第4章 区块链的分布式共识

【本章导学】

在区块链系统这样的非中心化系统中，并不存在中心权威节点，由于参与的各个节点的自身状态和所处的网络环境不尽相同，而交易信息的传递又需要时间，并且消息传递本身并不可靠，所以每个节点收到的需要记录的交易内容和顺序也很难保持一致。此外，由于区块链中参与的节点身份难以控制，还可能出现恶意节点故意阻碍信息传递或者发送不一致的信息给不同节点，从而扰乱整个区块链系统的记账一致性。因此，区块链系统的记账一致性问题，是一个非常关键的问题，关系到整个区块链系统的正确性和安全性。

【学习目标】

- 熟悉分布式共识的基础知识以及分布式共识的历史；
- 熟悉分布式共识的定理以及分布式算法的分类；
- 掌握常用分布式共识算法的原理、实现过程以及应用场景；
- 熟悉共识算法的实训案例。

4.1 分布式共识的基础

4.1.1 “分布式”与“共识”

可以把分布式共识分为“分布式”和“共识”两方面进行理解。

1. 分布式

分布式的字面含义是分散的，与其相对的概念是中心化。在区块链领域中，分布式的概念应用很广，但主要可以概括为每个节点有自主管理权(Self-Governance)，且成员之间可以点对点地完成信息的交换或资产的交易。能实现以上功能的系统称为

分布式系统。

2. 共识

共识问题实际上是社会科学和计算机科学领域共同的经典问题。最早可以追溯到 1959 年 Edmund Eisenberg 和 David Gale 发表的以“有主观思想的个体”为对象所做的“共识概率分布”的社科研究。而在计算机领域中的里程碑式的共识研究是在 1975 年由纽约州立大学的 E. A. Akkoyunlu、K. Ekanadham 和 R. V. Huber 在论文中提出的“计算机通信两军问题”。该研究主要论证了在不可靠的通信链路上试图通过通信达成共识是不可能的。

3. 分布式共识

区块链是典型的分布式系统。在区块链系统中，如何让每个节点通过一定的规则将各自的记账保持一致是非常关键的问题，这个问题的解决方案就是共识机制。为分布式系统设计的、目的是让所有参与记账的节点所记的信息保持一致的算法，就是分布式共识算法。

4.1.2 分布式系统

分布式系统(Distributed System)是相对于中心化系统的系统统称，是由一组“通过网络进行通信、为了完成共同的任务而协调工作的”计算机节点组成的系统。

分布式系统通常通过维护多个副本来进行容错，共识算法的核心问题是维护多个副本的一致性，即在部分副本由于各种原因发生错误的情况下，整体集群仍能正常对外提供服务。

在分布式系统中，多个主机通过异步通信的方式组成网络集群。在这样的异步系统中，需要主机之间进行状态复制，以保证每个主机处于一致的状态。然而在异步系统中，可能出现突然无法通信的故障主机，也可能出现网络拥塞和延迟，还可能出现错误信息。在传统的网络软件结构中，这几乎不是一个问题，因为有一个中心服务器存在，所以其他从库向主库看齐即可。但区块链是一个分布式的网络结构，在这个结构中每个节点是地位对等的，一切都要“商量着来”。

4.1.3 分布式计算

分布式计算(Distributed Computing)是研究分布式系统如何有效运算的计算机科学。大多数情况下，分布式计算的核心是把需要进行大量计算的工程数据分割成小块，由多台计算机分别计算，上传运算结果，并将结果统一合并得出数据结论。分布式

计算需要组件之间彼此进行交互以实现一个共同的目标。

目前分布式计算项目通常使用世界各地的千万个志愿者计算机的闲置计算能力，通过互联网进行数据传输，来解决一些对人类关系重大的科学问题。例如，分析计算蛋白质的内部结构和相关药物的 Folding@home 项目，该项目结构庞大，需要惊人的计算量，由一台计算机计算是不可能完成的。虽然现在有了计算能力超强的超级计算机，但这些设备造价高昂，而一些科研机构的经费却又十分有限，借助分布式计算可以花费较小的成本来达到目标。

4.1.4 分布式账本

分布式账本(Distributed ledger)又称共享账本，是一种能在网络成员之间共享和同步的数据库。分布式账本记录网络参与者之间的交易，比如资产或数据的交换。分布式账本不存在中央管理员或集中的数据库，并通过点对点网络和共识机制确保了跨节点的数据复制，降低了因协调不同账本所产生的时间和费用成本。区块链系统就是分布式账本的一种。

4.2 分布式一致性简史

分布式一致性(Distributed Consistency)是指分布式的系统在状态上达成一致。简单来说，就是不同的人对一件事情产生了共识。大家或许会对一致(Consistency)与共识(Consensus)含义的差别产生疑问。在这里仅作简要说明，一般认为，共识研究侧重于分布式系统达成共识的算法以及过程，一致性研究则更侧重于系统最终达成的稳定状态。不过，因为一致性的结果是共识算法的目标，两者是不可分割的，所以在很多文献和应用场景中，两个概念也并没有被严格区分，而是可互换的。

1. 计算机的两军问题

1975 年，纽约州立大学石溪分校的阿克云卢(E. A. Akkoyunlu)、埃卡纳德汉姆(K. Ekanadham)和胡贝尔(R. V. Huber)首次提出了计算机领域的两军问题。两军问题表明，在不可靠的通信链路上，试图通过通信达成一致是存在缺陷和困难的。两军问题是计算机领域的一个思想实验，开启了计算机科学界对用代码实现信任体系的研究。

2. 分布式计算的共识问题

分布式计算的共识问题于 1980 年由马歇尔·皮斯(Marshall Pease)等提出。该问题主要研究在一组可能存在故障节点且通过点对点消息通信的独立处理器网络中，如

