第 5 章

分布式系统与比特币网络

在第 2 章我们已经了解到，区块链其实就是一个通过哈希指针把各个区块关联起来、不可被篡改、同时也能被追踪溯源的链条。那么这个链条存储在哪里呢？在比特币网络中，不断生成的交易和区块又是通过怎样的方式在全网传播的呢？本章将介绍比特币系统另外两个重要的基本特征：分布式存储和点对点网络。这两个特性会帮助我们解答上述疑问。

5.1 分布式系统架构

在互联网高速发展的时代，各种类型的分布式系统被广泛应用。例如，搜索引擎、网上购物、社交媒体以及短视频等，背后都是由分布式系统支撑着日常业务的高效有序运转。可以说凡是涉及大规模数据的高并发量处理，都会不约而同地选择分布式系统。

在介绍分布式系统之前，我们先简单了解与之对应的集中式系统。

所谓集中式系统，顾名思义，就是集中处理，它有一个高性能和可扩充的中央处理系统，所有的数据控制、处理、运算甚至存储等任务都在中央处理系统上完成，与之连接的是各个终端设备，终端设备没有任何数据处理能力，只是用于数据的输入和处理结果的呈现。以我们在医院体检为例，如图 5-1 所示，体检的各个项目如果都集中在一个房间完成，就是集中式，这种体检模式非常低效。

虽然效率很低，但集中式系统也有它独特的优点：

（1）部署简单。一台或多台计算机组成中心节点，整个系统的业务和数据都集中部署在这个中心节点上。

（2）维护方便。中心节点往往都是性能优越的服务器，这类服务器在系统稳定性方面表现卓越，所以只需要维护好中心节点，不需要关心多节点部署，以及多节点之间负载均衡[①]、协同工作等问题。

集中式系统的缺点也很明显：

① 一种技术手段，将服务或业务分担给多台网络设备或多条链路，从而提高业务处理能力，保证业务的高可靠性。

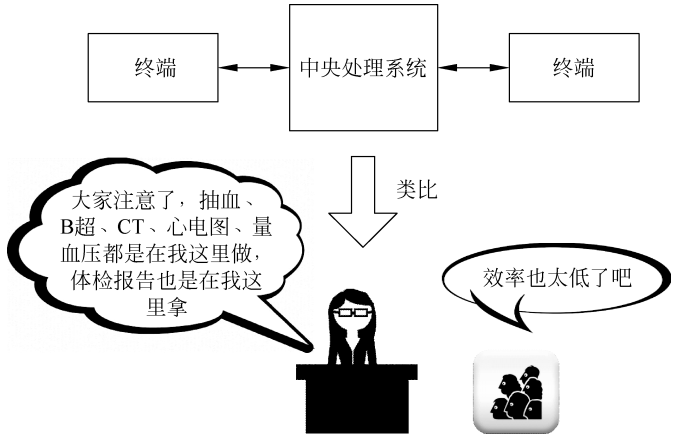


图 5-1　集中式系统示意图

（1）中心依赖性强。如果中央处理系统出现故障，那么整个系统也就崩溃了。

（2）可扩展性差。随着业务的不断发展，数据量不断增加，集中式系统升级扩容往往比较困难。

（3）安全性差。如果重要文件都存储在中央处理系统里，那么系统一旦出现不可修复的故障，文件自然也就损失了。

（4）可伸缩性差。数据量多时虽然可以通过升级硬件来解决，但是一旦数据量下降，之前升级的设备就浪费了。

（5）响应性差。集中式系统通常都是固定处理同一类任务，但是如果终端用户有特殊任务需求，那么需要对用户的程序和资源做单独的配置，这在集中式系统上实现起来很困难且效率不高。

那么什么是分布式系统？《分布式系统概念与设计》一书对分布式系统作了如下定义："分布式系统是一个硬件或软件组件分布在不同的网络计算机上，彼此之间仅仅通过消息传递进行通信和协调的系统。"

由上面的描述可知，分布式系统的本质就是建立在网络之上的计算机软硬件系统。首先，它是基于网络互联的；其次，在硬件层面上它由若干台独立的计算机组成，这些计算机是相互协同工作的，在软件层面上它就是建立在网络之上的支持分布式处理的软件系统，包括分布式操作系统、分布式程序设计语言、分布式文件系统、分布式存储系统等。

在一个分布式系统中，一组独立的计算机对于用户来说是一个统一的整体，用户不需要关心内部组织结构。各台计算机之间的差别，以及计算机之间网络通信方式等也都对用户屏蔽，用户就像在使用一种统一的方式与分布式系统进行交互，最终由这组计算机互相配合完成一个共同的目标。分布式系统如图 5-2 所示，好比体检的各个项目被分散到不同的房间，每个人根据体检中心的调度，有秩序地进入不同的房间，完成不同的体检项目，这样不会像集中式那样造成拥堵，从而高效地完成整个体检过程。

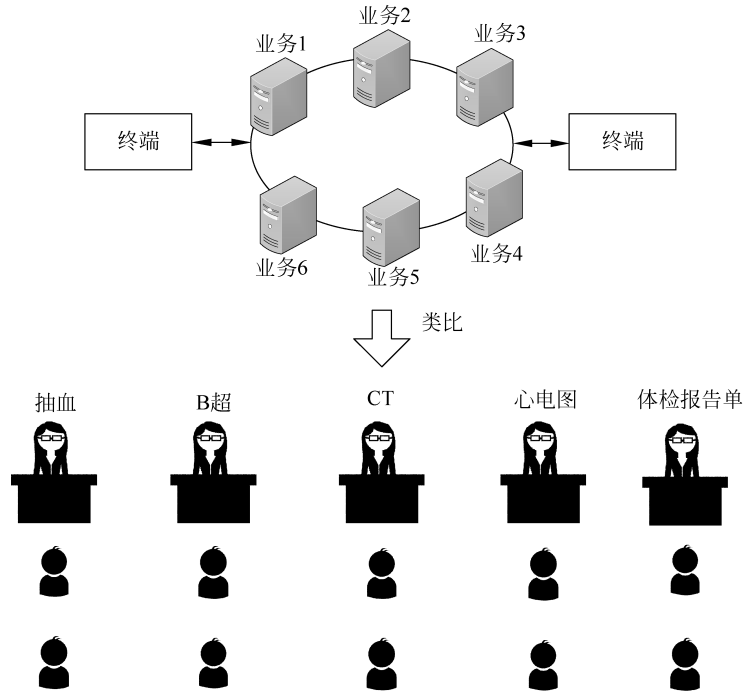


图 5-2　分布式系统示意图

分布式系统的优点：

（1）分布性：各台计算机在空间上是随意分布的，且其分布情况根据需要随时变动。

（2）对等性：计算机没有主和从之分，也就是说没有控制整个系统的中央计算机，也没有被控制的从计算机，在整个系统中计算机所构成的节点都是对等的，每个节点都有自己的中央处理系统、数据库等。

（3）容错性：分布式系统为数据提供了一种容错方式，即对数据进行副本（Replica）处理。所谓的数据副本处理是指在不同的节点持久保存同一份数据（即副本），如果某个节点出现故障导致数据丢失，或者新的节点加入系统，又或者节点因为网络问题脱离系统后再次加入系统，这些情况下都需要更新数据，此时就可以从其他节点的副本获取最新的数据。可以说，数据副本处理是分布式系统中解决数据丢失或数据更新的唯一手段。另外，分布式系统也为任务提供容错机制，与数据容错一样，存在一类任务副本，即多个节点提供相同的任务，如果有节点出现故障，那么其他节点可以接替该节点接受外部请求并做相应的处理。

（4）响应性好：每个节点可以独立地从事特定任务，这样方便接受多样化的外围终端处理请求接入。

（5）负载均衡：可以将负载从单个节点转移到多个节点来进行协同工作，从而提高效率。

万事万物都没有绝对的完美，分布式系统也一样存在缺点：

（1）部署复杂：部署单个任务比较快，但是如果一次性部署多个任务就比较复杂。

（2）需要有高效的协同工作机制，使得任务既能集中处理，也能分散处理。如果数据分散在各个计算机上，那么保证这些计算机之间的数据一致性也有很大难度。

（3）安全性差：对病毒比较敏感，一旦一个数据文件感染了病毒，那么可能因为副本处理而造成病毒在整个网络中扩散；此外，如果数据分散备份在分布式系统中，而非统一存储在一个中央存储系统，那么很难进行有效备份。

虽然分布式系统存在上述缺点，但是随着互联网业务的高速发展和智能终端的普及，产生了大量实时数据流，不断地冲击着现有系统的处理极限，各种应用在面对挑战时，无一例外地都会采用分布式系统。

延伸阅读

分布式和集群的关系

集群的概念：将一组松散的计算机软硬件连接起来高度紧密地协作完成计算工作。

有人会认为分布式系统同样由一组计算机共同完成计算工作，计算工作对用户来说是透明的，用户可以把分布式系统看作一台计算机系统，那么分布式系统不就是集群吗？实际上并不是这样。

集群是一种物理形态，分布式是一种工作方式，虽然两者都是由若干计算机组成的，但是有着本质区别。具体来说，集群一般是物理集中[①]、统一管理的一组计算机系统，实现同一业务，比如某服务提供商为了提前应对“双十一”可能出现的瞬时高并发访问请求，将若干台计算机搭建在一起构成集群，在每台计算机上部署相同的服务程序，这样可以分担来自客户端的访问请求压力；分布式系统主要解决中心化管理的问题，当所有的任务都叠加在一台计算机上处理时，其效率很低，同时有些任务是单台计算机无法独立处理的，这就需要融合多台计算机性能加以解决。分布式系统就是把一个任务拆分成若干子任务，并分配给不同的计算机来处理，同时要协调好计算机之间的通信。简单来说，集群通过堆加硬件资源来解决任务，对于任务本身不做任何拆解；分布式则是把一个任务分解为多个子任务，然后把子任务分配给不同的计算机来解决，最终由这些计算机协同完成这个任务。由此看来，集群通过提高单位时间内执行的任务数来提高效率，而分布式通过缩短单个任务的执行时间来提高效率。

此外，构成集群的计算机之间基本不需要通信协调，而分布式则需要协调各台计算机之间的通信，因为某台计算机的运行结果或运行状态需要通知其他计算机。集群里某台计算机出现故障，不会影响其他计算机提供服务，但是如果分布式里某台计算机出现故障，那么整个计算机组都会受到影响。

目前在实际应用中，分布式和集群有时会融合在一起，比如分布式的每个子任务可以用

① 所谓物理集中就是把若干台计算机资源堆积在一起。

集群来提高效率，而若干分布式系统也可以组成一个集群。

举例来看：学校举行考试，每个考场只有一位监考老师，验证件、发考卷和监考工作全由她来做，如图 5-3 所示。

后来考生多了，一位监考老师忙不过来，于是又安排一位监考老师做同样的事，如图 5-4 所示，这两位监考老师的关系就是集群。

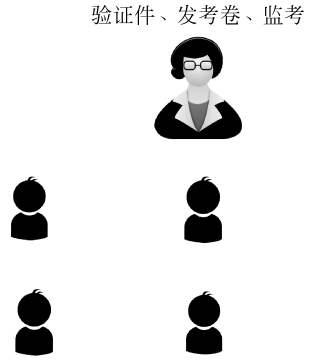


图 5-3　考场只有一位监考老师

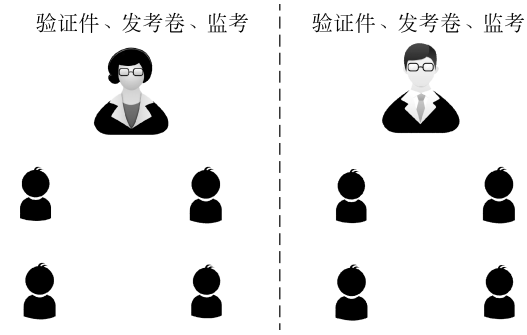


图 5-4　考场有两位监考老师

为了防止学生在验证件和发考卷的时候交头接耳，同时保障考试最核心的监考工作做到位，学校又安排了专门负责验证件和发考卷的考前老师，整个监考工作由考前老师和监考老师共同完成，这样考前老师和监考老师的关系就是分布式。如果一位考前老师忙不过来，那么请两位考前老师，这两位考前老师的关系就是集群，如图 5-5 所示。

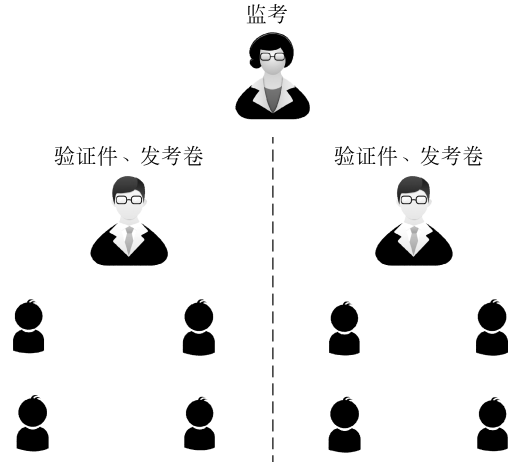


图 5-5　考场老师的“集群”和“分布式”

分布式系统的架构可以分为以下几种。

1. C/S 和 B/S 架构

C/S(Client/Server，客户端/服务器)架构如图 5-6 所示，服务器就是提供服务的端，其

内部有实现某种特定服务的进程，服务器根据提供服务的不同可以分为文件服务器、数据库服务器、Web 服务器等。客户端是请求服务的端，也就是用户使用的端，用户通过客户端向服务器发出请求，随后等待服务器的反馈。如图 5-6 所示，客户想下载某个文件，首先通过客户端向服务器发送下载文件请求，服务器响应该请求，将对应的文件下发给客户端。

