目录

1 背景

在 linux 史上，包管理器可以说是一个非常伟大的发明。曾经，当人们想安装软件时，要么去找编译好的二进制包，要么

2 重要性与前瞻性

2.1 包管理器的出现

在 linux 史上，包管理器可以说是一个非常伟大的发明。曾经，当人们想安装软件时，要么去找编译好的二进制包，要么自己下载源码进行编译。然而，前者数量极少，后者动辄 1 小时的编译时间和可能出现的各种错误令普通用户望而却步，加之版本与依赖的混乱，安装需要的软件成了一件非常耗时的事情。

若 Alice 想在自己的虚拟机上安装 linux kernel 4.9，如果她完全不使用包管理器，那么她必须进行以下操作（以下均为真实操作）

```
wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.9.tar.xz
make menuconfig
make
```

除此以外，menuconfig 中的复杂的内核编译选项对于非专业人士是极不友好的，编译过程中可能发生的种种错误也需要安装者自己去查找解决办法。

直到第一个包管理器的出现改变了这一切。

部分包管理器及平台

```
apt-get Ubuntu
yum CentOS
pip python
npm javascript
...
```

聪明的开发者们开始收集稳定且常用的软件，以特定规范打包放在一个服务器上，安装流程和依赖库自然都附带在其中，任何一台安装有包管理器的主机都可以从服务器获取这些软件并自动安装到本机，整个流程基本不需要人去干预，也很少会出错误。

现在，如果 Alice 想安装软件（例如 gedit），只需要：

```
sudo apt install gedit
```

聪明的包管理器就会完成一切工作。

2.2 软件源与镜像服务器

wiki 对于包管理器功能的介绍

- Working with file archivers to extract package archives

  与解压软件协同工作以打开被压缩的软件包

- Ensuring the integrity and authenticity of the package by verifying their digital certificates and checksums

  通过数字签名和校验码来确保软件包的完整性和权威性

- Looking up, downloading, installing or updating existing software from a software repository or app store

  对任意存在的软件包进行查找，下载，安装和升级等操作

- Grouping packages by function to reduce user confusion

  对软件包基于其功能进行分类，方便用户管理

- Managing dependencies to ensure a package is installed with all packages it requires, thus avoiding "dependency hell"

  管理软件包之间的依赖关系，确保每个软件包的依赖项都被正确安装，并能正确处理“特定依赖”

  注：“特定依赖”即指待安装软件包 A 依赖于特定版本的软件包 B，版本过高或过低均不能满足需求

从这里我们可以很清楚地看出，包管理器与相应的服务器是密不可分的。在这里，相应的服务器会保存许多常用软件的全部版本，并且根据稳定程度/用途/版权进行不同的划分，我们称之为软件源或者软件仓库。

随着包管理器开始流行，这种一键式安装方式收到大家的追捧。单一的主服务器已经开始难以满足越来越多的下载需求，镜像服务器随之诞生。

镜像服务器一般都会定时与主服务器进行同步以保证软件最新，这样做的好处是明显的：主服务器的压力减轻了。人们开始从离自己比较近的镜像服务器获取软件包。

2.3 镜像服务器的局限

在单个服务器带宽日益成为瓶颈的今天，同一地区数量不多的镜像服务器已经越来越难给所有用户提供稳定高速及时的同步更新。在默认服务器出现故障时，修改软件源

服务器这样的操作，对于普通用户也并非很便捷。除此以外，镜像服务器之间上游源和下游源之间同步的时间差，有时也会造成一定影响。

当前，面临这种情况，我们的解决方法是设立更多的镜像服务器，号召更多大学和机构的志愿者投入人力去维护。然而，当服务器增多时，新的问题出现了：负载难以均匀。

中国国内的部分镜像源

mirrors.ustc.edu.cn

mirrors.aliyun.com

mirrors.tuna.tsinghua.com

...

人们都有趋利避害的本能，然而每个镜像服务器性能是不同的，无论是从带宽还是从可容纳的最大并发量来说。不难理解，人们会聚集到下载速度最快的镜像服务器。而那些比较慢的镜像服务器则会无人问津。当大家都聚集在少数服务器时，最初的问题再次发生了。由此可见，【设立更多的镜像服务器】只能缓解下载压力，却并不能彻底解决这样的问题。

2.4 项目构想

让我们设想以下情况：

Alice 是一名个人开发者，她位于 North Korea。Alice 的国家对于互联网有很强的管制。显然，官方源无法访问，而她所在的地区也并没有公开的镜像源。那么，她想更新自己的 linux 软件，就只能使用虚拟专有网络(VPN)。然而，VPN 的低速与不稳定性让她非常难受。

Bob 是巨硬公司的一名程序员，他开发了一个基于 docker 的云平台。在一次 docker 的大更新时，Bob 采用了 docker 的新特性重写了部分组件，也发布了自己的新版本。但是很快，Bob 收到了很多反馈，许多用户投诉说云平台无法运行，奇怪的是也有部分用户表示一切正常。经过仔细的检查，Bob 终于发现，出错的那部分用户采用的是三级软件源 USTC mirror，官方源的 docker 更新还未同步到镜像上。

