。

5.3.1 以太坊接口开发

由于以太坊中只提供了 Json-RPC 调用接口，因此需要对以太坊编写 WebService 接口，实现与 Web 服务器的交互过程，其中接口管理信息如表 5-1 所示。

表 5-1 区块链管理接口信息

名称	参数及说明
AccountInfo	参数是账户地址。用于查询账户相关信息。
BlockNum	用于查询当前区块链区块个数。
BlockInfo	参数是区块号。用于查询区块的信息。
ActiveNode	返回当前结点认为的可靠节点。
Balance	参数是账户地址。返回该账户余额。
BatchTransactions	参数是账户地址和对应的账户金额。用于系统初始化账户金额。
BatchAccounts	参数是账户地址。用于初始化账户。

在智能合约模块中，同样需要开放 WebService 接口，开放 Web 服务器调用智能合约的相关接口。其中接口说明如表 5-2 所示。

表 5-2 任务发布平台接口信息

名称	参数及说明
CreateTask	参数任务的相关的 json 字符串。新建一个任务，并通过调用智能合约 ABI，将该任务以交易的形式写入到区块链中。智能合约根据任务内容将人物发起者账户对应信息转移到智能合约账户中，实现资金冻结。
AcceptTask	参数是接受任务者相关信息以及任务编号的 json 字符串。调用智能合约 ABI，找到对应的任务并对状态和接收信息进行修改
FinishTask	参数是完成任务相关的 json 字符串。与 AccptTask 类似，更改合约地址中对应任务的相关状态信息。
ConfirmTask	参数是任务发起者确认完成的 json 字符串。智能合约找对应任务完成者账户，从智能合约账户交易与合约对应的赏金，并将该任务从合约地址中移除。
DropTask	参数是任务发起者取消任务的 json 字符串。调用智能合约将任务从智能合约地址中删除

5.3.2 智能合约实现

智能合约结构体如下：

```
mapping (unit => Task) public taskMapping
event changeTask(uint taskId,address taskSender, address taskReceiver,uint
amount,uint state);
struct Task {
    uint taskId;
    address taskSender;
    address taskReceiver;
    uint amount;
    uint state;
}
```

智能合约中定义了全局变量 taskMapping 用于存放所有的交易，结构体中描述了该合约的参数信息，taskId 用于对任务进行区分，taskSender 和 taskReceiver 分别是任务发起者和接受者的账户地址，用于进行赏金交易，amount 是代表赏金的价值，state 是人物的当前状态。事件 changeTask 是每次触发时，根据触发的参数进行相应的操作，包括对任务状态的修改，以及赏金的交易过程。通过该智能合约实现了任务发布平台的去中心化。

