第 5 章

区块链 3.0 与超级账本

区块链技术是近几年各行业关注的热点，但区块链技术被很多人诟病的地方在于其实用性和落地，为了更好地将区块链技术应用起来，而不仅仅局限于数字货币和智能合约，目前金融、农业、教育、物流等领域都在积极开展区块链的研究，新的发展方向被称为区块链 3.0。它是继以比特币为代表的区块链 1.0 和以以太币智能合约为代表的区块链 2.0 之后的第三次区块链技术革新。本章将剖析各类区块链 3.0 项目的主要创新点，介绍被区块链 3.0 广泛采用的 DPoS 共识机制、分片技术、超级账本等技术。

5.1 区块链的演进路线

随着区块链技术的快速演变，新的技术和应用解决方案在不断出现，人们根据区块链的应用范围和处理速度将区块链的应用范围划分成了 4 个阶段，分别称其为区块链 1.0、2.0、3.0 和 4.0。如图 5-1 所示，以比特币为代表的数字货币能够支持任何时间和地点的快速跨国支付，实现了可编程货币，被称为区块链 1.0；以以太坊为代表的智能合约将区块链技术的应用范围扩展到其他金融领域，实现了可编程金融，称为区块链 2.0；以 EOS 和超级账本为代表的高速处理能力可以将区块链技术进一步应用到公证、仲裁、审计、物流、物联网等其他领域中来，实现了可编程社会，称为区块链 3.0；还有一批以互联价值(InterValue)、哈希图(Hashgraph)、纳尔图(Nerthus)、ALZA、Galaxygrap 为代表的公司，目前正尝试以全新的角度和理念去推进区块链技术的发展，可在交易吞吐量、可扩展性上实现质的飞跃，从而进一步支撑区块链作为某个行业的基础设施，并形成基于区块链的完善生态体系，将广泛而深刻地改变人们的生活方式和工业生产方式，被称为区块链 4.0。但是受限于底层协议的性能、适用范围和稳定性，目前区块链 4.0 还处于早期探索阶段，因此本章重点介绍区块链 3.0 的主要代表。

区块链技术的第一种形式，现在被称为区块链 1.0，由中本聪(Satoshi Nakamoto)于 2009 年提出并发布。区块链是比特币的核心组件，用作数据存储，即公共分类记账。区块链 1.0 最显著的成功是解决了“双重花费”的缺陷，并实现了去中心化的交易。比特币的出现第一次让区块链进入了大众视野，而后产生了莱特币、以太币、狗狗币等“山寨”数字货币。

可编程货币的出现，使得价值在互联网中直接流通成为可能。区块链构建了一种全新的、去中心化的数字支付系统，这个随时随地进行货币交易、毫无障碍的跨国支付以及低成本运营的去中心化体系，强烈地冲击了传统金融体系。

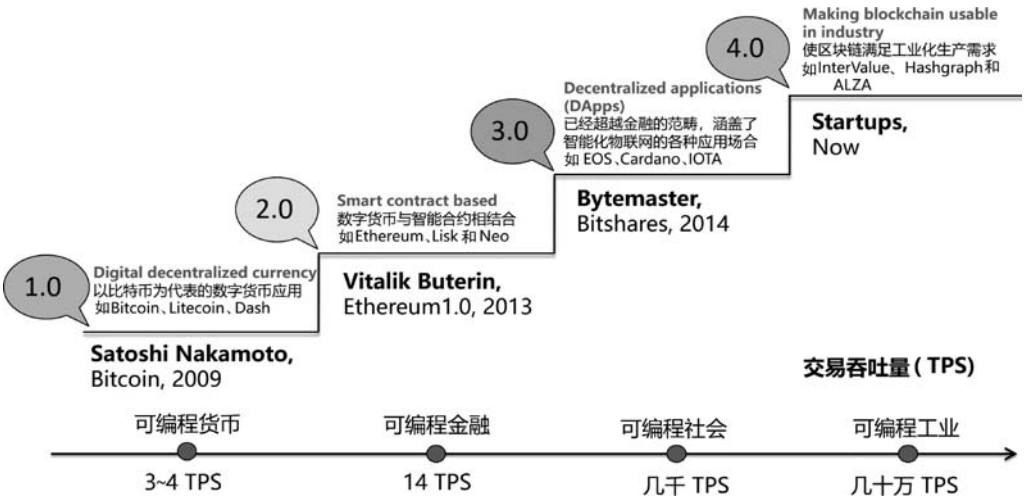


图 5-1　区块链的演进路线

区块链 2.0 的诞生是由于开发人员意识到该技术具有超越货币的应用，于是以太坊应运而生。以太坊平台的开发允许通过组合数据存储和智能合约来部署去中心化的应用程序，这些应用涵盖的范围从社交媒体网络到金融应用。区块链 2.0 面临的最重要的问题是易用性，例如使用以太坊，区块链上的每个节点都必须实时运行智能合约，这会占用大量的网络资源，也会大大影响交易速度；另外，智能合约从开发到发布到网络中需要一定的技术基础和开发周期，换句话说，开发一个功能完善的去中心化应用是很复杂的。

有人认为在比特币和以太币之后，区块链 3.0 的时代需要更高速的交易速度、更多的落地应用。目前基于区块链技术的应用层出不穷，尤其 2018 年更是区块链技术爆发的一年。在比特币和以太坊的讨论中，最大的问题在于它们的交易速度。2017 年诞生的基于以太坊 CryptoKitties 应用，也叫区块猫，一度将以太坊网络拥堵至瘫痪。在以太坊中，每笔交易处理的时间和花费的交易费有关，一旦交易时间过长或者网络堵塞就要花费更多的手续费，这也暴露了区块链 2.0 技术的一个致命问题。另外，作为用户，需要更加实用方便的 DApp，甚至期望能够取代现有的应用。

目前的区块链 2.0 技术无法处理全球规模的海量交易，对于区块链技术有了更多的要求，截至目前，有几种不同的方法可以弥补现有区块链技术的缺点，包括闪电网络、网状网络和块网格结构。区块链的未来是一个分散的互联网，结合了数据存储、智能合约、云节点和私有链网络，所以区块链 3.0 面临的挑战还有很多。区块链技术从 2.0 到 3.0 需要解决的问题主要有如下几方面。

1. 更快速安全的交易

比特币每秒处理的交易最快7笔左右，以太坊是平均每秒30～40笔，而普通信用卡每秒能处理几千笔交易，支付宝在高峰期每秒达到几万笔交易，所以交易速度的极大提升是区块链3.0最重要的方向。网络去中心化的一个牺牲是处理速度的降低，但如何在分布式的前提下提升速度？目前有一些很有效的方向，如**分片技术**（sharding）、**石墨烯技术**（graphene）以及**闪电网络**（lightning network）等。

除了速度，分布式网络的安全性也是区块链3.0的关键。黑客会利用智能合约的漏洞进行攻击，因为互联网的应用可以在底层并不稳定的时候进行开发，很多互联网应用对系统的可用性要求不高，即便发生错误所带来的损失并非不可接受。但是区块链上或者说价值网络上的应用直接关系到金钱、信用、所有权、认证、资产、控制权等，远比互联网承载的信息更有价值，一旦发生错误带来的损失也是不可同日而语的。

北京时间2016年6月17日发生了一起攻击事件，在区块链历史上留下沉重一笔。由于其编写的智能合约存在重大缺陷，区块链业界最大的众筹项目The DAO（被攻击前拥有1亿美元左右资产）遭到攻击，导致300多万以太币资产被分离出The DAO资产池。最后，不得已将以太坊网络做了软分叉（关于分叉的概念在第7章有具体的讲解）。软分叉将从块高度1 760 000开始把任何与The DAO和child DAO相关的交易认作无效交易，以此来阻止攻击者在27天之后提走被盗的以太币。

2. 更多实用的DApp

如今许多人都在质疑，除了炒作和投机，区块链还有多少真正的价值。一方面，这项技术的处理速度太慢，无法大规模应用。另一方面，挖矿十分耗电，现在爱尔兰用在挖比特币的电力比全国的日常家庭电耗还多。有人对区块链的真正价值感到困惑。事实上很明显地可以看出，目前的区块链应用仍然处于一个拿着钥匙找锁的状态，用户和投资者实际上并不很了解区块链到底能做什么，甚至都不知道这是什么，正如最开始的互联网一样，成为互联网公司最大的目的就是上纳斯达克交易。

区块链3.0旨在将区块链应用到目前的领域中，如金融领域，虽然当前很火的互联网金融浪潮在全球范围改变了传统金融业务模式，但直销银行、互联网保险、互联网券商等平台业务的重点还是在于渠道的争夺、经营模式的改变，而区块链技术有望将金融业的下一个发展阶段推向更加接近金融本质的层面——信用。理论上，在技术识别能力足够的情况下，它能让交易双方无须借助第三方信用中介开展经济活动，从而实现全球低成本的价值转移。

虽然目前已经出现很多基于以太坊、基于区块链的应用，但实际上真正做到代替性的产品还没有出现。一个新技术的出现，需要市场和用户慢慢地认识，就像20世纪90年代的互联网，有无数人认为是泡沫，直到今天人们才看到互联网给生活带来的巨大改变。区块链也一样，在3.0时代，人们希望看到更多实用的DApp，让区块链走进人们的日常生活中。比特币引入的全新的去中心化组织和共识机制，已经成功地衍生出了几百项不可思议的创新，这些创新很有可能影响社会的不同领域，从分布式系统科学到金融、经济、货币、中央银行、企业治理等。很多原本要求中心机构或组织作为权威或信用点进行控制的人类行为，现在

都可以实现去中心化了。区块链和共识系统的发明将大大降低大型系统的组织和协调成本，同时消除了权力集中、腐败和逃避监管的隐患。

3. 更便捷的应用开发

虽然以太坊的诞生使用户能够在其上部署智能合约，但是智能合约能够完成的工作还是有限的，有时一个 DApp 需要几个智能合约，而且在以太坊上部署智能合约需要用户花费手续费，功能越复杂，所需要的手续费越多；另外，智能合约的编写也需要开发人员掌握以太坊开发语言。目前区块链行业诞生了“公链”，它类似以太坊，方便用户在上面开发 DApp。目前公链比较著名的有 EOS、ADA、AE 和 Zil 等，公链也被认为是最符合区块链 3.0 的一类技术。

实际上公链能否成为区块链 3.0 还有待时间的验证，公链的很多性能在实际运行时能否达到预期是用户考察的关键。

4. 创新性技术

一个区块链项目是否有价值，与它的底层技术有很大关系。目前排名前十的数字货币，有五个都是因为有自己的创新技术。而也正是因为这些创新技术的出现，区块链技术能够与越来越多的行业结合，例如区块链＋人工智能、区块链＋大数据。

区块链技术是非常有前景的，通过技术特性的落地，解决现实中的问题，把传递信息的互联网和传递价值的区块链应用结合起来，可以极大地推动人类社会的进步。目前，区块链从业者从底层核心技术实现，到链上应用，再到各类落地场景应用等各个层面，都开展了全方位的探索。

总而言之，区块链 3.0 一定是高效实用的，是区块链技术实现落地应用的关键时间点。目前也有很多具有突破性技术的区块链平台出现，如 EOS、ADA 和 Zil。下面详细介绍这三个区块链平台。

5.2　商用操作系统

商用操作系统(enterprise operation system，EOS)是区块链奇才 Daniel Larimer(被称为 BM，即 Bit Master)领导开发的类似操作系统的区块链架构平台，旨在实现分布式应用的性能扩展。简单讲，通过 EOS 这一区块链操作系统，用户可以快速开发出属于自己的 DApp，相比于以太坊的智能合约，EOS 能使用户更容易地进行开发。有人把以太坊比作计算机的 CPU，而把 EOS 比作 Windows 系统，程序运行在 CPU 上，EOS 更进一步为用户提供开发 DApp 的工具。EOS 提供了一个开源软件，使用的区块链架构可扩展性更强，消除了用户费用，并允许轻松部署 DApp。

EOS 提供账户、身份验证、数据库、异步通信以及在数以百计的 CPU 或群集上的程序调度。该技术的最终形式是一个区块链体系架构，该区块链每秒可以支持数百万个交易，同时普通用户无须支付使用费用。EOS 最终的目标是达到每秒处理百万次请求的性能。EOS 创始人 Daniel Larimer 以前开发过的两个产品 BitShare 和 Steemit 现在每秒能够处理

2000 请求，日平均处理 60 万次请求。虽然 EOS 将会使用与 Steemit 和 BitShare 相同的架构——石墨烯架构和委托股权证明法。但是许多人认为，EOS 的性能绝对会大大超越 BitShare 和 Steemit。目前，EOS 已经主网上线，正在逐步更新代码，解决上线后的问题。

EOS 还有一个特点是每次交易不需要手续费，这是相比于比特币和以太币以及其他数字货币很大的改变。这也是为什么 EOS 能在短短时间内成为人们关注的焦点，一跃成为市值第五的数字货币。比特币和以太币通过手续费保证本身的价值，那么没有手续费的 EOS 怎么保证其自身的价值呢？EOS 上开发 DApp，需要用到的网络和计算资源是按照开发者拥有的 EOS 的比例分配的。当拥有了 EOS，就相当于拥有了计算机资源，随着 DApp 的开发，可以将手里的 EOS 租赁给别人使用，单从这一点来说 EOS 也具有广泛的价值。

不同于以太坊的 PoW 机制，EOS 的共识机制称为 DPoS，它规定系统由 21 个节点负责运行，而每次只需一个节点处理交易产生区块，各个节点不是竞争关系，而是合作关系。每个节点循环产生区块，但每轮的顺序是不一定的，而且每轮有新节点产生，有老节点淘汰，但总数需要维持在 21 个。这样做的最大好处是能大大提高网络处理交易的速度，减少能源浪费。

随着 EOS 的上线，随之而来的一个重要问题是 RAM 资源的获取。RAM 是 EOS 的内存资源，而不是一种代币或通证。RAM 在 EOS 软件平台上对应内存数据库资源。作为 DApp 开发者，RAM 是一项宝贵资源，数据库记录需要消耗 RAM。在 EOS 网络上，大量的操作也都需要消耗 RAM 来存储数据，例如创建一个 EOS 账号、创建一个 EOS 智能合约、进行 EOS 转账等。

RAM 的分配本来是按照 EOS 持有量分配的，但是考虑到很多持有者并不是开发者，也没有动机买卖，可能会造成有开发需要的开发者因为没有资源而放弃。为了创造流动性，BM 创造了一个 RAM 的交易市场，并且制定了一套玩法，使得 RAM 的流动性和交易性得到了保证。

RAM 的总量为 64GB，非常有意思的一点：当前 RAM 使用率只有 1.70%，但是 RAM 占有率已经接近 86%，很显然，炒币的人已经成功地把 RAM 市场占领了，这也是为何 RAM 价格可以飙升如此之快。

为了保持超级节点的高效运行，节点、RAM、内存总量有上限（以后会扩容），如果要保持区块链数据可以随时存储、修改，就需要这部分数据存储在内存中，而内存的使用需要用户自己去 EOS 系统中购买，不需要的时候再卖给系统，换回 EOS 代币。而随着 RAM 不断地被租用，剩余可用的 RAM 越来越少时，RAM 所需要抵押的 EOS 就会越来越多，也就是说 RAM 的价格会越来越贵。

开发 DApp 需要抵押 EOS、超级节点投票也需要抵押 EOS、连开户都需要抵押 EOS……什么都要抵押，这样说来，EOS 实在是太紧缺了。这里的 EOS 指的是 EOS 代币，下面来详细介绍 EOS 代币。

5.2.1　EOS 简介

根据 EOS 官网的介绍，EOS 是由 Block.one 公司主导开发的一种全新的基于区块链智能合约平台，旨在为高性能分布式应用提供底层区块链平台服务。EOS 代币目前是 EOS 区块链基础设施发布的基于以太坊的代币，主要有三大应用场景：带宽和日志存储（硬盘）、计算和计算储备（CPU）、状态存储（RAM）。EOS 在中国还有一个昵称，叫作“柚子”，是 EOS 英文发音的谐音。上面说过，EOS 的市值已经跃居数字货币市场的第五名，EOS 的 ICO 持续一年，从 2017 年 6 月到 2018 年 6 月，总发行量 10 亿个。