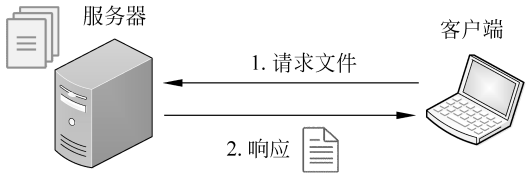


图 5-6　C/S 架构

B/S(Browser/Server，浏览器/服务器)架构类似于 C/S 架构，两者本质上都是服务器和客户端之间通信，只是表现形式不同，B/S 架构的客户端固定为浏览器，我们通过浏览器上网其实就是采用的 B/S 架构。

看到这里，可能有人会有疑问，C/S(B/S)架构怎么跟分布式有关呢？分布式不是由一组计算机构成的吗？事实上，C/S(B/S)这种简单的由两台计算机构成的系统也属于分布式，因为客户端完成请求和显示任务，服务器完成响应和计算任务，这样一个服务的最终实现是由两台计算机协同工作完成的。

下面我们举例说明。每年全国研究生入学考试初试成绩的公布都会牵动几百万学子的心，在网上发布成绩时会出现瞬时高并发查分请求，教育部门为了应对这种突发情况，将 Web 服务部署在若干台计算机上，且每台计算机都完成同样的 Web 服务，这些 Web 服务器就组成了集群。当 Web 内容发生更新时，会将更新同步到每台 Web 服务器上。如果有大量并发访问服务出现，系统会根据负载均衡策略将这些接入请求平均转发到各个 Web 服务器上，浏览器和与之建立连接的 Web 服务器就组成了分布式架构，如图 5-7 所示。

2. 分布式对象架构

在分布式对象架构中，所有的计算机个体不存在特定的客户端或服务器功能，每台计算机既可以作为服务端接收处理其他计算机发来的业务请求，又可以作为客户端向其他计算机发送业务请求，计算机之间通过一种称为对象请求代理的中间件[①]实现相互之间的通信，对象请求代理能在对象(即计算机)之间建立客户端/服务器联系。分布式对象架构如图 5-8 所示。

3. 面向服务的架构

面向服务的架构(Service-Oriented Architecture，SOA)是一个组件模型，它将应用程序的不同功能单元(又称为服务)通过这些服务之间定义好的接口和契约联系起来，接口应该独立于提供服务的软硬件平台，这样不同系统中的服务可以以统一和通用的方式进行交

① 所谓的中间件，是指一种独立的系统软件或服务程序。

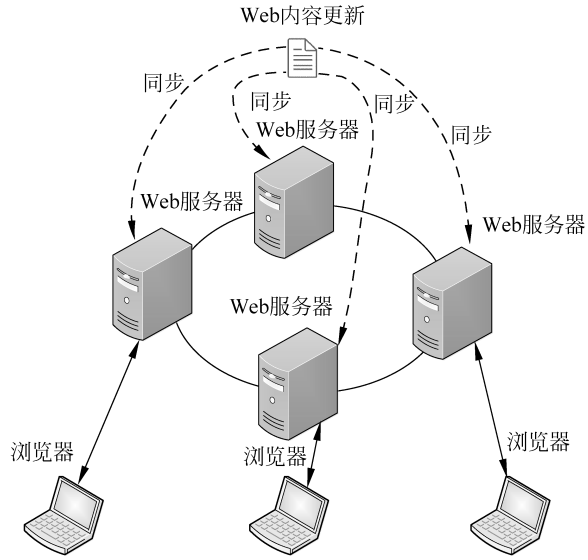


图 5-7　B/S 分布式架构示意图

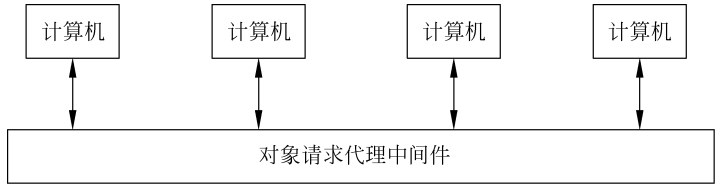


图 5-8　分布式对象架构

互。SOA 架构和工作流程如图 5-9 所示，其中服务是指可实现某种功能的单元，一个服务通常以独立的形式存在于操作系统进程中，多个服务之间通过配合最终提供一系列功能；服务描述定义了可提供的服务功能，指定服务请求者和服务提供者之间的交互方式。

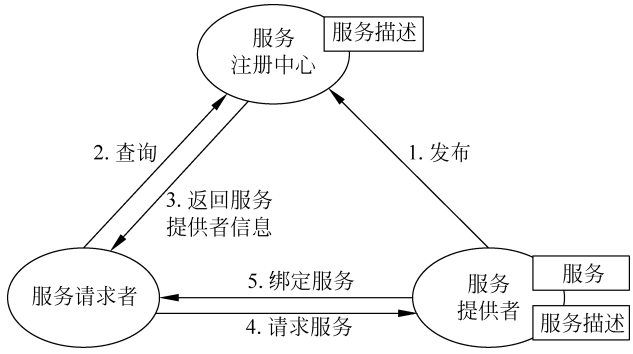


图 5-9　SOA 架构及其工作流程

由图 5-9 可知，SOA 架构中有三个角色：

(1) 服务提供者。一个可以通过网络寻址发现的实体，它主要实现服务功能，用 WSDL

(Web Services Description Language)描述服务，并用 UDDI(Universal Description Discovery and Integration)在服务注册中心发布 WSDL 文档，另外对服务的请求进行响应。

(2) 服务注册中心。一个接收并存储服务描述的实体，为服务请求者提供搜索服务。这里的服务描述是在服务注册表里存储的提供服务的 WSDL 文档。

(3) 服务请求者。利用服务注册中心搜索所需要的服务描述，即通过访问服务注册表发现所需服务的 WSDL 文档，在 WSDL 文档基础上通过 SOAP 协议与要访问的服务提供者通信，然后使用服务。

SOA 的基本流程如下：

(1) 由服务提供者发布自己的服务描述到服务注册中心，这样方便服务请求者发现并调用它。

(2) 服务请求者通过查询服务注册中心来寻找满足其需求的服务描述。

(3) 如果服务注册中心有满足其需求的服务描述，就将服务提供者信息返回给服务请求者。

(4) 服务请求者根据服务注册中心返回的信息给相应的服务提供者发起服务请求。

(5) 服务提供者响应请求，并绑定服务给服务请求者，这样服务请求者就能调用服务。

4. 点对点网络架构

点对点(Peer-to-Peer，P2P)网络，又称端对端网络或对等网络，一般简称为 P2P 网络，它也是一种分布式网络架构。P2P 网络中的所有节点都处于对等的地位，没有主从之分，每个节点既可以充当服务器，为其他节点提供资源；也可以充当客户端，从其他节点那里享受资源。这些资源包括信息的共享、计算资源共享、存储空间共享、网络共享等。整个网络一般不依赖中心服务器，所有节点通过 P2P 协议实现点对点通信并通过特定协议实现相互之间资源的共享。

P2P 网络架构与前面介绍的分布式对象架构一样，都不存在客户端和服务器的界限，两者的区别在于分布式对象架构需要对象请求代理的中间件来实现节点之间的相互通信，而 P2P 网络则允许节点之间直接通信并共享资源。

P2P 网络根据其结构的不同，可以分为集中式 P2P、纯分布式 P2P、混合式 P2P 和结构化 P2P 这四种类型。

(1) 集中式 P2P。它存在一个类似中心服务器的索引服务器，这个服务器独立于 P2P 网络中的其他节点，它只负责保存节点的索引信息，包括路由信息(IP 地址和端口号)和节点资源等，其他节点都与索引服务器相连。当一个节点要访问某个资源时，请求索引服务器，索引服务器将资源所在的路由信息返回给请求节点，然后请求节点根据路由信息直接访问资源所在的节点。需要说明的是，索引服务器只保存资源索引信息，并不存储其他资源。集中式 P2P 的优点是结构简单、容易实现，它实现了资源查询和资源传输的分离，能够有效节省索引服务器的带宽消耗，并且减少资源传输的时延。但其缺点也很明显，首先，这种类型本质上还是存在一个类中心，即索引服务器。该索引服务器需要存储所有节点的资源信息，当节点规模扩大的时候，就很容易出现性能瓶颈，比如搜索资源的性能、存储更多资源的性能等；其次，如果索引服务器崩溃，那么节点之间无法相互访问资源，整个 P2P 网络也就

瘫痪了。集中式 P2P 流程图如图 5-10 所示，在线活跃的节点资源信息被存入索引服务器，当某节点需要一项资源时，向索引服务器发送查询请求；索引服务器检测和查询后，将符合条件的资源所属节点信息列表返回给请求节点；请求节点收到列表，根据当前的网络流量和延时等信息选择合适的节点，并向该节点发起连接请求；资源所属节点接收连接请求后，两个节点之间就开始共享资源了。集中式 P2P 是最早出现的 P2P 应用模式。

（2）纯分布式 P2P。相对于集中式 P2P，纯分布式 P2P 没有所谓的索引服务器，每个节点是完全对等的，节点之间的连接也是任意的，所以其拓扑结构是随机的、没有规则的。纯分布式 P2P 结构如图 5-11 所示。如果一个新节点想要加入这个网络，最简单的方法就是随机选择网络中的一个节点，并与之建立邻居关系，接着向邻居节点广播消息，邻居节点收到广播消息后，继续发送给自己的邻居节点，这样呈波浪状地将消息在全网广播，让整个网络知道这个新加入节点的存在，这样新节点就通过这个邻居节点与其他节点关联起来了，这种广播方式又称为泛洪机制。但是泛洪机制存在两个较大的问题：一是容易形成泛洪循环，即发起广播的节点最后又收到该广播信息，从而形成一个闭环；二是如果请求节点所需的资源在很多节点都有，那么这些节点会同时给请求节点发送响应信息，这样请求节点如果处理不过来就很容易发生拥塞，造成节点瘫痪。

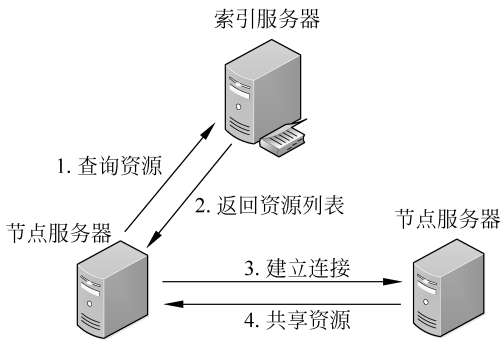


图 5-10　集中式 P2P 流程图

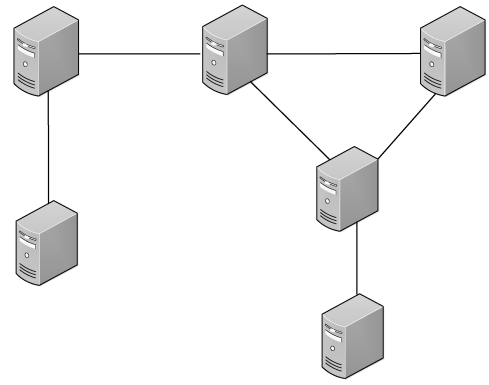


图 5-11　纯分布式 P2P 网络架构图

（3）混合式 P2P。混合式 P2P 其实就是混合了前面介绍的集中式 P2P 和纯分布式 P2P 结构，如图 5-12 所示。混合式 P2P 中的节点分为超级节点和普通节点两种类型，其中超级节点之间组成纯分布式 P2P 网络，每个超级节点与若干个与之相连的普通节点构成局部的集中式 P2P 网络。当一个新的普通节点需要加入混合式 P2P 网络时，它首先会随机选择一个超级节点进行通信，然后该超级节点会把其他超级节点列表信息推送给该普通节点，该普通节点根据超级节点列表信息中的超级节点状态来决定具体哪个超级节点作为其最终加入的父节点。这种加入方式所引起的泛洪广播只是在超级节点之间发生，不会扩大到系统中所有的普通节点，从而尽力避免了泛洪机制存在的问题。目前在实际应用中，大多数系统都是基于混合式 P2P 结构搭建的。

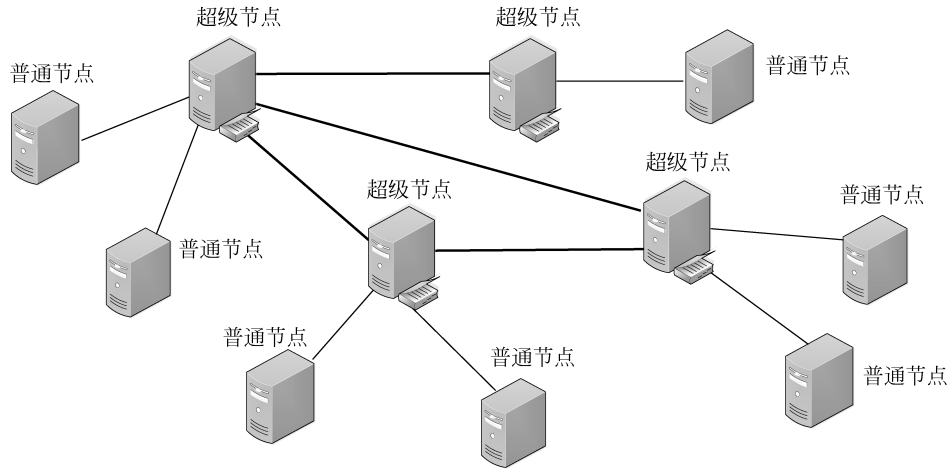


图 5-12　混合式 P2P 网络架构图

混合式 P2P 的流程图如图 5-13 所示，普通节点向与之相连的超级节点发送资源查询信息，如果该超级节点没有相应的资源信息，则向其他超级节点转发此请求，直到找到所需资源为止，最后请求节点与资源所在节点直接通信来共享资源。