5.3.3 Web 服务端接口开发

Web 服务端主要负责用户的登录和任务发布的相关功能，本文主要描述任务发布的相关功能。其接口如表 5-3 所示

表 5-3 TaskService 接口信息

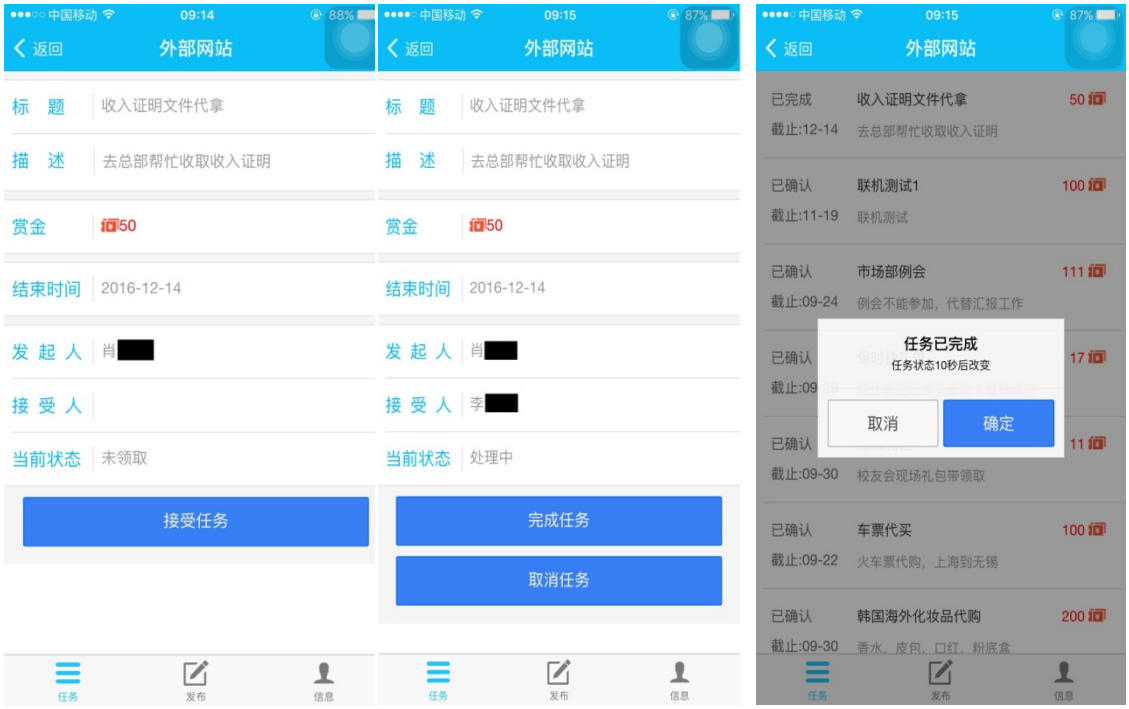
名称	请求类型	参数及说明
addTask	Post	参数是包含任务 Id、任务发布者地址、任务标题、任务内容、任务截止日期、赏金和当前日期的 JSON 字符串。用于向区块链 WebService 接口发送添加任务请求。
deleteTaskById	Delete	参数是任务发布者地址、任务 Id、任务状态。用于调用区块链 Webservice 接口调用 DropTask 方法。
updateTask	Put	参数是任务 Id、发布者地址和任务状态。用于更改合约地址中对应任务的状态。
queryAccount	Get	参数是发布者地址。向区块链 WebService 接口发送请求，查询账户相关信息。

5.3.4 任务发布系统实现

本文针对上述描述的接口完成了系统的实现。搭建联盟链启动时一个节点的日志信息如图 5-5 所示。联盟链中选取的视图是从 0 开始，生成区块时间间隔为 10s。从日志中可以看到，首先接收到了新区块的信息，并打印区块号视图等相关信息，同时可以看到该节点完成了第 28 号区块的共识校验过程，并添加到该节点的区块链上的日志打印，说明区块链可以正常运行。

```
22:42:04.659 DEBUG [net]  From:     3973cb86 [redacted]   Recv:   NEW_BLOCK [ number: 28 hash:8a70e2 view: 0 ]
22:42:04.659 DEBUG [sync]  New block received: block.index [28]
22:42:04.938 DEBUG [net]  To:   0947751[redacted] Send:   [PING]
22:42:04.939 DEBUG [net]  From:     094[redacted]   Recv:   [PONG]
22:42:04.941 DEBUG [net]  From:     094[redacted]   Recv:   [PING]
22:42:04.941 DEBUG [sync]  New block Imported:
BlockData [ hash=8a70e27789dbf6b8b7159a20514dd045bdf9469d52329a49f0acebb92288606f
  parentHash=6fba3d7187afe1b5ac65ea736c402e60f7e33ece9f94c0622651462764b22b02
  unclesHash=
  coinbase=0000000000000000000000000000000000000000
  stateRoot=668bf7d6961087250e1e3bf6bdd1a88e51c9105bd80b79149a443af33d27fc4c
  txTrieHash=56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421
  receiptsTrieHash=56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421
  difficulty=0
  number=28
  gasLimit=1000000000
  gasUsed=0
  timestamp=1480776134 (2016.10.03 22:42:04)
  extraData=
  mixHash=
  nonce=
Uncles []
Txs []
]
```