很明显，上面所提到的镜像服务器的局限是真实存在的。

那么，我们需要一个分布式，相对公平，动态调整的全新的包分发机制，最好还要是能自动同步，不需要人力去维护，而且能在少数节点出现故障时不影响整个网络的工作。

首先，我们把用户节点分为 A 类和 B 类：

A 类储存以下内容：

所有 B 类节点的地址

部分软件包（可能并不完整）

B 类储存以下内容：

软件包索引（每个 A 类节点对应储存的软件包）

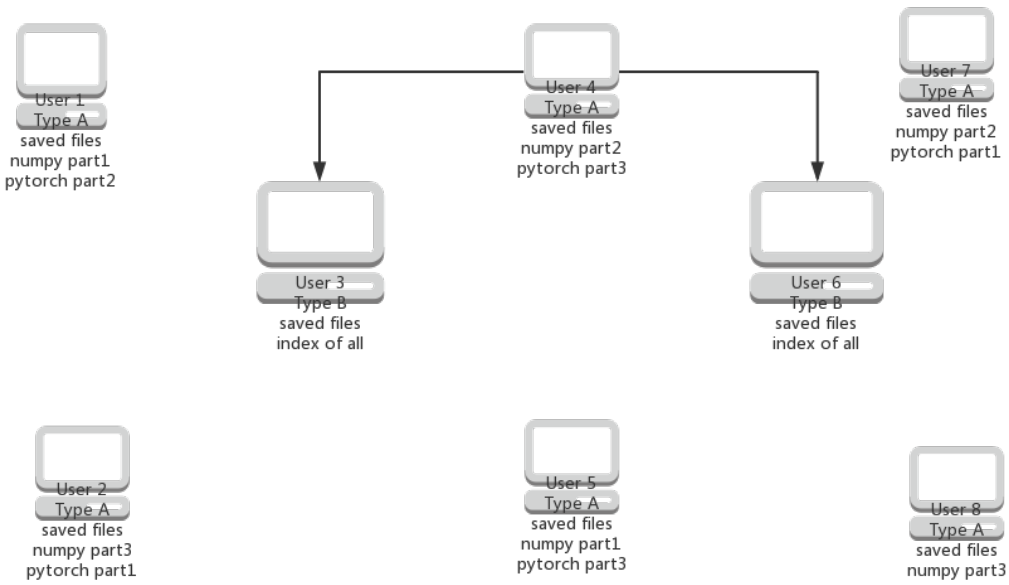
在一个小型的有 8 个节点的网络中，如果 4 号节点想要下载 numpy：

1. 它会根据储存的地址访问节点 3 和节点 6，询问储存 numpy part1，

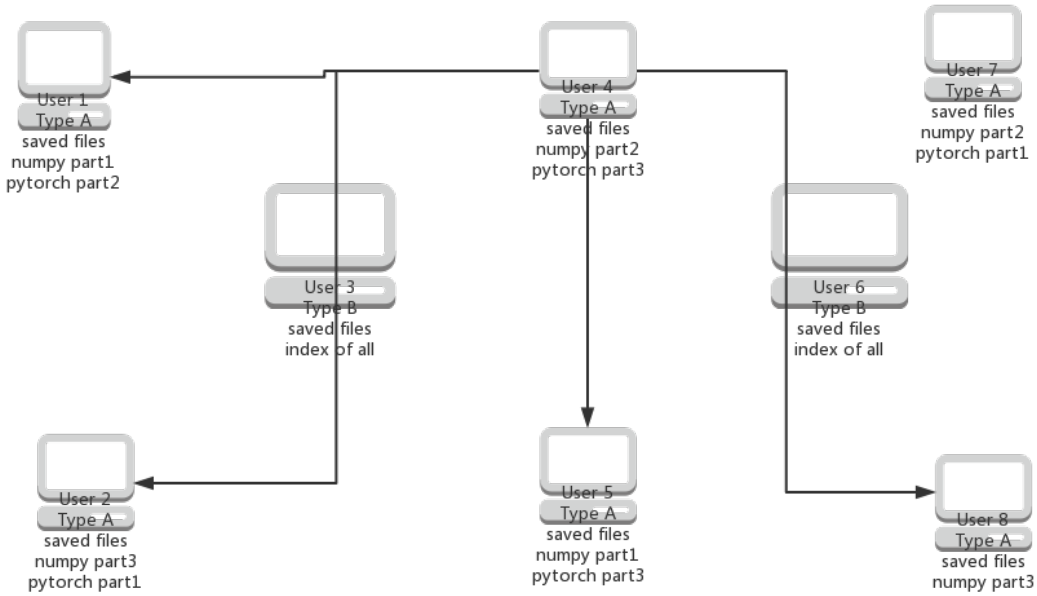
   3 的节点，返回

   节点 1，5--part1—hash

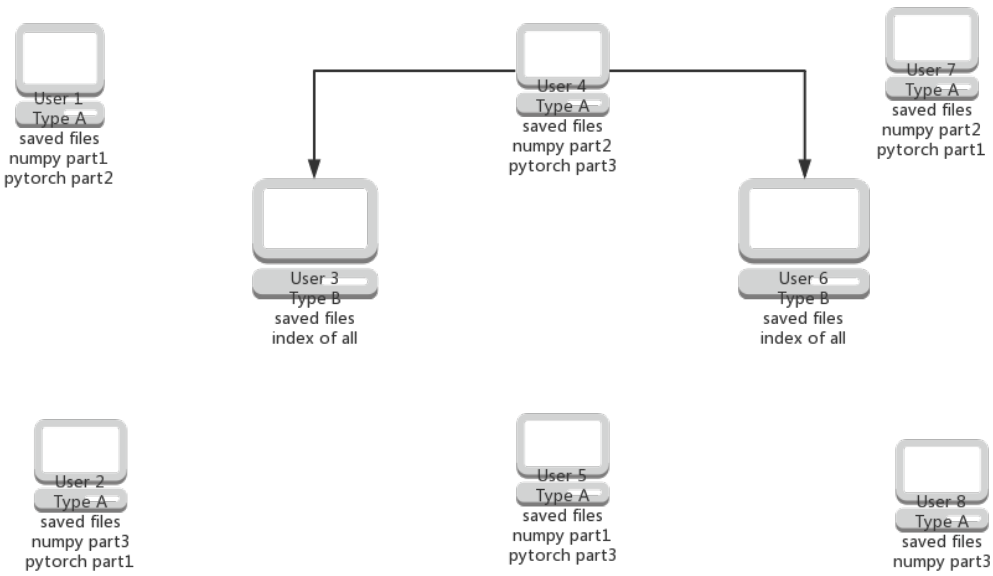
   节点 2，8--part3--hash



2. 它访问上述节点，获取 numpy 的剩余两部分，并使用 hash 进行验证



3. 下载完成后，4 号节点将 1，2，5，8 号节点的文件可用性/网络速度等连接参数返回给 3，6 号节点



现在，Alice 可以通过别的节点获得 linux 的更新，Bob 的云平台的用户也能获得最新的更新（因为只要有一个用户进行了更新，更新便会被推送到整个网络中）。

我们期望，通过这个项目，能够构建一个至少由几千个节点构建的包分发网络。所有的软件包，被分散储存在各个节点上。而每个节点，都能迅速地从其他节点获取软件包。当一次下载请求成功或失败后，请求节点会给来源节点打分，分数则作为激励，决定在网络拥塞的时候，哪些节点能获得更高的速度。那些比较稳定的节点会变成一级服

务器给其他节点提供索引。对于每一个软件包，它会被储存多少份完全由它的热度与重要性决定。

同步是由节点完成的，当官方源发生更新时，新的软件包在被确认后会加入这个网络，并被部分节点储存。

而最有趣的地方是，这一切将完全由网络本身计算出来并且动态调整。设定好规则之后，它不需要人力去维护，也不可能被摧毁（只要按照规则运行的节点还有一定数量）。当然，对于伪造的节点和中间人攻击，这个网络也应有足够的健壮性来判断和驳回虚假的更新请求。

linux 与开源界的核心哲学是"自由"，与其依赖一个传统的镜像源，使用分布式的 P2P 分发网络想必更加能贯彻"人人为我，我为人人"的自由思路。

3 立项依据

3.1 关于中心化与镜像源的现象与弊端

3.1.1 关于中心化

众所周知，互联网是建立在 HTTP 协议上的. HTTP 协议是个伟大的发明，让我们的互联网得以快速发展.但是互联网发展到了今天 HTTP 逐渐出来了不足.

1. 中心化是低效的，并且成本很高
2. Web 文件经常被删除
3. 中心化限制了 web 的成长
4. 互联网应用高度依赖主干网

关于第一点：使用中心化的服务器下载完整的文件(网页，视频，图片等)，速度慢，效率低. 如果改用 P2P 的方式下载，可以节省近 60%的带宽. P2P 将文件分割为小的块，从多个服务器同时下载，速度非常快.