何针对一个特定事件达成一致共识。

3. 拜占庭将军问题

拜占庭将军问题(The Byzantine generals problem)也称为拜占庭容错,是在1982年由图灵奖得主莱斯利·兰伯特(Leslie Lamport)为具象描述分布式系统研究提出的一个著名的虚拟问题。

拜占庭位于如今土耳其的伊斯坦布尔,是东罗马帝国的首都。由于当时拜占庭帝国国土辽阔,为了达到防御目的,每个军队都分隔很远,将军与将军之间只能靠信差传递消息。在发生战争的时候,拜占庭军队内所有将军和副官必须达成共识,决定是否有赢的机会才去攻打敌人的阵营。但是,在军队内有可能存在叛徒和敌军的间谍,他们能够影响将军们的决定,又会扰乱整体军队的秩序。而达成共识的结果并不代表大多数人的意见。这时候,在已知有成员谋反的情况下,其余忠诚的将军要想办法在不受叛徒的影响下达成一致的协议,拜占庭将军问题就此形成。

拜占庭将军问题是一个协议问题,拜占庭帝国军队的将军们必须全体一致地决定是否攻击某一支敌军。问题是这些将军在地理上是分隔开来的,并且将军中存在叛徒。叛徒可以任意行动以达到以下目标:欺骗某些将军采取进攻行动;促成一个不是所有将军都同意的决定,如当将军们不希望进攻时促成进攻行动;或者迷惑某些将军,使他们无法做出决定。如果叛徒达到了这些目的之一,则任何攻击行动都是注定要失败的,只有完全达成一致的努力才能获得胜利。图4-1所示为拜占庭将军问题。

图4-1 拜占庭将军问题

拜占庭假设是对现实世界的模型化,将拜占庭将军问题延伸到互联网生活中来,其内涵可概括为:在互联网大背景下,当需要与不熟悉的对方进行价值交换活动时,人们如何才能防止不会被其中的恶意破坏者欺骗、迷惑从而作出错误的决策。进一步将拜占庭将军问题延伸到技术领域中来,其内涵可概括为:在缺少可信任的中央节点

和可信任的通道的情况下，分布在网络中的各个节点应如何达成共识。

现如今，拜占庭将军问题仍是公认的最难问题。科学家们认为，在存在消息丢失的不可靠信道上试图通过消息传递的方式达到一致性是不可能的。

拜占庭将军问题最终想解决的是互联网交易、合作过程中的 4 个问题，分别为信息发送的身份追溯、信息的私密性、不可伪造的签名和发送信息的规则。

可以说，拜占庭将军问题是区块链技术想要解决的核心问题，直接影响着区块链系统共识算法的设计思路和实现方式，因而在区块链技术体系中具有重要意义。

4.3 分布式共识的定理

4.3.1 FLP 定理

1985 年，由 Fischer、Lynch 和 Patterson 3 位科学家发表的论文 *Impossibility of Distributed Consensus with One Faulty Process* 指出：在网络可靠，但允许节点失效（即便只有一个）的最小化异步模型系统中，不存在一个可以解决一致性问题的确定性共识算法（No completely asynchronous consensus protocol can tolerate even a single unannounced process death）。以上结论被称为 FLP 不可能定理。该定理被认为是分布式系统中重要的原理之一。

此定理实际上告诉人们，不要浪费时间去为异步分布式系统设计在任意场景下都能实现共识的算法。

4.3.2 CAP 定理

1. CAP 理论的由来

CAP 理论最早是 2000 年由 Eric Brewer 在 ACM 组织的一个研讨会上提出的猜想，后来 Lynch 等人进行了证明。该原理被认为是分布式系统领域的重要原理之一。

2. CAP 的定义

CAP 理论中的字母“C”为一致性（Consistency）。这里的一致性指的是强一致性。强一致性意味着，当系统的更新操作成功并返回客户端完成后，所有节点在同一时间的数据完全一致。

CAP 理论中的字母“A”为可用性（Availability），指的是分布式系统可以在正常

响应时间内提供相应的服务。

CAP 理论中的字母“P”为分区容错性(Partition Tolerance)。分布式系统在遇到某节点或网络分区故障的时候,仍然能够对外提供满足一致性和可用性的服务。关于分区的含义是指在各分布式系统里面,由节点组成的网络本来应该是连通的。然而可能因为一些故障,使得有些节点之间不连通了,整个网络就分成了几个区域。数据散布在这些不连通的区域中,这就叫分区。

分区容错性的含义是当一个数据项只在一个节点中保存时,分区出现后,和这个节点不连通的部分就无法访问这个数据了,这时分区就是无法容错的。提高分区容错性的办法就是将一个数据项复制到多个节点上,出现分区后,这一数据项就可能分布到各个分区中,容错性就提高了。

CAP 理论实际上是指一个分布式系统最多只能同时满足一致性(Consistency)、可用性(Availability)和分区容错性这 3 项中的两项,如图 4-2 所示。

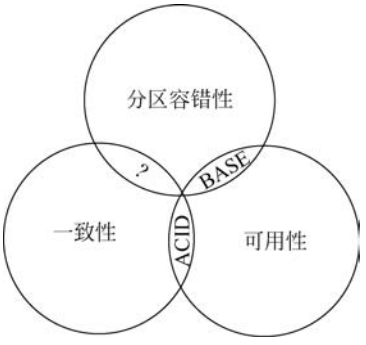

图 4-2　CAP 理论三要素

通过 CAP 理论,如果无法同时满足一致性、可用性和分区容错性,那么要舍弃哪个呢?对于多数大型互联网应用的场景,主机众多、部署分散,而且现在的集群规模越来越大,所以出现节点故障、网络故障是常态,而且要保证服务可用性达到 N 个 9,即保证 P 和 A,舍弃 C(退而求其次保证最终的一致性)。虽然某些地方会影响客户体验,但没达到影响用户流程的严重程度。对于涉及钱财这种不能有一丝让步的场景,必须保证 C。网络发生故障时宁可停止服务,这是保证 C 和 A,舍弃 P。还有一种是保证 C 和 P,舍弃 A。例如,网络故障是只读不写,孰优孰劣,没有定论,这时只能根据场景定夺,适合的才是最好的。