图 5-5 联盟链节点启动日志



（a）任务接受界面　（b）任务确认完成界面　（c）信息提交界面

图 5-6 用户展示界面

图 5-6 是任务发布系统的前台展示界面，本文设计的前端界面是应用于移动端浏览器，其中图 a 是任务接受界面，该界面通过登陆界面进入，登陆界面信息由 Web 服务器管理，后台通过对登陆账号关联找到该账户对应的区块链账户地址。该页面中显示了发布任务并显示该任务的完成赏金以及该任务的截止日期。图 b 是任务确认完成界面，可以看到任务名称、任务描述、赏金、任务状态和截止时间等信息，信息是每次刷新查询区块链信息返回的结果，当点击确认完成则赏金就会转到任务接受者账户中。图 c 中是点击确认完成后的信息，由于本文搭建的联盟链生成区块的时间是 10s，因此一笔交易执行最多在 10s 内写入到区块链中，并永久保存，因此提示任务状态 10s 后改变，此时在区块链中该任务的状态会变成已完成，而赏金则会从发布者的账户交易到接受者账户中，实现交易。图 5-7 是在任务发布前与任务接收完成后，发布者和接受者账户余额信息，可以看到通过接受任务，赏金成功的从任务发布者账户中转到了任务接受者账户中，实现了任务发布系统交易过程的去中心化。

（a）任务发布前任务发布者账户信息　　（b）任务完成后任务发布者账户信息

（c）任务发布前任务接受者账户信息　　（d）任务完成后任务发布者账户信息

图 5-7 赏金交易信息

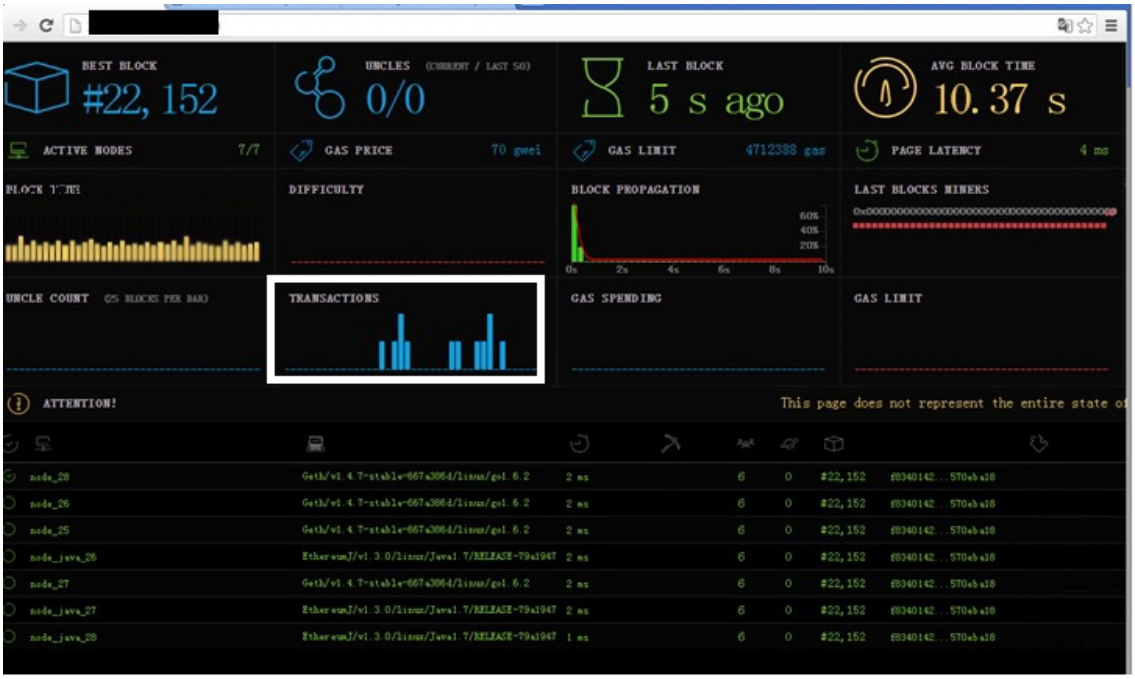


图 5-8 区块链检测界面

图 5-8 是本文基于开源前端框架搭建的以太坊检测界面，主要通过 JSONRPC 调用实现与区块链进行数据交互。其中因为本文以太坊取消了挖矿机制，同时方便应用在代码中取消了 Gas 消费，因此在图中涉及到挖矿相关的数据为初始写入

数据。从图中右上角可以看到区块的生成时间均值是 10.37 秒，与区块生成设置时间一致，图中左上角显示现在生成的总区块数，在图中框住的区域代表交易笔数的柱状图，可以看到有交易发生，说明有智能合约被执行。