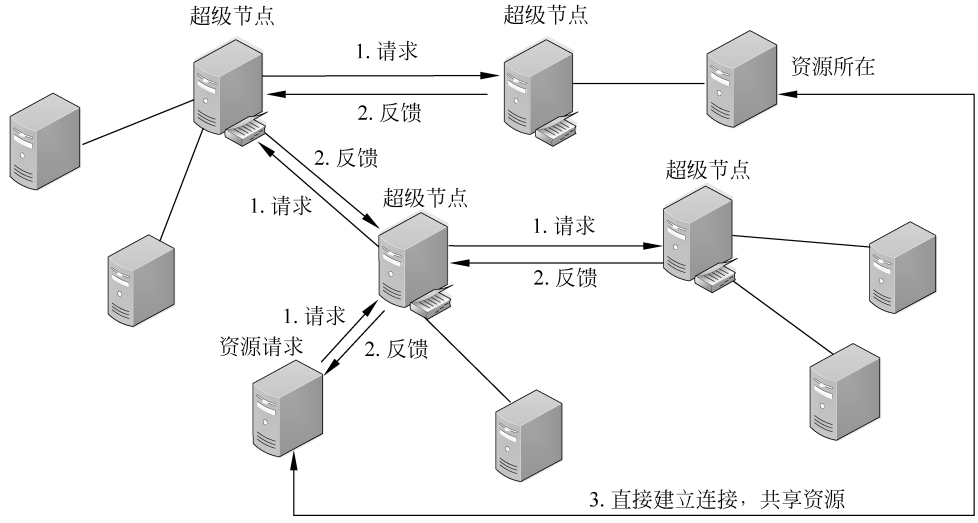
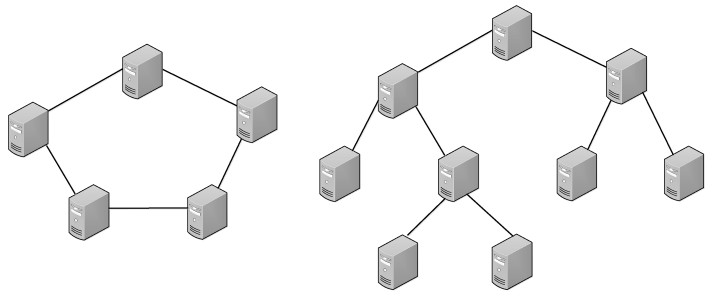


图 5-13　混合式 P2P 流程图

(4) 结构化 P2P。它也是一种分布式结构，与纯分布式 P2P 结构的区别在于，结构化 P2P 网络中的节点之间按照一定的规则选择邻居节点并进行连接，最终呈现出的网络结构是规则有序的；而纯分布式 P2P 结构是随机组网连接，没有规则可言。结构化 P2P 网络按照固定结构的不同，可分为环状网络、树状网络等，如图 5-14 所示。

下面我们总结 P2P 网络的特性。



(a) 环状网络结构　　(b) 树状网络结构

图 5-14　结构化 P2P 网络架构图

（1）去中心化。

所有的资源都分布在 P2P 网络中的节点上，资源的传输共享都是直接在节点之间进行的，这样就没有中心服务器介入资源传输共享，进而消除了中心化可能带来的性能瓶颈。虽然集中式和混合式 P2P 中都有类似中心作用的索引服务器，但是这个索引服务器只起到资源索引的作用，实际的资源传输共享最终还是直接在节点之间完成，这样就大大削弱了索引服务器的中心作用，降低了对其性能的要求。可以说，去中心化是 P2P 网络最基本的特性。

（2）可扩展性。

传统的 C/S 和 B/S 架构中，系统能容纳的用户数量和所提供的资源性能主要由服务器自身性能决定，当大量用户集中访问资源导致数据量井喷时，服务器就不得不采用高性能计算机和高带宽网络。当有更多资源访问需求时，又不得不重复建设升级，所以这种高投入和高开销的模式限制了系统规模的扩展。但是 P2P 网络并不限制用户节点的加入和资源需求的增加，相反，它随着节点的增加，也同步扩充自身资源处理能力，始终能够较容易地满足用户需求。纯分布式 P2P 的网络是全分布式的，不存在性能瓶颈，理论上其扩展性是无限的；集中式和混合式 P2P 中的索引服务器处理索引请求相比资源访问处理的数据量小很多，所以也方便进行扩展。

（3）健壮性。

当网络出现异常情况，比如节点脱网、网络阻塞或网络中断等导致资源请求和执行失败等，如何能及时发现并处理这些异常，关乎着系统的稳定性和资源提供的持续性。在传统的集中式网络中，如果中心服务器出现异常，那么整个网络就瘫痪了，根本谈不上什么健壮性。而 P2P 网络天生具有抗干扰、高容错的特性，由于资源请求和执行是分散在网络各个节点上进行的，就算部分节点失效也不会影响其他资源的请求执行；此外，在部分节点脱网失效后，P2P 网络会自动收敛，重新调整拓扑结构，进而保证在线节点的连通性，这样确保任何节点的脱网都不会影响信息在网络中传播。实际上，P2P 能以自组织方式建立网络，节点可以随意加入或退出，有一些 P2P 网络结构还能根据网络带宽、节点规模、当前网络负载等信息进行实时动态调整。

（4）私密性。

目前在互联网上有很多工具和方法能搜集用户私密信息，根据用户的 IP 地址追踪到用户本人。但是在 P2P 网络中，由于信息的传输分散在各个节点之间实现，不通过某个中心节点，所以用户私密信息很难被窃听和泄露。此外，互联网上解决私密问题主要采用中继转发的技术手段，将用户隐藏在众多网络参与者之中，而 P2P 网络中所有节点都具有中继转发的能力，因此大大提高了用户的隐匿性。

（5）高性能。

通常用户所拥有的节点只是作为客户端连接到互联网中，仅仅作为资源的消费者，其自身的资源和能力，诸如计算能力、存储空间等，始终无法被互联网使用。但是 P2P 网络能够有效地利用互联网中这些分散的用户节点，将计算任务或存储任务布置到节点上，利用这些闲置节点的计算能力和存储空间达到高性能计算和海量数据存储的目的，这与分布式计算的思想一致。这种利用空闲资源的方法能以更低的成本提高计算和存储能力。

P2P 网络架构的典型应用是早期的互联网本身，那个时候网络上的节点都是基于 IP（网络互联协议）来进行互联互通的。当前最成功的 P2P 技术应用是文件分享，比如最著名的内容分发协议比特流（BitTorrent）。

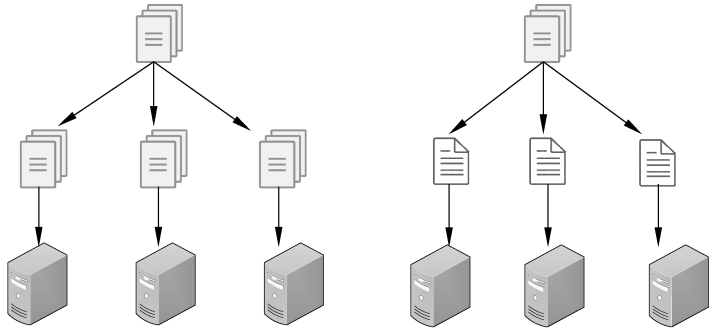
5.2　分布式存储

上一节介绍了集中式和分布式架构的一些基础理论，特别对分布式架构进行了详细的描述。本节引出本章开头提到的问题，即区块链数据如何存储在网络之中。有了分布式架构技术的基础，我们再继续介绍分布式存储技术。

分布式存储，就是将数据存储在网络中的各个节点里，其原理是通过网络将分散的各个节点的空余存储空间利用起来，形成一个虚拟的存储设备，并将数据分散存储在这些节点上。

数据的分布式存储分为两类，如图 5-15 所示。一类是一份数据复制形成好几份副本，将副本分别存储在不同的节点上；另一类是将同一份数据进行数据切割后分为好几份子数据，这些子数据分别存储在不同的节点上。比特币系统采用的是数据复制模式，在分散的不同节点存储整个区块链。

对于数字货币来说，交易数据是很重要的，所以一定要存储下来，同时又不能只存储在中心节点上，万一中心节点故障，所有的交易数据可能就此丢失，所以一定要去中心化存储，这样也就只能分散存储到各个节点上。比特币系统是在全世界范围内的交易，按照上面的理论，节点都要存储全世界范围内的所有交易数据。实际上，比特币系统并不要求每个节点存储完整的交易数据，而只是部分节点存储，即完全节点。矿工通过挖矿来获取记账权，将打包的交易数据挂到区块链上。也许有人会提出异议，如果完全节点背叛了我们，任意篡改



(a) 数据复制　　(b) 数据切割

图 5-15　数据分布式存储

交易数据，那么这种部分节点保存全部交易信息的模式不就不可靠了吗？实际上，交易信息是由成千上万的不同机构、不同个人共同维护的同一份信息，只有共同维护的这份信息才被认为是最终正确的信息，而单个或少数完全节点篡改的信息未得到大多数完全节点的认可，是没有任何意义和价值的。可以说，去中心化的分布式存储是区块链的特色之一。分布式存储大大降低了对于数据存储的投入，特别是建设或租用数据中心，因此在实际应用中大多采用分布式存储。

5.3　比特币网络

前面我们重点介绍了 P2P 网络架构的基本原理，现在我们知道，比特币系统中连接各个对等节点的组网方式采用的就是 P2P 网络架构，P2P 网络是区块链核心技术之一。不同的区块链设计可能会采用不同的 P2P 网络模型，但是最基本的原理是一致的。下面我们将重点介绍最具代表性的区块链 P2P 网络：比特币网络。

5.3.1　比特币网络架构

比特币采用前面介绍的 P2P 网络架构，网络中所有的节点关系对等，节点之间通过一种扁平的网状拓扑结构来互联互通，它们并不存在层次结构，也就没有哪个节点属于所谓的上层节点或中心节点。正因为 P2P 网络具有去中心化、可扩展性以及健壮性等特点，所以中本聪选择 P2P 网络作为比特币底层数据交换的网络技术平台。比特币网络中的节点共同承担着比特币网络服务的责任，比如承担组网、发现新节点、验证和传递交易及区块数据等功能。比特币网络中所有的节点都是直接或间接连通的，一个节点发出的数据，短时间内可以扩散到全世界比特币网络中大部分的节点；如果网络中部分节点发生故障，也不会影响

到其他节点的正常工作，除非所有节点同时脱网或出现故障，那么 P2P 网络也就不存在了。

比特币网络本质上是一种点对点的数字货币交易系统，它的特点就在于去中心化的设计理念，通过一种扁平化、去中心化的 P2P 共识网络[①]来维持整个比特币网络。

在整个比特币网络中，除了比特币节点之间使用 P2P 协议建立网络互联之外，还存在很多其他的协议，比如用于挖矿以及轻量级 Stratum 钱包的 Stratum 协议。这些额外协议由网关服务器提供，网关服务器通过运行 P2P 协议加入比特币网络，其提供的网关功能实现两个不同协议的网络互联，这样就将比特币网络扩展到运行其他协议的节点上。比如 Stratum 服务器通过 Stratum 协议，将所有 Stratum 矿工节点与主比特币网络[②]相连，这样 Stratum 协议也就与 P2P 协议互连了，Stratum 服务器就是网关服务器。

真实的比特币网络不仅包含主比特币网络，还包含运行其他专门协议的节点。它们共同组成了整体网络结构，被称为“扩展比特币网络”。

5.3.2　节点类型和作用

比特币网络中的每个节点都是对等的，但是它们各自所具备的功能却是不一样的。总体来看不外乎四大功能：网络路由、区块链数据库、钱包、挖矿，如图 5-16 所示。

（1）网络路由。主比特币网络中所有节点都必须具备路由功能，没有路由就无法接入网络，只有接入网络才能成为节点，进而接收和转发交易及区块数据，并且发现和维持与其他节点的连接。

（2）区块链数据库。在比特币网络中，所有的交易数据都需要进行分布式存储，区块链数据库用于存储数据，有了整个区块链完整数据就可以独立验证收到的交易是否有效，而不需要用其他节点的区块链数据作为参考，这样做最安全。截至 2017 年 12 月，比特币节点需要接近 170GB 的存储空间来存储整个区块链数据。

（3）钱包。钱包是用户管理比特币资产的工具，保存着用户控制的所有私钥，同时具备查看用户比特币资产、发起交易、查看历史交易记录等功能。钱包可以理解成我们现在使用的手机银行客户端，私钥相当于银行账户密码，由私钥可以生成对应的公钥，而由公钥可以计算得到比特币地址，比特币地址相当于银行卡卡号。

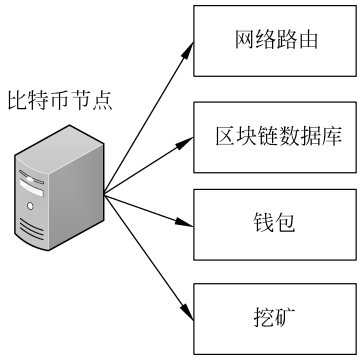


图 5-16　比特币节点拥有的四项功能

（4）挖矿。挖矿是一种行为，由矿工来完成。在比特币网络里，每产生一个交易，并不是立即写入区块链数据库，而是将交易广播后由矿工节点收集起来，等待矿工节点挖矿成功获得交易的记账权，从而把交易最终写入区块链数据库，完成对交易的确认，挖矿成功的矿工获得比特币奖励。