关于第二点：回想一下是不是经常你收藏的某个页面，在使用的时候浏览器返回 404(无法找到页面)，http 的页面平均生存周期大约只有 100 天. Web 文件经常被删除(由于存储成本太高)，无法永久保存. 而我们做的包分发系统，会类似 IPFS 一样提供了文件的历史版本回溯功能(就像 git 版本控制工具一样)，可以很容易的查看文件的历史版本，数据可以得到永久保存

关于第三点：我们的现有互联网是一个高度中心化的网络．互联网是人类的伟大发明，也是科技创新的加速器．各种管制将对这互联网的功能造成威胁，例如：互联网封锁，管制，监控等等．这些都源于互联网的中心化.而分布式的系统可以克服这些 web 的缺点.

主干网受制于诸多因素的影响，战争，自然灾害，互联网管制，中心化服务器宕机等等，都可能是我们的互联网应用中断服务．使用分布式系统可以是互联网应用极大的降低互联网应用对主干网的依赖.

由以上可以看出，不仅仅是我们要做的包分发系统，各种应用，只要需要投入到互联网当中去，都会产生类似的问题。

3.1.2 关于软件包管理与分发的现状

现在我们自己使用的软件包的管理软件都要配置一个好的，稳定的软件源

我们用的这些源一般就是镜像源，比如：科大镜像，清华镜像，豆瓣等。
那么这就会引出来一个相应的问题：这些镜像由谁来维护呢?

这些镜像站是公益的！

引用科大镜像源的一段历史：

科大最早的镜像站诞生于 2001 年 3 月。2002 年的时候，开始使用 debian.ustc.edu.cn 域名。2005 年，科大 LUG 发出 fund raising campaign，利用捐款买了一台机器，而 2008 年时，吴峰光先生捐献了一台整机给科大，又有一人捐赠了硬盘，使一个新的镜像站点诞生，那就是 oss.ustc.edu.cn。

引用一位曾经科大镜像的维护人员张成的说明：

我在 2010 年左右接手维护 {debian,oss}.http://ustc.edu.cn。而从最初一直到我接手的这段时间，两个镜像站点一直在打游击战，一般都放在当时维护者所在的实验室（因此当时的维护者也基本都是研究生），每次当期维护者毕业之后就要搬迁一次。在 2011 年左右，由于 debian.ustc 的访问量越来越大，老的机器数次宕机。同时，硬盘仓位也已经填满，无法继续添加新的硬盘，老机器已经不再适合继续服役了。此时，网络中心的 jameszhang 老师捐赠了一台新的服务器，我们将 {debian,oss}.ustc 两台机器上的内容全部迁移到了新的服务器上，并且启用了新的域名：http://mirrors.ustc.edu.cn （瀚海星云 精华区文章阅读:mirrors.ustc.edu.cn 正式开张）。

就此，我们可以看出镜像的第一点弊端，硬盘空间要不停扩张，到一个程度甚至需要增添新的服务器。

访问量过大也会对镜像站提出越来越高的要求

精华区在维护镜像站期间，为了找到合适的上游镜像，也找过许多国外的镜像站。我发现大多数规模比较大的镜像站，都是由学校学生维护的，也托管于学校。在国外，由于人力成本比较高，而学生维护则基本没有人力成本（不需要支付工资），带宽成本基本可以忽略。常常是一些组织给学校赞助机器，然后由学生来维护镜像站。国内的情况稍不同，但是结果是一样的。

从这里可以看出为了维护一个镜像站，需要专业人员的不停维护（包含学生），对人力的有持续的需求

而且国内由商业组织维护的镜像站点，我知道的最早的是 http://mirrors.anheng.com.cn/debian/ （当年也是 Debian 在中国的官方镜像之一）。其实严格来说，这个镜像站并不算是商业组织维护的。维护者是该公司的刘世伟（似乎有江湖名号龙芯三剑客之一，另外两个是张乐和孙海勇）。刘世伟利用其公司办公室的带宽提供这个镜像服务。大概在 3 年前，我参观了一次安恒公司，他们的带宽只有 4Mbps，现在估计也没有增加。所以，这个镜像其实在许多年前就已经没有能力提供优质的服务了。在 debian.ustc 流行起来之后，这个镜像站就已经很少为人所知了。然后 163、sohu 的镜像也出来了。现在由商业公司维护的开源镜像就越来越多了。例如各云主机厂商一般都会提供，还有一些文艺范的公司，比如豆瓣之类的，也会提供。

现在软件源的需求可见一斑，由此，若能够使得包的分发摆脱了传统镜像源所固有的缺陷，这便是需求的所在，也正是我们开发分布式包分发系统的意义

国内的带宽成本非常高，而相对来说，一些高校的带宽很富裕，对带宽成本并不敏感，所以镜像站点大多数也都在高校中。而科大的新的镜像站点，不知为何知名度挺高，上线没多久压力就非常大，一度打满千兆网卡。系统各方面都碰到了瓶颈，为此不得不做更多的优化，尝试更多的技术。

由于镜像站也是属于趋中心化的，其巨大的访问，会导致其对服务器带宽的要求额外的高。一个很讽刺的例子就是使大多百度云盘用户感到困扰的限速问题：现在使用百度云的用户都知道，要从百度的云端上将自己存储的大文件下载下来是一件多么有难度的事，就前两年来说，使用百度云下载（如果没有 VIP），带宽会被限速至 100Kbps。在如今人均带宽都在逐年提升的进程中，这是多么让人难过的事。为什么？因为百度云盘的用户过于庞大，如果人人都要以自家最大带宽来访问百度云，那么服务器就无法承受了。

再说商业公司。首先，服务器比较多一些的商业公司，一般都会自己搭建镜像站（只是大多数没有对公开放）。上百上千台服务器，要装个软件都得用一下公网镜像，也是很可怕的事情，一方面网速慢，另一方面，万一公网镜像挂了就悲剧了，不能依赖于不可控的第三方镜像。所以，这些商业公司都会自己在内网搭建一个镜像站。而云主机厂商，更进一步

的，不仅自己要用，自己的客户也要用。像阿里云，算上所有的虚拟机，数量可能在几十万到几百万这个量级，如果他们不提供镜像，那么任何一个第三方的镜像站都可能被压挂。

由此看来，不仅有学校科研的需求，各大商业公司也都有对软件源的需求，为了防止公网镜像失效产生的后果，不得不投入资金在自己搭建镜像站。同样，这也需要人维护。

镜像站本身是无法盈利的。因此提供镜像站点本身肯定都是公益的。开源镜像服务却并没有合适的盈利模式（至少现在还没有发现），对用户收钱肯定是不合适的，这会极大的打击大家参与开源活动的积极性。广告也是不可能的，开源站虽然流量大，但是几乎没有PV。因为没有合适的盈利模式，我认为在接下来的很长的时间内，开源镜像服务可能主要还是由高校以及对开源服务依赖比较强的商业公司提供。然而，学校的带宽是有上限的，随着开源社区越来越庞大，学校的资源会无法满足开源社区的需求，因此，接下来，商业公司会成为开源镜像服务的主力。