5.4 本章小结

本章对前文提出的以太坊相关改进进行应用，搭建 8 个节点的改进以太坊系统，并初始化正常运行。然后对于本文提到的去中心化任务发布平台应用进行实现，开放以太坊相关接口，让 Web 服务器可以与以太坊节点进行交互，并实现对区块链的监控管理。介绍了智能合约的相关开发过程，并通过编写 DApp 实现去中心化的任务发布平台，完成不同联盟体账户之间的交易，对改进的以太坊系统进行应用。本章最后分别给出联盟链搭建系统启动日志、任务发布平台用户展示界面以及对区块链监控的界面实现的截图。

第 6 章 总结与展望

6.1 工作总结

本文以计算机金融领域中的区块链技术作为本文的研究背景。区块链技术从2009年首次以比特币的形式提出之后，在计算机金融领域内引起了强烈的反响。区块链技术提供了一种去中心化系统的实现方式，以太坊更是在区块链技术上创新的提出了去中心化的应用。本文以任务发布平台系统作为研究背景，实现联盟链形式的去中心化任务发布平台的搭建。现阶段流行的区块链不论比特币、以太坊都是以公链模式进行搭建，因此都采用了 PoW 共识机制，然而 PoW 共识机制并不适用于联盟链环境中，一方面会造成资源不必要的浪费，另一方面针对节点数量少的联盟链中，安全性几乎为零。而近些年，区块链的发展却正向着联盟链方向前进，因此基于联盟链的区块链研究对于推动区块链技术的应用与发展具有巨大的意义。

本文基于联盟链为背景，以对区块链研究作为大方向，主要针对以太坊在联盟链场景中共识机制和账户存储优化两个方面进行研究，同时将研究内容与现实应用结合，实现去中心化的任务发布平台，本文主要研究内容如下：

（1）以以太坊为例，详细地介绍了区块链的架构模型以及区块链内核心技术。

（2)分析了现在区块链技术中常用的共识机制的实现原理，其中包括PoW、PoS、DPoS 和拜占庭共识机制，分别分析每种机制的优劣并指出机制中存在的问题。

（3）介绍智能合约的概念，并针对实现去中心化应用的以太坊面临的不足提供解决办法。

（4）设计一种基于改进 PBFT 的以太坊共识机制，并实现该系统，解决了基于联盟链中，以太坊系统对算力浪费的问题。其中本文针对 PBFT 机制进行改进，结合以太坊区块结构，采用超时机制来解决 PBFT 机制在视图切换以及证书回收时，为了保证各节点一致性需要相互通信，增大网络开销的问题。

（5）详细介绍了以太坊中状态树的实现过程，并针对状态树效率问题，提出改进关联账户状态同时修改的办法解决。其中针对于关联账户的筛选采用 k-means 聚类算法实现。

（6）针对以太坊的改进，实现去中心化的任务发布平台，其中包括前端设计、后台接口以及智能合约的编写，并在局域网搭建的联盟链中实现运行。

6.2 展望

虽然本文所提出改进 PBFT 机制解决了以太坊中算力浪费的问题，同时针对状态树的效率提出改进方案，但是由于作者本身的知识水平有限和时间的限制，本课题研究还需要在很多地方进行改进和完善，这也将是作者本人的下一步工作，总结起来一共有以下几点：

（1）针对以太坊的共识机制，本文设计的模型中，不适用于动态的添加或删除联盟中的节点，该功能有待改进并实现。

（2）改进 PBFT 以太坊共识机制中，出块时间要大于网络延迟，虽然本文设计的共识机制达到了以太坊现有出块速度，并以交易列表维护的本地库提供服务，但如何快速达成共识仍然是进一步研究的目标。

（3）本文结合以太坊中 Merkle Patricia 树提出了关联账户的概念，虽然对于关联账户的操作可以提高效率，但是非关联账户之间的交易并没给出相关的解决办法，同时该算法只适用于联盟链中。

参考文献

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Consulted, 2009.

[2] 李林. 当前比特币行业发展现状及政策研究[J]. 清华金融评论, 2015(8).

[3] Becker J, Breuker D, Heide T, et al. Can we afford integrity by proof-of-work? Scenarios inspired by the Bitcoin currency[M]. The Economics of Information Security and Privacy. Springer Berlin Heidelberg, 2013: 135-156.

[4] Wright A, De Filippi P. Decentralized Blockchain Technology and the Rise of Lex Cryptographia[J]. Social Science Electronic Publishing, 2015.