4.4　共识算法

区块链用代码和算法在虚拟世界里建立了一个很稳固的信用系统。要理解这个信用系统,就需要掌握区块链的共识机制。

区块链通过全民记账来解决信任问题,但是所有节点都参与记录数据,那么最终以谁的记录为准呢?或者说,怎么样保证所有节点记录的是一份相同的正确数据呢?这就是如何达成共识的问题。

在区块链系统这样的非中心化系统中，并不存在中心权威节点。由于参与的各个节点的自身状态和所处网络环境不尽相同，而交易信息的传递又需要时间，并且信息传递本身并不可靠，所以每个节点收到的需要记录的交易内容和顺序也难保持一致。此外，由于区块链中参与的节点身份难以控制，还可能出现恶意节点故意阻碍信息传递或者发送不一致的信息给不同节点，从而扰乱整个区块链系统的记账一致性。因此，区块链系统的记账一致性问题是非常关键的，关系着整个区块链系统的正确性和安全性。

在区块链系统中，如何让每个节点通过一定的规则将各自的记账保持一致是一个很关键的问题，这个问题的解决方案就是共识机制。

4.4.1 共识的过程

为了更好地理解共识过程模型，先介绍一下节点的分类及其负责事项。

(1) 数据节点：生产数据或者交易的节点，即大多数使用区块链应用的用户。数据节点是最普通的节点，数据节点的集合是区块链网络节点的集合。

(2) 矿工节点：对生成的数据进行验证、打包、更新上链的节点(矿工)，矿工节点通常会全体参与公式竞争过程，记为 M(Miners)。

(3) 代表节点：在特定算法中会选举节点作为代表来参与共识过程并竞争记账权，记为 D(Delegates)。

(4) 记账节点：通过共识过程竞争和选出记账节点，记为 A(Accountants)。

如图 4-3 所示为区块链过程的基础模型中的 4 个具体步骤：选主(Leader Election)、造块(Block Generation)、验证(Validation)和上链(Chain Updation)。

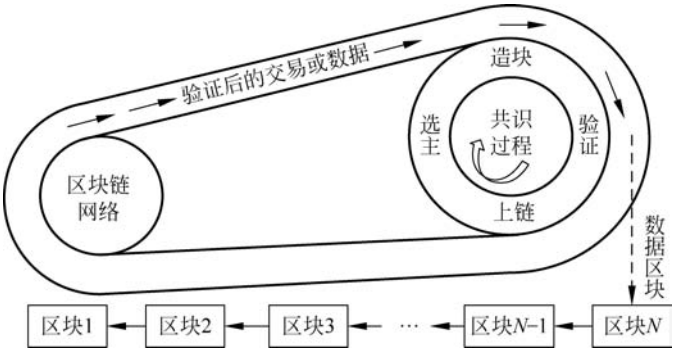

图 4-3 区块链过程的基础模型

1. 选主

选定一个区块的验证者(记账者)是共识过程的第一步也是最核心的一步。选主是通过证明、联盟、随机、选举或者混合等方式从全体矿工节点中选出记账节点的过

程，可以表达为公式，用函数代表共识机制的具体实现方法。

2. 造块

记账节点将交易打包到一个区块中，并将生成的新区块广播给全体矿工节点（或代表节点）。这些交易或者数据通常根据交易费用、交易优先级、等待时间等多种因素综合排序后，依序打包进入新区块。

3. 验证

矿工节点/代表节点收到广播的新区块后，将各自验证区块内数据的正确性和合理性。如果新区块获得大多数代表节点的认可，则该区块正式作为下一个区块部署到链上。

4. 上链

由记账节点将新区块添加到主链最末端，与父区块通过哈希值相连，形成一个新的记录完整账本数据的区块链。如果主链存在分叉，则需要根据相关共识规则，判别并选出一条合适的分支作为主链。如果把共识过程看作一个黑箱，那么输入的是数据或交易信息，输出的是封装好的数据区块以及更新了新区块的区块链。

4.4.2 共识算法的分类

共识算法有非常多不同的分类依据，且不同分类之间又有所交叉，很容易让人觉得混乱，这是分布式共识学习早期非常正常的现象。我们可将共识算法的分类作为了解共识算法的工具手段，更重要的是了解每一种共识算法的思路（选主策略、容错类型、具体步骤等）。

1. 基于选主策略的分类

根据选主策略，可以大致将区块链共识算法分为选举类、证明类、随机类、联盟类以及混合类。

1）选举类共识

矿工节点在每一轮选举过程中通过投票的方式选出本轮的记账节点（因此也叫投票类共识），获得半数以上选票的矿工获得记账权。常见于传统分布式一致性算法，例如 Paxos 和 Raft。常用拜占庭容错共识算法也是一种选举类共识算法，只是 PBFT 需要满足的是 1/3 的容错率。投票式共识的局限性是，如果参与人数太多，那么投票过程会很慢。此外，如果区块链系统的共识节点是一个开放的网络，不断有共识节点加入和退出，投票式共识也面临挑战。