如此重要的镜像站，无法盈利，却需要自身投入大量成本，这固然已经成为其缺陷之一了

3.1.3 结论

由以上对镜像站的说明，我们可以得到以下几点：

- 镜像站需要不停更新硬件，以满足对存储空间的逐步扩大的需求。包括（硬盘，服务器）
- 镜像站会由于访问量的提升涉及到：
    - 系统稳定性要求，不至于使服务器崩溃。
    - 带宽的量与成本的需求
- 选定镜像站后，由于是单一源，下载速度受到限制
- 对于软件包分发的需求是足够大的，各个镜像站的存在对资源产生了过度浪费，对其优化甚至革新是有价值的
- 镜像站没有合适的盈利机制，需要单方面投入成本

针对以上问题，可以得出分布式包分发系统的优越性

镜像站存在的问题	分布式包分发系统给予的方案
镜像站需要不停更新硬件，以满足对存储空间的逐步扩大的需求。包括（硬盘，服务器）	分布式包分发系统的站点是每个用户，硬件是否更新取决于用户自身，新的空间来源来自于新的用户，无需强制更新

镜像站会由于访问量的提升涉及到：1.系统稳定性要求，不至于使服务器崩溃。2.带宽的量与成本的需求	1.由于系统是分布性的，采用区块链模式，并不要求每个节点都随时在线，即使个人电脑出现故障，也不会影响整个系统，因此系统是稳定的。2.由于下载来源各个节点，不会对单一节点造成过大带宽负担。
由于是单一源，下载速度受到限制	由于是多个源，在来源方面不会受到带宽限制，可以达到自身最大带宽
对于软件包分发的需求是足够大的，各个镜像站的存在对资源产生了过度浪费	此系统使用的资源都只是各个电脑节点中的一小部分，不会造成大的资源浪费
镜像站没有合适的盈利机制，需要单方面投入成本	这个系统不需要盈利，符合开源理念，其哲学为“有付出才有回报”，自身提供小部分硬盘空间，得到优质的包分发服务。

3.2 关于设计方向

系统必须拥有的性质：

- 离散型(Autonomy and decentralization)：构成系统的节点之间都是对等的，没有中央控制机制进行协调
- 伸缩性(Scalability)：不论系统有多少节点，都要求高效工作
- 容错性(Fault tolerance):不断有节点加入和离开,不会影响整个系统的工作

对于方向问题，采取 Q&A 方式阐明：

此处对比其他分布式系统：namecoin、洋葱网络、storj、还有现在大热的 ipfs

他们共同面临的问题有两个：

1. 法律问题，这种东西无法监管，有害信息深恶痛绝

   A: 我们的分布式包分发系统会于官方软件源保持同步，不会存在除了软件包以为的有害信息。
2. 标准问题，没有大型组织来统一制定标准，比如 IE 不支持，便很难对普通大众普及使用。

   A: 我们初步使用的是命令行使用方式，制作成软件，制作全新的一套包分发软件系统，不需要 IE 等支持。

更多疑问：

a. 节点的所有者可以自由选择想要维持的数据，如果有个数据大家都不愿意存储，是不是意味着如果上传者删除了该数据，那么这个数据也就从此消失了呢？

A: 我们的系统采取区块链的方式存储，并不需要节点存储完整软件的所有块，所有者并不是自由选择想要维持的数据，而是有系统同样分配，节点只需要提供存储空间即可

b. 相同内容的文件在系统中只会存在一份，节约存储空间。如果这份文件都没有人 Pinning，是不是意味着也有消失的可能？是不是 IPFS 还有一部分主动备份所有数据的功能？

A: 我们的数据不止存储一份，至于存储的份数，这是由系统中的特殊评判标准决定的

c.如何保证节点的所有者可以自由选择想要维持的数据？共享了硬盘后，写入的数据貌似我们是不知晓的，只是做了一个传播的媒介？

A: 不保证自由选择，这没有什么意义，但我们会采取相应的奖励机制，使得有付出有回报。

由以上 Q&A，我们已经可以得出做这个项目的必要依据了：

- 虽是分布式系统，但只涉及包分发，可以避免很多分布式文件系统的缺陷
- 分布式包分发系统对镜像源的一系列缺陷的避免，是对企业，学校教育，科研需求的正确考量，具有足够的价值。
- 符合 CAP 理念：扩展性好。支持线性扩容，当存储空间不足时，可以采用热插拔的方式增加存储设备，扩展方便。可靠性强。分布式文件系统包含冗余机制，自动对数据实行备份，在数据发生损坏或丢失的情况下，可以迅速恢复。可用性好。用户只需要拥有网络就可以随时随地的访问数据，不受设备、地点的限制。

4 相关工作

4.1 星际文件系统-IPFS

4.1.1 什么是 IPFS

星际文件系统(InterPlanetary File System, IPFS)是一种永久的、内容可寻址的、版本化的、点对点的超媒体的分布式存储、传输协议。它的核心是梅克尔有向无环图(MerkleDAG)，图的链接是哈希值。其目标是补充甚至取代当前的超文本传输协议--HTTP，以构建更快、更安全、更自由的互联网环境。
它由 Juan Benet 在 2014 年 5 月份发起。

IPFS 综合了以前的对等系统的成功想法，包括 DHT，BitTorrent，Git 和 SFS。它将这些技术简化、发展并将成熟的技术连接成一个单一的内聚系统。
IPFS 是完全传输中立的，平台上的任何部分都不会存储到一个集中的服务器上，这有利于传输的自由并且保证高速。

4.1.2 IPFS 的特点

4.1.2.1 三大主要特点

1. 内容可寻址：IPFS 用由文件内容生成唯一哈希值来标识文件，而非文件保存位置。并有删除冗余的机制，节约存储空间。
2. 版本化：整合 Git，可追溯文件修改历史。
3. 点对点超媒体：P2P，保存各种各样类型的数据。总之，IPFS 类似于互联网，也可以被看作是一个独立的 Bittorrent 群在同一个 Git 仓库中交换对象。

4.1.2.2 一些其他特点

- IPFS 提供了一个高吞吐量、按内容寻址的块存储模型，及与内容相关的超链接。
- IPFS 结合了分布式散列表、鼓励块交换，是一个自我认证的空间
- 没有单点故障，节点之间不需要相互信任
- 分布式内容传递可以节约带宽，提高传输速度
- 防止 HTTP 方案可能遇到的 DDoS 攻击
- 可以通过多种方式访问，包括 FUSE 和 HTTP
- 将本地文件添加到 IPFS 文件系统可使其面向全世界可用
- 有一个称为 IPNS 的名称服务，它是一个基于 PKI 的全局命名空间，用于构建信任链，并与其他 NS 兼容，并可以映射 DNS、.onion、.bit 等到 IPNS。