[5] Ametrano F M. Bitcoin, Blockchain, and Distributed Ledger Technology[J]. Social Science Electronic Publishing, 2016.

[6] Dennis R, Owen G. Rep on the block: A next generation reputation system based on the blockchain[C]. International Conference for Internet Technology and Secured Transactions. IEEE, 2015.

[7] Buterin V. Ethereum white paper: a next generation smart contract & decentralized application platform[J]. www3. ethereum. org Nick Szabo, Formalizing and Securing Relationships on Public Networks, http://szabo. best. vwh.net/formalize. html, 2013.

[8] 袁勇, 王飞跃. 区块链技术发展现状与展望[J]. Acta Automatica Sinica, 2016, 42(4):481-494.

[9] Smith B, Christidis K. IBM Blockchain: An Enterprise Deployment of a Distributed Consensus-based Transaction Log[J].

[10]陈何清. 基于区块链的 IMIX 传输系统的设计与实现[D]. 南京大学, 2016.

[11]Underwood S. Blockchain beyond bitcoin[J]. Communications of the Acm, 2016, 59(11):15-17.

[12]Pilkington M. Blockchain Technology: Principles and Applications[J]. Social Science Electronic Publishing, 2016.

[13]Lee L. New Kids on the Blockchain: How Bitcoin's Technology Could Reinvent the Stock Market[J]. Social Science Electronic Publishing, 2016.

[14]Mainelli M, Milne A. The Impact and Potential of Blockchain on Securities Transaction Lifecycle[J]. Social Science Electronic Publishing, 2016.

[15]King S, Nadal S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake[J]. 2012.

[16] Larimer D. Delegated proof-of-stake white paper [Online], available: http://www.bts.hk/dpos-baipishu.html, 2014

[17]D Schwartz，N Youngs，A Britto. The Ripple protocol consensus algorithm [Online], available: https://ripple.com/files/ripple_consensus_whitepaper.pdf,2015
[18]Martin J P. Fast Byzantine Consensus[J]. IEEE Transactions on Dependable & Secure Computing, 2005, 3(3):202-215.
[19]Cachin C. Architecture of the Hyperledger Blockchain Fabric[J]. 2016.
[20]Amir Y, Coan B, Kirsch J, et al. Prime: Byzantine Replication under Attack[J]. IEEE Transactions on Dependable & Secure Computing, 2011, 8(4):564-577.
[21]Garay J, Kiayias A, Leonardos N. The Bitcoin Backbone Protocol: Analysis and Applications[M]// Advances in Cryptology - EUROCRYPT 2015. Springer Berlin Heidelberg, 2015:281-310.
[22]Zamfir V. Introducing Casper “the friendly ghost”[J]. Ethereum Blog, 2015.
[23]Brown R G, Carlyle J, Grigg I, et al. Corda: An Introduction[J]. R3 CEV, August, 2016.
[24]G Wood, Ethereum: a secure decentralised generalised transaction ledger[online],available: http://gavwood.com/Paper.pdf
[25]Bahga A, Madisetti V K. Blockchain Platform for Industrial Internet of Things[J]. Journal of Software Engineering and Applications, 2016, 9(10): 533.
[26]Zohar A, Sompolinsky Y. Accelerating Bitcoin’s Transaction Processing[J]. 2013.
[27]Bahga A, Madisetti V K. Blockchain Platform for Industrial Internet of Things[J]. Journal of Software Engineering and Applications, 2016, 9(10): 533.
[28]Huckle S, Bhattacharya R, White M, et al. Internet of Things, Blockchain and Shared Economy Applications[J]. Procedia Computer Science, 2016, 98:461-466.
[29]Carter J L, Wegman M N. Universal Classes of Hash Functions (Extended Abstract) [J]. Stoc ’77 Proceedings of the Ninth Annual Acm Symposium on Theory of Computing, 1979, 18(2):143-154.
[30]Decandia G, Hastorun D, Jampani M, et al. Dynamo: amazon's highly available key-value store[J]. Acm Sigops Operating Systems Review, 2007, 41(6):205-220.
[31]Ferguson P, Senie D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing[J]. RFC, 1998.
[32]Nikolić I, Biryukov A. Collisions for Step-Reduced SHA-256[M]. Fast Software Encryption. Springer Berlin Heidelberg, 2008:1-15.
[33]J. Göbel, A.E. Krzesinski, H.P. Keeler, et al. Bitcoin Blockchaindynamics: The selfish-mine strategy in the presence of propagation delay[J]. Performance Evaluation, 2015, 104:23-41.