2）证明类共识

矿工节点需要证明自己具有某种特定的能力，证明的方式通常是竞争性地完成某

项难度很高的任务，在竞争中胜出的矿工节点将获得记账权。例如，PoW 和 PoS 分别是基于矿工的算力和权益来完成随机数搜索任务。

3）随机类共识

矿工节点随机决定每一轮的记账节点。此类算法现在的应用比较少见，例如 Algorand 和 PoET。

4）联盟类共识

矿工节点首先基于某种特定方式选出一组代表节点（Delegates），然后代表节点轮流或抽签作记账节点，例如 DPoS 共识算法。

5）混合类共识

混合类共识采用多于一种共识算法来选择记账节点，例如 PoW＋PoS 混合共识、DPoS＋BFT 混合共识。

2. 基于容错类型的分类

区块链共识算法也可分为拜占庭容错和非拜占庭容错两类。

1）拜占庭容错共识

拜占庭容错共识，即能够处理拜占庭故障的共识算法，例如最典型的建立在拜占庭将军问题上的 PBFT 算法和 PoW 算法。

2）非拜占庭容错共识

非拜占庭容错共识，即不能容忍或处理拜占庭故障的共识算法，这类算法只能容忍故障-停止或者故障-恢复等普通的崩溃故障。其中最著名的是 Paxos 和 Raft。

3. 基于部署方式的分类

根据区块链的部署方式，可以把共识算法分为公有链共识、联盟链共识和私有链共识。

1）公有链共识

公有链共识适用于公有链系统，这类共识去中心化程度高，人人都可参与，因此需要能解决包括恶意攻击在内的拜占庭故障，技术效率偏低。

2）联盟类共识

联盟类共识适用于联盟链，去中心化程度低于公有链共识，但技术效率远远高于公有链共识。例如超级账本 Fabric 使用的 PBFT 共识算法。

3）私有链共识

私有链共识适用于私有链的共识算法，通常是完全中心化的，即由一个中心节点来提出共识建议，其他节点按照这个建议执行。例如经典的 Paxos 和 Raft 共识算法。

图 4-4 所示为区块链共识算法的分类。

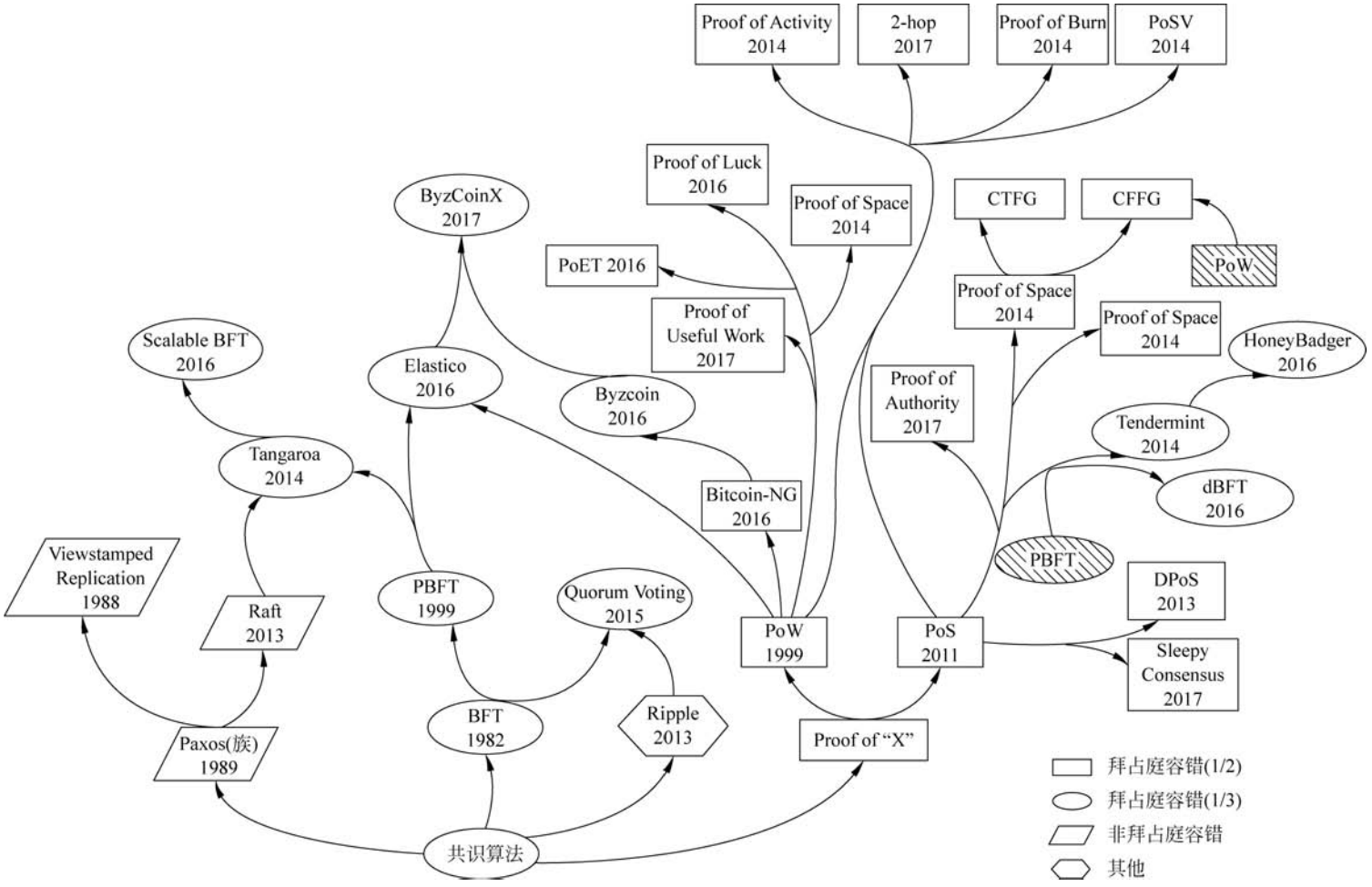

图 4-4　区块链共识算法的分类

4.4.3 主流的区块链共识算法

1. 工作量证明

PoW 是历史最悠久和迄今为止最安全可靠的共识算法，也是比特币系统采用的共识算法。同时，PoW 也是一种应对服务与资源滥用或是阻断服务攻击的经济对策。它的本质是利用工作方和验证方所需工作量的不对称性，以耗用的时间、设备与能源作为担保成本，来确保服务与资源用于满足真正的需求。工作量证明的主要特征是客户端需要花不少时间来做有一定难度的工作，然后得出一个结果；相反地，验证这个结果是非常容易的。它最常采用的技术原理是哈希函数。如比特币的工作量证明（挖矿）就是去找到一个 SHA256 哈希值的输入值。