下面简单介绍与 EOS 一些的相关概念。

1. Block.one

Block.one 是一家注册在开曼群岛的区块链开发公司，这家公司发起了 ICO，募集的钱由 Block.one 公司保管，用于 EOS.io 的开发和运营。EOS 的 ICO 为期一年，一年之后即软件开发完毕之后，Block.one 不负责软件的上线，而是由各个节点首先上线。也就是说，Block.one 只负责技术开发和维护，但是对于用户的 EOS 能否从代币变成数字货币，Block.one 不保证。

2. EOS.io

EOS.io 是由 Block.one 公司负责开发的一套开源的、去中心化软件，软件的名字就叫 EOS.io software。任何人都可以使用这套免费开源的代码，去创造自己的公链。每个区块生产者都需要运行 EOS.io，并且上线后软件还会不断更新。

3. EOS Token

通称 EOS 代币，主网上线之前钱包里的 EOS 都是代币。按照官方的解释，EOS 代币是没有任何价值的。存储 EOS 代币的以太钱包需要注册，生成 EOS 账户的公钥和私钥，以便以后在 EOS 区块链网络上生成 EOS coin。

4. EOS Coin

EOS platform 和 EOS 区块链网络运转之后，要把拥有公钥和私钥的 EOS Token 转移到 EOS 区块链上，如此才能变成 EOS coin，届时，EOS 才是真正的 EOS，到这里，你才算真正地拥有了 EOS。所有的价值也是通过 EOS Coin 才能展现出来。

5. EOS Community

EOS Community，即 EOS 社区。EOS 的价值主要取决于 EOS 公链的发展，EOS 公链的发展则取决于 EOS 社区，待到 EOS 网络运转之后，就要看有多少人采用 EOS.io 开发自己的公链，用户越多，社区越大，价值越大，所以 EOS 什么时候能达到或超过 ETH 的高度就要看 EOS 社区的发展。

6. EOS block producer

不同于比特币和其他电子货币，EOS 的节点数维持在 21 个，但是一般节点可以成为节点候选者，每次的 21 个节点称为区块生产者（block producer）或者超级节点（super node）。

EOS.io 为了给 EOS 币用户参与的机会，也为了规范区块生产者，EOS.io 给每个持有

EOS币用户投票的权利。一个EOS币可以投30票给节点，根据获得的票数选出21个节点和其他候选节点。

EOS.io架构中区块产生是以21个区块为一个周期，在每个出块周期开始时，21个区块生产者会被投票选出。前20名出块者首先自动选出，第21个出块者按所得投票数目对应概率选出。所选择的生产者会根据从块时间导出的伪随机数进行混合，以便保证出块者之间的连接尽量平衡。EOS.io里预计每3s生产一个区块。任何时刻，只有一个生产者被授权产生区块。如果在某个时间内没有成功出块，则跳过该块。如果出块者错过了一个块，并且在最近24小时内没有产生任何块，则这个出块者将被删除。这确保了网络的顺利运行。EOS是没有手续费的，因此普通受众群体更广泛。

5.2.2 DPoS共识机制

任何共识过程必须回答的问题包括但不限于以下几方面：

（1）谁应该产生下一个更新块应用于数据库？

（2）下一个块何时应该生产？

（3）什么交易应该包括在该块？

（4）协议的变化如何应用？

（5）竞争的交易历史应该如何解决？

共识机制的目标是找到这些问题的解决方案，DPoS共识算法于2014年4月由Bitshares的首席开发者Daniel Larimer（现为EOS的CTO）提出并应用。当时Daniel Larimer观察到比特币系统共识算法PoW的一些问题，如矿池导致算力越来越集中、电力耗费过大等。所以他提出了一种更加快速、安全且能源消耗比较小的算法，这就是后来的DPoS。而EOS的目标之一是提高交易速度，所以提出DPoS机制。

DPoS区块生产者是合作关系，在任一个时间点，只有一个生产者有权产生区块选择，而在比特币的工作量证明算法（PoW）中，区块生产者是竞争关系，最终只有一个生产者获胜，其他生产者的算力会被白白浪费。DPoS平均每3s产生一个区块，平均10分钟产生200个区块；而PoW平均10分钟产生一个区块。200∶1在效率上是极大的提升。DPoS交易本身可以作为股权证明：每个交易都包含前一个区块头的哈希值。一轮有21个区块生产者，前20个是自动产生，第21个是根据投票数产生的。DPoS在正常情况下不会产生分叉。如果产生了分叉，有更多生产者支持的分支的长度会增加得更快。区块生产者不可能同时在两个分支上生产区块，一旦被发现就会被投票罢免。

在EOS之前BM已经开发出了两个很成功的区块链项目，它们也是EOS的技术基础，尤其是石墨烯技术。石墨烯可追溯到Daniel Larimer最早创立的Steemit，在创立EOS之前他还创立过BitShare，EOS是Daniel Larimer创建的第三个区块链项目。石墨烯是EOS底层和关键的技术，实际上是一个工具库，如图5-2所示。

图5-2中包括插件库、函数库和石墨烯插件，它们提供了三个顶层模块：钱包模块、观察者（节点）模块和初试区块模块。通过这些基础技术，给用户和开发者提供了一个完整的

系统。同时，石墨烯技术实现了多线程并行工作。另外，石墨烯也是实现 DPoS 的技术基础，而 BM 最初开发石墨烯也是为了解决 PoW 机制处理交易过慢的问题。

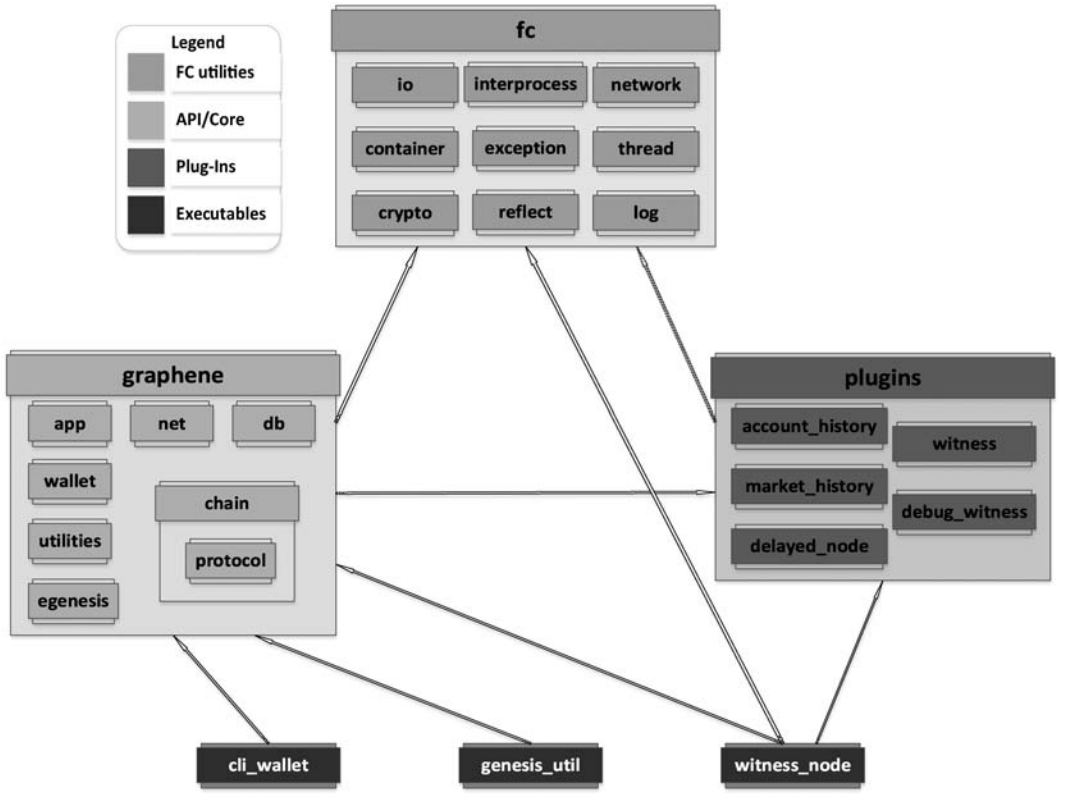


图 5-2　石墨烯库结构

DPoS 共识算法维护的区块链出块者都是 100％在线的，一个交易平均 1.5s 后会被写入区块链中，同时被所有出块节点知晓这笔交易。但是在一些非常情况下，如软件 bug、网络拥塞或恶意出块者出现，区块链可能出现分叉。为了确保一个交易是不可逆转的，等待 15(15/21)个区块确认。根据 EOS.io 软件的配置，在正常情况下 15 个区块确认时间平均需要 45s。在分叉产生的 9s 内，出块节点就可能发现这个分叉可能并警告用户。一个节点观察网络的时候如果发现连续 2 次的丢块事件，这意味着该节点有 95％可能性在区块链的分叉分支上。在连续出现 3 次丢块以后，该节点有 99％的可能性在一条分叉出来的区块链上。据此可以生成一个预测模型，它将利用节点丢失的信息、最近的参与率以及其他因素来快速地警告用户出现的问题。

DPoS 算法和比特币的 PoW 算法有很大区别，在 PoW 算法中，矿工只要发现交易信息就开始打包区块，需要消耗巨大的算力，而且交易确认时间很长。而 DPoS 算法则通过提前选举出可信节点，避免了信任证明的开销，同时生产者数量的减少(21 个)也极大提升了交易确认效率，防止性能差的节点拖慢整个区块链生产速度。DPoS 的时间片机制能够保证

可信区块链的长度始终比恶意分叉的区块链长（恶意节点数量不大于 1/3 总节点数量），例如，如图 5-3 所示，假设在 DPoS 机制中有四个区块生产者，生产区块的顺序是 A—B—C—D—A，节点 B 想自己构造一个分叉链，但是由于每次只有等 C、D 和 A 生产完区块才能产生一个区块，所以始终没有主链长，因为系统只会选择一个最长的链作为主链。

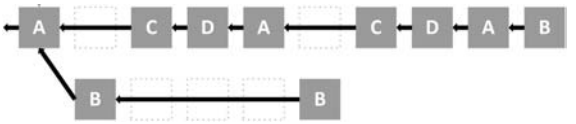


图 5-3　EOS 避免分叉

本质上，DPoS 和代议制民主以及董事会制度一样，都是一种精英制度。但这种精英制度受制于基层的民众。在美国，议员是被全民选举出来；在 DPoS 中，代币持有者至少有权决定见证人，或者说矿工的身份。相比于 PoW 与 PoS，DPoS 反而更加符合去中心化和平等的精神。

目前基于 DPoS 机制的区块链项目除了 EOS，还有 Bitshares 和 Asch。可见，这样一种新的共识机制会被越来越多的开发者关注。

5.2.3　EOS 测试网络搭建

EOS.io 在 EOS 主网上线之前就提供了 EOS 的测试网络，并不断更新，通过测试网络不断地完善 EOS 的技术。任何对 EOS 技术有兴趣的开发人员都可以在 GitHub 上下载测试网络在个人计算机上感受 EOS 的实际运行。本书采用的是 EOS 2.0 的版本，各个版本之间的运行方式是类似的。

运行环境：Ubuntu 16.04。

系统内存：8GB。

存储空间：30GB。

从 GitHub 上下载源代码：

```
git clone https://github.com/EOSIO/eos.git -b DAWN-2018-02-14 --recursive
```

DAWN-2018-02-14 代表的是版本号，读者可以下载最新的版本，只需更改版本号。

等待提示下载完成后，会自动生成一个名称为 eos 的文件夹。在 Terminal 中进入 eos 文件夹开始编译：

```
cd eos
./build.sh ubuntu full
```

根据计算机的网络和运行速度，这一步大约需要两个小时。当提示完成之后，就可以进行安装：

```
cd build
```

```
make install
```

当安装好之后，会生成 programs 文件夹，其中包含 eosd、eosc 和 eos-walletd 等子文件夹。同样地，最新的版本生成的文件夹的名称不一样，但功能是类似的。

下面对自己的 EOS 网络进行配置，修改 config.ini 文件，具体方法是：

```
# Load the testnet genesis state, which creates some initial block producers with
the defaultkey
genesis-json = /path/to/eos/source/genesis.json
# Enable production on a stale chain, since a single-node test chain is pretty much
always stale
enable-stale-production = true
# Enable block production with the testnet producers
producer-name = inita
producer-name = initb
producer-name = initc
producer-name = initd
producer-name = inite
producer-name = initf
producer-name = initg
producer-name = inith
producer-name = initi
producer-name = initj
producer-name = initk
producer-name = initl
producer-name = initm
producer-name = initn
producer-name = inito
producer-name = initp
producer-name = initq
producer-name = initr
producer-name = inits
producer-name = initt
producer-name = initu
# Load the block producer plugin, so you can produce blocks
plugin = eosio::producer_plugin
# Wallet plugin
plugin = eosio::wallet_api_plugin
# As well as API and HTTP plugins
plugin = eosio::chain_api_plugin
plugin = eosio::http_plugin
```

修改完后保存，这里面规定了超级节点的名字，如 inita；规定了创世区块的位置。之后进入 program 文件夹，在 Terminal 上输入：

```
cd eosd
./eosd
```

如果没有错误，则会如图 5-4 所示，从中会看到 EOS 开始生产区块，因为是测试网络，生产区块的流程和实际的不同，但也是按照配置文件里的超级节点进行。这时要保持 EOS 网络的运行，再重新打开一个新的 Terminal。

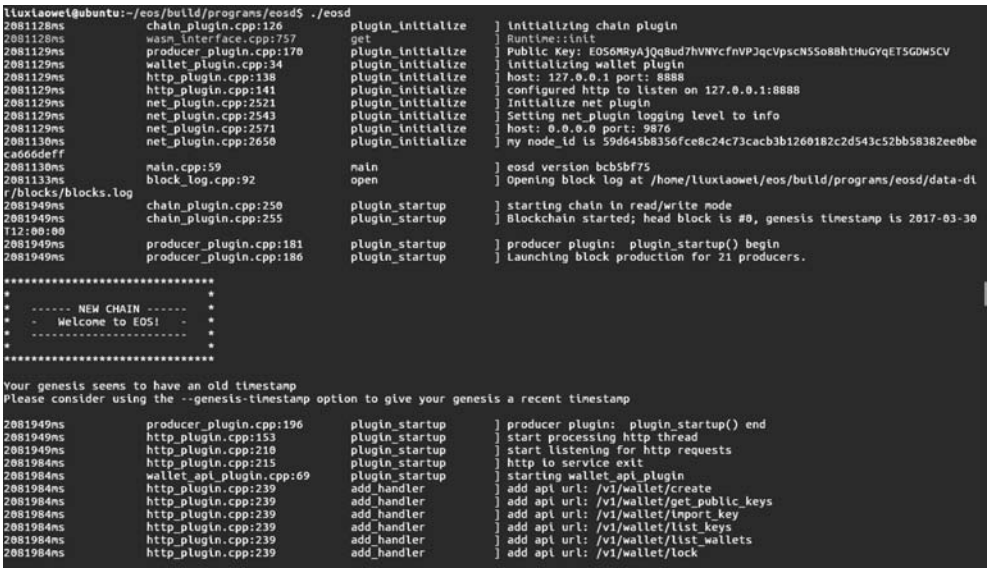


图 5-4 EOS 测试网络开始产块

所有的操作都是基于 eosd 完成的。首先要保持 nodeos 的运行。然后，用下面的命令创建一个钱包。

```
cd build/programs/eosc
./eosc wallet create
```

创建成功后如图 5-5 所示，并会展示私钥。

图 5-5 EOS 测试网络创建钱包

为货币合约创建一个账户 currency，首先生成两组 key，分别对应 OwnerKey 和 ActiveKey。

在 eosc 目录下，把生成的 key 做好备份，两组 key（OwnerKey 和 ActiveKey）分别会有公钥和私钥，操作语句如下：

```
./eosc create key # owner_key
./eosc create key # active_key
```

然后，将两组 key(OwnerKey 和 ActiveKey)的私钥导入钱包，操作语句如下：

```
./eosc wallet import (OwnerKey 的私钥)
./eosc wallet import (ActiveKey 的私钥)
```