---

① P2P 共识网络是指将共识机制运用到 P2P 网络中。

② 主比特币网络是由运行比特币 P2P 协议的节点组成的网络，也称“比特币网络”。

在早期的比特币网络中，每个节点都具备这四项功能，但是随着比特币的进一步发展，特别是区块链数据的爆发性增长和矿池的出现，为了满足不同的应用场景需求，不需要每个节点都具备全部功能，不同类型的节点可能只包含部分功能，一般只有比特币核心（Bitcoin Core）节点才会包含所有功能，这样就大大增加了比特币网络的灵活性。目前最主要的四种比特币网络节点类型如下。

1. 比特币核心节点（标准客户端）

比特币核心节点，又称标准客户端，包含全部的四项功能，如图 5-17 所示。该类型节点参与全网的路由功能，发现和维持与其他节点之间的连接；验证并传递交易和区块数据；通过挖矿获取记账权，将交易记入区块链数据库；核心节点还具备钱包功能，提供比特币交易工具。

早期的比特币网络全部由该类型节点组成，节点之间通过比特币协议来交互，比特币协议[①]包括 P2P 网络互联协议、交易与区块的传播协议和验证协议、区块链底层数据结构，以及相关的哈希密码、数字签名等一系列协议和标准。早期比特币网络示意图如图 5-18 所示。

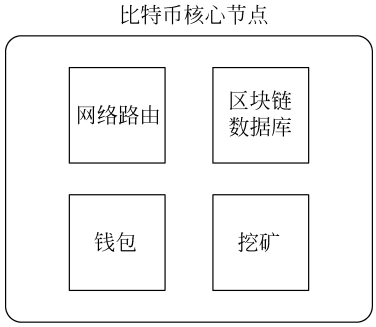


图 5-17　比特币核心节点功能

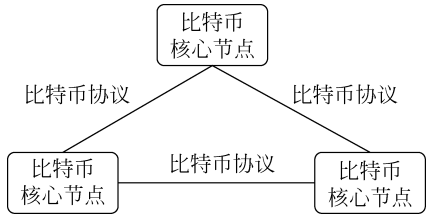


图 5-18　早期比特币网络示意图

2. 完全区块链节点

完全区块链节点包含完整的区块链数据并具有网络路由功能，如图 5-19 所示。

完全区块链节点存储有最新、最完整的区块链数据，所以它能独立地完成对所有交易数据的校验，而不需要借助外部其他节点的数据或信息；同时依靠网络获取其他节点发来的交易数据，以及具有挖矿功能的节点发出的区块数据，交易验证后继续广播，区块验证之后写入本地区块链数据库。

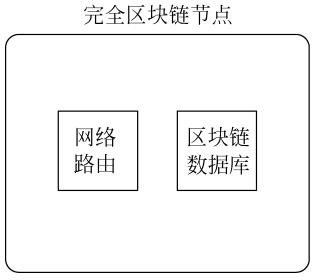


图 5-19　完全区块链节点功能

完全区块链节点到底有什么作用呢？如果只是为了保存完整区块链数据，那么凡是具有区块链数据库功能的节点都可以用来存储数据，并不需要设计一个单独的完全区块链节点。实际上，完全区块链节点起到对资产的保护作用。我们

① 详见 https://en.bitcoin.it/wiki/Protocol_documentation。

知道，涉及资产操作，特别是对转账这种操作，需要保证绝对的安全，而实际应用中查询资产比转账操作使用频率高很多，所以对于查询类操作应尽量避免访问包含钱包功能的节点，完全区块链节点就是用来处理类似查询资产这样的业务操作。

另外，完全区块链节点组成名义上的边缘路由器[①]。除了专业挖矿公司外，其他想在区块链顶层做开发的公司基本上会运行该类型节点，这样可以基于完全区块链节点构建顶层的应用服务，从而实现资产查询、交易支付处理、交易机构搭建、区块链信息调阅等功能，这些功能都是顶层应用服务商提供给用户使用的，能与比特币系统交互。前面提到的“边缘”并非指完全区块链节点的地理位置位于比特币网络的边缘，而是指用户只有使用基于完全区块链节点搭建的顶层应用，才能接入比特币网络，从而实现操作，所以可以简单地认为完全区块链节点类似于边缘路由器。

完全区块链节点通过自身的路由功能与比特币核心节点相连，如图 5-20 所示。

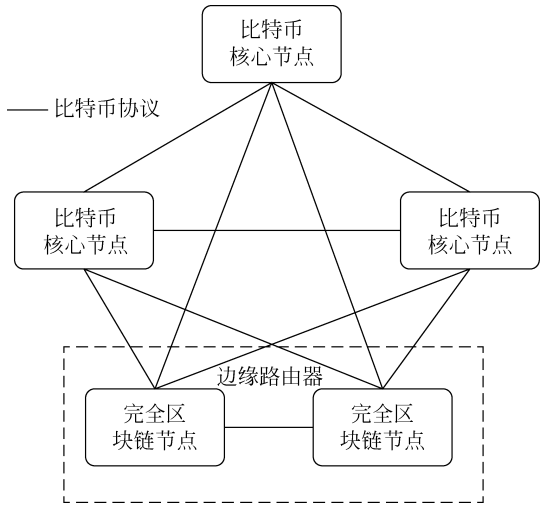


图 5-20　边缘路由器与比特币核心节点互联

3. 个体矿工

个体矿工(Solo Miner)包含完整的区块链数据、网络路由和挖矿功能，如图 5-21 所示。

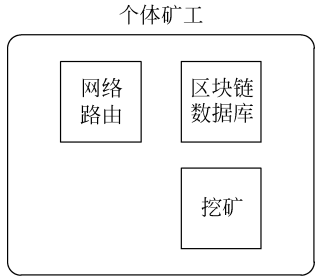


图 5-21　个体矿工功能

个体矿工的主要任务就是挖矿，随着交易生成并在全网广播，每个个体矿工都会把这些交易打包，同时增加一笔自己收款的币基交易在里面。打包完成后构成一个区块，矿工通过挖矿竞争记账权，这是一段需要经过大量哈希计算才能挖出有效区块的工作量证明过程，第一个成功挖出符合工作量证明

① 边缘路由器又称接入路由器，一般位于网络外围，作为业务数据的出入口。

要求的区块的个体矿工会把新区块添加到自己本地的区块链副本顶端，同时把这个新区块通过网络发送给其他节点，其他节点接收后验证区块的真实性和准确性，如果验证通过则添加该区块到自己的区块链副本顶端。因为要校验收到的区块是否正确，所以个体矿工需维护一个完整的区块链副本。个体矿工接收到新区块会立刻验证区块头部及其包含的交易，验证通过则更新本地区块链副本，并放弃当前的挖矿工作，重新打包交易进入下一轮挖矿竞赛。

个体矿工强调“个体”二字，是因为在比特币诞生的早期，矿工都是利用自己的计算机来运行比特币标准客户端，每个矿工之间在网络层面上都互不认识，没有任何关联性。那个时候每个矿工都能挖到矿，但是随着比特币价值的不断增加，挖矿竞争越来越激烈，个体挖矿变得难上加难，于是由很多个体矿工组建的矿池应运而生。矿池能增加挖矿算力，个体矿工逐渐转为依赖于矿池的矿工。不过从广义上来讲，也可以把独立个人或公司经营的矿池看作个体矿工。

个体矿工一般通过比特币协议与比特币核心节点相连，如图 5-22 所示。

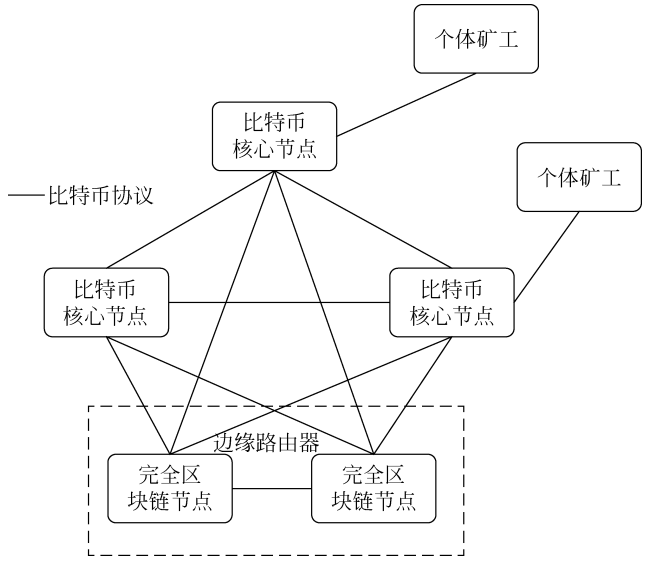


图 5-22　个体矿工与比特币核心节点网络互联

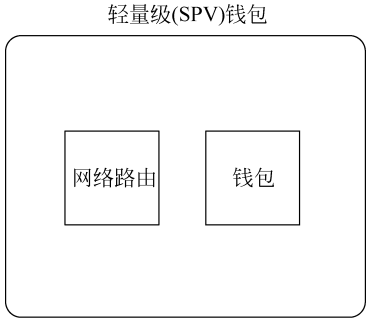


图 5-23　轻量级(SPV)钱包功能

4. 轻量级(SPV)钱包

轻量级(SPV)钱包包含钱包和网络路由功能，如图 5-23 所示。

轻量级(SPV)钱包最主要的作用就是具有钱包功能，这也是绝大多数普通用户使用的功能，其节点数量远远大于前面三类节点。在现实生活中，我们随身携带的钱包都是轻巧的，没有人愿意把大量现金全部带在身上，而只愿意携带少量现金。同样，比特币钱包也不会把整个区块链副本都放在里面，而只存储区块链

副本的一个子集，确切地说，一般只存储区块头部数据和自己关心的交易，而不存储所有交易数据。我们在资金交易时，最关注的无非就是自己是否收到钱以及自己支付的钱是否成功，对于比特币交易来说，就是需要验证向自己的钱包转入资金和自己付款是否成功。因为该节点没有完整区块链副本，所以需要给具有区块链副本的节点发送查询消息，通过询问获取该交易所在区块的相应梅克尔路径，使用一种称为简单支付验证（Simplified Payment Verification，SPV）的方法来验证交易是否成功，从而确认向自己付款或自己支付的交易是否成功，所以称之为轻量级（SPV）钱包。这种轻量级钱包适合运行在存储和计算资源有限的设备上，比如智能手机、iPad 等，越来越多的移动钱包选择该节点类型。这种模式扩大了比特币的应用范围，并促使比特币交易被更多的人接受。

轻量级（SPV）钱包都能与具有区块链副本的节点相连，但是由于个体矿工主力于挖矿，完全区块链节点主力于构建顶层应用，所以理论上它是与比特币核心节点相连。实际上在扩展比特币网络中，轻量级（SPV）钱包是与完全节点客户端相连的。关于完全节点客户端在本节后面会详细介绍。

上面介绍了比特币网络的四种基本节点类型，其中前三种节点具有完整区块链数据库，这三种节点也是我们在本书前面章节提过的完全节点。完全节点支撑起整个比特币网络的主要业务和应用，成为比特币网络的骨干。目前比特币网络中所有能找到的可达节点大约有 9294 个（截至 2019 年 10 月），可以在 https://bitnodes.earn.com 查询比特币节点的数量和分布。图 5-24 是从该网站获取的比特币节点信息，节点数量最多的前三个国家分别是美国、德国和法国。

GLOBAL BITCOIN NODES DISTRIBUTION

Reachable nodes as of Mon Oct 28 2019 08:54:31 GMT+0800 (中国标准时间).

9294 NODES

24-hour charts »

Top 10 countries with their respective number of reachable nodes are as follow.