4.1.3 IPFS 的使用场景

- 在/ipfs 和/ipns 下挂载全球文件系统
- 文件加密，数据共享系统
- 挂载的个人同步文件夹，拥有版本功能
- 可用于所有软件的带版本的包管理器
- 可以作为虚机的根文件系统
- 可以作为数据库：应用可以直接操作 Merkle DAG，拥有 IPFS 提供的版本化、缓存以及分布式特性
- 可以做（加密）通讯平台
- 各种类型的 CDN
- 永久的 Web，不存在不能访问的链接
- IPFS 提供了编写和部署应用程序的新平台
- 一个新的版本化大数据分发系统

4.1.4 IPFS 的架构

IPFS 协议分为几组负责不同功能的子协议(自上而下)：

子协议	功能简述
身份	管理节点身份生成和验证
网络	管理与其他对等体的连接，使用各种底层网络协议
路由	维护信息以定位特定的对等体和对象，响应本地和远程查询
交换	一种支持有效块分配的新型块交换协议(BitSwap)，模拟市场，弱化数据复制
对象	具有链接的内容寻址不可更改对象的 Merkle DAG。用于表示任意数据结构
文件	由 Git 启发的版本化文件系统层次结构

子协议	功能简述
命名	自我认证的可变名称系统

这些子系统不是独立的;它们是集成在一起，互相利用各自的属性。

4.1.4.1 身份

IPFS 节点由 NodeId 标识，是使用 S/Kademlia 创建的公钥的密码散列。节点存储其公私钥，
首次连接时，对等体交换公钥，并检查：hash(other.PublicKey)是否等于 other.NodeId。如果没有，则连接被终止
关于加密函数的注意事项：

IPFS 不是将系统锁定到一组特定的功能选择，而是支持自我描述的值。哈希摘要值以多重哈希格式存储，其包括指定使用的哈希函数的头和以字节为单位的摘要长度。这允许系统可以：

1. 选择最佳功能用例（例如，更强的安全性与更快的性能），
2. 随着功能选择的变化而演变。自描述值允许兼容使用不同的参数选择。

4.1.4.2 网络

IPFS 节点与数百个其他节点进行定期通信时，可能跨越广域网络。网络层比较核心，其使的 LibP2P 可以支持任意传输层协议。
而 NAT 技术能让内网中的设备共用同一个外网 IP，这和家庭路由器是同一个原理。
IPFS 网络堆栈功能：

1. 传输层： IPFS 可以使用任何传输协议。
2. 可靠性： 如果底层网络不提供可靠性，IPFS 可使用 uTP 或 SCTP 来提供可靠性。
3. 可连接性：IPFS 还可以使用 ICE NAT 穿墙打洞技术。
4. 完整性：可以使用哈希校验和来检查邮件的完整性。
5. 可验证性：可以使用发送者的公钥使用 HMAC 来检查消息的真实性。

4.1.4.3 路由

路由系统可用于查找:

1. 其他同伴的网络地址,
2. 专门用于服务特定对象的对等节点。

IPFS 使用基于 S/Kademlia 和 Coral 的 DSHT。在对象大小和使用模式方面，IPFS 类于 Coral 和 Mainline。
它根据其大小对存储的值进行区分:

1. 小的值(等于或小于 1KB)直接存储在 DHT 上。
2. 更大的值，DHT 只存储值索引，这个索引就是一个对等节点的 NodeId，该对等节点可以提供针对该类型的值的具体服务。

IPFS 路由系统可以根据用户的需求替换的，例如广域网中使用 DHT，局域网中使用静态 HT。

4.1.4.4 块交换 - BitSwap 协议

IPFS 中的 BitSwap 协议受到 BitTorrent 的启发，通过对等节点间交换数据块来分发数据的。像 BT 一样，每个对等节点在下载的同时不断向其他对等节点上传已下载的数据，但 BitSwap 不局限于一个 torrent 文件中的数据块。BitSwap 协议中存在一个永久的所有节点组成的市场。这个市场包括各个节点想要获取的所有块数据。

在基本情况下，BitSwap 节点以块的形式彼此提供直接的值。当跨节点的块的分布是互补的时候，BitSwap 会工作的很好。在某些情况下，节点必须为自己块而工作。在节点没有所需的对等节点情况下，它会以更低的优先级去寻找对等节点想要的块。这会激励节点去缓存和传播稀有片段， 即使节点对这些片段不感兴趣。

激励机制，BitSwap 策略，BitSwap 账本等保证其达到应有的功能

4.1.4.5 Merkle DAG 对象

DHT 和 BitSwap 允许 IPFS 构造一个庞大的点对点系统用来快速稳定的分发和存储。IPFS 建造了一个 Merkle DAG，对象之间的 links 都是 hash 加密嵌入在源目标中。这是 Git 数据结构的一种推广。
Merkle DAGS 给 IPFS 提供了很多有用的属性:

1. 内容可寻址：所有内容都是被多重 hash 校验和来唯一识别的，包括 links。
2. 防止篡改：所有的内容都用它的校验和来验证。如果数据被篡改或损坏，IPFS 会检测到。
3. 重复数据删除：所有的对象都拥有相同的内容并只存储一次。这对于索引对象非常有用，比如 git 的 tree 和 commits，或者数据的公共部分。

4.1.4.6 文件

IPFS 在 Merkle DAG 上还为模型化版本文件系统定义了一组对象。这个对象模型与 Git 比较相似：

模型	简介
Block	一个可变大小的数据块
List	块或者其他链表的集合
Tree	块，链表，或者其他树的集合
Commit	树在版本历史记录中的一个快照

4.1.4.7 IPNS：命名以及易变状态

目前为止，IPFS 栈形成了一个对等块交换组成一个内容可寻址的 DAG 对象。这提供了发布和获取不可改变的对象。这甚至可以跟踪这些对象的版本历史记录。但是，这里有一个关键成分遗漏了：易变的命名。没有这个，发送 IPFS 的 links，所有新内容的通信肯定都会有所偏差。现在所需就是能有某些方法可以获取相同路径的的易变状态。

IPFS 使用 IPNS，SFS 来解决这个问题，这个巧妙的设计来使得加密后的 DAG 对象名可定义，增强可阅读性。

4.1.5. 工作流程

以一个电影为例，假设 Tom 之前已经下载过这部电影，他启动 IPFS 节点，将这个视频加入 IPFS 网络。 接着他会得到一个哈希指纹 m,同时发布到公共网关，并得到一个/IPFS/m 的路径名。

然后，他把哈希指纹会和路径名告诉 Bill，Bill 只用启动一个本地节点，对该网关发一个寻址 PIN 的请求，IPFS 自动索引分布式哈希表的哈希值，找到指纹 m 对应的节点列表。

对于大的文件，比如这样一部电影，可能分片存在其他一些子节点上，IPFS 把这些节点的列表并行抓取，最后有本地的 manager 拼成完整的文件。因为并行的速度十分之快，Bill 很快就可以看到这部电影，同时他还可以继续分享给别人。