运行结果如图 5-6 所示。

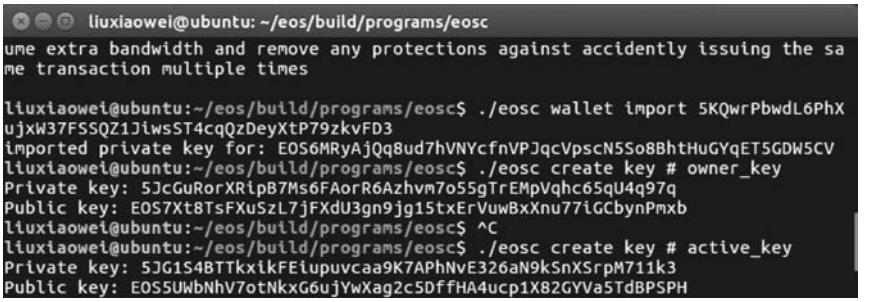


图 5-6　EOS 钱包导入私钥

接下来，用 eosc create account 命令创建账户 currency 并导入两组 key(OwnerKey 和 ActiveKey)的公钥，操作语句如下：

```
./eosc create account eosio currency (OwnerKey 的公钥 ActiveKey 的公钥)
```

使用 get account 命令可以查看 currency 是否已经创建成功：

```
./eosc get account currency
```

图 5-7 所示是创建智能合约的界面。关于 EOS 测试网络的更多操作可以在 https://github.com/EOSIO/eos 查看，EOS 上线以来代码更新速度越来越快，也说明 EOS 项目的价值。通过测试网络也能对 EOS 技术有一个更清楚的了解。

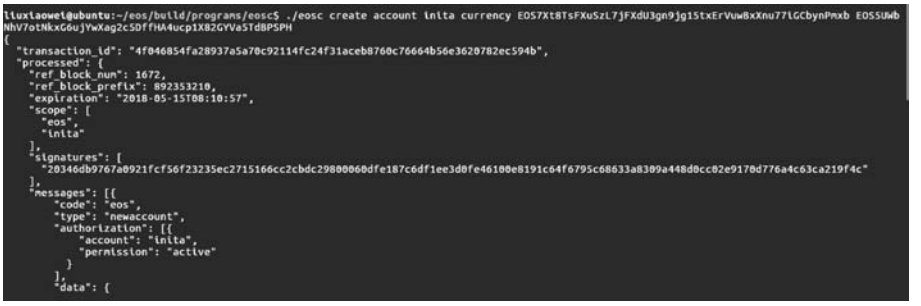


图 5-7　EOS 测试网络创建合约

没有比 EOS 更能完美说明加密货币与 ICO 狂潮的例子。EOS 在过去一年时间融资约 700 万个 ETH，相当于 42 亿美元，创下史上规模最大、耗时最长的 ICO 纪录。EOS 绝对是眼下区块链世界的一艘航空母舰。若拿来与股票市场上科技巨头的 IPO 规模类比，EOS 规

模足可列名史上第三大，还远远超过谷歌、推特等科技巨头。EOS 预计的处理速度能达到百万级别，但是有些人认为这是在牺牲了去中心化的前提下，因为只有 21 个节点参与，并不是真正的去中心化。以目前的技术来看，中心化与处理速度是两个对立的因素，往往节点的分布性越强，交易速度越慢，如最早期的以太币。因为节点之间共识的建立时间影响交易速度，因此便提出了 PoS（权益证明）机制，艾达币就是 PoS 的典型代表。

5.3 艾达币

艾达币（Cardano）是一个开源的、分散的公共区块链和加密货币项目，由其本地加密货币 ADA 推动。目前的基础公链在规模化、交互性、可持续性三方面普遍存在缺陷。Cardano 的哲学是在学习和继承现有基础公链优点的基础上，进行概念和技术的创新，希望能最终解决上述三方面问题，成为更便捷、更高速、更智能的新一代底层基础公链，也就是常说的区块链 3.0。Cardano 不仅是加密货币，也是一个完全开源的区块链平台。其中心思想是通过构建一个分层次的区块链生态系统，将整个体系划分为结算层和计算层两个层次，分别来解决货币和智能合约两个层面的问题。

Cardano 专注于解决当今区块链所面临的 4 个关键问题：

（1）可扩展性。

（2）互通性。

（3）可持续发展。

（4）治理。

本着许多开源项目的精神，Cardano 并没有从全面蓝图开始，甚至没有一个权威的白皮书。相反，它包含一系列的原则、工程实践和探索途径。同时，官方发表了很多基于 Cardano 技术的学术论文，使其具有研究价值。

5.3.1 ADA 简介

ADA 是 Cardano 的代币，中文名称：艾达币。ADA 的发行总量共计 450 亿枚，在 2015 年 10 月到 2017 年 1 月进行了众筹。截至 2018 年 7 月，艾达币的价格稳定在 1.10 元左右，而其 ICO 均价只有 0.0163 元，上涨了近 70 倍！值得注意的是，日本投资者占到了 ADA 众筹总额度的 90%以上，这种筹码过于集中的状况与区块链去中心化的思维是有所背离的，并不利于 ADA 全球化的生态建设，同时，也显现出 ADA 的全球化营销还不够充分。但换个角度来看，这也是一种机遇，在中国、韩国、美国等主流玩家都还没有深度参与的情况下，ADA 就已经成为市值前十的币种，后期随着项目的不断发展，若全球广泛参与进来，币价还有很大的升值空间。

区块奖励每 3.5 分钟发放一次，发放频率参考如下：最初每个区块产生 2000 艾达币，共计 3 744 961 区块；第二阶段每个区块产生 1000 艾达币，共计 3 744 961 区块；第三阶段每个区块产生 500 艾达币，共计 3 744 961 区块。以此类推，以每 3 744 961 区块为单位陆续

减半，以每分钟产生 3 个区块的速率，直到全部释放完，大概需要 24 年的时间。每个区块产生的艾达币 75%用于持有者的奖励，25%作为项目启动后技术开发、生态建设者的奖励。只要参与 Cardano 应用的研发建设，就有可能获得奖励。作为 Cardano 生态的创新激励机制还在开发中，项目启动后会在一段时间内完成。

目前，主流的公链中比较普遍的设计是在一个链上存储各方面的信息。但这样带来的问题是无法满足实际运行中的需求，并且会给未来网络速度拓展、治理优化以及可持续性带来阻碍。Cardano 结算层与计算层分开运行的方式，可以针对不同的分层进行有针对性的部署和升级。针对结算层，可以通过软分叉对数字货币交易中遇到的问题进行升级和换代，而对于计算层，则可以根据 DApp 的运行需求进行针对性的拓展和改良。因此，分层的方式实现了在一个生态内建立清晰、有边界的系统运行秩序，实现更好的可拓展性和交互性。也可以把它简单理解为区域自治的概念，货币和应用程序可以分别根据各自运行特点采用不同的治理策略。

卡尔达诺结算层需要交易费主要有两个原因：

(1) 人们运行卡尔达诺结算层完整节点需要花费时间、金钱和精力来运行协议，为此他们应得到补偿和奖励。在卡尔达诺结算层中与比特币不同的是，当新货币在每个区块被挖出时，交易费用是协议参与者的唯一收入来源。

(2) 为了防止 DDoS(分布式拒绝服务攻击)。在 DDoS 攻击时，攻击者尝试用虚假交易来冲击网络，但如果他必须为每个虚假交易支付足够高的费用，这种攻击形式对于他来说就过于昂贵了。

ADA 的分层与 EOS 的分片技术是不同的概念。分片是同类型链之间的信息交互，而分层则是两条治理理念和治理方式完全不同的链，在同一个生态体系下运行。

一个公链的共识算法相当于它的价值观。无论工作量证明(PoW)、权益证明(PoS)还是 EOS 的委任权益证明(DPoS)，都有各自的优点和缺点，也都是其平台治理价值观的显现。在这方面，Cardano 采用的是独特的**乌洛波罗斯**(Ouroboros)算法，或者叫作权益算法，在下一节做详细介绍，并宣称这是第一个经过“同行评审”并“可证明安全”的股权证明共识算法。

5.3.2　权益证明

之前提到过，传统的比特币和以太币的共识机制通过工作量证明产生一个节点生产区块，其他节点则没有机会，这样就浪费了大量资源。权益证明是为了解决资源浪费问题而提出的。“权益”指的是节点上的地址所拥有的相对价值。“相对价值”指的是卡尔达诺结算层系统中某个节点钱包上的价值除以总价值。

如图 5-8 所示是 Cardano 生产区块的过程。首先，在时间域上，是一个个连接的时隙(epoch)。每个时隙中又分成很多 slot，如 slot1、slot2、…、slot$N$，slot 的个数可以改变，每个 slot 里有一个 slot_leader 负责生产区块。在每个 epoch 中最开始有一个最初始的块(genesis block)，它规定这个 epoch 中每个 slot 的区块生产者，这些内容在前一个 epoch 中

就要完成。所以权益证明要解决的问题是怎么选出每个 slot_leader。

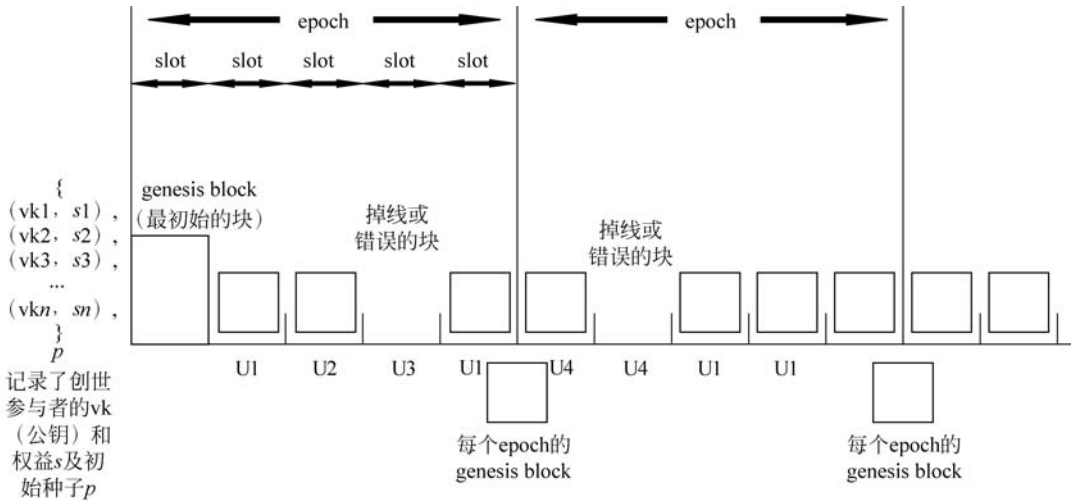


图 5-8 Cardano 生产区块

数字货币共识机制一个基本理念是各个陌生节点之间建立信任，以及各个节点间互相协作。但是问题就是如何在陌生节点间快速建立信任，并尽可能减少通信，PoW 机制采用 VSS(verifiable secret sharing)机制。举例来说，如图 5-9 所示，A 和 B 之间相互不认识，A 先给 B 发送一个随机字符串 s，B 收到后再给 A 发送一个新的字符串 s'。当 A 收到 s'后再给 B 发送一个 open(s, n)，B 通过比较之前的字符串 s 和 open(s, n)，如果没有变化，则 B 通过将 s 与 s'做运算得到随机字符串 S，A 也做相同的运算，同样也会得到 S，这样 S 就是 A 和 B 公认的字符串，或者说通过 S，A 和 B 建立起了共识。但是这一机制存在的问题是，在实际的网络中，两个节点之间如果通信延迟或者网络中断，B 没有收到 A 发送的 open(s, n)，这样就无法进行 s 和 s'的运算；另外，网络中的节点数量非常多，两两之间建立共识的计算量会随着节点的增多呈指数增长。

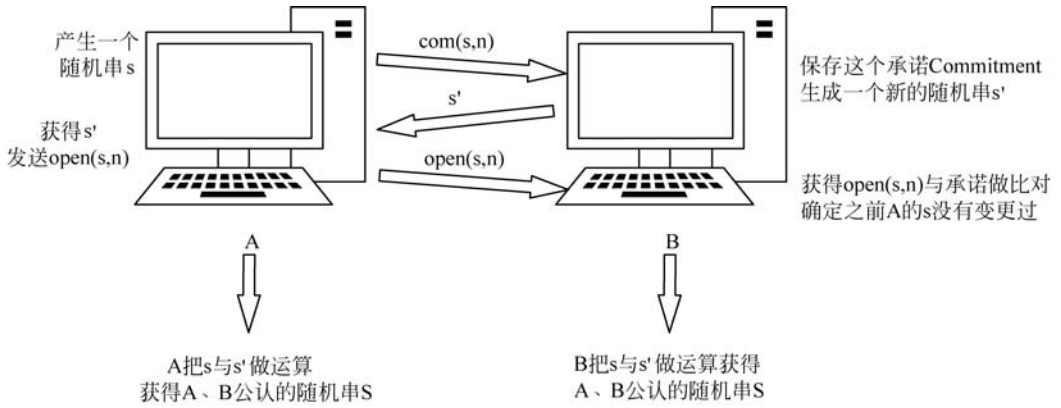
Cardano 系统中，需要获得所有人的字符串(这个字符串也被称为密钥)，最终所有人根据这些密钥形成一个统一的字符串 ρ(随机种子)。一旦有节点掉线，或者网络延迟，那么就无法获得相应节点的字符串。为了解决网络延迟带来的问题，在 PoW 机制中，将一个字符串分成很多份，如果节点数有 m 个，相应地拆分成 m 份，如图 5-10 所示是 Cardano 系统产生随机种子过程。

将每个 epoch 分为三个阶段：

(1) 第一阶段(4/10)。这一阶段每个节点将自己的字符串向全网广播，同时发送一份处理过的小片段，也就是这个字符串的一部分 σ_1、σ_2、…、σ_i($m=i$)。这些发送字符串的节点叫作别选节点。Stakehoders 在这个阶段把自己的 com() 和 σ_1、σ_2、…、σ_i 广播打包到链上。

(2) 第二阶段(4/10)。每个节点广播自己的 open()，open()负责解码上个阶段发送的

字符串。这时一部分节点的 open()有可能没有收到。Stakeholders 广播自己的 com()对应的 open()打包到链上。



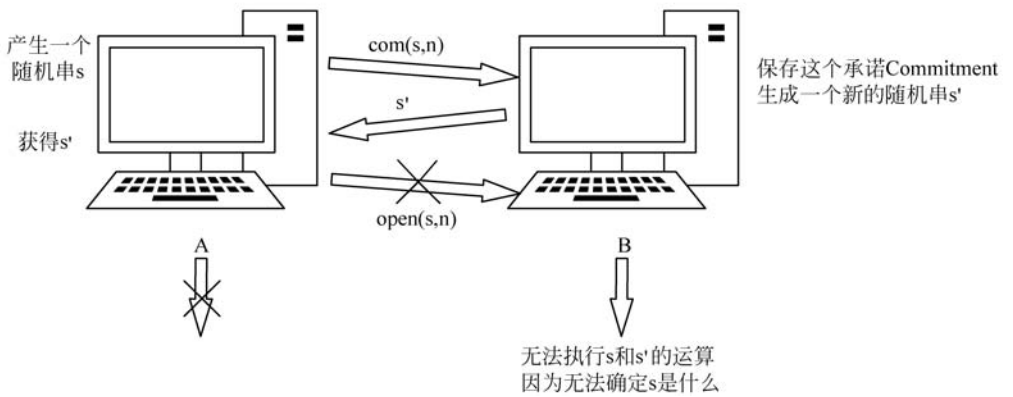


图 5-9　两个节点建立信任

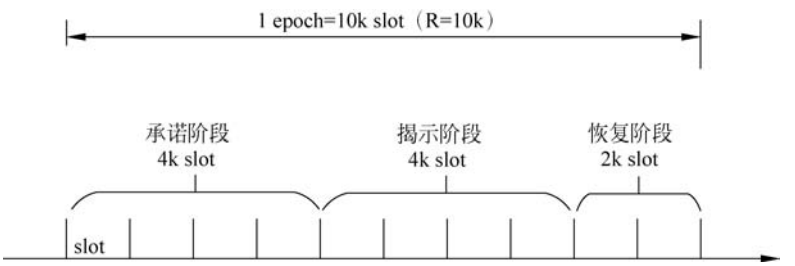


图 5-10　Cardano 产生随机种子的三个阶段

（3）第三阶段(2/10)。根据对应节点的 open()解码密钥，如果没有解码的密钥，则从链上拿到 σ，用自己的密钥解出后广播到链上，其他节点可以收集足够多的密钥进行恢复。Stakeholders 检查没有被 open()的 com()，若有，从链上拿到 σ 用自己的密钥解出后广播到链上，此时别人可以收集出足够人数的密钥进行 rec()。