RANK	COUNTRY	NODES
1	United States	2411 (25.94%)
2	Germany	1868 (20.10%)
3	France	608 (6.54%)
4	Netherlands	488 (5.25%)
5	Singapore	331 (3.56%)
6	Canada	322 (3.46%)
7	United Kingdom	316 (3.40%)
8	China	305 (3.28%)
9	Russian Federation	237 (2.55%)
10	Japan	190 (2.04%)

More (94) »

图 5-24　比特币在线节点数量

在扩展比特币网络中，还包含其他类型的节点。

1. 矿池协议服务器

前面提到过，有些挖矿节点不是个体矿工，而是和其他矿工一起连接到矿池，参与集体

挖矿，这种节点称为矿池矿工。这些节点形成了一个局部的集中式矿池网络，在集中式矿池网络存在一个矿池协议服务器，矿工与其连接。矿池矿工和矿池协议服务器之间的通信采用的不是标准比特币协议，而是 Stratum 协议或其他矿池挖矿协议，同时矿池协议服务器使用比特币协议与其他比特币节点通信。这里的矿池协议服务器也就是前面提到的网关服务器。

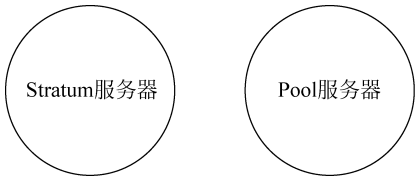


图 5-25　矿池协议服务器

矿池协议服务器分为两类，如图 5-25 所示，一类是 Stratum 服务器；另一类是 Pool 服务器。前者是 Stratum 矿工通过 Stratum 协议连接到 Stratum 服务器，从而构成 Stratum 矿池，其中 Stratum 协议是矿工与矿池之间的传输控制协议；后者用于 Pool 矿工通过 Pool 矿池协议与 Pool 服务器相连，构成去中心化的矿池组织形式。Stratum 矿池是一种托管矿池，Stratum 矿工加入其中共同挖矿，一旦有一个矿工挖到矿，比特币奖励就会发到矿池地址，当奖励达到一定规模，矿池按照每个矿工的算力贡献来分配比特币收益。但是 Stratum 矿池本质上也是带有中心化色彩的，Stratum 矿工全部连接到 Stratum 服务器，并且有一个负责人管理整个 Stratum 矿池以及比特币收益，一旦这个 Stratum 服务器发生故障，那么矿池中的矿工也要被迫停工；此外，矿池负责人也有可能监守自盗，私自卷走所有的比特币收益。所以，为了弥补这些缺陷并解决个体矿工自身算力不足，但又不想加入中心化矿池的困境，采用了一种折中方案——Pool 矿池。例如 P2P 矿池(P2Pool)，这是一个点对点矿池，没有一个中心管理方，它允许个体矿工通过专业矿池协议和算法把运算量组织起来共同挖矿。P2P 矿池通过将 Pool 服务器的功能去中心化，实现了一个类似比特币区块链但难度低于它的份额链(Share Chain)。份额链允许矿工在去中心化的矿池中协同工作，Pool 矿工以每 30 秒产生一个份额区块的速度在份额链上挖矿并获得份额，每个区块记录了 Pool 矿工的算力贡献，并继承之前份额区块上的算力贡献。当某个份额区块同时达到比特币网络难度目标时，这个区块就被传播到比特币网络上，然后根据每个矿工对份额区块的算力贡献来分配区块奖励，同时每个矿工可自行跟踪所有的算力贡献，不需要一个中心管理方来记录。Stratum 矿池、Pool 矿池及其相关的挖矿协议在这里不作深入介绍，有兴趣的读者可以自行查看相关文献。

矿池协议服务器运行比特币协议来加入比特币网络，这样与其相连的矿工节点也间接与比特币网络通信，如图 5-26 所示。

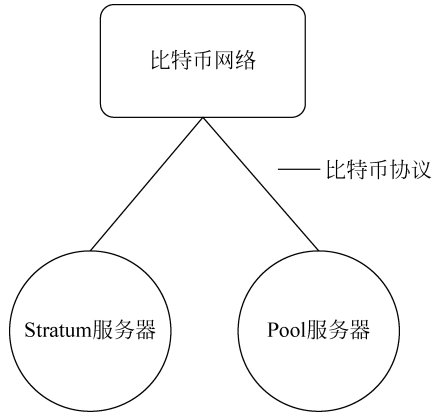


图 5-26　矿池协议服务器与比特币网络相联

2. 矿池矿工

矿池矿工具备挖矿功能，它与前面介绍的个体矿工的区别在于，该节点既没有网络路由，也没有完整区块链数据，所以它既不能独立加入

比特币网络，也不能实现交易验证，只能通过加入矿池，依赖矿池协议服务器来发挥作用，所以该节点需要运行 Stratum 协议或其他矿池挖矿协议与相关服务器建立联系，如图 5-27 所示。

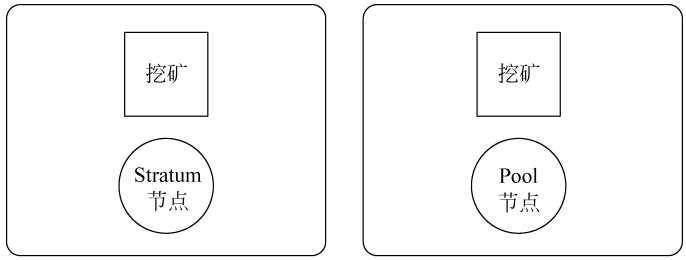


图 5-27　矿池矿工

由图 5-27 可知，矿池矿工主要分为两大类，即 Stratum 矿工和运行其他矿池挖矿协议的矿工，在这里运行其他矿池挖矿协议的矿工主要指 Pool 矿工。矿池矿工与矿池协议服务器通过相关的矿池协议相连，如图 5-28 所示。

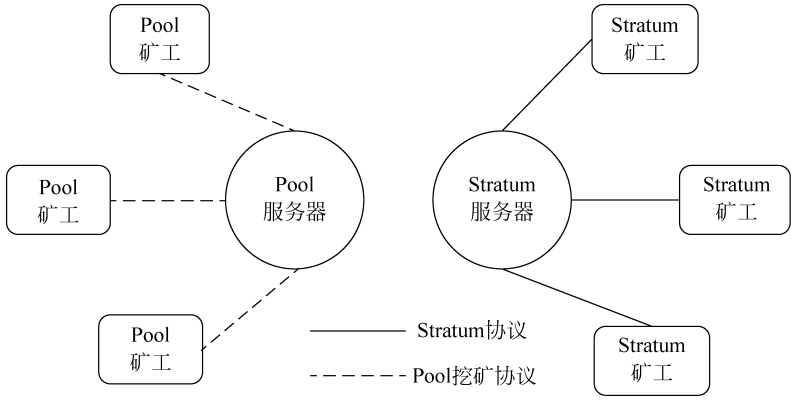


图 5-28　矿池矿工与矿池服务器的网络互联

个体矿工只有自身挖到矿后才能获得比特币收益，没有挖到矿，收益就为零。但是加入矿池的每个矿工只需要一起协作挖矿，就能共享比特币收益，从而获得更稳定的挖矿收益。

3. 轻量级(SPV)Stratum 钱包

轻量级(SPV)Stratum 钱包包含钱包功能，并运行 Stratum 协议，如图 5-29 所示。

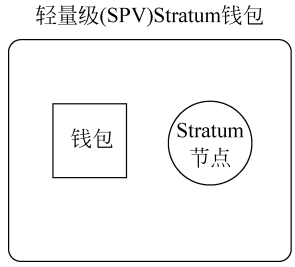


图 5-29　轻量级(SPV)Stratum 钱包功能

轻量级(SPV)Stratum 钱包没有网络路由功能，所以不能直接与比特币网络相连，只能通过 Stratum 服务器实现接入，而该服务器又连接完全节点客户端，从而间接实现与比特币网络的连接，进而实现钱包功能，如图 5-30 所示。

轻量级(SPV)Stratum 钱包和前面介绍的轻量级(SPV)钱包,又称为 SPV 节点或轻量节点(Lightweight Node)。

还有一种被称为完全节点客户端的节点类型,它包含网络路由、区块链数据库和钱包功能,如图 5-31 所示。该节点不从事挖矿,主要功能是管理用户的钱包,可以独立验证交易。它作为 Stratum 服务器接入比特币网络的接入端,同时也为 SPV 节点提供区块链数据查询服务。

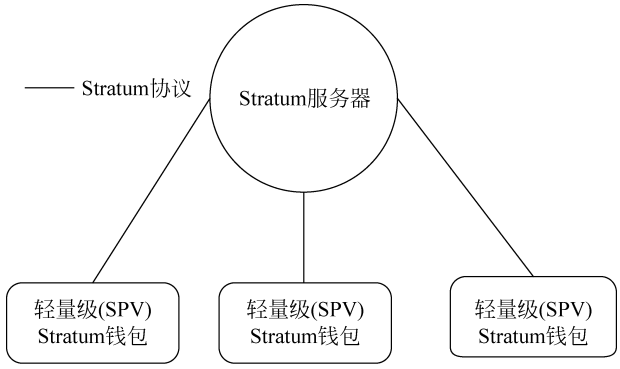


图 5-30　Stratum 钱包与 Stratum 服务器互联

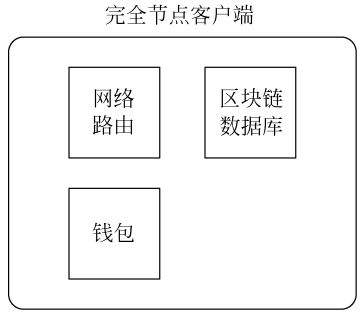


图 5-31　完全节点客户端功能

5.3.3　扩展比特币网络

本节详细介绍扩展比特币网络。我们先来看一张经典架构图,如图 5-32 所示。扩展比特币网络不仅运行比特币协议,用于各个节点之间的网络互联和数据交互,还运行其他协议,相关节点通过网关服务器与比特币网络互联互通,主要是矿池矿工节点和轻量级(SPV)Stratum 钱包节点。

有了前面比特币节点类型和功能的介绍,我们再看这张经典图就比较容易理解了,下面对图中一些框图和英文作解释,具体的中文理解可以参考前面的介绍。

整个扩展比特币网络由三种协议类型组成:

(1) Bitcoin Protocol:比特币协议;

(2) Stratum Protocol:Stratum 协议;

(3) Pool Mining Protocol:Pool 挖矿协议。

图中每个小方框代表一种节点,实际应用中,可能是矿机、个人笔记本、台式机或者移动终端。除了 Stratum 矿工、Pool 矿工和轻量级(SPV)Stratum 钱包外,其他节点之间都用比特币协议来互相通信。

节点的四大功能:

(1) 钱包,如图 5-33 所示。

(2) 挖矿,如图 5-34 所示。

(3) 区块链数据库,如图 5-35 所示。

(4) 网络路由,如图 5-36 所示。

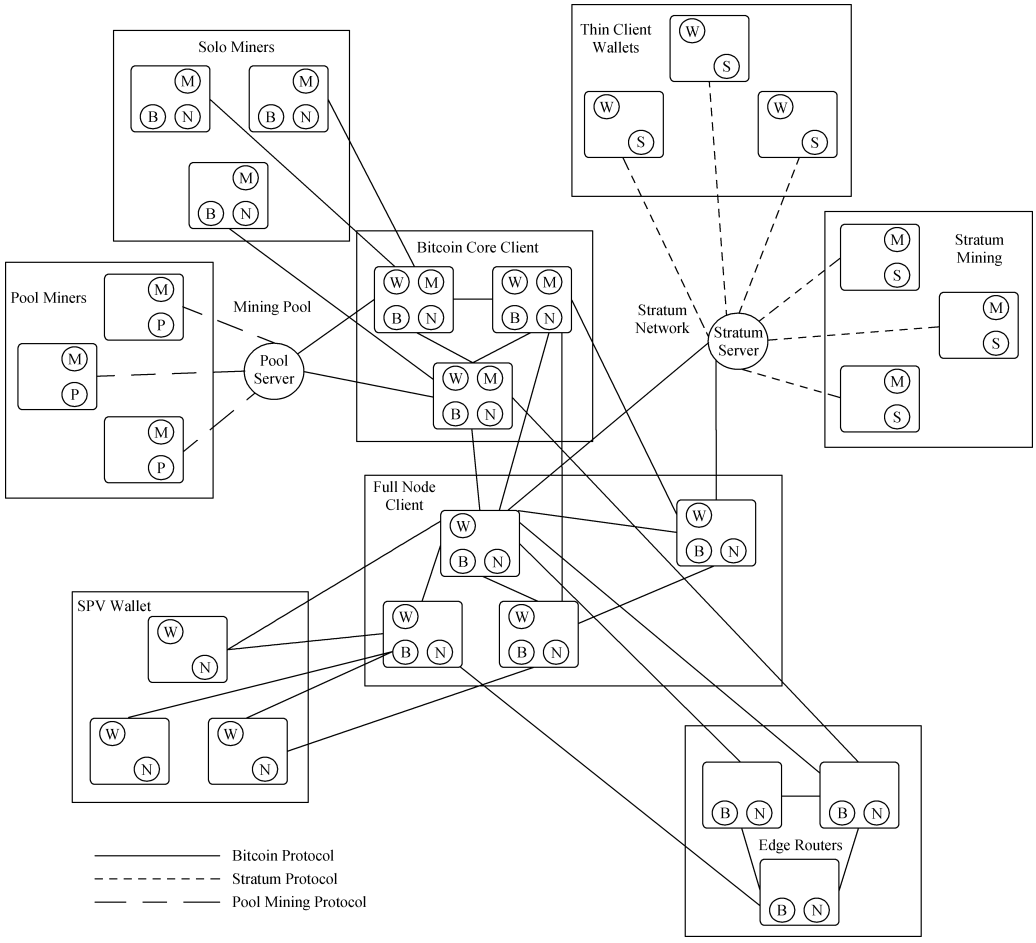


图 5-32　扩展比特币网络经典架构图

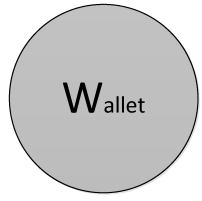


图 5-33　钱包

图 5-34　挖矿

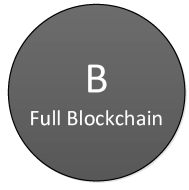


图 5-35　区块链数据库

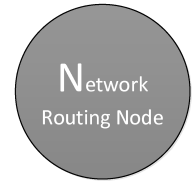


图 5-36　网络路由

节点类型如下：

(1) Reference Client (Bitcoin Core)：标准客户端(比特币核心)，也就是比特币核心节点(图 5-37)。

(2) Full Block Chain Node：完全区块链节点(图 5-38)。

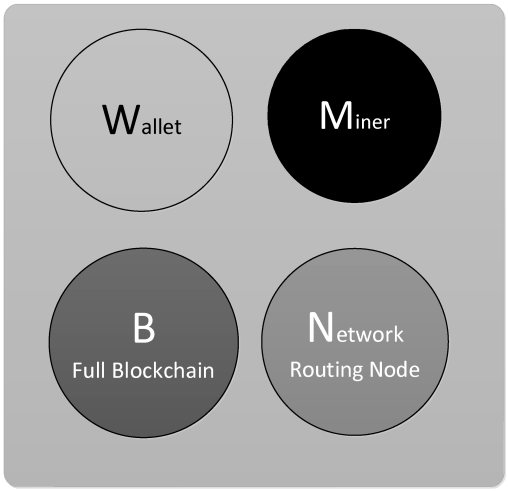


图 5-37　标准客户端(比特币核心)