4.1.6. IPFS 的应用意义

1. 可以为内容创作带来一定的自由。如 Akasha，它是一个基于以太坊和 IPFS 的社交博客创作平台，用户创作的博客内容通过一个 IPFS 网络进行发布，而非中心服务器。它没有太多监管的限制，也没有中间商抽成，内容收益直接归创作者所有。
2. 可以降低存储和带宽成本。如 Dtube，它是一个搭建在 Steemit 上的去中心化视频播放平台，其用户上传的视频文件都经过 IPFS 协议进行存储，具有唯一标识。相较于传统视频网站，它降低了同资源冗余程度，同时大大节约了海量用户在播放视频时所产生的带宽成本。
3. 可以与区块链完美结合。区块链的瓶颈之一就是账本的存储能力，目前大部分公链的最大问题是没法存储大量的超媒体数据在自己的链上。比特币至今全部的区块数据也才 30-40G 左右。运用 IPFS 技术解决存储瓶颈是目前来看的过渡方案。
4. 可以为传统应用提供分布式缓存方案。 如 IPFS-GEO，它是一个为传统 LBS 应用提供分布式缓存的项目，可以将地理位置坐标数据通过 GeoHash 算法转化成一维字符串，并将与之相关联的具有检索价值的数据存入 IPFS 网络，由 IPFS 网络标识唯一性，并分布在各个邻近节点上。当检索请求到来时，系统先通过字符串近似度范围比较，缩小检索范围，加快检索效率，通过 NodeID 从附近节点拿到超媒体数据，达到类似分布式缓存的效果，大大提高了 LBS 应用整个检索动作的效率。

4.1.7. IPFS 的未来

IPFS 的思想是几十年成功的分布式系统的探索和开源的产物。IPFS 综合了很多迄今为止很成功的系统中优秀的思想。除了 BitSwap 新协议之外，IPFS 最大的特色就是系统的耦合以及设计的综合性。

IPFS 是去中心化网络基础设施的一个野心设想，很多不同类型的应用都可以建立在 IPFS 上。最低限度，它可以用来作为一个全局的，挂载性，版本控制文件系统和命名空间，或者作为下一代的文件共享系统。而最好的情况是，IPFS 可以让 Web 升级一个层次，当发布一个有价值的信息时，
任何感兴趣的人都可以进行发布而不会强迫性的必须只允许发布机构进行发布，用户可以信任信息的内容，信不信任信息的发送者都是无关紧要的，
还有一个特点就是，一些重要但很老的文件也不会丢失。

4.2 区块链

4.2.1 什么是区块链

4.2.1.1 概述

区块链是一种去中心化、通过对等网络方式进行交易的账本或列表。通过统一的共识机制共同维护一份账本。每个节点都有一份完整的数据记录。区块链，成块的交易通过密码学算法连接在一起，使得整个账本公开透明、可追踪、不可篡改。
简言之，区块链的主要作用是储存信息。任何需要保存的信息，都可以写入区块链，也可以从里面读取，所以它是数据库。所有节点最后都会同步，以保证区块链一致。
区块链没有管理员，它是彻底无中心的。采用该技术，使用者可以在不需要中心化第三方的前提下，通过互联网转移价值。买方和卖方通过互联网直接进行交易，而无需可信第三方中介机构的验证。交易不是匿名的，但使用者在该技术下采用的是代用名。该技术在创建交易记录时对身份识别信息进行加密，因此不会造成个人信息的公开。

4.2.1.2 区块

区块链由一个个区块组成。区块很像数据库的记录，每次写入数据，就是创建一个区块。

每个区块包含两个部分:

区块头（Head）：记录当前区块的特征值
区块体（Body）：实际数据

区块头包含了当前区块的多项特征值:

生成时间
实际数据（即区块体）的哈希
上一个区块的哈希
...

每个区块都有一个唯一的哈希，哈希由区块头唯一决定:

Hash = SHA256(区块头)

如果有人修改了一个区块，该区块的哈希就变了。为了让后面的区块还能连到它(因为下一个区块包含上一个区块的哈希)，该人必须依次修改后面所有的区块，否则被改掉的区块就脱离区块链了。哈希的计算很耗时，短时间内修改多个区块几乎不可能发生，除非有人掌握了全网 51%以上的计算能力。正是通过这种联动机制，区块链保证了自身的可靠性，数据一旦写入，就无法被篡改。

4.2.2 区块链的现实意义

区块链的产生是应时代发展要求的。

1. 人们需要真实、有价值的信息、够降低信任成本。计算机和互联网让信息分享更加便宜、更加便捷，利用信息透明，优化价值链，提升协作效率。但是，无法杜绝的虚假信息、违约行为也让人头疼不已，基于互联网的传播和复制也极为容易，人们为信任所投入的成本已经越来越大，必然阻碍效率的进一步提升。
2. 人们需要一个将共识、行为和价值激励相互连接的生产关系网。相比工业革命带来生产力巨大飞跃，生产关系的改变就不那么巨大。人类的生产活动以组织为中心开展，依旧是自上而下、金字塔层级的中心化结构。组织业务越复杂，层级越多，要实现客观公正的利益分配就越难，因此，效率提升也就难上加难。
3. 当前金融系统仍然内生性地受制于“基于信用的模式”的弱点。我们无法实现完全不可逆的交易，因为金融机构总是不可避免地会出面协调争端。缺乏不可逆的支付手段，互联网的贸易大大受限。

   基于此，我们非常需要这样一种电子支付系统，它基于密码学原理而不基于信用，使得任何达 成一致的双方，能够直接进行支付，从而不需要第三方中介的参与。

   而区块链将分布式存储、加密技术、P2P 网络等技术融为一体，有去中心化、

去信任化 的技术优势，被人们称之为价值互联网。区块链最有可能解决人与人之间的信任问题，并缔造出新的生产关系网络——点对点价值交换。

4.2.3 区块链的应用场景

1. 数字货币:提高货币发行及使用的便利性。比特币的崛起颠覆了人类对货币的概念。比特币及其他数字货币的出现与扩展正在改变人类使用货币的方式。数字货币能够替代实物现金，降低传统纸币发行、流通的成本，提高支付结算的便利性；并增加经济交易透明度，减少洗钱、逃漏税等违法犯罪行为提升央行对货币供给和货币流通的控制力；同时，通过发展数字货币背后的区块链技术应用，扩展到整个金融业及其他领域，确保资金和信息的安全，提升 社会整体效能。
2. 跨境支付与结算:实现点到点交易，减少中间费用。通过区块链的平台，不但可以绕过中转银行，减少中转费用，还因为区块链安全、透明、低风 险的特性，提高了跨境汇款的安全性，以及加快结算与清算速度，大大提高资金利用。
3. 票据与供应链金融业务:减少人为介入，降低成本及操作风险。供应链金融也能通过区块链减少人工成本、提高安全度及实现端到端透明化。
4. 证券发行与交易:实现准实时资产转移，加速交易清算速度
5. 物联网
    - 货物运输：通过多家运输公司转移货物，确保透明性和及时送达
    - 组件跟踪和合规性：存储原件和用于车队维护的替换部件的来源记录
    - 记录运营维护数据：存储运营和维护记录，以便在业务合作伙伴之间共享或将它们用于监管
6. 身份管理
    - 构建值得信赖的数字身份
7. 供应链
    - 提高食品安全网络中的可跟踪性、透明性和效率
8. 金融服务
    - 了解您的客户：访问可信的最新客户信息，这能提高金融机构中的客户服务的准确性
    - 清算和结算：在金融机构之间实时点对点转移资金，这可以加速结算
9. 游戏、音乐等