至此，所有人都获得了随机字符串，并从这些密钥中形成种子（随机生成的字符串），那么就可以得出一个统一的随机数 ρ。为了保证每个 slot 的随机性，Cardano 通过 FTS 产生每一个 slot_leader，如图 5-11 所示。

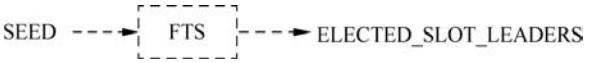


图 5-11　FTS 函数生成 slot_leader

也可以把它看成一个函数，输入是备选 slotleader、随机种子 ρ 和 slot 数量，输出是每个 slot 的 slotleader。FTS 函数实现了权益证明的最终目标，即节点的权益越多，被选中成为 leader 的概率越大。通过权益证明机制让所有节点参与进来，每个节点都会对最终的结果产生影响。

5.4　Zilliqa

目前现有的加密货币和智能合约平台或多或少都有扩展性问题，每秒能够处理的交易数量是有限的，一般少于 20 次。但随着使用公共加密货币和智能合约平台的应用和用户数量的增长，用于每秒处理数百和数千次数量级交易的需求正在增加，而 Zilliqa 就是问题的可能解决方案。目前大概有 1000 个左右的 DApp 运行在以太坊上，不过以太坊的处理效率显然不能正常运行。Zilliqa 采用分片技术，声称能够达到每秒数千次的交易速度，并且随着网络的发展只会继续变得更快。

Zilliqa 是一种旨在扩大交易速度的新的区块链平台，通过分片技术将传统的区块链分成一个个小的网络，随着矿工人数的增加，分片的数量会增大，其交易速度也会上升。在以太坊现有的 30 000 名矿工的规模下，Zilliqa 预计处理的交易速度是以太坊的 1000 倍。

分片简单地说是将采矿网络分成更小的碎片，每个都能够并行处理事务。使用更小的网络节点子集来验证每一个事务，而不是等待网络中的每个节点，这节省了大量时间。Zilliqa 称自己为“下一代高通量区块链平台”。Zilliqa 团队在一篇文章中提到，在使用 2400 个节点、4 个分片的情况下，每秒处理的交易量达到 1389 个。一个月后，又发布了第二组测试结果：在使用 3600 个节点、6 个分片的情况下，每秒处理的交易量达到 2488 个。

5.4.1　Zil 简介

Zilliqa 的代币为 Zil，一共 210 亿个，30％代币公开销售，40％挖矿产生，30％给到团队及投资机构。代币的主要功能类似以太币，用于智能合约开发的消耗和交易的手续费。

40%由挖矿产生的币将在10年内挖完，前4年挖出其中的80%，后6年挖出剩余的20%。

只要有节点参与挖矿，系统会自动将其分配到某个分片，因此就有收益，不用靠算力进行竞争，这样一来普通的GPU矿机也能参与挖矿。而且，随着网络的扩展，每个分片网络单独处理每笔交易，所以每笔交易的总能源成本将保持不变；Zilliqa通过执行一个PoW，能把多个区块写入链中，从而给予矿工更高的奖励，因此可刺激更多的矿工加入网络，依靠分片技术，带来更高的交易速度和吞吐量，进而更加安全。这样也分摊了用户的交易费成本，使其远低于比特币或以太坊的交易费用。

团队成员主要来自新加坡，CEO董心书毕业于华中师范大学计算机专业。2008年毕业后他去了新加坡国立大学读博士，一直做网络安全相关的研究工作，从区块链到网络浏览器和应用程序，可以说是建立安全系统的科学家和实践者。他还是新加坡等几个国家网络安全项目的技术负责人，研究成果已在顶级国际会议上发表。现在，他正在领导用于金融和电子商务领域的安全区块链的研究和开发。技术总监贾瑶是2012年从华中科技大学交换到新加坡国立大学的，与董心书为师兄弟关系。而Zil联合创始人兼首席科学顾问PrateekSaxena为新加坡国立大学的计算机科学研究员，助理教授，拥有加州大学伯克利分校的计算机科学博士学位，董心书视其为老师，曾表示是受到他的影响才进入区块链行业。PrateekSaxena是首席科学顾问，拥有加州大学伯克利分校计算机科学博士学位。

目前，虽然说Zil的价格一路上涨，技术也在不断更新，市场看好，但是对于公有链项目而言，最重要的就是谁能解决底层公有链的性能瓶颈，实现系统的可扩展性，大幅提高交易处理速度，谁就能成为下一个区块链3.0。Zil团队正在积极进行网络升级，用技术手段解决这些核心问题，赢得大众的认可。Zil的愿景是要实现安全数据驱动的分散式应用程序，满足机器学习和金融算法的扩展需求。

5.4.2　分片技术

在Zil网络中，大的网络按照预先规定的节点数被划分为多个较小的网络，如图5-12所示，网络中有两个分片，每个分片独立处理一笔交易，然后打包成区块，称为**微块**(micro block)。然后网络中有一些特殊的节点，来将微块组合成**终块**(final block)，这些节点称为**委员会节点**，后面会详细介绍目录服务**委员会**(directory service committee)的概念。最终，终块组成**交易区块链**(TX-blockchain)。也就是说，分片技术可以通过多个小的网络独立的处理交易，每个交易在特定的分片中打包进区块，最后再组合成区块链。图5-12中，微块1和微块2分别对应网络1和网络2，所以在实际中，网络分片越多，微块数越多，能同时处理的交易也越多。

为了统筹和管理所有的分片，Zilliqa中提出**目录服务委员会**(简称DS committee)，DS committee也会生产区块，也就是DS Block。DS区块不同于交易区块，它不会存储交易信息，而会存储每笔交易对应的分片。DS区块和一般的区块结构是一样的，在Zilliqa中它和交易区块链是两种不同的区块链，两者没有连接。那么一个节点加入Zilliqa中就有两个选

择，加入 DS committee，或者加入普通分片里的节点，想加入哪个组织，只需完成对应的工作量证明即可，简单来说就是有两种工作量证明机制，节点选择一个完成后即加入对应的两个不同的组织。

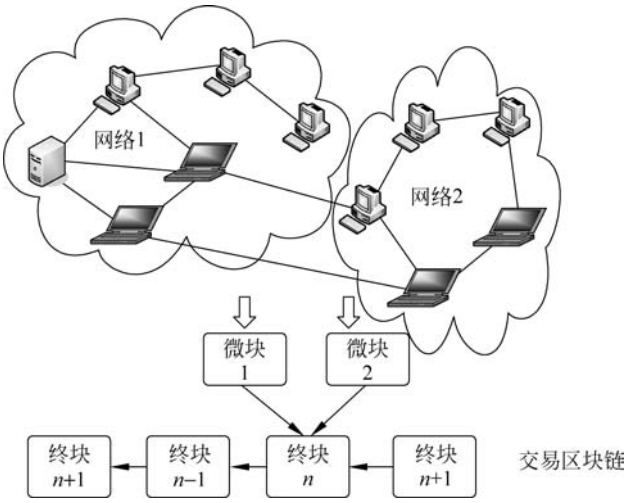


图 5-12　Zilliqa 分片结构

比特币的 PoW 共识算法（工作量证明）解决了长期困扰着计算机科学家的拜占庭将军问题，基于算力的系统是可靠的。在 PoW 区块链中，每个事务都是某个块的一部分，每个块的创建都需要巨大的计算量，但是同时也造成了巨大的资源浪费。Zilliqa 的替代方法利用了 PoW 的优点以及实用的拜占庭容错（PBFT）协议。矿工们使用 PoW 在 Zilliqa 区块链上建立他们的身份。一旦确定了身份，矿工就被分配到一个共识组，其中可以运行多轮 PBFT 共识。执行一个 PoW 将多个块写入链中，从而提供更大的保证奖励。在 Zilliqa，每个月需要执行大约 12 小时的 PoW，这时显卡满负载运行。在其余时间内，显卡将以闲置模式运行，消耗最少的电量。矿工将消耗更少的能量，从而使开采成本比其他基于 PoW 的区块链低得多。

Zilliqa 为矿工和用户带来了另外两项好处。首先，随着网络的扩展，每笔交易的总能源成本将保持不变。其次，Zilliqa 的交易费用将远低于比特币或以太坊的交易费用。在以太坊网络上矿工会优先处理较高交易费用的交易，而 Zilliqa 区块链上矿工只会受交易能够处理的规模的大小所激励。因此大大降低用户的交易费用。

前面说过，在 Zilliqa 中节点可以有两种选择——加入 DS committee 或者成为分片中的普通节点，只需完成对应的工作量证明。具体的方法分为如下两步。

1. 加入 DS committee

节点加入 DS committee 需要完成 PoW_1，其工作量证明内容如图 5-13 所示。

Algorithm1：PoW_1 for DS committee election

Input：i：Current DS-epoch，DS_{i-1}：Prev. DS committee composition

Output：header：DS-Block header.

1 **On each competing node**：

　　//从 DS 区块链中随机抽取一个 epoch

　　//DBi－1：第 i 个 epoch 前的最新一个 DS-Block

2 　r_1←**GetEpochRand**($\mathbf{DB}_{i-1}$)

　　//从交易区块链中随机抽取一个 epoch

　　//TBj：第 i 个 epoch 前的最新一个 TX-Block

3 　r_2←**GetEpochRand**($\mathbf{TB}_j$)

　　//pk：节点公钥，IP 为节点 IP 地址

4 　nonce，mixHash←**Ethash-PoW**(pk，IP，r_1，r_2)

5 　header←**BuildHeader**(nonce，mixHash，pk)

　　//header 包括 pk、nonce 以及其他字段

　　//IP. header 发送给 DS committee 的成员

6 　**MulticastTo** $\mathbf{DS}_{i-1}$(**IP**，header)

7 　**return** header

图 5-13　加入 DScommittee 的 PoW 内容(引自 Zilliqa 技术白皮书)

很好理解，根据当前的 DS-Block 获得一个随机数 r_1，再根据 TX-Block 获得一个随机数 r_2，再加上节点的公钥和 IP 地址，这四项同时作为参数，计算 nonce 值，从而完成 PoW_1。

2. 加入分片节点

节点加入分片节点需要完成 PoW_2，其工作量证明内容如图 5-14 所示。

Algorithm 2：$\mathbf{PoW_2}$ for shard membership

Input：i：Current DS-epoch，DS_i：Current DS committee composition

Output：header：DS-Block header.

1 **On each competing node**：

　　//从 DS 区块链中随机抽取一个 epoch

　　DBi-1：第 i 个 epoch 前的最新一个 DS-Block

2 　r←**GetEpichRand**($\mathbf{DB}_{i-1}$)

　　//pk：节点公钥，IP 为节点 IP 地址

3 　nonce，mixHash←**Ethash-PoW**(pk，IP，r)

　　//IP，header 发送给 DS committee 的成员

4 　**MulticastTo** $\mathbf{DS}_{i-1}$(**IP**，header)

5 　**return** nonce，mixHash

图 5-14　成为普通节点的 PoW 内容(引自 Zilliqa 技术白皮书)

不同于比特币和以太币中的 PoW，Zilliqa 中工作量证明不是生产区块的资格证明，而是用来生成节点身份和避免女巫攻击(sybil attacks)的手段。

当有一笔交易时，会根据交易的地址位数，将交易分配到对应的分片中。每个分片中有一个 leader，负责在每个分片中发布处理交易的消息。leader 发布消息后，每个节点开始检

验信息的正确性。当有 2/3 以上的节点通过后，leader 将交易打包进区块中。在 Zilliqa 共识协议中，如果领导者是诚实的，它可以不断地推动共识小组中的节点就新的交易达成协议。但是，如果领导是拜占庭，它可以有意地延迟或丢弃来自诚实节点的消息，并减慢协议。为了惩罚这些恶意领导者，协议会定期更改每个分片的领导和 DS committee。这可以防止拜占庭领袖在无限期的时间内拖延共识协议。由于所有节点都是有序的，下一个领导者将以循环方式进行选择。事实上，产生每一个微块后分片的领导者都会改变，并且在每产生一个区块之后 DS committee 的领导者也会更改。

5.5 超级账本

超级账本（hyperledger）是区块链 3.0 最典型的代表，它是由非营利性组织 Linux 基金会发起成立的，致力于企业级区块链开发及应用的开源项目，该项目试图打造一个透明、公开、去中心化的分布式账本项目，作为区块链技术的开源规范和标准，让更多的应用能更容易地建立在区块链技术之上，打造可以跨行业的区块链应用解决方案。在此之前，全球大多数区块链项目都要从底层开始摸索，在没有执行标准的情况下，缺乏行业间的通力协作。超级账本项目的目标是建立一个跨行业的开放式标准及开源代码开发库，允许企业根据自己需求创建自定义的分布式账本解决方案，以促进区块链技术在各行各业的应用。

在众多应用中，区块链分布在几个兴趣社区中。第一个社区由比特币这样的特定项目专注于为网络上所有想要在开放区块链上测试、构建和使用替代数字货币的人提供完全开放的技术。

包括去中心化应用程序在内的第二个社区由 Solidity 开发人员在以太坊虚拟机（ethereum virtual machine，EVM）上构建，为开发智能合约提供了无授权技术，为参与者提供绝对的开放和隐私保护。在以太坊上构建 DApp，需要熟练掌握特定技能，如使用 Solidity 语言进行编程。

此外，第三批区块链创新者试图克服无限制去中心化的常见问题，他们的目标是开发“去中心化瓶颈问题解决方案”。业务合作伙伴将相互合作，以 KYC（know your customer）概念为基础创建信任关系并进行交流。Linux 基金会的超级账本项目就属于第三个社区。

超级账本项目的所有源代码均托管在 Gerrit 和 GitHub 上，有兴趣的读者可以前往查阅。目前，超级账本主要包括以下项目：

（1）Fabric：最早由 IBM 和 DAH 于 2015 年年底发起研究的区块链平台架构，以区块链基础核心平台为目标，包括 Fabric CA、Fabric-API、Fabric SDK 等子部件。该框架具有组件可拔插、授权通道通信机制及支持 Java、Python 和 Go 等多种语言的特点，便于研究人员开发。该架构支持使用 PBFT、Raft 等新型共识机制，适用于企业级业务。作为最受大众

关注的超级账本项目，Fabric 的发展非常迅速，在某些领域的应用也取得了非常好的成绩，具有很大的发展空间和开发潜力。

(2) Sawtooth：是由 Intel 主导开发并贡献的区块链平台架构，以数字金融资产管理为目标，包括 arcade、core、dev-tools、validator、mktplace 等子项目。基于 Sawtooth 框架的项目可以根据实际业务场景自主选择不同的共识机制，默认使用时间流逝证明共识机制(Proof of Elapsed Time，PoET)。这种机制杜绝了通过 ASIC 专用硬件“作弊”的可能，可靠性由 Intel CPU 硬件保障。Sawtooth 分布式账本平台支持运行智能合约，同时具有模块化程度高、可定制能力强等特点，有很大的开发价值。

(3) Iroha：由 Soramitsu 公司和 Hitachi 公司等多家日本企业联合开发，目标是为超级账本提供 C++开发环境，为移动和网页应用提供基础构架支持。Iroha 项目的核心是为分布式账本提供基础设施，构建数据会员服务、共识算法、P2P 网络传输及智能合约等组件的基础构架，利用 iOS、Android 和 JavaScript 资源库的便利资源保证网络安全运行。受 Fabric 的影响，Iroha 项目的构架设计和智能合约服务形式等方面与 Fabric 项目很相似。

(4) Blockchain-explorer：由 DTCC 牵头发起，与 IBM、Intel 等公司合作研发的区块链浏览器。该项目提供 Web 操作界面，通过界面可快速查看、调用、部署区块或事务，读取相关数据、网络信息、链码和事务序列，以及任何其他保存在账本中的相关信息。