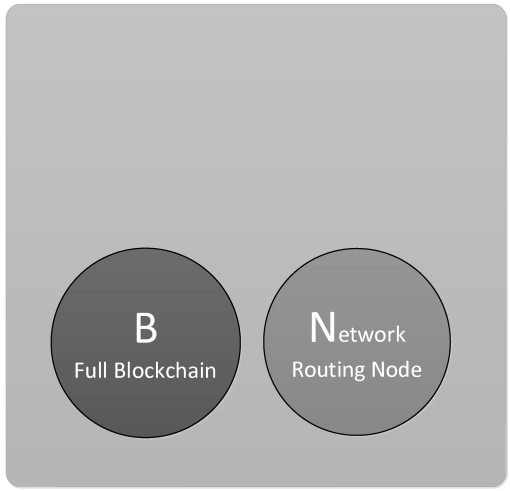


图 5-38　完全区块链节点

(3) Solo Miner：个体矿工(图 5-39)。

(4) Lightweight(SPV) wallet：轻量级(SPV)钱包(图 5-40)。

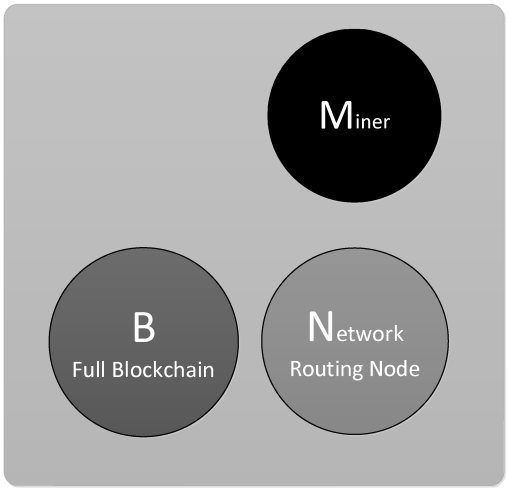


图 5-39　个体矿工

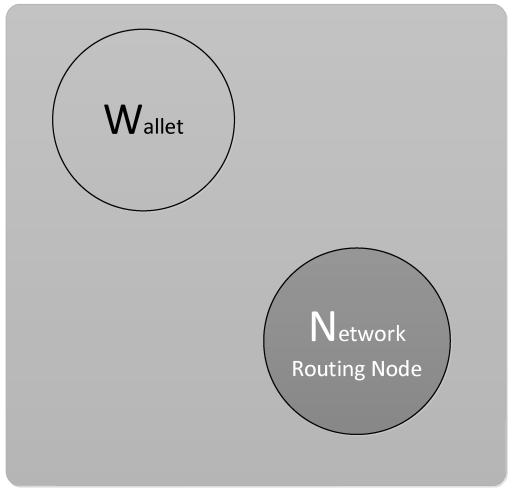


图 5-40　轻量级(SPV)钱包

(5) Pool Protocol Servers：矿池协议服务器，如图 5-41 所示。前者为 Pool 服务器，后者为 Stratum 服务器。

(6) Mining Nodes：矿池矿工节点，如图 5-42 所示，前者为 Pool 矿工，后者为 Stratum 矿工。

(7) Lightweight(SPV) Stratum wallet：轻量级(SPV)Stratum 钱包(图 5-43)。

(8) Full Node Client：完全节点客户端(图 5-44)。

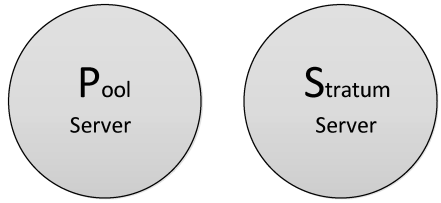


图 5-41　矿池协议服务器

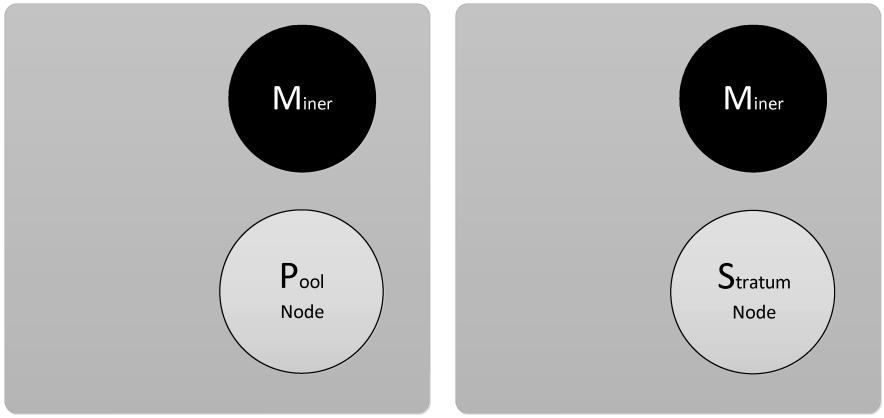


图 5-42　矿池矿工节点

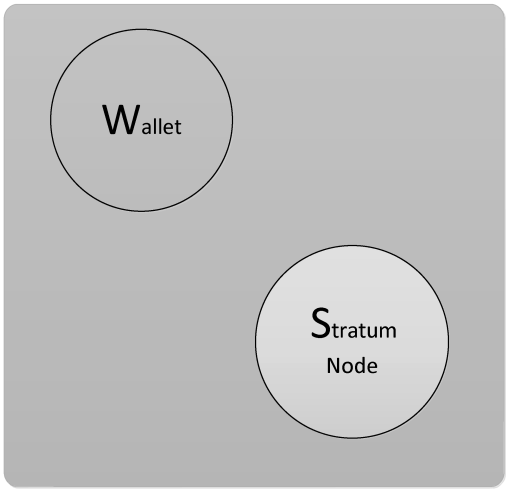


图 5-43　轻量级(SPV)Stratum 钱包

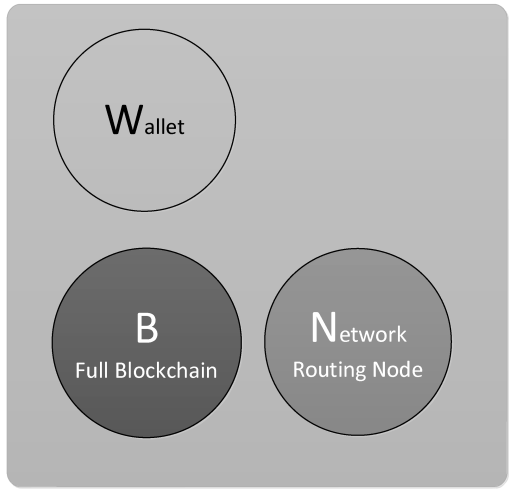


图 5-44　完全节点客户端

图 5-32 英文标识分别表示：

(1) Stratum Mining：Stratum 挖矿。

(2) Stratum Network：Stratum 网络，由 Stratum 服务器和运行 Stratum 协议的节点组成。

(3) Full Node Client：完全节点客户端。

(4) Thin Client Wallets：瘦客户端钱包，由轻量级(SPV)Stratum 钱包组成。

(5) Edge Routers：边缘路由器，由完全区块链节点组成。

(6) Mining Pool：Pool 矿池。

(7) SPV Wallet：轻量级(SPV)钱包。

(8) Bitcoin Core Client：比特币核心客户端。

(9) Solo Miners：个体矿工。

(10) Pool Miners：Pool 矿工。

重新来回顾这张图，如果我们把整个网络看成人体，那么每一个节点可以看成人体内的各个细胞，而各种协议就是连接细胞、遍布全身的神经。比特币核心节点相当于大脑，它能实现比特币网络的所有核心功能，比特币开发者一般都是在比特币测试网络 Testnet 运行核心节点对比特币新的功能进行开发和测试，同时对比特币协议进行改动升级；完全节点客户端支撑了比特币的主要业务，相当于比特币的骨架；完全区块链节点构成的边缘路由器像肌肉一样保护着内部的节点；钱包就像是皮肤，与外界直接接触；不管是个体矿工还是矿池矿工，他们都不停地挖矿来创建新的区块，然后像血液一样，把新区块源源不断地输送到其他节点，矿工维持着整个比特币网络的生命。

5.3.4 网络发现与同步

下面介绍新节点如何加入比特币网络并最终达到网络同步，以及新节点是如何从其他节点获取区块实现区块同步的。

1. 网络同步

1) 通过握手协议交换“握手”信息

新节点要加入比特币网络，首先随机选择比特币网络中至少一个在线节点作为接入节点，接入节点和新节点没有地理位置的约束。新节点使用 TCP 协议与该接入节点建立连接，端口号通常为 8333(或其他约定好的端口号)，两个节点间连接一旦建立成功，新节点就立即给接入节点发送 version 进行“握手”，version 里包含“握手”信息，见表 5-1。接入节点收到 version 信息后，返回给新节点一个应答信息(verack)，这样接入节点就“认识”了新节点；如果接入节点想让新节点“认识”自己，也可以将自己的“握手”信息发送给新节点，这样两个节点就实现互换“握手”信息，彼此相互“认识”了。

表 5-1　“握手”信息主要内容

字　　段	定　　义
PROTOCOL_VERSION	节点采用的 P2P 协议版本号
nLocalServices	节点支持的本地服务列表
nTime	当前时间
addrYou	从本地节点看到的远端节点 IP 地址
addrMe	节点的 IP 地址
Subver	节点运行的软件子版本号
BestHeight	节点当前存储的区块链高度，用于区块同步

那么新节点是如何寻找到接入节点的呢？比特币系统采用两种方式来获取接入节点。一种方式是利用 DNS 服务器(称为“DNS 种子”)来查询比特币节点，“DNS 种子”一般是硬编码[①]到代码里的，一些“DNS 种子”提供稳定的比特币节点的静态 IP 地址列表，新节点会首先尝试连接这些节点；还有一些“DNS 种子”是 BIND(Berkeley Internet Name Daemon)协议的定制实现，它返回利用爬虫技术[②]或长期运行的比特币节点收集的比特币节点地址列表中的一个随机子集。另一种方式是节点引荐，新节点先连接一个节点，让该节点作为介绍节点，使新节点与更多的节点相连。介绍工作完成后，新节点就与介绍节点断开连接。

通过上述方式，新节点和接入节点建立连接通道后，它们就开始交换“握手”信息。两个节点间的握手交互过程如图 5-45 所示。新节点和接入节点就形成邻居关系。

2) 地址传播与发现

新节点与一个或多个邻居节点建立连接后，就向其邻居节点传播包含自身 IP 地址的 addr 消息。为了确保让更多的节点知道这个新节点的加入，邻居节点收到 addr 消息后，再继续把该 addr 消息向自己的邻居节点传播。另外，新节点也会发送 getaddr 消息给它的邻居，以此获取邻居所知的对等节点的 IP 地址列表，新节点通过这种方式找到新的对等节点并进行连接。地址发现协议流程如图 5-46 所示。由此可见，新节点启动时，只需要连接一个对等节点，对等节点会将它知道的其他对等节点告诉新节点，那么新节点又可以与这些对等节点建立连接，这些对等节点又会把它们知道的对等节点告诉新节点，以此类推。每个节点都必须与不同的节点建立连接，这样才能在不断有节点离开的网络中保证建立到达比特币网络的多条路径。连接的节点可能会因为网络问题或其他原因脱网，所以节点总是在不停地寻找新的节点来代替脱网节点。实际上，新节点不必连接太多的节点，否则会浪费网络资源。

① 硬编码是指将数据直接嵌入到程序或其他可执行对象的源代码中。硬编码的数据一般为不变的信息。

② 爬虫技术，又称网页蜘蛛，这是一种按照一定规则自动抓取万维网信息的程序或脚本，爬虫技术最常用于搜索引擎，但也有人用此技术来非法获取用户隐私。

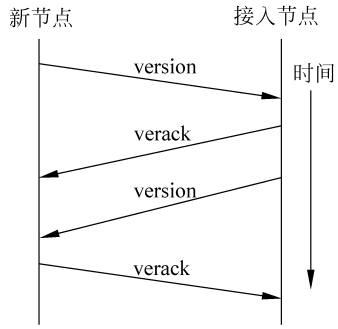


图 5-45　两个节点的握手交互过程

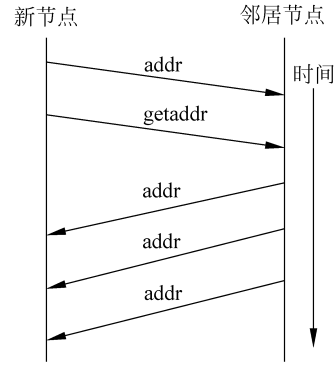


图 5-46　地址发现协议流程

3）节点的重启

新节点首次加入比特币网络后，会记录最近成功连接的节点信息作为节点备份，一旦该节点重启，就可以利用备份的节点信息迅速与备份节点建立连接，只有当备份节点都没有反应时，重启的节点再利用“DNS 种子”或节点引荐的方法来重新连接到比特币网络。

4）节点的保活

如果一个连接上没有通信，那么节点会定期发送保活消息给与之连接的对端节点，用来维持两点间的连接。一旦节点在某个连接上超过一定时间（一般为 90 分钟）没有收到对方应答，那么可以认定对端节点脱网，这时就需要重新选择一个新的节点来代替这个脱网节点。如图 5-47 所示，由此整个比特币网络都会根据节点的变化和网络问题来自适应动态调整，从而不依赖于中心控制实现自主伸缩。