简单来说，工作量证明就是一个抽签类共识。工作量证明的主要特征是计算的不对称性，节点需要做一定难度的工作来得到一个结果，而验证方很容易通过结果来检查节点是不是做了相应的工作。

这类算法的核心思想实际上是所有节点竞争记账权，而对每一批次的记账（或者说，挖出一个区块）都赋予一个“难题”，要求只有解出这个难题的节点挖出的区块才是有效的。同时，所有节点都不断地通过试图解决难题来产生自己的区块，并将自己的区块追加到现有的区块链之后，但全网络只有最长的链才被认为是合法且正确的。

具体来说，每个矿工节点不断猜测一个随机数，利用该随机数，结合原有交易数据，使得产生的块中的哈希值满足一定条件。由于哈希计算是一个不可逆的过程，所以除了反复猜测随机数进行计算验证外，没有有效的方法能够逆推出符合条件的随机数。在比特币系统中，可以通过调整计算出来的哈希值所需要满足的条件来控制计算出新区块的难度。

通过控制区块链的增长速度，工作量证明还保证了若有一个节点成功解决难题完成出块，该区块能够以更快的速度（与其他节点解决难题的速度相比）在全部节点之间传播，并且得到其他节点验证的特性；这个特性再结合它所采用的“最长链原则”的评判机制，就能够在大部分节点都是诚实（正常记账，认同最长链原则）的情况下，避免恶意节点对区块链的控制。这是因为在诚实节点占据了全网 50％以上算力时，可以预见，当前最长链的下一个区块很大概率也是诚实节点产生的，并且该诚实节点一旦解决了“难题”并生成了区块，很快就会告知全网的其他节点，而全网的其他节点在验证完该区块后，便会基于该区块继续解决下一个难题并生成后续的区块，这样一来，恶意

节点很难完全掌控区块的后续生成。

总的来说，工作量证明共识算法有以下的优点。

(1) 架构简洁，容易理解。

(2) 攻击的成本非常高昂，难以实现。这是由于要获得多数节点承认，攻击者必须投入超过总体一半的运算量(>50%的攻击)，才有机会篡改结果。

(3) 在一定程度上保证了公平性，投入的算力与获得打包权的概率成正比。

虽然 PoW 是业界公认安全可靠和使用最广泛的共识算法，在几乎完美地实现了诸如比特币的加密货币的发行和流通的同时，保障了系统的去中心化。但与此同时，其计算开销和能源消耗(主要是电力消耗)一直饱受诟病，同时矿池不断增大规模而引发的“中心化问题”也争议不断。

2. 权益证明

权益证明(Proof of Stake)也称凭证类共识，这类算法引入了“凭证”的概念，根据每个节点的属性(持币数、持币时间、可贡献的计算资源、声誉等)，定义每个节点出块的难度或者优先度，并且取凭证排序中最优的节点，或者取凭证排序中比较靠前的小部分节点进行加权，随机抽取某个节点进行下一段时间的记账出块。

这类型共识算法在一定程度上降低了整体的出块开销，同时能够有选择地分配出块资源，即根据应用场景选择“凭证”的获取来源，是一个改进的方向。然而凭证的引入提高了算法的中心化程度，有可能造成“贫者愈贫，富者愈富”的马太效应。

3. 委托权益证明

委托权益证明(Delegated Proof of Stake，DPoS)是紧接着 PoS 提出的共识算法，弥补了 PoS 算法中拥有记账权益的参与者未必希望参与记账的缺陷。DPoS 共识算法的基本思路类似于“董事会决策”：节点可以将其持有的权益(股份)授予一个代表，获得票数最多且愿意成为代表的前 N 个节点将进入董事会，轮流对交易进行打包结算，并且签署(即生产)新区块。如果说 PoW 和 PoS 分别是“算力公平竞争”和“权益角逐”的话，则 DPoS 可被认为是“民主集中式的记账方式”。

4. 实用拜占庭容错

实用拜占庭容错(Pratical Byzantine Fault Tolerance，PBFT)算法由 Miguel Castro 和 Barbara Liskov 于 1999 年提出。

PBFT 算法解决了之前 BFT 算法容错率较低的问题，它可以在恶意节点少于三分之一的情况下，保证系统的正确性(避免分叉)。且它降低了算法复杂度，与原始的 BFT 算法相比，其算法复杂度从指数级降低到了多项式级，这使得 BFT 算法可以实

际应用于分布式系统。

PBFT 算法的正常运行有 5 个步骤：请求(Request)、预备(Pre-prepare)、准备(Prepare)、确认(Commit)和回复(Reply)，如图 4-5 所示为 PBFT 算法的典型案例。

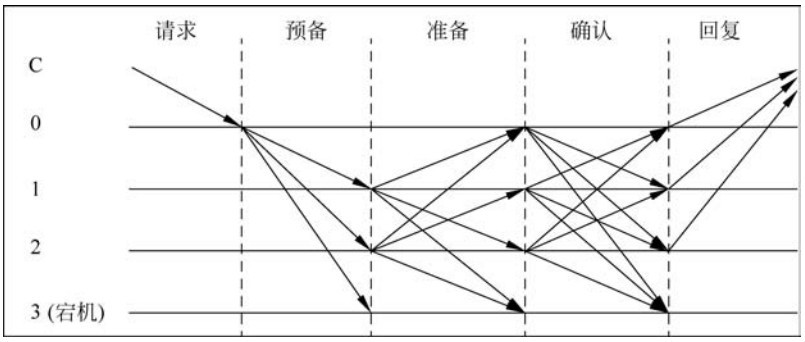

图 4-5　PBFT 算法的典型案例

步骤 1：请求。请求端 C 向主节点 0 发送请求信息＜REQUEST，o，t，c＞，其中，o 为操作，t 为时间戳，c 为客户端编号。在示例中，为客户端 C 发送请求到主节点 0。