(5) Cello：由 IBM 团队于 2017 年 1 月提出，目标是将“区块链即服务”部署模式应用于实际开发情景，减少创建、管理和终止区块链所需要的工作量。Cello 项目使应用开发者不需要学习如何逐步搭建和维护超级账本网络，可直接通过可视化面板来部署和管理多条区块链。

(6) Indy：由 Sovrin 基金会于 2017 年 3 月发起，提供基于分布式账本技术的数字身份管理机制。

(7) Composer：由 IBM 团队发起并维护，定位为区块链开发协同工具，提供面向链码开发的高级语言支持。该项目可以简化区块链网络创建过程，加速智能合约开发和部署。

(8) Burrow：由 Monax 公司于 2017 年 4 月发起，定位是一个通用的智能合约执行引擎。该项目的前身为 eris-db，基于 Go 语言实现。Burrow 项目提供了支持以太坊虚拟机的智能合约区块链平台，并支持 Proof-of-Stake 共识机制和权限管理，可以提供高效的区块链交易。

(9) Quilt：是对 W3C 支持的跨账本协议 Interledger 的 Java 实现，Interledger 协议也被称为 ILP，是一种账本间交易的协议，用于跨系统间(包括分布式账本和非分布式账本)的价值传输。Quilt 通过 ILP 的实施来提供账本系统间的互操作性。

(10) Caliper：由华为公司于 2018 年 3 月发起，定位是区块链平台性能的测试工具。用户可以在定义好测试集的情况下针对自己的区块链网络进行性能测试，获取一系列的测试结果并生成测试报告。

（11）Ursa：由 Intel、Sovrin、IBM 等企业的开发者共同研发，是一个共享加密库，提供密码学相关组件。该项目旨在提高网络安全性，同时也以避免重复的加密工作为目标。

（12）Grid：由 Cargill、Intel 和 Bitwise IO 公司发起，是一个框架和库的集合，提供帮助快速构建供应链应用的框架。

在这些项目中，Fabric 是最出名的一个，一般谈到超级账本指的都是 Fabric。下面介绍 Fabric 的架构和关键技术，包括多链多通道、账本设计等，以及链码的编写和部署流程，最后简单说明 Fabric 的一些不足。

5.5.1 公有链、私有链和联盟链

在具体介绍 Fabric 之前，先来简单介绍一下区块链的分类。根据不同的用户需求及适用场景，区块链可以分为公有链（public blockchain）、私有链（private blockchain）和联盟链（consortium blockchain）这三类。

1. 公有链

全世界任何个体或团体可以随时自由加入公有区块链网络来读取网络中的数据信息并发起交易，且合法交易可以获得该区块链的有效确认。任何个体都能参加公有链的共识过程，以此为依据，决定哪些区块可以被添加到区块链中，明确当前网络状态。公有链通常被认为是“完全去中心化”的，任何个人或团体都不能控制或篡改其中的数据。多数公有链通过代币奖励机制来鼓励参与者竞争记账，以此来保证数据的安全。公有链是出现最早的区块链类型，也是目前世界上应用范围最广的类型，典型的公有链有：比特币区块链、以太坊。

2. 私有链

与公有链完全相反，私有链网络的读写权限由某个组织或个人完全控制，数据读取规则由网络所属组织规定，通常具有很大程度的访问限制。与公有链相比，私有链中的节点数目少且可信度较高，因此达成共识的时间相对较短、交易速度更快、效率更高、成本更低。私有链属于“许可链”，这意味着每个参与节点都需要经过认证，这样可以更好地保护隐私，并且能做到身份认证等金融行业必需的要求。相比传统的中心化数据库，私有链能够防止组织内部某些节点故意隐瞒或篡改数据，并在发生事故后能够迅速追责。因此，很多金融机构更倾向应用私有链技术。私有链主要应用于企业内部，代表有 Linux 基金会、R3 Corda 平台。

3. 联盟链

联盟链介于公有链和私有链之间，可实现“部分去中心化”。联盟链由若干个组织或机构共同参与管理，每个组织或机构控制一个或多个节点，所有节点共同记录交易数据，采用 PBFT、Raft 等新型共识机制保证账本的唯一性和安全性。联盟链属于“许可链”，参与者需要经过许可才能加入，并且只有这些参与的组织和机构能够对区块链中的数据进行读写和发起交易。从某种角度来看，联盟链也可以被看作特殊的私有链，只是私有化程度相较私有链更低，但是其同样具有成本低、效率高的特点，适用于不同领域的业务。典型的联盟链有：

R3 区块链联盟、超级账本、俄罗斯区块链联盟等，其中为大众所熟知的就是开源的超级账本项目中的 Hyperledger Fabric 子项目。

5.5.2　Fabric 网络架构

Fabric 的架构历经了两个版本的演进，最初接触的 0.6 版本只能被用来进行商业验证，无法被应用于真实场景中，主要原因就是结构简单，基本所有的功能都集中在 peer 节点，在扩展性、安全性和隔离性方面有着天然的不足。因此，在后来推出的 1.0 正式版中，peer 节点的功能被分拆，共识服务从 peer 节点剥离，独立为排序节点提供可插拔共识服务。更为重要的一个变化就是 1.0 正式版中加入了多通道（multi-channel）功能，可以实现多业务隔离，相比于 0.6 版本发生了质的飞跃。

如图 5-15 所示，由多个分布式节点组成的网络层位于 Fabric 1.0 网络架构的最底部。这些节点共同构成一个 P2P 的网络，采用 Gossip 协议进行节点间的数据传输及定位感知，同时使用 gRPC 框架实现接口调用功能。

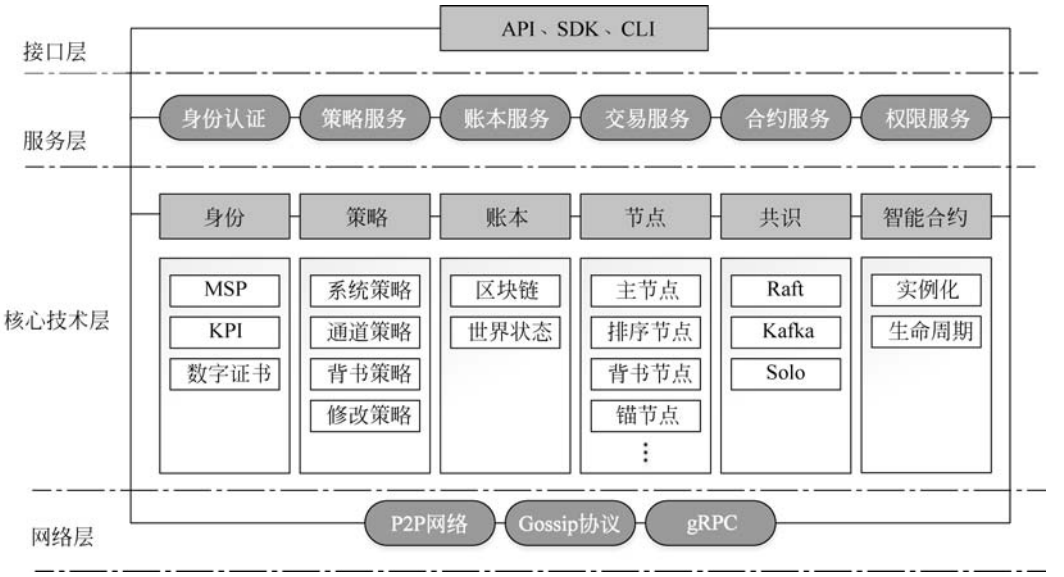


图 5-15　Fabric 1.0 架构图

核心技术层中的身份认证技术确保加入网络的用户有一定的信任基础，同时根据身份确定参与者在区块链网络中拥有的各项权限。策略是一组规则，这些规则定义了如何制订决策和达到特定目标的步骤，是基础架构管理的机制。分布式账本存储有关业务对象的重要事实信息、对象属性的当前值，以及导致这些当前值的交易历史。节点是网络的一个基本元素，它们承载账本和智能合约。在 Hyperledger Fabric 网络中，节点分类较多，每种节点都有特定的权限和工作。共识简单来说就是概率一致性算法，这些算法最终以高概率确保

账本的一致性，Fabric 的设计依赖于确定性共识算法，因此可以保证经过排序节点的区块都是一致的和正确的。智能合约定义了生成添加到账本的新交易的可执行逻辑。

服务层包括 Fabric 所提供的多项服务，其中最基本的记账服务，用于记录交易信息或重要数据；身份认证服务用于确定参与者身份，增强信任基础。策略服务可以定义管理机制，帮助管理者更好地控制网络的运行状态；智能合约服务帮助用户更好地编写适用于特定场景的合约代码，目前支持用 Go、Java 或者 Node.js 开发；权限服务用于分配不同的权限给不同的用户。

在接口层方面，Fabric 提供 API 接口方便使用者进行应用开发。对于客户端应用，Fabric 目前提供 Node.js、Java SDK 及 Go SDK 等开发工具集合。开发者还可以通过命令行界面（command line interface，CLI）高效便利地测试 chaincode 或查询交易状态。

通过对 Fabric 架构的分析，可以看出技术层是整个架构的核心部分，且其中的技术具有很强的专业性，对于想学习 Fabric 的人来说，一定要对这些技术有一定的了解。下面将逐项介绍 Fabric 的关键技术。

5.5.3 Fabric 关键技术

1. 身份认证

Hyperledger Fabric 区块链网络中有许多不同的参与者，每个参与者地位并不完全等同，为了区分不同的参与者并保证参与者的可信基础，Fabric 引入身份认证机制。每个参与者都有一个封装在 X.509 数字证书中的数字身份，这些身份确定了参与者对区块链网络资源的确切权限和拥有的信息访问权限。

Fabric 中数字身份的颁发是通过公钥基础设施（public key infrastructure，PKI）实现的，PKI 是为网络提供安全通信的软件、硬件、规程及人员等要素的集合。虽然区块链网络并非简单的通信网络，但它也依赖 PKI 标准来确保各类网络参与者之间的安全通信，并确保发布在区块链上的消息得到正确的身份验证。

PKI 有四个关键要素：数字证书、公钥和私钥对、证书颁发机构及证书吊销列表。

（1）数字证书：数字证书是一个文档，其中包含与证书持有者有关的一组属性。最常见的证书类型是符合 X.509 标准的证书，该证书对网络内部参与者的身份详细信息进行编码。证书通过密码学技术来记录证书拥有者的所有属性，避免证书被篡改，所以在 Fabric 中一般将各类节点的 X.509 证书视为无法更改的数字身份证。

（2）公钥和私钥对：也称非对称加密，每个用户有两个密钥，这两个密钥之间可以互相解密，一般情况下用户保留私钥并将公钥公布。私钥和相应公钥之间的独特关系是安全通信的关键技术。用私钥签名的消息可以由公钥进行验证，以此来保证信息来源受信任；用公钥加密的信息，只能由拥有私钥的用户来解密，以此保证信息的私密通信。

（3）证书颁发机构：为参与者提供可验证数字身份的机构，在 Fabric 中称证书颁发机

构为 CA。

(4) 证书吊销列表：证书吊销列表(certificate revocation list，CRL)是一个对证书的引用列表，CRL 中都是由于某些原因被吊销的证书。验证者可以通过 CRL 检查被检验者证书是否仍然有效。

前面已经介绍了 PKI 如何提供可验证的身份，下一步是了解如何使用这些身份来代表区块链网络的受信任成员，这就需要用到管理服务提供商(managed service provider，MSP)。

MSP 是 Fabric 中提供抽象化成员操作的组件。MSP 的实现表现为一组文件夹，这些文件夹被添加到网络的配置中，并用于定义身份，以及身份的管理与认证。MSP 通过列出成员的身份或授权某些 CA(certification authority)可以为其成员颁发有效的身份，从而定义信任区间，并将身份转换为角色。

为了更好地理解 MSP 与 PKI 之间的关系，举一个简单的例子。可以将前面介绍的 PKI 中的证书颁发机构比作发行信用卡的提供商，它分配了许多不同类型的可验证身份，由 MSP 确定商店接受哪些信用卡提供商(可验证身份)。这样，MSP 就将身份(信用卡)转换为了角色(在商店购买商品的能力)。

在 Fabric 网络中，根据作用域的不同，可以将 MSP 分为本地 MSP 和通道 MSP，这二者功能几乎一样，但作用范围不同。本地 MSP 定义节点的权限，通道 MSP 在通道级别定义了管理权和参与权。

2. 策略

策略是使 Hyperledger Fabric 与以太坊或比特币等前两代区块链网络有所不同的主要原因之一。Fabric 是经过许可才能参加的区块链网络，其用户身份可以被基础结构识别，有一定的信任基础，因此这些用户能够在启动网络之前决定网络的治理策略，或在网络运行过程中随时修改策略。

策略是一组规则，这些规则定义了如何制定决策及实现特定目标的步骤。策略通常描述为“谁”和“什么”，例如用户对资产的访问权和控制权。在 Fabric 区块链网络中，策略是一种管理基础架构的机制。在通道中添加或删除成员，更改区块的形成方式和智能合约上链所需签名数目的标准，这些行为及谁可以执行这些行为均由策略定义。简而言之，各级参与者在 Fabric 网络上执行的所有操作均由策略控制。

在 Fabric 网络中，策略是实施在不同层次上的。每个策略域管理着网络运行的不同方面。图 5-16 是 Fabric 策略层次结构的直观表示。

策略有很多分类，例如系统通道配置、应用通道配置、背书策略、签名策略及修改策略。下面简单介绍这些策略的具体内容。

(1) 系统通道配置：每个网络的创建都是从创建排序系统通道开始的，系统通道配置中的策略控制排序服务所使用的共识机制，定义如何创建新区块，同时规定了联盟中哪些成

图 5-16　Fabric 策略层次结构图

员有权创建新应用通道。

（2）应用通道配置：应用程序通道是为联盟组织提供的专用私密通信通道。应用通道配置中的策略控制在通道中添加或删除成员的能力，且规定了需要经过哪些组织批准才能将链码上传到通道上。

（3）背书策略：Fabric 网络中交易需要经过背书节点背书（拟运行），每个智能合约的背书策略规定了调用该智能合约进行交易需要哪些背书节点进行背书并签名。

（4）签名策略：签名策略定义了需要多少成员认可签名后才能认定业务有效。这种策略是最常见的策略之一，它们可以构建出非常具体的规则，在语法上支持 AND、OR 和 NOutOf 的任意组合。例如，OR（Org1，Org2），意味着要满足该策略，至少需要来自 Org1（组织 1）中的一个成员，或者 Org2（组织 2）中的一个成员的签名；如果是 AND（Org1，Org2）则要求两个组织的成员必须同时签名。

（5）修改策略：定义了如何更新策略，同时也规定了批准策略更新时需要的签名。每个通道配置都包括对修改策略的引用。

3. 账本

账本是 Hyperledger Fabric 中的一个关键概念。它存储有关业务对象的重要事实信息、对象属性的当前值及导致这些当前值的交易历史。在 Hyperledger Fabric 网络中，账本由两个不同但相关的部分组成，如图 5-17 所示，一个是世界状态，另一个是区块链。

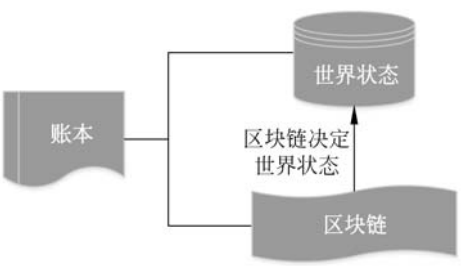


图 5-17　Fabric 账本

世界状态：一个保存账本状态当前值的数据库。用户可以通过应用程序直接访问状态的当前值，而不需要遍历所有交易日志来计算。一般情况下，状态表示为键-值对。由于状态值可以被创建、更新和删除，所以世界状态可以频繁发生更改。世界状态数据库目前主要包括 LevelDB 和 CouchDB 两种，默认使用 LevelDB 数据库，该数据库适用于账本状态为简单键值

对的情况。CouchDB数据库支持查询和更新复杂的数据类型，适合应用于账本状态是JSON文档的情况。

区块链：以区块的形式记录决定当前世界状态的所有交易日志。区块链的数据结构与世界状态不同，里面内容一旦写入就无法篡改。由于区块链数据结构只涉及极少量的简单操作，因此一般以文件的形式实现。