2. 区块同步

新节点如果属于完全节点类型，成功加入比特币网络后，它所做的第一件事就是同步区块链，从比特币网络中下载从创世区块到当前最新区块的所有区块，在本地构建一个完整的区块链副本。节点客户端软件中已经内嵌了创世区块，这样新节点从创世区块开始下载区块，最终完成区块同步。一些节点在本地已经存有区块数据，但因为种种原因退网，不管它与网络断开多长时间，只要再一次入网，都要同步区块，这时就从它自身存储的最近的区块开始，同步到网络当前最新的区块。

图 5-45 和表 5-1 中，节点加入网络时，会给对端发送 version 消息，该消息里的 BestHeight 字段表明了该节点所包含的区块链最新区块高度；对端有了新节点的区块链高度，就可以与自身区块链高度比较。接着两个节点间互发 getblocks 消息，该消息包含了各自区块数据中最新区块的哈希值，如果其中一个节点发现对端最新区块的哈希值是自己区块链里面的旧块，那么可以认定自己的区块链更长，需要给对端同步区块，并且也知道同步哪些区块，这样该节点将对端需要更新的区块的哈希值通过 inv 消息发送过去。如果需要更新的区块太多，为了减轻网络传输压力，可以采用分批次发送的方式。需要更新的节点收到 inv 消息后，给拥有区块多的节点发送 getdata 消息，接着拥有区块多的节点通过发送

block 消息把更新区块发过去，需要更新的节点通过之前 inv 消息中的哈希值来确定是否收到正确的区块，最终完成节点与全网络的区块同步。

区块同步的过程如图 5-48 所示。

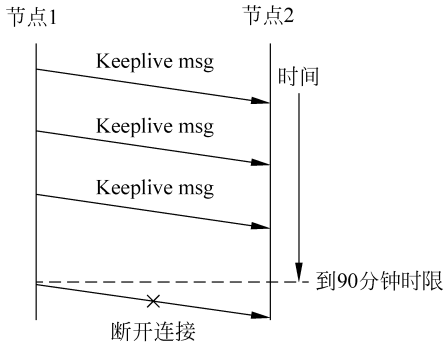


图 5-47　节点判断连接示意图

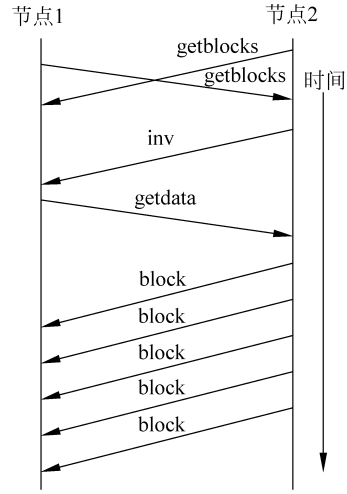


图 5-48　区块同步过程

5.3.5　简单支付验证(SPV)

完全节点拥有完整的区块链副本，导致需要较大的存储空间。越来越多的比特币客户端都是运行在资源有限的设备上，如手机、iPad 等。对于这些设备，使用简单支付验证(SPV)方法可以保证它们在不保存完整区块链副本的前提下，也能验证自己所关心的交易，通常是与自身钱包相关的收款和付款交易。下面介绍简单支付验证。

SPV 节点只下载区块头部和关心的交易数据，而不保存全网所有的历史交易数据，这样需要存储的数据量大大减少，只是完整区块链副本的 1/1000。由于没有完整的交易数据，节点无法构建和维护更新全网所有未花费交易输出 UTXO，但是 SPV 提供的方法能让其他节点提供自己关心的交易在区块链上的梅克尔路径。

打个比方，节点就像第一次到南京来旅游的游客，要找朋友推荐的一家在凤凰西街 30 号的美食餐馆。对于完全节点，他有一张详细地图，标注了主干道、每条小街小巷的地址，很容易找到凤凰西街 30 号。而 SPV 节点只知道主干道，并没有详细的地图。于是他就通过询问街上的路人，获取如何从主干道到达凤凰西街 30 号的路线。这里的每个主干道对应不同的区块头部，每个具体的地址就是挂在某个区块头部下面的交易，路线就是相应的梅克尔路径。对于 SPV 节点来说，凤凰西街 30 号是他关心的交易信息，街上的路人就是其他节点，通过询问节点获取该交易信息对应所属区块的梅克尔路径。SPV 节点需要防止有一部分陌生人指路不正确，不管出于恶作剧还是其他原因，所以最好是向足够多的路人问路。完全节点和 SPV 节点查询交易区别如图 5-49 所示。

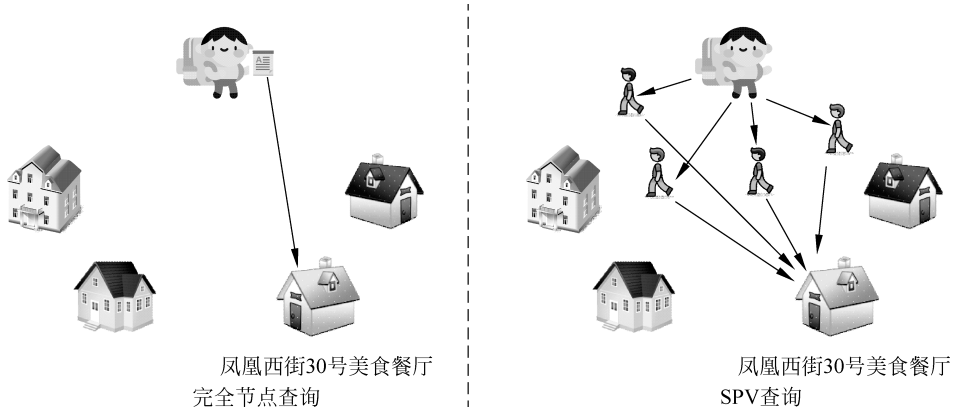


图 5-49　完全节点和 SPV 节点查询交易区别

由上面的比喻我们可以知道，完全节点拥有完整的区块链副本，维护在交易中产生的、未被消费的 UTXO 完整列表，接收到交易时，它只需要通过验证 UTXO 是否已被花费，即：验证引用的关联交易输出是否在未被消费的 UTXO 完整列表里来确认交易是否为双重支付。SPV 节点没有完整的区块链副本，也没有完整的 UTXO 列表，所以不能通过 UTXO 来验证交易是否为双重支付，但是它采用了一种独特的方法，即先利用梅克尔树来确定交易和包含该交易的区块的头部之间关联关系定位交易在区块链上的位置，然后等待基于该区块产生的后续区块被记录进区块链副本中。因为后续记录的区块和它所包含的交易都是经过全网验证通过的，只要等到至少 6 个区块对交易的确认[①]，就能判断交易是有效的，证明该交易不是双重支付且能永久留在区块链上，如图 5-50 所示，交易 1 是被记入永久区块链的交易。

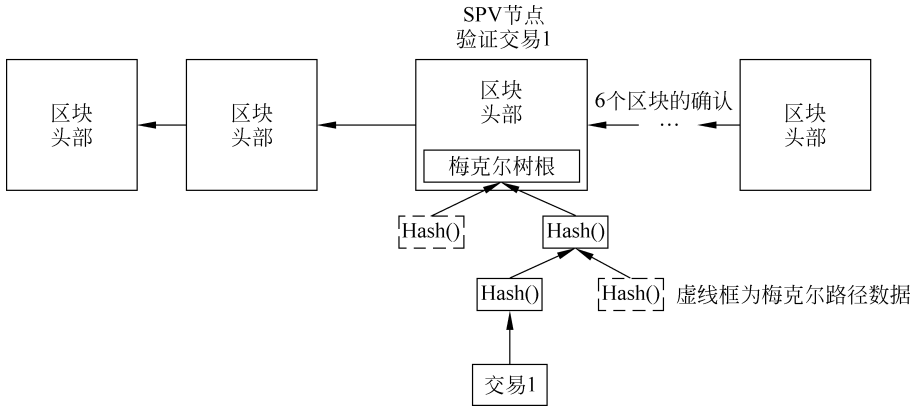


图 5-50　SPV 节点验证交易

① 维基百科：一般是 6 个或更多的确认数，有时也是 7 个区块。《精通比特币（影印版）》（第 2 版）里关于 SPV 的描述为经过 7 个区块的确认。

SPV 实际上做了两件事：第一，找到要验证的这笔交易在哪个区块。SPV 节点通过询问存有完整区块链副本的节点，获取所关心交易在链上对应的梅克尔路径和相应的区块头部，这就是向路人询问凤凰西街 30 号从哪条主干道走的例子。这时 SPV 节点根据梅克尔路径，计算出梅克尔树根哈希值，从而验证交易隶属于该梅克尔树，将交易与区块相连，且利用区块头部信息将区块定位于本地所存的区块链（确切地说是区块头部链），以此验证交易被记录进区块链。第二，确认这笔交易是否在区块链上被 6 次确认了。

SPV 节点能够验证区块的有效性和交易存在性（即交易存在于该区块中），但是因为它没有所有交易的记录，所以无法验证某一个交易不存在，比如同一个 UTXO 的双重支付。SPV 节点需要其他节点配合做验证，可能会被诱导连接到虚假网络中，从而被攻击所利用。所以从安全性角度来说，拥有完整区块链副本的完全节点是最安全的。

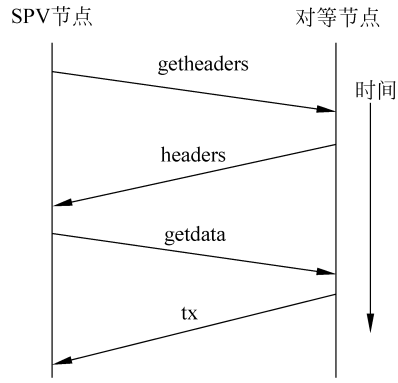


图 5-51　SPV 节点获取区块头部和交易流程

SPV 节点获取区块头部和交易的流程如图 5-51 所示，它给对等节点发送 getheaders 请求消息，对等节点收到请求后返回 headers 消息，消息中包含区块头部信息。对于感兴趣的交易，SPV 节点发送 getdata 消息，对等节点则反馈包含交易的 tx 消息。

由于 SPV 节点对特定交易的获取请求会暴露其自身的钱包地址，如果有第三方对网络进行持续监控，就能跟踪这个钱包发生的所有交易请求，然后根据 SPV 钱包地址等数据进行关联，从而侵犯用户隐私。如图 5-52 所示。

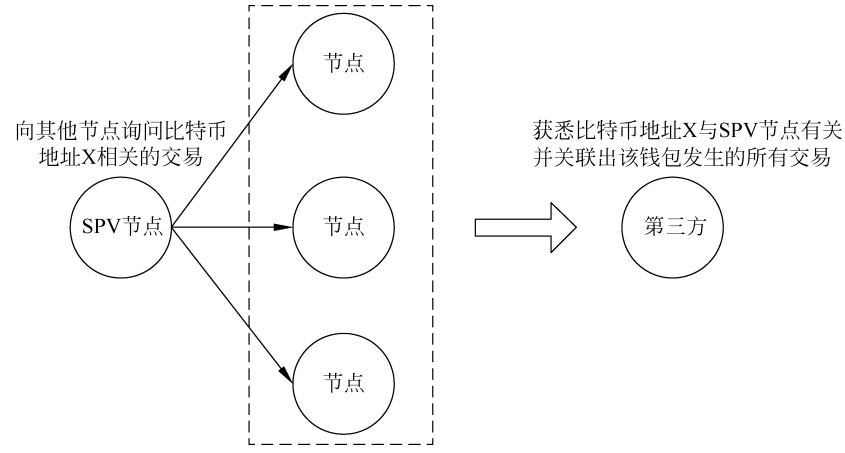


图 5-52　用户隐私暴露

为了解决 SPV 节点隐私的问题，比特币系统的开发者们发明了一个称为“布隆过滤器”(bloom filters)的功能，该功能使得 SPV 节点可以接收交易信息子集，而不用暴露确切的地址。

在前面图 5-49 的例子里，一个没有详细地图的人直接询问路人某个具体地理位置在哪里，这本身就暴露了他要寻找的地理位置信息，如果使用布隆过滤器，其提问方式能有效地隐藏他要找的信息，可以通过提问关键词来实现，比如以“西街”结尾的街道。如果想要更精确的结果，那么提问就需要更精确些，但这样暴露的隐私也越多；要想暴露的隐私少，提问就需要含糊，这样会从路人那得到更多与提问相关的地理位置信息，但并不都是自己感兴趣的，以此提高私密性。总之，布隆过滤器允许 SPV 节点通过调节搜索精度来调节私密性，由 SPV 节点提供给对等节点过滤出它感兴趣的交易，而不会泄露地址等隐私。

布隆过滤器包含一个 N 位的二进制可变长度数组和 M 个哈希函数，哈希函数的输出固定在 1～N 范围内，也就是与二进制数组的每个位对应。下面以 16 位数组和 3 个哈希函数为例，展示布隆过滤器工作机制。