4.2.3 区块链的架构

4.2.3.1 交易

一枚电子货币是这样的一串数字签名:

每一位所有者通过对前一次交易和下一位拥有者的公钥签署一个随机散列的数字签名。并将这个签名附加在这枚电子货币的末尾，电子货币就发送给了下一位所有者。而收款人通过对签名进行检验，就能够验证该链条的所有者。

为了防止重复交易又想要在电子系统中排除第三方中介机构，那么交易信息就应当被公开宣布，我们需要整个系统内的所有参与者，都有唯一公认的历史交易序列。收款人需要确保在交易期间绝大多数的节点都认同该交易是首次出现。

4.2.3.2 时间戳服务器

时间戳服务器通过对以区块(block)形式存在的一组数据实施随机散列而加上时间戳，并将该随机散列进行广播。每个时间戳将前一个时间戳纳入其随机散列值中，每一个随后的时间戳都对之前的一个时间戳进行增强，这样就形成了一个链条。

4.2.3.3 工作量证明

为了在点对点的基础上构建一组分散化的时间戳服务器，仅仅像报纸或世界性新闻网络组一样工作是不够的，还需要一个类似于亚当•柏克提出的哈希现金。在进行随机散列运算时，工作量证明机制引入了对某一个特定值的扫描工作。

我们在区块中补增一个随机数，我们通过反复尝试来找到这个随机数，直到找到为止，这样我们就构建了一个工作量证明机制。
这一证明机制的优点在于:

- 保证区块的信息不被修改，由于之后的区块是链接在该区块之后的，所以想要更改该区块中的信息，就还需要重新完成之后所有区块的全部工作量。
- 解决了在集体投票表决时，谁是大多数的问题。而工作量证明机制的本质则是一 CPU 一票。“大多数”的决定表达为最长的链，因为最长的链包含了最大的工作量。如果大多数的 CPU 为诚实的节点控制，那么诚实的链条将以最快的速度延长，并超越其他的竞争链条。

- 如果想要对业已出现的区块进行修改，攻击者必须重新完成该区块的工作量外加该区块之后所有区块的工作量，并最终赶上和超越诚实节点的工作量。而一个较慢的攻击者试图赶上随后的区块，那么其成功概率将呈指数化递减。
- 解决节点参与网络的程度的起伏问题。为了解决这个问题，工作量证明的难度将采用移动平均目标的方法来确定，即令难度指向令每小时生成区块的速度为某一个预定的平均数。如果区块生成的速度过快，那么难度就会提高。

4.2.3.4 网络

运行该网络的步骤如下:

1. 新的交易向全网进行广播
2. 每一个节点都将收到的交易信息纳入一个区块中
3. 每个节点都尝试在自己的区块中找到一个具有足够难度的工作量证明
4. 当一个节点找到了一个工作量证明，它就向全网进行广播
5. 当且仅当包含在该区块中的所有交易都是有效的且之前未存在过的，其他节点才认同该区块的有效性
6. 其他节点表示他们接受该区块，而表示接受的方法，则是在跟随该区块的末尾，制造新的区块以延长该链条，而将被接受区块的随机散列值视为先于新区快的随机散列值

   节点始终将最长的链条是为正确的链条

4.2.3.5 激励

每个区块的第一笔交易进行特殊化处理，该交易产生一枚由该区块创造者拥有的新的电子货币。这样就增加了节点支持该网络的激励，并在没有中央集权机构发行货币的情况下，提供了一种将电子货币分配到流通领域的一种方法。

另外一个激励的来源则是交易费。如果某笔交易的输出值小于输入值，那么差额就是交易费，该交易费将被增加到该区块的激励中。只要既定数量的电子货币已经 进入流通，那么激励机制就可以逐渐转换为完全依靠交易费，那么本货币系统就能够免于通货膨胀。

激励系统也有助于鼓励节点保持诚实。如果有一个贪婪的攻击者能够调集比所有诚实节点加起来还要多的 CPU 计算力，那么他就面临一个选择:要么将其用于诚实工作产生新的电子货币，或者将其用于进行二次支付攻击。那么他就会发现，按照规则行事、

诚实工作是更有 利可图的。因为该等规则使得他能够拥有更多的电子货币，而不是破坏这个系统使得其自身 财富的有效性受损。

4.2.3.6 回收硬盘空间

如果最近的交易已经被纳入了足够多的区块之中，那么就可以丢弃该交易之前的数据，以回收硬盘空间。为了同时确保不损害区块的随机散列值，交易信息被随机散列时，被构建成一种 Merkle 树的形态，使得只有根被纳入了区块的随机散列值。通过将该树的分支拔除的方法，老区块就能被压缩。而内部的随机散列值是不必保存的。

4.2.4 智能合约

简而言之，智能合约就是一段用来直接控制电子资产交易的计算机代码，智能合约可以和区块链技术无缝对接，使区块链可编程化、可定制化，智能合约因此赋予了区块链智能，使区块链可以突破汇款这一传统的应用，让区块链可以应用在更复杂的逻辑中。智能合约区别于普通程序代码的强大之处，在于它被公开而不可更改地储存在区块链之上，在定义好的内外部条件下得到区块链全网节点的忠实执行。任何人都不可能单方面篡改和阻止智能合约的执行。但是这个性质也使智能合约的漏洞不能得到及时修复，利用漏洞的攻击行为也难以被及时阻止，从而造成实实在在的危胁。

4.2.5 区块链的现状和未来

4.2.5.1 良好的发展态势

比特币自 2008 年诞生以来，以此为原型衍生出区块链技术，无数技术爱好者参与贡献，发展方向百花齐放，如：

- 以太坊(Ethereum)
- 发展数字货币为主的比特币(Bitcoin)、莱特币(Lite Coin)
- 以信息存档为方向的公证通(Factom)
- 为保护用户隐私目的的 Zcash 和 Dash
- 专注于去中心交易所的比特股(Bitshare)
- 分布式账本平台 Corda
- …