区块链的结构是互连区块的顺序日志，其中每个区块包括三个主要部分。

(1) 块头：包含当前区块编号、当前区块的哈希值及上个区块块头的哈希值。

(2) 区块数据：包括按顺序排列的交易列表。

(3) 区块元数据：包括区块创建者的证书和签名，用于其他节点验证该区块。

4. 节点

节点是区块链网络最重要的基本元素之一，Hyperledger Fabric网络中的节点与比特币、以太坊等公有链中的节点有所不同。公有链中所有节点的功能、地位、权限几乎完全相同，而在Fabric中，节点的分类非常具体，每种节点都有自己的工作与权限。下面介绍Fabric网络中的各类节点。其中，前四种节点都属于组织内部的节点，统称为peer节点。

(1) 主节点：组织内部唯一和排序节点通信的节点，一般由配置文件指定，如果下线则可以通过选举策略重新选举。

(2) 背书节点：运行交易、智能合约得到模拟结果，再将模拟结果返回给用户进行验证，为交易做保障。Fabric区块链网络中的交易和智能合约一般只在背书节点上运行一遍。当所有的检验、排序完成后，交易结果将记录于区块链上，形成不可篡改的记录。

(3) 记账节点：最普通的节点，除记账外没有其他功能。主节点和背书节点同时也可以是记账节点。

(4) 锚节点：锚节点用于组织之间通信，一个组织可以有0个或多个锚节点。如果一个节点需要与其他组织中的节点联系，它可以通过在该组织的通道配置中定义的锚节点进行通信。

(5) 排序节点：排序节点属于组织外的节点，并不属于peer节点。Fabric的共识机制是由排序节点来完成的。它主要做两件事：①从全网客户端接收交易，将交易按规则排序；②将排完序的交易按固定时间间隔打包成区块，发放给主节点。

peer节点托管了账本副本实例和链码副本实例，使Fabric网络可以避免由于单点故障导致的网络瘫痪。每个peer节点可以通过加入多个通道来托管多个账本和智能合约，应用程序和管理员想要访问这些资源，就必须先与peer节点进行交互。这就是为什么peer节点被认为是Fabric网络最基本的结构。当节点刚创建时，它既没有账本也没有链码，需要创建和安装。

图5-18中，Peer 1和Peer N在通道a，组成区块链A；Peer 2和Peer N在通道b，组成区块链B；Peer 1、Peer 2和Peer N在通道c，组成区块链C。三个通道组成了三个相互

独立的链，peer 节点只需维护自己加入链的账本信息，感应不到其他链的存在。这种模式与现实业务场景有诸多相似之处，不同业务有不同的参与方，不参与某业务时不应看到该业务相关的任何信息。

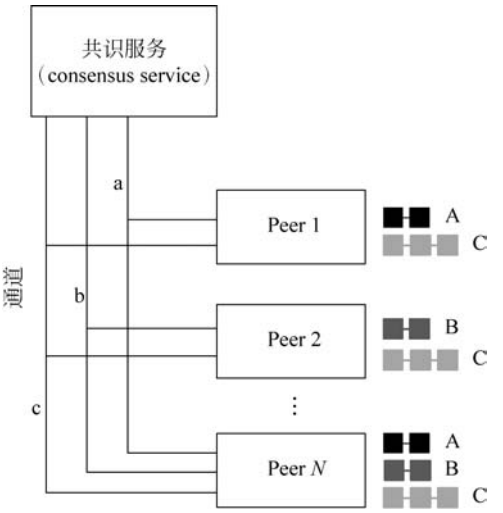


图 5-18　peer 节点托管多个账本和智能合约示意图

5. 排序服务和共识机制

如以太坊和比特币等公有区块链网络，任何人都可以不经过许可就加入并参加共识过程。因此，这些系统依赖概率一致性算法，这些算法最终以高概率确保账本的一致性，但仍然可能产生账本“分叉”情况。分叉的原因可能是网络中的不同参与者对交易排列顺序有不同的规则。

Hyperledger Fabric 的工作原理有所不同。它具有一个名为 orderer 的特殊节点（也可称为“排序节点”）来执行对交易的排序，所有排序节点一起工作构成排序服务。Fabric 的设计依赖确定性共识算法，因此可以保证经过排序节点的区块都是一致且正确的。账本不会像比特币等公有链网络中那样出现分叉。

Fabric 交易流程如图 5-19 所示，共分为 6 步，首先客户端需要经过 CA 的认证（图 5-19 中的“0. 登记注册”），许可后才能成为区块链的一个节点。之后的五个步骤分为三个阶段（交易提议、排序打包、验证提交），这三个阶段确保区块链网络中所有 peer 节点的账本保持一致。这里重点介绍的是排序服务在交易流程中起到的作用。

第一阶段：交易提议。

第一阶段包括流程图中步骤 1 和步骤 2，客户端将交易建议发送给背书节点，背书节点将调用智能合约以生成交易建议响应，但并不会直接将模拟交易结果应用于账本，而只是将建议响应返回给客户端查看。

第二阶段：排序打包。

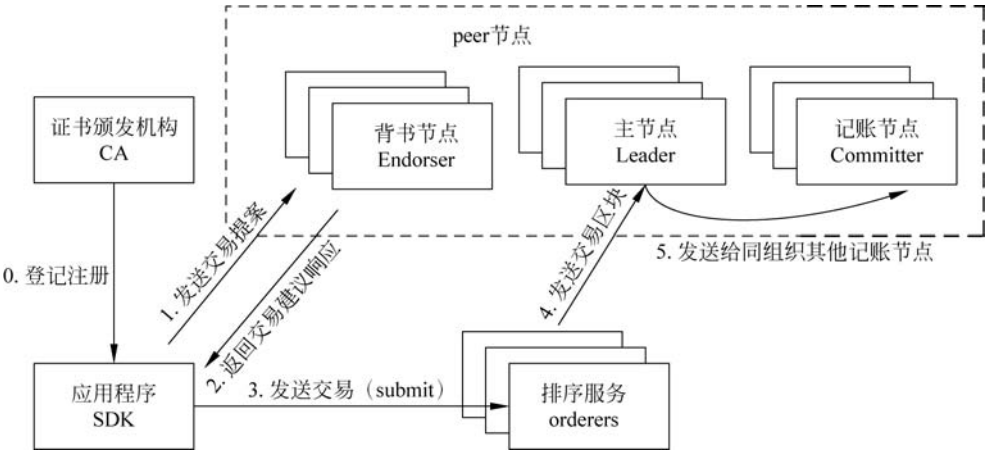


图 5-19　Fabric 交易流程

交易的第一阶段完成后，客户端收到一组来自背书节点的已背书的交易建议响应。在第二阶段（步骤 3），客户端将包含已背书的交易建议响应发送给排序节点。排序节点同时从许多不同的客户端应用程序接收交易，并将交易按事先定义的顺序排序打包成区块。

第三阶段：验证提交。

第三阶段包括步骤 4 与步骤 5，将排序打包好的区块分发给通道上各个组织的主节点，以便进行最终验证和提交。主节点收到排序节点发来的区块，将区块再分发给组织内部其他有记账权力的节点，记账节点分别验证交易的背书签名是否满足条件，但一般不会检验交易本身。当条件满足时，记账节点将所有区块中的交易应用于账本进行记录；当条件不满足时，依旧会保留交易记录，但不会将交易结果应用于世界状态。

Hyperledger Fabric 网络中，排序节点之间对交易的排序达成共识的实现方法有下面这几种。

（1）Solo：排序服务的 Solo 实现仅用于测试，并且仅由单个排序节点组成。它在 Fabric v2.0 中已被弃用，并可能在将来的版本中完全删除。

（2）Kafka：Apache Kafka 是一种碰撞容错（crash fault tolerance，CFT）实现，它使用“leader 和 follower”节点配置。Kafka 利用 ZooKeeper 集合进行管理，从 Fabric v1.0 开始便可以使用基于 Kafka 的排序服务了，但许多用户发现管理 Kafka 集群的额外开销较大，不适合中小型用户使用。

（3）Raft：Raft 是从 Fabric v1.4.1 开始使用的新版共识机制，它是一种基于 Etcd 中 Raft 协议实现的 CFT 排序服务。Raft 同样遵循“leader 和 follower”模型，每个通道选举一个 leader 排序节点，follower 排序节点复制其决策。Raft 排序服务应该比基于 Kafka 的排序服务更易于设置和管理，并且它的设计允许不同的组织为分布式排序服务贡献节点。

6. 智能合约

智能合约以可执行代码的形式定义了不同组织用户之间交互的规则。应用程序调用智能合约来生成、查询记录在账本上的交易。智能合约可以为任何类型的业务对象实施管理规则，执行智能合约即可在满足条件的情况下自动执行各项条款，比人工业务流程高效得多。

智能合约可以以编程方式访问账本的两个不同部分：一个是不可变地记录所有交易历史的区块链，另一个是保存这些状态当前值的世界状态。智能合约主要在世界状态下放置、获取和删除状态，或者查询不可变的区块链交易记录。

在 Fabric 区块链网络中，通道为组织之间提供了完全独立的通信机制。若使用智能合约则需要先将其提交到通道上，经过审核和实例化后，可供通道上的参与者进行调用。各组织在使用智能合约于通道中进行交易之前，需要对智能合约的管理达成一致，如确定背书策略。同时智能合约还可以调用同一通道内或不同通道间的其他智能合约，使智能合约的开发更加简便高效。

5.5.4 Fabric 样本网络构建过程

前面介绍了 Hyperledger Fabric 区块链网络的网络架构与关键技术，接下来对 Fabric 网络的形成过程进行简单讲解。整个样本网络构建过程大致分为以下六部分。

1. 建立网络

构建 Hyperledger Fabric 区块链网络的首要工作是由网络创建者定义网络配置，再通过网络配置定义排序服务。

如图 5-20 所示，组织 R1 是整个 Fabric 网络 N 的创建者，由该组织定义网络配置 NC，

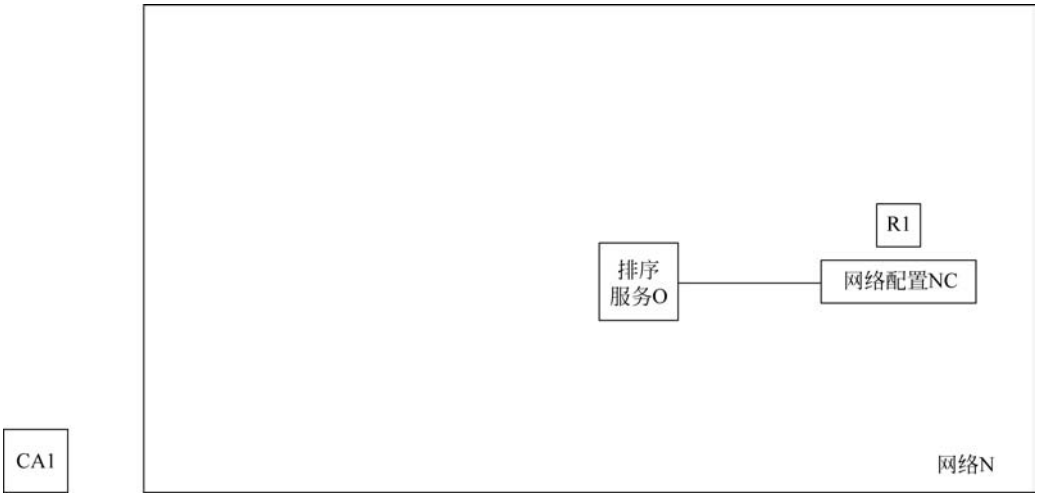


图 5-20 Fabric 网络构建——建立网络

再通过网络配置 NC 定义排序服务 O，启动排序服务后 Fabric 网络初步形成。根据策略，排序服务 O 目前由组织 R1 中的管理员进行配置和启动，并托管在组织 R1 下。证书颁发机构 CA1 用于为组织 R1 的管理员和网络节点颁发数字证书。网络配置 NC 规定了组织 R1 在整个网络中的各项权限，同时也定义了排序服务的各项策略。

2. 添加网络管理员

如图 5-21 所示，网络建立初期，由组织 R1 单独定义网络配置 NC，并对网络进行管理。在这一阶段，将添加组织 R2 作为网络管理员，与 R1 共同管理网络。此后，R1 和 R2 在网络配置中享有同等的权利，共同管理网络 N（实际情况可能需要添加多个组织作为网络管理员，这里为了简化仅添加一个组织 R2）。同时添加证书颁发机构 CA2，用于为组织 R2 中的用户颁发数字证书。

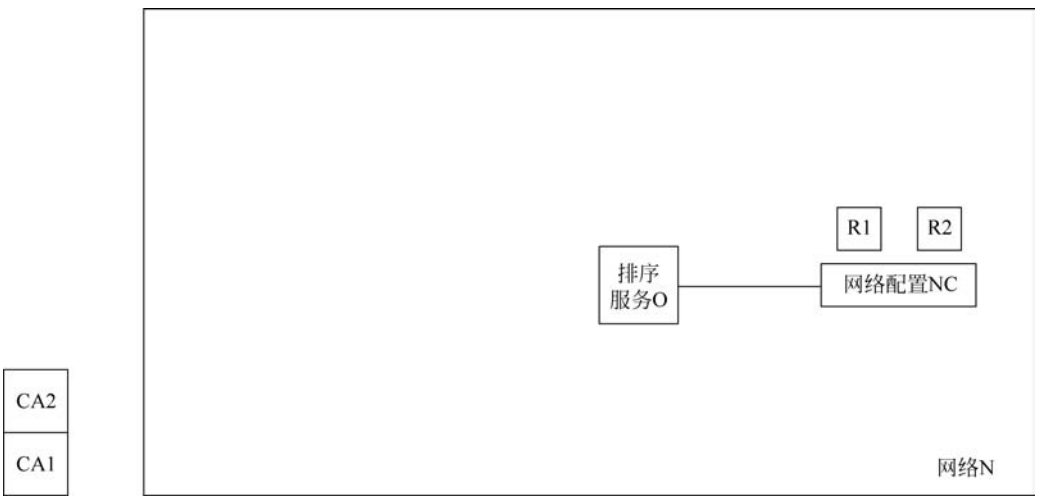


图 5-21　Fabric 网络构建——添加网络管理员

前面介绍的构成排序服务的节点运行在组织 R1 的基础结构上，由 R1 管理和启动。但在 R2 加入网络并成为管理员后，R1 可以分出一部分排序节点交由 R2 进行管理，这样就可以构建一个多组织、多节点的排序服务，更有利于网络的去中心化。

3. 定义联盟并创建通道

Hyperledger Fabric 属于联盟链，需要定义联盟来进行交易和数据传输。在本阶段，将定义一个联盟，并为该联盟创建一个专用通道进行通信。

如图 5-22 所示，网络管理员定义一个联盟 X1（根据网络配置 NC 的规定，只有作为管理员的组织 R1 或 R2 可以创建新的联盟）。该联盟包含两个成员，组织 R1 和 R3，联盟的定义存储于网络配置 NC 中。请注意，一个联盟可以有任意数量的组织参加，这里为了简化，只定义两个组织作为一个联盟。

定义完联盟后，要为该联盟创建一个专用通信通道。通道通信是 Hyperledger Fabric

主要的通信机制，联盟的成员可以通过该机制相互通信，一个网络中可以有多个通道。

如图 5-22 所示，使用联盟定义 X1 创建通道 C1，通道 C1 由独立于网络配置 NC 的通道配置 CC1 控制。CC1 由组织 R1 和 R3 共同制定和管理，网络管理员 R2 在通道 C1 中没有任何权限。通道 C1 使用排序服务 O 进行交易排序打包。这里强调一点，虽然网络配置 NC 看起来级别较高，但它并不能直接影响通道配置 CC1，通道配置仅由通道管理员管理，这也体现了 Fabric 网络去中心化的特点。而通道之所以重要，是因为它为联盟成员之间的私密通信提供了一种保密机制，允许组织共享基础结构的同时保持数据隐私。