[34]Stoica, Ion, Morris, Robert, Liben-Nowell, David. et al.: Chord: a scalable peer-to-peer lookup protocol for internet applications[J]. IEEE/ACM Transactions on Networking, 2003, 11(4):149-160.
[35]Kamvar S D, Schlosser M T, Garcia-Molina H. The Eigentrust algorithm for reputation management in P2P networks[C]. International Conference on World Wide Web. ACM, 2003:640-651.
[36]Herbert J, Litchfield A. A Novel Method for Decentralised Peer-to-Peer Software License Validation Using Cryptocurrency Blockchain Technology[C]. Australasian Computer Science Conference. 2015.
[37]Stoica I, Morris R, Karger D, et al. Chord: A scalable peer-to-peer lookup service for internet applications[J]. ACM SIGCOMM Computer Communication Review, 2001, 31(4): 149-160.
[38]Omohundro S. Cryptocurrencies, smart contracts, and artificial intelligence[J]. AI matters, 2014, 1(2): 19-21.
[39]Castro M, Liskov B. Practical byzantine fault tolerance and proactive recovery[J]. Acm Transactions on Computer Systems, 2002, 20(4):398-461.
[40]Platania M, Obenshain D, Tantillo T, et al. On Choosing Server- or Client-Side Solutions for BFT[J]. Acm Computing Surveys, 2016, 48(4):1-30.
[41]Cachin C, Schubert S, Vukolić M. Non-determinism in Byzantine Fault-Tolerant Replication[J]. 2016.
[42]Castro M, Liskov B. Practical Byzantine fault tolerance[C]. Symposium on Operating Systems Design and Implementation. USENIX Association, 1999:173--186.
[43]Delmolino K, Arnett M, Kosba A E, et al. Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab[J]. IACR Cryptology ePrint Archive, 2015, 2015: 460.
[44]Kõlvart M, Poola M, Rull A. Smart Contracts[M]. The Future of Law and eTechnologies. Springer International Publishing, 2016.
[45]田丰,马仲蕃. 图与网络流理论[M].科学出版社.1987.
[46]Macqueen J B, Macqueen J B. On convergence of k-means and partitions with minimum average variance[J]. Annals of Mathematical Statistics, 1965, 36.
[47]Bradley P S, Fayyad U M. Refining Initial Points for K-Means Clustering[C]. Fifteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc. 1998:91--99.

[48]Wagstaff K, Cardie C, Rogers S, et al. Constrained k-means clustering with background knowledge[C]. ICML. 2001, 1: 577-584.

致谢

时光荏苒，白驹过隙，转眼间两年年半的研究生生涯就要结束了。回想这两年半的时间，不禁感慨万千，两年半的点点滴滴都还历历在目。在完成论文之际，想表达这两年半时间内对身边人的感谢。

首先我要感谢我的导师黄秋波老师，黄老师对科研教学的一丝不苟，对学生的认真负责给我留下了深刻的印象。黄老师在我的研究生生涯中，从学习和生活上给予了我耐心的指导和细心的帮助，从论文的选题开题，到论文的撰写，都给予了我悉心的指导和帮助。除了学习上的帮助，生活上也对我帮助很多，将他的社会经验与我分享。黄老师严谨的治学态度、科学的工作作风、胸怀坦然的为人品格和诲人不倦的敬业精神令我敬佩。在此，特向恩师黄秋波老师表示由衷的感谢。

其次，我要感谢刘飞、孙志峰学长，是你们对我的帮助让我快速融入了研究生生活，实验室同学张志翔、钟征祥、杨延彬和刘天扬，这两年半的时间内每当我有困难的时候都会给予我帮助，是你们陪伴我度过了这两年半美好的研究生生涯。

感谢身边的每一位同学和辅导员，是你们的陪伴让我感受到了班级的团结与温暖，与你们相聚在这里是我莫大的荣幸。

在这里，我也要对我的父母和我的家人表示感谢，你们对我的关怀是我前进的最大动力。

最后，感谢抽出宝贵时间对本论文进行评审的专家们，谢谢你们给予的宝贵意见和指导。