步骤 2：预备。主节点 0 向所有副本节点广播预备消息＜PRE-PREPARE，v，n，d，m＞这一步的目的是让每个节点都获取原始消息。其中，v 为视图编号，n 是主节点为该请求分配的编号，d 是原始信息摘要，m 是客户端发送的原始请求信息。在示例中，主节点 0 收到 C 的请求后进行广播，扩散至备份节点 1、2、3。

步骤 3：准备。备份节点 i 收到并验证通过预备信息后，立即进入准备阶段并向其他节点广播准备消息＜PREPARE，v，n，d，i＞，其中，v、n、d 与预备阶段相同；主节点和所有备份节点在收到准备消息后验证其有效性，如果验证通过，则将消息写进日志(Blog)。

在示例中，备份节点 1、2、3 收到消息后，记录并再次广播。路径为 1→0→2→3，2→0→1→3，而 3 因为宕机无法完成广播。

步骤 4：确认。当 PREPARED＜m，v，n，i＞为真时，备份节点 i 将确认消息＜COMMIT，v，n，$D(m)$，i＞ 向除自己以外的其他节点广播，其中，$D(m)$为 m 的信息摘要。所有备份节点在收到确认消息后，验证其有效性，若有效，则写入日志。

在示例中，若节点 0、1、2、3 在准备阶段收到超过一定数量的相同请求，则进入确认阶段，记录并广播确认请求。

步骤 5：回复。当＜COMMITED-local (m，v，n，i) ＞为真时，系统将按序号依次执行请求。执行结束后，备份节点向客户端发送回复信息＜REPLY，v，t，c，i，r＞，其中，v 为视图编号，t 为时间戳，c 为客户端编号，i 为备份节点编号，r 为操作结果。客

户端 C 等待来自 $f+1$ 个不同备份节点的相同响应，这些消息需要具备验证签名、相同的时间戳 t 和操作结果 r，才能被视为正确有效（因为失效的备份节点不超过 f 个）。

在示例中，若节点 0、1、2、3 在确认阶段判断确认为真，或者说收到一定数量的相同请求，则对 C 进行回复。

解释说明：在准备和确认阶段，在 $2f+1$ 个状态复制机的沟通内节点就要做出决定，这样才可以确保一致性。考虑最坏的情况：假设收到的请求有 f 个来自正常节点，也有 f 个请求来自恶意节点，那么，第 $2f+1$ 个只可能来自正常节点（因为此处限制最多只有 f 个恶意节点）。由此可知，“大多数”正常节点是可以让系统工作下去的。所以 $2f+1$ 这个参数和 $n>3f+1$ 的要求是逻辑自洽的。

5. 混合共识算法

PoW 类共识虽然安全性高，但会浪费大量资源，并且交易非常缓慢，随着以太坊智能合约的兴起也渐渐失去了吸引矿工留下的激励作用。BFT 类算法能很好地解决 PoW 的“历史遗留问题”，但由于成本较低，所以安全性得不到非常高的保障。

为了得到更安全且不会耗费太多资源的算法，产生了一些基于已有共识算法的混合式共识算法。凭证类-拜占庭容错共识算法（PoS-BFP）就是其中一种。

在早期的 PoS 算法中，成功挖出区块的人会向全网广播，但其他见证人无法对此进行确认，不仅如此，他们必须在自己生产区块时才能确认之前的区块。在加入了拜占庭容错算法的新算法中，每个见证人在出块时依然进行全网广播，不同的是其他见证人将在收到广播后立刻对其进行验证，并将签名事实传回出块见证人手里。在全网达到 2/3 确认的瞬间，区块产生，交易成立，这使得交易时间大大缩短。

4.4.4 常见共识算法的比较

常见共识算法的比较如表 4-1 所示。

表 4-1 常见共识算法的比较

算法	关　键	实现途径	优　势	适用场景
PoW	算力	投资矿机	验证方易核验，求验方（持币人）公平竞争	稀缺资源竞争，如拍卖
PoS	所持资源	囤积代币	快速、高效、分散、灵活	大多数交易，分布式云
BFT	信誉与安全性	在自己专业的领域持续做对的事	激励制度完善	选举投票

4.4.5 共识算法实训案例

1. 实训目的

本实训将模拟工作量证明(PoW)算法在区块链的使用。根据之前学习的内容，PoW 算法为所有矿工节点竞争账本记账权的过程，在智谷区块链沙箱平台(http://env.zhiguxingtu.com/)可以模拟此过程。选择进入“分布式共识实训”(如图 4-6 所示)，显示内容如图 4-7 所示。

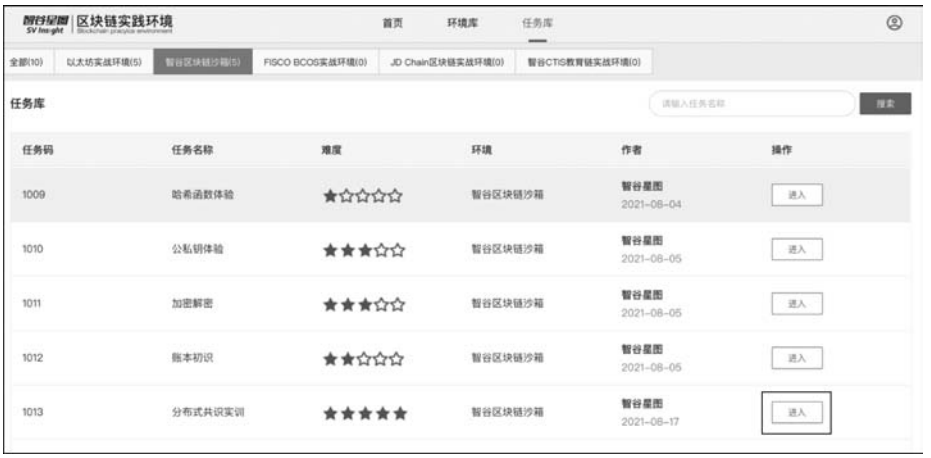

图 4-6 选择“分布式共识实训”