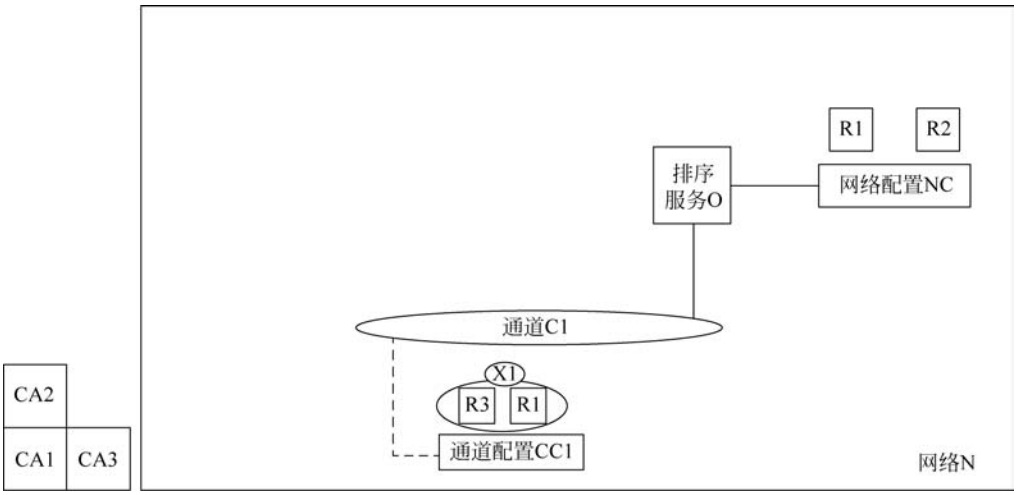


图 5-22　Fabric 网络构建——定义联盟并创建通道

4. 添加节点和客户端

刚建立的通道 C1 只连接了排序服务 O，没有连接其他内容，在本阶段将用通道连接各组件，如客户端和 peer 节点，并在节点上添加账本和智能合约。

如图 5-23 所示，在通道 C1 中添加节点 P1、节点 P3 和客户端 A1、客户端 A3，节点 P1 和客户端 A1 归组织 R1 管理，节点 P3 和客户端 A3 归组织 R3 管理。节点 P1、节点 P3、客户端 A1、客户端 A3 及排序服务 O 中的排序节点可以通过通道 C1 相互通信。

区块链网络中节点的主要作用就是托管账本和智能合约，供各类成员访问。如图 5-23 所示，节点 P1 和 P3 均托管了账本 L1 和智能合约 S1。可以认为账本在物理上托管于各个节点，但逻辑上托管于通道。例如，节点 P1 和节点 P3 都托管了账本 L1（存储于本地物理设备上），但通道 C1 中每个节点的账本 L1 都应该是完全一样的（逻辑上唯一），所以说账本逻辑上托管于通道。

当智能合约安装到节点上并在通道中实例化后，客户端便可以调用智能合约进行交易和数据访问。如图 5-23 所示，节点 P1 和节点 P3 均安装了智能合约 S1（实际每个节点可以安装多个智能合约，这里为了简化仅展示智能合约 S1），客户端 A1 或客户端 A3 便可以通

过调用智能合约进行交易或访问申请。

至此，已经建立了一个比较完整但相对简单的 Hyperledger Fabric 区块链网络。

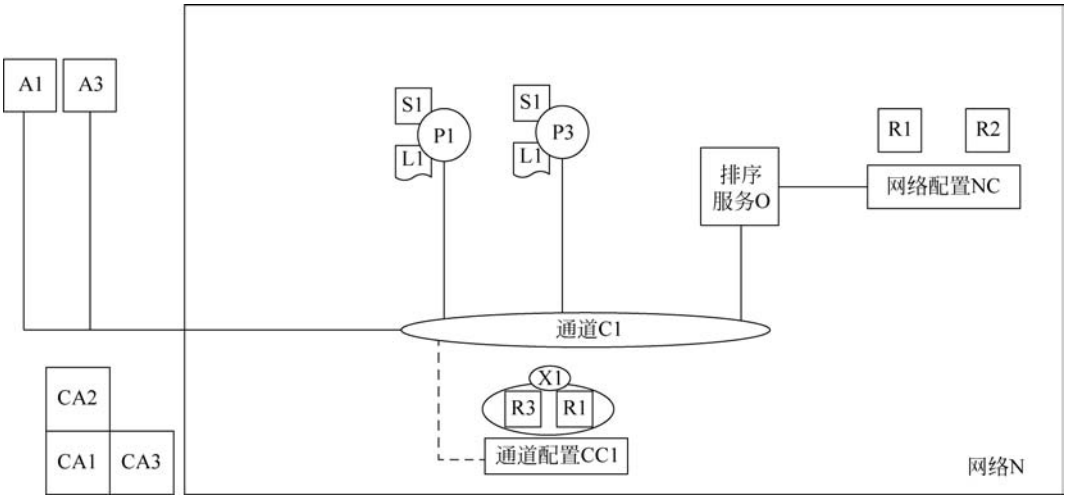


图 5-23　Fabric 网络构建——添加节点和客户端

5. 添加另一个联盟定义并创建通道

一个 Fabric 网络可以包含多个联盟和相对应的多条通道。目前网络 N 仅定义了一个联盟和一个通道，在本阶段将定义另一个联盟并为其创建专用通道。这样可以更直观地了解网络、联盟和通道之间的关系。

如图 5-24 所示，将组织 R4 添加进网络中，然后由网络管理员 R1 或 R2 在网络配置 NC 中添加新的联盟定义 X2，其中包括组织 R3 和 R4。接下来为联盟 X2 创建一个新的专用通

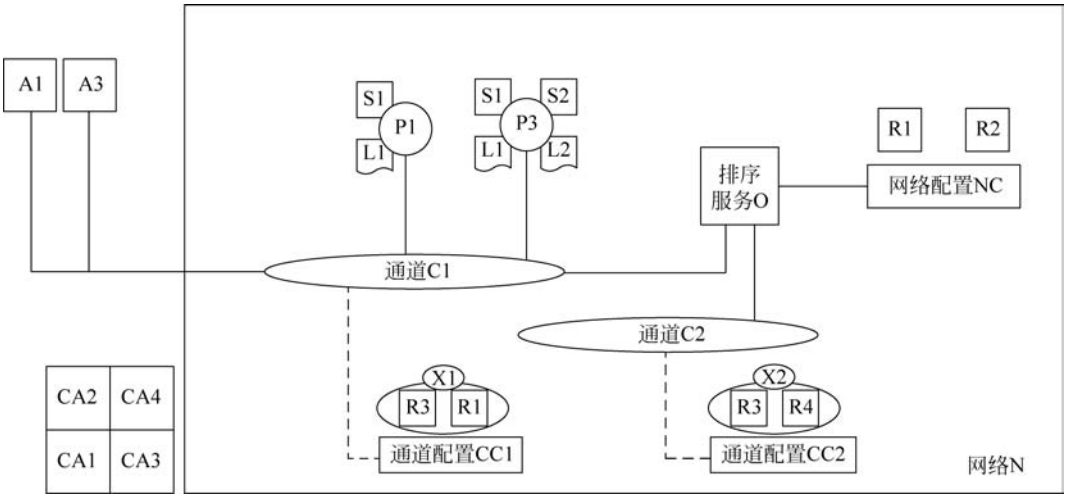


图 5-24　Fabric 网络构建——添加 X 2 联盟定义并创建通道

道 C2，该通道具有独立于网络配置 NC 的通道配置 CC2，该通道配置由组织 R3 和 R4 定义。再次强调，通道配置 CC2 并不受网络配置 NC 的直接控制，也和另一个通道配置 CC1 无关。CC2 仅由 R3 和 R4 制定，规定了通道 C2 的一些管理策略，也定义了 R3 和 R4 的管理权限。如果想要对 CC2 进行修改，则只能由组织 R3 或 R4 进行。

可以看出，一个网络中可以定义多个联盟，每个联盟有自己的专用通道，其他非联盟组织无权干预联盟通道内的管理和通信（即使是网络管理员）。这种模式体现了 Fabric 网络的去中心化性质。

6. 添加节点并将节点加入多个通道

定义完新联盟并为其创建专用通道后，本阶段将节点和客户端连接到通道，使其能进行正常通信。

如图 5-25 所示，与之前添加节点和客户端的过程一样，将属于组织 R4 的节点 P4 加入网络并将其连接到通道 C2，将属于组织 R4 的客户端 A4 也连接到通道 C2。到这里可以发现，组织 R3 比较特殊，它既是联盟 X1 的成员也是联盟 X2 的成员，而属于组织 R3 的节点 P3 也比较特殊，该节点加入了两个通道。在这里强调一点，一个节点是可以加入多个通道的，但是节点上不同通道的账本和智能合约是不能直接共通的，它们在节点内部是被隔离开的，不同通道的账本和智能合约可以交互但不是在一个节点内部进行。

节点 P3 和 P4 均加入了通道 C2，并托管了账本 L2 和智能合约 S2。客户端 A3 和 A4 连接到通道 C2 后，可以调用智能合约进行交易和访问。这里要强调的是，智能合约不仅要在节点上安装，还必须在通道上进行实例化之后才能被其他成员调用。同时，每个节点和通道可以安装多个智能合约，这里为了简化所以只展示了一个。

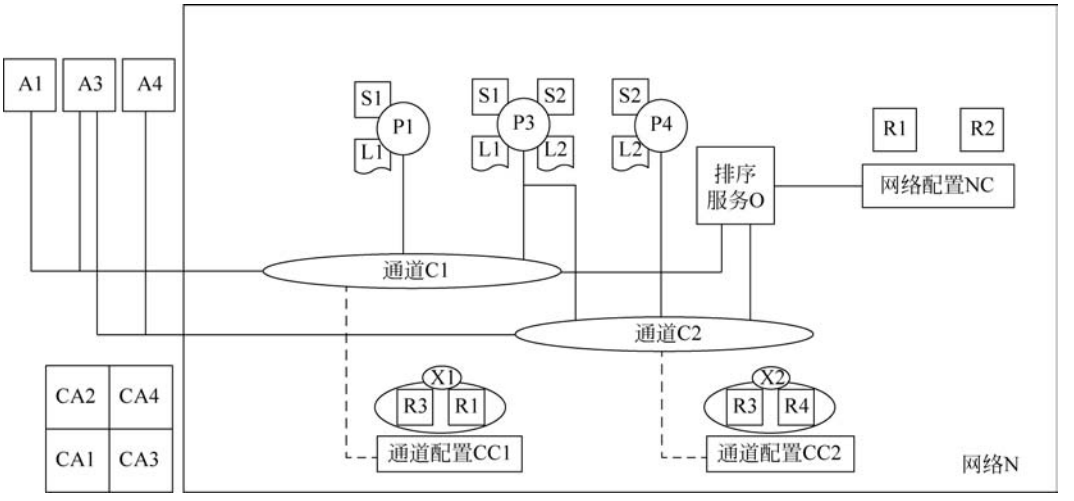


图 5-25　Fabric 网络构建——添加节点并将节点加入多个通道

至此，创建了一个具有两个通道、三个 peer 节点、两个智能合约和一个排序服务的四组

织网络。该网络由四个证书颁发机构支持，并为三个客户端提供账本和智能合约服务，客户端与节点通过两个通道分别进行交互。

5.5.5　Fabric 多机部署

Fabric 单机部署一般只用于学习和测试，实际应用时建议使用多机部署网络。这里说的多机部署网络，是指在真正的多台服务器上，使用配置和可执行文件来启动不同角色，以此构成的网络。

构建多机网络最少需要 2 台服务器，一台作为 orderer，一台作为 peer 节点。这里为了更贴近实际，使用 5 台服务器，一台作为 orderer，设置两个组织，每个组织拥有 2 台 peer 服务器，另外还需准备一台服务器作为客户端，并在每台设备上都安装 go 和 docker 软件。

首先，进行角色 IP 规划：

```
cli(客户端)IP: 192.168.2.5
orderer(排序节点)IP: 192.168.2.11
peer0.org1(组织 1peer 节点 0)IP: 192.168.2.21
peer1.org1(组织 1peer 节点 1)IP: 192.168.2.22
peer0.org2(组织 2peer 节点 0)IP: 192.168.2.31
peer1.org2(组织 2peer 节点 1)IP: 192.168.2.32
```

接下来配置域名解析，在所有机器的/etc/hosts 中追加以下信息：

```
192.168.2.11 orderer.example.com
192.168.2.21 peer0.org1.example.com
192.168.2.22 peer1.org1.example.com
192.168.2.31 peer0.org2.example.com
192.168.2.32 peer0.org2.example.com
```

关闭所有服务器的防火墙：

```
systemctl stop firewalld.service
systemctl disable firewalld.service
```

在启动各个服务器之前，要提前准备好 MSP、创世块及各个角色的相关配置文件等材料。这些材料均在客户端生成，然后按角色定义分配给各个服务器，注意私钥文件等隐私文件不要泄露。可以在 cli 中创建目录“/opt/work/deploy”，用于存放各类相关文件。

生成 MSP 文件的命令是 cryptogen，需提供配置文件 crypto-config.yaml：

```
OrdererOrgs:
  - Name: Orderer
    Domain: example.com
    Specs:
      - Hostname: orderer
PeerOrgs:
  - Name: Org1
```

```
    Domain: org1.example.com
    EnableNodeOUs: true
    Template:
      Count: 2
    Users:
      Count: 1
  - Name: Org2
    Domain: org2.example.com
    EnableNodeOUs: true
    Template:
      Count: 2
    Users:
      Count: 1
```

运行以下命令生成 MSP 材料：

```
bin/cryptogen generate --config=./crypto-config.yaml
```

运行结束后，会得到一个 crypto-config 目录，共有 109 个目录，107 个文件。可以用 tree 命令查看结果。接下来运行下列命令，生成创世块、通道配置块和两个锚节点更新文件。

```
mkdir channel-artifacts
# 生成创世块
bin/configtxgen -profile TwoOrgsOrdererGenesis -channelID byfn-sys-channel \
-outputBlock ./channel-artifacts/genesis.block
# 生成通道配置文件
bin/configtxgen -profile TwoOrgsChannel -outputCreateChannelTx \
./channel-artifacts/channel.tx -channelID mychannel
# 生成 Org1 的锚节点更新文件
bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate \
./channel-artifacts/Org1MSPanchors.tx -channelID mychannel \
-asOrg Org1MSP
# 生成 Org2 的锚节点更新文件
bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate \
./channel-artifacts/Org2MSPanchors.tx -channelID mychannel \
-asOrg Org2MSP
```

完成之后，会在 channel-artifacts 目录中生成 4 个文件：

```
channel.tx
genesis.block
Org1MSPanchors.tx
Org2MSPanchors.tx
```

接下来需要准备角色相关的配置文件。orderer 对应的配置文件名为“orderer.yaml”，可以到官方源代码中去复制，然后根据自身需求进行修改。完成后可以执行“start.sh”命令

启动 orderer 服务。所有的 peer 服务器对应的配置文件都称为“core.yaml”，同样可以从官方源代码中复制、调整。完成后执行“start.sh”启动服务。

客户端是指连接 peer 服务器的客户端，所以在使用客户端时，需指定待连接服务器的名称和 MSP 材料。本范例中使用的客户端是终端，建议为每个 peer 服务器准备独立的客户端目录，这样可以分开启动黑屏窗口作为客户端，互不干扰，使用起来比较方便。

后面的操作基本上都是在客户端中完成的，首先进入 peer0.org1.example.com 的客户端目录，运行第一条命令，查看客户端是否可用：

```
cd Admin@org1.example.com
./peer.sh channel list
```

检测完成后，执行下列命令创建通道。通道只要创建一次就行，不限制在哪个客户端中操作，创建命令完成之后，会生成通道创世块文件，默认命名为“通道名.block”。

```
cd Admin@org1.example.com
./peer.sh channel create -c mychannel -f ../channel-artifacts/channel.tx \
-o orderer.example.com:7050 --tls --cafile tlsca.example.com-cert.pem
```

命令中需要输入通道名称(-c)和通道配置文件(-f)。

完成后在当前目录中生成通道创世块文件“mychannel.block”，所有节点在加入该通道时都要用到该文件。接下来，在所有节点的客户端中，执行加入通道的命令：

```
./peer.sh channel join -b mychannel.block
```