(1) 初始化：首先进行初始化，将 16 位数组的每个位设置为 0，如图 5-53 所示。

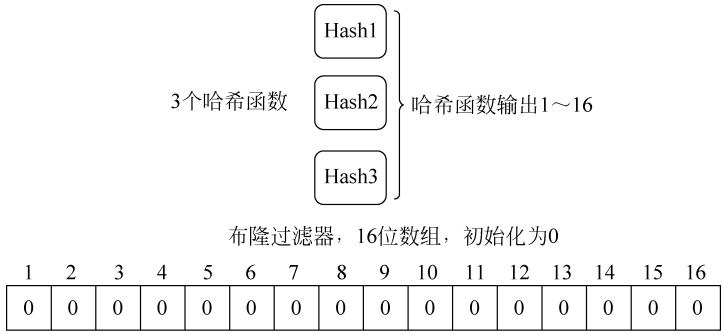


图 5-53　初始化布隆过滤器

(2) 单个模式记录进布隆过滤器：这里的模式是指搜索条件的关键词，比如公钥哈希值，符合搜索条件的交易就能被识别。模式要逐个添加进布隆过滤器。添加单个模式的方法是将单个关键词依次输入 3 个哈希函数进行计算，每个哈希函数的输出是一个 1～N 的数字，然后将 3 个数字所对应的二进制数组比特位逐个置 1，这样就完成了该模式记录进布隆过滤器。过程如图 5-54 所示，模式 B 经过 3 个哈希函数计算所得分别为 3、7、15，然后把二进制数组对应位逐个置 1，现在的布隆过滤器二进制数组为 0010001000000010。

(3) 新的模式继续添加进布隆过滤器：搜索条件的关键词不止一个，新的模式的添加方法即在前一个模式记录进布隆过滤器之后重复第(2)步。需要注意的是，哈希值范围在 1～16，根据鸽巢原理，随着输入模式的增多，越容易出现相同的哈希值。由此我们约定：对于新模式中哈希值与前面模式中的哈希值一样的情况，对应的二进制数组位维持为 1，这样约定的好处就是模糊了输入模式。因为某二进制数组位为 1 可能是很多模式哈希计算的

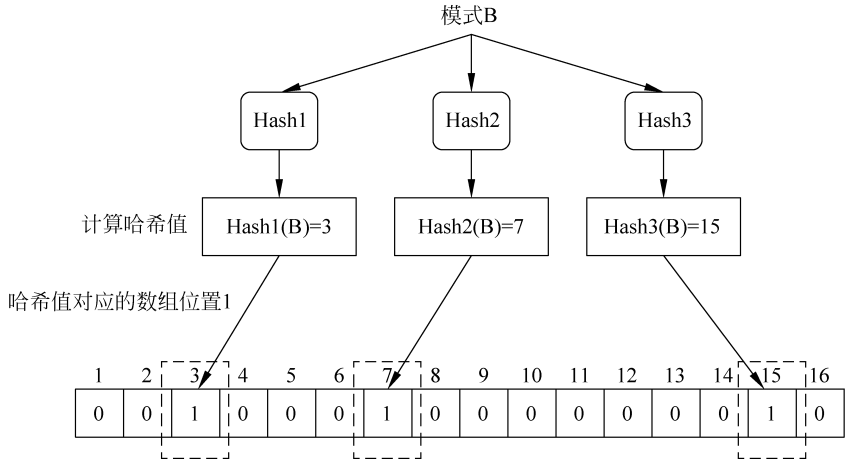


图 5-54　模式记录进二进制数组

结果，这也意味着，随着更多的模式输入，其结果指向同一个位，导致布隆过滤器更模糊、更不精确。总的来说，布隆过滤器的精确度是随着输入模式数量、哈希函数数目 M 和二进制数组位长 N 的变化而变化。哈希函数越少、二进制数组位长越短，可记录的模式就越少，精确度也就越低；相反，哈希函数越多、二进制数组位长越长，就可以实现在较高精确度下记录越多的模式。如图 5-55 所示，模式 I 经过 3 个哈希函数计算所得分别为 8、7、2，然后再把二进制数组对应位置 1。因模式 B 的第二个哈希函数输出是 7，随后模式 I 的第二个哈希函数输出也是 7，所以二进制数组第 7 位仍置为 1，添加模式 I 后的布隆过滤器二进制数组变为 0110001100000010。

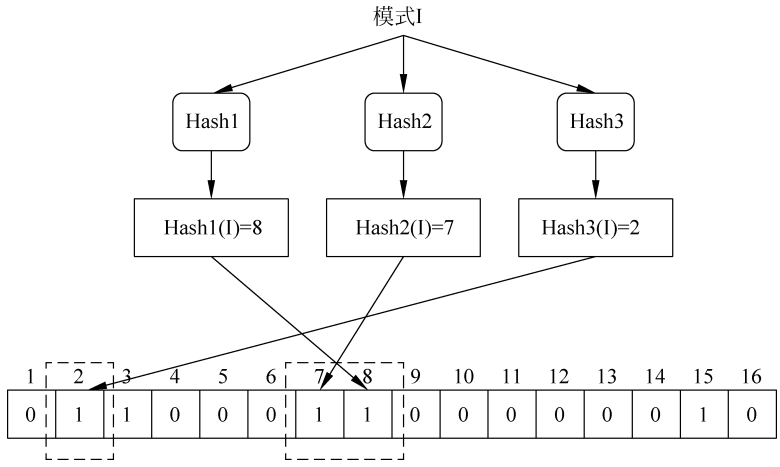


图 5-55　新的模式记录进二进制数组

(4) 对特定模式是否为布隆过滤器的一部分进行验证：

验证一个模式是否为布隆过滤器的搜索条件之一，只需要把模式输入哈希函数，将计算

的结果结合布隆过滤器的二进制数组比对，看二进制数组中对应哈希结果的位置是否为1，如果是，那么该模式“可能”已经被记录进布隆过滤器。因为这些位也可能是其他模式的结果所记录的，所以说是“可能”，并不一定是。这样，节点将可能符合布隆过滤器搜索条件的交易相关信息过滤出来发给SPV节点，如图5-56所示。

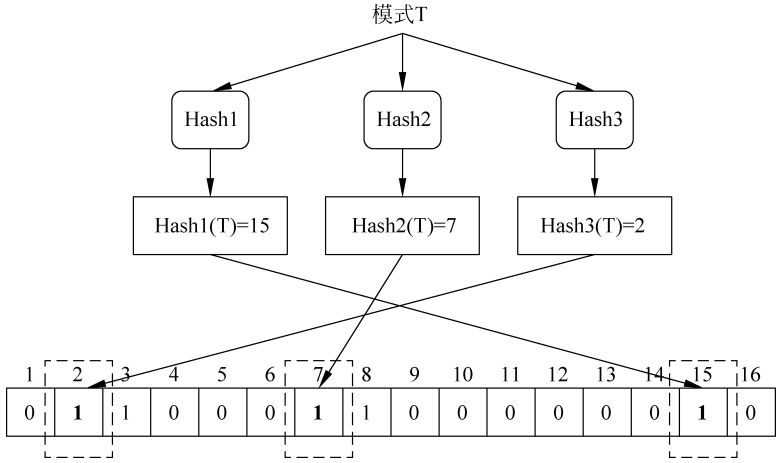


图5-56　模式T经过验证可能是布隆过滤器记录的搜索条件之一

只要布隆过滤器二进制数组中对应哈希结果的位置有一个不为1，则说明此模式一定不匹配，注意，是“一定”。如图5-57所示，模式X经过3个哈希函数计算得到的哈希值分别为16、2、8，而二进制数组中对应16位置的值为0，所以可以肯定模式X不存在于布隆过滤器。这样，使用布隆过滤器的节点就知道相应的交易不是SPV节点感兴趣的交易，从而不会将其发给SPV节点。

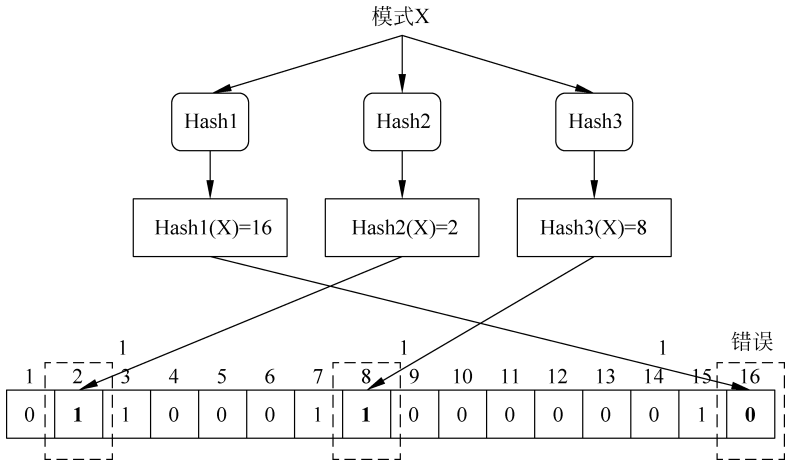


图5-57　模式X经过验证肯定不存在于布隆过滤器

SPV 节点使用布隆过滤器来实现简单支付验证的流程如图 5-58 所示。

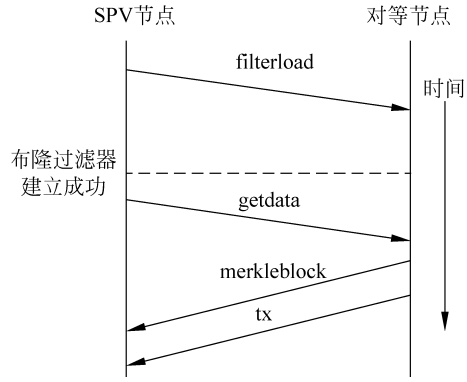


图 5-58　SPV 节点使用布隆过滤器来实现 SPV

(1) 首先 SPV 节点创建一个自己感兴趣交易的布隆过滤器，从钱包的 UTXO 列表中提取公钥哈希、脚本哈希、交易 ID，从而列出它感兴趣的所有的地址、公钥、哈希值等，将每一个模式添加进布隆过滤器。

(2) SPV 节点发送 filterload(过滤器加载)消息给对等节点，filterload 消息中包含了与该 SPV 节点连接时使用的布隆过滤器。

当布隆过滤器建立起来后，对等节点就可以通过布隆过滤器检查交易的每个部分，包括交易 ID、每个交易输入、每个交易输出锁定脚本中的数据(公钥、哈希)等，以此过滤出相应 SPV 节点的付款交易和收款交易，当然也包括一些混淆视听的交易。对于匹配过滤器的交易，对等节点就将其相关信息通过消息发送给 SPV 节点。

(3) SPV 节点向对等节点发送 getdata 消息来获取交易。

(4) 对等节点收到 getdata 消息后，把匹配布隆过滤器的每个交易所对应的区块头部以及该交易连接到区块头部里的梅克尔树根的梅克尔路径，通过 merkleblock 消息发送给 SPV 节点，同时也把该交易信息通过 tx 发送给 SPV 节点。

(5) SPV 节点收到区块头部、梅克尔路径和交易后，丢弃它不关心的交易相关信息，利用梅克尔路径信息把感兴趣的交易和区块头部里的梅克尔树根关联起来，验证交易是否存在于该区块中。同时 SPV 节点也能利用区块头部信息将区块与区块链关联来验证区块，这样通过交易关联区块的验证、区块关联区块链的验证这两个方法确保了交易已经被记录进了区块链中。目前一个区块的大小约为 1MB，而 SPV 节点收到的区块头部和梅克尔路径的大小不到 1KB。

(6) SPV 节点根据接收到的感兴趣的交易更新与自己相关的 UTXO 列表和钱包余额，并修改布隆过滤器，以确保后续如果有引用该 SPV 节点 UTXO 列表中新增加的 UTXO 的交易，新的布隆过滤器能够匹配到。SPV 节点通过发送 filteradd 消息给完全节点将模式新增到过滤器中。布隆过滤器的这种交互式修改只支持模式增加，不支持模式删除。SPV 节点如需删除某个模式，只能先通过 filterclear 消息将整个过滤器从完全节点清除，再重新发送新的过滤器给完全节点。

5.3.6　交易池

在比特币网络中，几乎每个节点都会维护一个临时的未确认交易列表，我们称之为交易池(transaction pool)或内存池(memory pool)。那些已经创建并发布到网络上，且通过节点的交易验证但是尚未被写入区块链的交易，都会被接收的节点放入各自的交易池中，如

图 5-59 所示，这样便于节点跟踪交易。

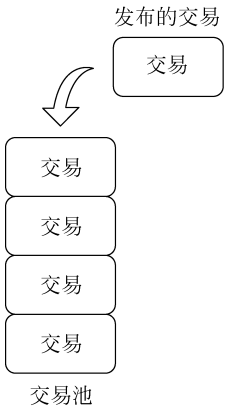


图 5-59　交易池示意图

交易池是一块内存[①]，当节点上电启动时，交易池为空，随着节点不断接收交易而动态填充，又随着交易被纳入区块链而动态减少。

在本书第 3 章提到过孤儿交易池，孤儿交易池和交易池一样也是存储在内存中，孤儿交易池里面存放的都是那些暂时找不到所引用的前序交易的交易。当新交易被纳入交易池时，节点首先检查孤儿交易池，判断里面是否有引用该交易的输出的孤儿交易，一旦发现有孤儿交易引用该交易的输出，则立刻将该孤儿交易从孤儿交易池中移至交易池，同时搜索孤儿交易池，检查是否有孤儿交易引用这个不再是孤儿的交易的输出，以此类推，直至补全整条交易链。

有些节点还保存有 UTXO 列表，这个列表看起来和交易池类似，但实质上却大相径庭。首先，两者存放的内容不同，交易池里面存放的是没有写入区块链的未确认交易，而 UTXO 列表里面存放的是已经确认过的交易输出；节点与节点之间的交易池可能由于节点的全新加入或重启而差别很大，但它们之间的 UTXO 列表因为代表着网络当前的共识而差异很小。其次，与交易池不同，UTXO 列表不会被初始化为空集合，它里面保存着数量庞大的未花费交易输出。最后，交易池存放在内存中，而 UTXO 列表既可以存放在内存，也可以存放在永久性存储空间中。

① 非永久性存储存储器，里面存放的数据一旦遇到设备断电就会永久丢失。