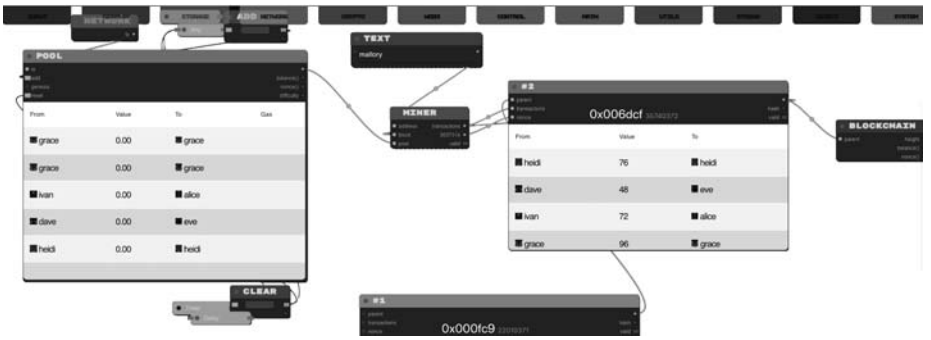

图 4-7 分布式共识实训显示内容

图 4-7 内有诸多功能模块，主要功能如下所述。

(1) POOL 模块模拟每个节点内部交易池的生成内容，在实训平台中已经将 POOL 模块内部的交易动态制作为动态生成，大家可以观察到模块内交易会动态改变，可以直接引用此模块作为矿工节点交易池使用。

（2）MINER 模块用于竞争账本记账权，此模块会不断生成哈希值以匹配新区块的要求。在 MINER 模块中有诸多属性，包括 address、block 等，address 代表 MINER 模块竞争到新区块的奖励地址，block 代表选择要竞争的新区块，transactions 代表新区块中存储的交易数据，不断变化的数字代表竞争哈希的输入值。

（3）BLOCK 模块：如图 4-7 中的“＃2”为 Block 模块，“＃2”代表高度为 2 的区块，在实训界面还有“＃1”的区块，两个区块通过 parent 输出相连从而形成“区块链”。另外，transaction 代表区块中要存储的交易，nonce 表示匹配的竞争哈希值。接下来开始操作模拟 PoW 竞争记账。