参数输入只有一个。创建的通道是生成的通道创世块文件 mychannel.block，如果是在其他客户端目录下，需要输入正确的文件路径。命令执行成功将会返回“Successfully submitted proposal to join channel”的信息。

一个组织一般情况下设置一个锚节点，所以一个组织只需执行一次更新锚节点命令。

更新组织 1 锚节点：

```
cd Admin@org1.example.com
./peer.sh channel update -f ../channel-artifacts/Org1MSPanchors.tx \
-o orderer.example.com:7050 -c mychannel --tls \
--cafile tlsca.example.com-cert.pem
```

命令成功会返回“Successfully submitted channel update”信息。

至此，多机网络已经部署完成。如果想进一步验证网络是否可用正常工作，可以使用官方提供的测试智能合约进行检验。

5.5.6　超级账本智能合约开发

这里以超级账本 Fabric 1.1 和 Mac 系统为例，逐步安装超级账本的依赖库。Go 的版本要求大于 1.10，在 Mac 中可直接使用 Homebrew 安装，十分方便。这里比较特殊一些，NodeJs 9.0 以上的版本暂时不兼容，支持版本为 8.9 以上 9.0 以下。推荐使用 nvm 进行

NodeJs 的版本控制。

首先安装 nvm，安装命令为：

```
brew install nvm
```

nvm 安装成功后，可以用 nvm 安装 NodeJs：

```
nvm install 8.10
```

最后，切换 NodeJs 版本：

```
nvm use 8.10
```

切换到自己的工作区，运行下面的命令下载 Fabric 相关的运行包：

```
curl -sSL https://goo.gl/6wtTN5 | bash -s 1.1.0
```

这里下载的文件包含一些节点镜像，文件体积较大，可能需要等待较长时间。

安装完成后，启动网络。

切换到例程所在目录：

```
cd fabric-samples/first-network
```

生成网络配置文件：

```
./byfn.sh -m generate
```

启动网络：

```
./byfn.sh -m up
```

Chaincode 是部署在 Hyperledger Fabric 网络节点上，可以用来与分布式账本进行交互的一段代码。Chaincode 在验证节点上的隔离沙盒（Docker 容器）中执行，并通过 gRPC 协议被相应的验证节点或客户端调用和查询。

下面以一个简单的例程进行说明。

```
func (t *Chaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    function, args := stub.GetFunctionAndParameters()
    switch function {
    case "creator":
        return t.creator(stub, args)
    case "creator2":
        return t.creator2(stub, args)
    case "call":
        return t.call(stub, args)
    case "append":
        return t.append(stub, args)
    case "attr":
        return t.attr(stub, args)
```

```
    case "query":
        if len(args) != 1 {
            return shim.Error("parametes's number is wrong")
        }
        return t.query(stub, args[0])
    case "history":
        if len(args) != 1 {
            return shim.Error("parametes's number is wrong")
        }
        return t.history(stub, args[0])
    case "write": //写入
        if len(args) != 2 {
            return shim.Error("parametes's number is wrong")
        }
        return t.write(stub, args[0], args[1])
    case "query_chaincode":
        if len(args) != 2 {
            return shim.Error("parametes's number is wrong")
        }
        return t.query_chaincode(stub, args[0], args[1])
    case "write_chaincode":
        if len(args) != 3 {
            return shim.Error("parametes's number is wrong")
        }
        return t.write_chaincode(stub, args[0], args[1], args[2])
    default:
        return shim.Error("Invalid invoke function name.")
    }
}
```

以上合约代码可以通过以下命令来获取。

```
mkdir -p $GOPATH/github.com/introclass
cd $GOPATH/github.com/introclass
githttps://github.com/introclass/hyperledger-fabric-chaincodes.git
```

下载完成后，通过以下命令进行安装。

```
cd /opt/app/fabric/cli/user/member1.example.com/Admin-peer0.member1.example.com
./3_install_chaincode.sh
```

超级账本可以采用多种语言进行来发，包括Go、Java等，相比于以太坊开发过程相对复杂。超级账本和以太坊也有相似的地方，都允许合约的相互调用。总的来说，超级账本的资料比较少，能够参考的企业级项目也很少，但是有IBM等公司的技术支持，超级账本目前的问题也会被解决。

5.6 区块链即服务

区块链即服务(blockchain as a service,BaaS)是一种帮助用户搭建、维护及管理企业级区块链网络和应用的服务平台。BaaS将计算、通信和存储等资源整合,连同上层的区块链记账功能、应用开发功能、配套设施功能一起转换为可编程接口,让区块链应用的开发及部署过程变得简单高效。同时将网络构建标准化,保障区块链应用的安全可靠,为区块链业务的稳定运营提供有力支撑,最大程度地缓解运营过程中出现的弹性不足、安全性较低及性能不够高等难题,让开发者可以专注开发。

BaaS服务形态极大地加速了区块链技术在各行业的落地应用,促进实体经济与区块链深度融合。区块链云服务是BaaS中最流行的模式,也可以将BaaS狭义地比作区块链云服务。

如图5-26所示,在整个区块链云服务体系中,底层最基础的是IaaS(infrastructure as a service,基础设施即服务),将CPU、内存和网络等计算资源作为服务提供给消费者,用户可以在这些基础设施上自行部署和运行软件。PaaS(platform as a service,区块链平台即服务),提供多种区块链底层平台,如联盟链Fabric、公有链Ethereum等。用户可以直接基于底层区块链平台进行应用开发,而无须费力搭建平台。位于顶层的是SaaS(software as a service,软件即服务),将区块链技术的具体软件应用作为服务提供给用户,用户可在各种设备上直接访问,而不需要管理控制任何云计算基础设备。

图5-26 区块链云服务体系

BaaS本质上是区块链设施的云端租用平台,因其特性使得计算、平台、软件等资源得到了最大程度的利用和共享。BaaS不仅可以提供节点租用、通道租用及工具租用等服务,必要时还将提供关键技术支持。BaaS的具体服务能力包括区块链节点及整链搭建能力、区块链应用开发能力、区块应用部署能力及区块链运行监控能力。

BaaS致力于提供企业级区块链基础设施、技术平台,基于面向服务的基础设计原则。BaaS具有简单易用、扩展灵活、安全可靠、云链结合、运维可视及合作开放等特点,能够为用户低成本、高效地搭建安全、灵活、可靠的企业级区块链。

5.6.1　BaaS 总体架构

BaaS 总体架构主要分为管理平台和运行态两个部分。如图 5-27 所示，BaaS 管理平台主要分为资源管理、区块链管理和平台管理三个部分。

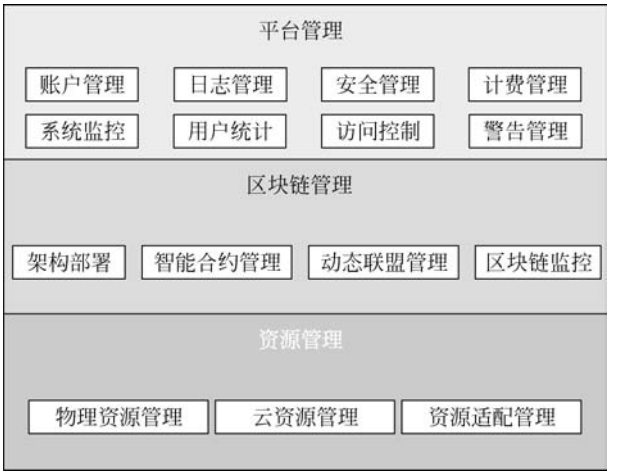


图 5-27　BaaS 管理平台

资源管理是对底层资源的管理，其中包括云资源管理及物理资源管理等。

区块链管理是针对区块链技术组件的管理，如智能合约管理、动态联盟管理、架构部署及区块链监控。

平台管理为使用 BaaS 的用户提供一些通用的管理服务，其中包括日志管理、账户管理、安全管理及计费管理等常用服务。

如图 5-28 所示，BaaS 的运行态主要包括四个层面：资源层、基础层、业务层及应用层。

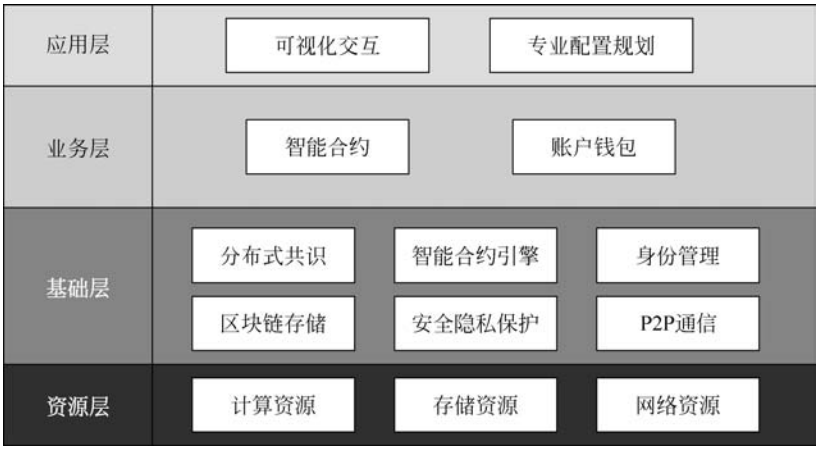


图 5-28　BaaS 运行态

位于底层的资源层包括计算资源、存储资源、网络资源等 IaaS 服务，为上层区块链基础提供强大的计算能力、巨大的存储空间及高速的网络，支撑整体运行态。

区块链基础层提供各类区块链技术组件，其中包括开源的 Hyperledger Fabric、Corda 及非开源的蚂蚁区块链、TrustSql 等区块链架构，为上层业务应用提供高性能、低成本、安全可靠的企业级区块链系统。

业务层向用户提供标准智能合约接口，用户可以根据实际应用环境构建专门的智能合约。

应用层为用户提供一些安全、快捷、可靠的区块链应用。

5.6.2 BaaS 基本模块设计

BaaS 的设计主要分为两部分：管理平台设计及底层关键技术设计。

1. 管理平台设计

如图 5-27 所示，BaaS 管理平台主要分为三个层次：资源管理层、区块链管理层及平台管理层。资源管理层中各个模块均是管理基础资源的，如计算力、网络、云平台等，主要与 IaaS 云平台进行交互。区块链管理层的服务面向的是平台需求用户，负责提供区块链平台的创建、管理及安全监控等，且一般情况下该层支持多种底层区块链。平台管理层的各个模块向用户和平台管理员提供常用的日志、计费和账户等管理功能。

在资源管理层中，物理资源管理模块负责管理 CPU、服务器等物理设备；云资源管理模块负责实现云资源的管理调度；资源适配管理模块负责各种资源调度 API 的封装，为区块链管理层各模块提供统一的资源管理接口。

在区块链管理层中，架构部署模块负责构建区块链系统整体架构，其中包括对各类节点的安装部署、配置和软件升级等操作；智能合约管理模块主要对智能合约的上传、安装、实例化、升级、调用权限等功能进行管理，某些区块链平台提供智能合约购买服务，用户可以在智能合约商店直接购买；动态联盟管理模块主要应用于联盟链场景，动态联盟管理包括联盟链的创建、新成员的加入、策略设置等功能；区块链监控模块负责对区块链网络和节点的运行状况进行监控。

在平台管理层中，账户管理一般分为管理员账户和用户账户，管理员账户用于管理平台自身设置，用户账户由客户注册后根据需要创建区块链。

2. 底层关键技术设计

如图 5-29 所示，区块链底层技术架构可分为三个部分：基础设施、技术组件及应用组件。其中，基础设施包括开发环境、运营管理和测试环境等；技术组件包括共识机制、数据存储、安全通信和交易模型等；应用组件主要指智能合约及成员管理等上层应用。

作为 BaaS 底层技术模块的核心，技术组件主要包括可插拔的共识机制、多类型数据存储支持、加密安全通信及区块链网络交易模型。

可插拔的共识机制：所谓共识是指多个节点对网络中某些操作、流程达成一致，而共识机制就是达成共识所用到的算法及规则。区块链服务技术架构中需要支持可拔插的共识机

制，用户可根据自身需求选择共识算法。目前比较常用的共识算法有 PoW、PoS、DPoS、PBFT、Raft 等。

图 5-29　区块链底层技术架构

多类型数据存储支持：区块链账本的分布式存储是指每个节点都维护逻辑唯一的账本的副本。其中，每个节点都按照特定的存储模式存储完整的交易数据，传统区块链的存储模式是块链式存储。但随着区块链 3.0 时代的到来，某些新型区块链项目提出了非块链式的存储模型，其中最典型的就是有向无环图（DAG），该模式可以有效解决安全、高并发、可扩展性低和数据量过大等问题。某些区块链还使用数据库来保存账本状态当前值，如以太坊和 Fabric，目前比较流行的数据库有 LevelDB 和 CouchDB。

加密安全通信：作为区块链的底层传输模式，P2P（peer-to-peer，点对点）是无中心、依靠对等节点交换信息的互联网通信技术。P2P 网络具有较强的并行处理能力和较高的信息交换效率，同时设备资源消耗也比较低，可以满足区块链节点的多项通信需求。区块链系统还使用多种密码学技术对数据进行加密保护，如使用哈希算法将不能明文的数据转换为哈希值进行记录，保护数据隐私安全；使用非对称加密技术，产生公私钥对进行加密数据传输，保证数据在通信过程的安全；采用零知识证明对某些无法直接展示的信息进行安全验证。

区块链网络交易模型：目前主要存在两种区块链网络交易模型，一个是以比特币为代

表的 UTXO(unspent transaction output,未花费的交易输出)模型,另一个是账户/余额模型。账户/余额模型是生活中最常见的一种模式,银行使用的就是这种模式。UTXO 模式没有余额概念,只有一系列交易记录,通过追溯交易记录判断是否有能力进行目前的交易。

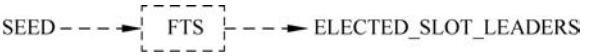
随着区块链技术的推广普及,越来越多的企业注意到其商业价值并开始尝试将区块链技术应用到自身业务中。但是一些中小型企业在开发区块链技术的过程中常常遇到技术门槛高、安装部署复杂、管理运维成本高等问题。为了解决这些问题,加快区块链的落地应用,占据未来市场,很多大型厂商开始推出各种类型的区块链服务平台,提供区块链基础设施和服务,打造企业级的区块链产业生态。

5.7 小结

本章介绍了区块链 3.0 相关的项目和技术方案,解决交易速度问题,使得区块链技术更好地落地应用。EOS 通过 DPoS 共识机制替代传统的 PoW 机制,减少节点的竞争从而加快交易速度；Zilliqa 是通过分片技术提高并行处理速度；Cardano 通过权益证明机制实现价值转换；超级账本以联盟链的形式推动区块链跨行业应用与协作,通过框架方法和专用模块提高可扩展性,用开放协议和标准发展性能和可靠性。BaaS 平台为企业级区块链应用提供基础设施和服务,解决使用区块链过程中的各种问题,加快企业区块链业务的落地速度。这些项目都代表着目前区块链的最新技术和未来发展方向。

思考题

1. 区块链的演进路线分为哪几个阶段？每个阶段各有什么特点？
2. 区块链 2.0 面临什么问题？
3. 商用操作系统有什么主要特点？
4. 商用操作系统是否为去中心化的区块链技术？
5. 商用操作系统采用的 DPoS 算法和比特币采用的 PoW 算法在效率上有什么区别？
6. DPoS 的时间片机制有什么作用？
7. Cardano 是一个开源的、分散的公共区块链和加密货币项目,它的目标是什么？
8. 如何理解 Cardano 的“区域自治”？
9. ADA 的分层与 EOS 的分片技术这两种概念有何区别？
10. 如何理解 Cardano 系统中的下述过程。



11. 简述 Zil 网络中分片技术的流程。
12. 为了统筹和管理所有的分片,Zilliqa 提出目录服务委员会,那么 DS Block 和交易区块有什么相同点和不同点？

13. Zil 在交易过程中，如果领导者是诚实的，它可以不断推动共识小组中的节点就新的交易达成协议，如果领导是拜占庭会怎么样？

14. Fabric 架构在 1.0 正式版中的主要改变是什么？

15. 简述 Fabric 的特点、架构和运行机制。

16. 简述 Fabric 中的“多通道”功能。