2. 实训步骤

步骤一：设置 MINER 模块 Properties 的 difficulty 为 3，如图 4-8 所示。

图 4-8　设置挖矿的难度示例

设置 BLOCK 模块的 Properties 的 difficulty 为 3，并且设置为允许挖矿，如图 4-9 所示。

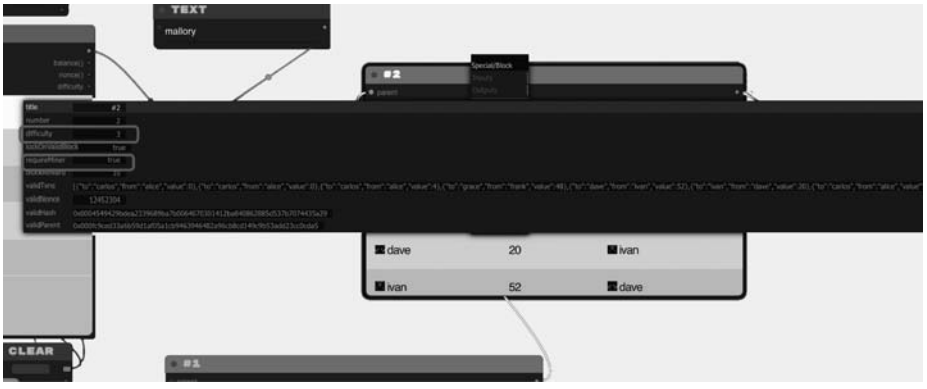

图 4-9　设置区块的难度和允许挖矿设立

步骤二：复制 MINER 模块和 BLOCK 模块生成两组竞争账本的对象，并按照“＃2”的模式连接，如图 4-10 所示。

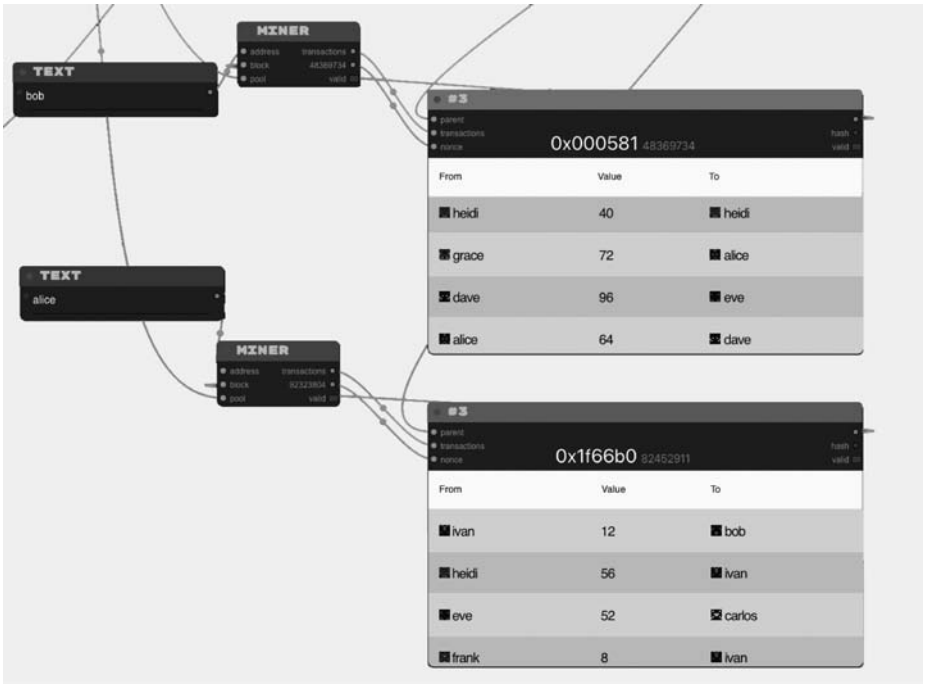

图 4-10　生成两组竞争账本的对象示例

需要注意的是，新区块的 parent 输入为“＃2”的输出才能形成链，并且两个 MINER 的 pool 都要与 POOL 模块相连，如图 4-11 所示。

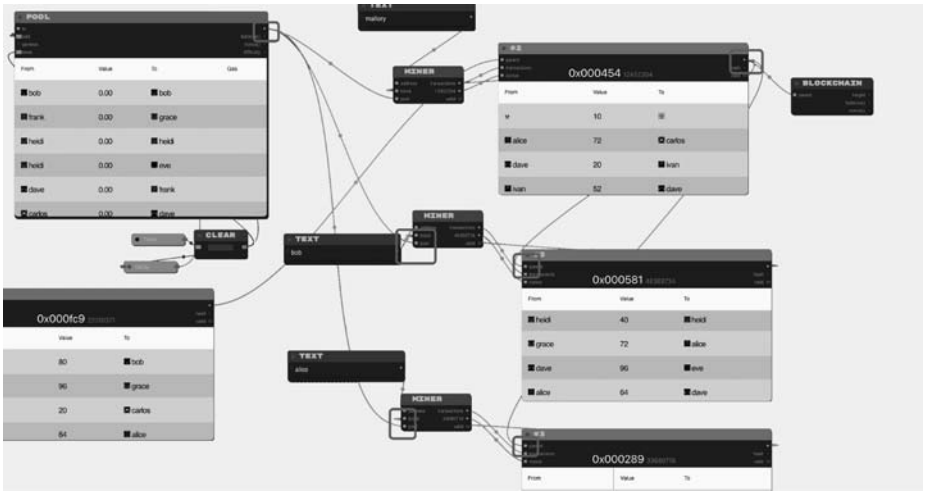

图 4-11　竞争账本的设置示例

步骤三：确认 BLOCK 模块的 properties 的 difficulty 为 3，并且已允许挖矿。两个矿工节点将不停地进行运算，匹配区块难度。如图 4-12 为一个矿工节点先算出了新区块“＃3”，这个区块将被其他节点背书形成共识。

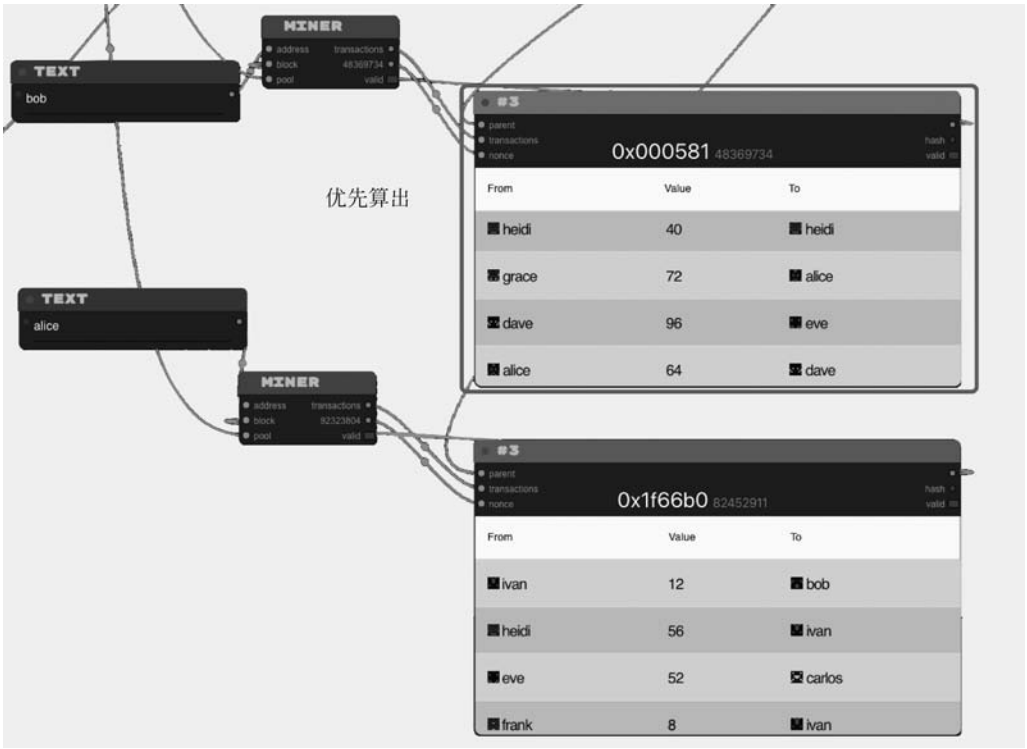

图 4-12　新区块“＃3”产生示例

3. 实训总结

使用沙箱平台可以模拟 PoW 竞争的方式，以上实训为模拟高度为 3 也就是“＃3”区块的生成示例，大家也可以继续模拟更多高度的竞争，加深对 PoW 共识算法的理解。同时，关于实训的内容，大家也可以思考如下两个问题：

（1）没有将两个 MINER 模块连接为一个 BLOCK 模块的具体原因是什么？

（2）什么情况下，竞争产生的新区块会被共识？矿工竞争计算得出的新区块最后都会如何处理？