4.2.5.2 仍需解决的问题

尽管行业发展生机勃勃，但区块链无论从技术创新还是商业应用，还面临很多挑战。

1. 智能合约仍存在安全隐患，黑客可利用漏洞盗取用户的数字资产
2. 以不同应用目标而建立的区块链平台，彼此之间存在兼容性问题
3. 区块链缺少和现实物理世界的交互，让许多应用创新不得不流于形式，如商品
4. 目前，区块链应用仍有较高的技术门槛，导致大规模商用的成本太高
5. 存在性能瓶颈，目前分布式系统的吞吐量和容量均不足以支撑全球大范围高频次使用，其性能还难以赶超中心式系统，分布式系统还难以实现大规模商用
6. 以工作量证明为基础的共识机制低效、耗能、不绿色环保
7. 任何人都不可能单方面篡改和阻止智能合约的执行这一性质使智能合约的漏洞不能得到及时修复，利用漏洞的攻击行为也难以被及时阻止，从而造成实实在在的危胁。

4.2.5.3 技术改进

1. 以太坊提出基于权益证明的共识机制，并配合一套设计精巧、赏罚分明的经济学激励措施，这一新的共识机制有望使以太坊公链变得更安全、更高效和更绿色。
2. 缩短区块产生间隔时间和分区来破除吞吐量和容量的局限
3. 对智能合约进行形式化验证，在智能合约中内建危机应对机制，建立发现和修补智能合约漏洞的激励机制以有效防范智能合约自身性质带来的安全风险

4.3 BT 技术

4.3.1 概述

BitTorrent（BT）是用于点对点文件共享（“P2P”）的通信协议，用于通过 Internet 分发数据和电子文件。

BitTorrent 是传输大文件的最常用协议之一，例如包含电视节目或视频剪辑的数字视频文件或包含歌曲的数字音频文件。要发送或接收文件，某人在其连接到互联网的计算机上使用 BitTorrent 客户端。BitTorrent 客户端是实现 BitTorrent 协议的计算机程序。

4.3.2 特点

BitTorrent 协议可用于减少分发大型文件的服务器和网络影响。BitTorrent 协议不是从单一源服务器下载文件，而是允许用户同时加入一组“主机”以相互上载/下载。该协议是用于分发数据的较早的单一来源，多镜像源技术的一种替代方案，并且可以在较低带宽的网络上有效地工作。使用 BitTorrent 协议，几台基本计算机（如家用计算机）可以替代大型服务器，同时有效地将文件分发给许多收件人。这种较低的带宽使用也有助于在一个给定的区域内防止互联网流量出现大幅上升，不论是否使用

BitTorrent 协议，所有用户的网络速度都会更高。想要上传文件的用户首先创建一个小型的洪流描述符文件，通过传统方式（网页，电子邮件等）分发这些文件。然后他们通过充当种子的 BitTorrent 节点使文件本身可用。那些具有 torrent 描述符文件的人可以将它提供给他们自己的 BitTorrent 节点，这些节点作为对等节点或者监听者，通过连接到种子和/或其他节点来下载它。

正在分发的文件被分成段称为碎片。当每个同伴接收到一个新的文件时，它就成为其他同伴的来源（该片断），从而减少原始种子，且不必将该片段发送给每台希望拷贝的计算机或用户。使用 BitTorrent，分发文件需要它的人共享; 种子完全有可能只发送文件本身的一个副本，并最终分发给无限数量的同伴。每一块都由包含在洪流描述符中的加密哈希来保护。这确保了任何修改都可以可靠地检测到，从而防止意外和恶意修改在其他节点处接收到的任何文件。如果一个节点以一个真实的 torrent 描述符副本开始，它可以验证它收到的整个文件的真实性。

片段通常是非顺序下载的，并且由 BitTorrent 客户端重新排列成正确的顺序，该客户端监视它需要的片段，以及它具有的片段并且可以上传到其他对等端。在单个下载过程中，部分的大小是相同的（例如，一个 10 MB 的文件可能会以十个 1 MB 的块或 40 个 256 KB 的块的形式传输）。由于这种方法的性质，任何文件的下载都可以在任何时候暂停并在以后恢复，而不会丢失先前下载的信息，这使得 BitTorrent 在传输更大的文件时特别有用。这也使得客户能够立即找到可用的部分并立即下载它们，而不是停止下载并等待下一个（并且可能不可用的）部分，这通常会减少下载的总体时间。一旦对等体完全下载了一个文件，它就成为一个额外的种子。这种从同伴到播种者的最终转变决定了文件的整体“健康”

BitTorrent 客户端是实现 BitTorrent 协议的任何程序。每个客户端都能够使用该协议通过网络准备，请求和传输任何类型的计算机文件。对等体是运行客户机实例的任何计算机。要共享一个文件或一组文件，首先创建一个名为“ torrent ”的小文件。该文件包含有关要共享的文件以及有关跟踪器（即协调文件分发的计算机）的元数据。想要下载

文件必须首先获取它的 torrent 文件并连接到指定的跟踪器，该跟踪器告诉他们从哪个其他同伴下载该文件的各个部分。

尽管两者最终都通过网络传输文件，但 BitTorrent 下载与传统下载（例如，通常以 HTTP 或 FTP 请求为例）在几个基本方面有所不同：

1. BitTorrent 通过不同 IP 机器上的不同 IP 连接发出很多小数据请求，而传统的下载通常是通过单个 TCP 连接到单台机器上。
2. BitTorrent 随机下载或以“最稀有的优先” [11]方式下载，以确保高可用性，而经典下载是顺序的。

综合起来看，这些差异使得 BitTorrent 能够为内容提供商实现更低的成本，更高的冗余度，以及比常规服务器软件更大的抵抗滥用或“ 快速拥挤 ”的能力。然而，理论上这种保护的代价是：下载需要一定的时间才能达到全速，因为建立足够的对等连接可能需要时间，并且节点可能需要一段时间才能获得足够的数据以成为有效的上传。这与常规下载相反，尽管更容易受到超载和滥用，但可以非常迅速地达到全速，并始终保持这种速度。

4.3.3 创建和发布种子

分发数据文件的对等体将文件视为多个尺寸相同的文件，通常字节大小为 2 的幂，并且通常在每个 32 kB 和 16 MB 之间。对等体使用 SHA-1 散列函数为每个片段创建一个散列，并将其记录在 torrent 文件中。大小大于 512 kB 的部分将减小 Torrent 文件的大小以用于非常大的有效负载，但声称会降低协议的效率。[16]当另一对等以后接收特定片，所述片的哈希相比较所记录的散列来测试片是无差错的。[6]提供完整文件的对等称为播种机，提供初始副本的对等体称为初始播种机。包含在 torrent 文件中的确切信息取决于 BitTorrent 协议的版本。按照惯例，一个 torrent 文件的名字有后缀.torrent。torrent 文件具有“宣布”部分，它指定了 URL 跟踪器，以及一个“信息”部分，包含用于文件，它们的长度，所使用的块长度（建议的）名称，和 SHA-1 哈希码对每个所有这些都被客户用来验证他们收到的数据的完整性。虽然 SHA-1 已经显示出密码方面的弱点，但 Bram Cohen 并没有把足够大的风险看作是后向不兼容的变化，例如，SHA-3。

Torrent 文件通常在网站或其他地方发布，并至少注册一个跟踪器。该跟踪器维护当前参与该洪流的客户端的列表。或者，在无跟踪器系统（分散跟踪）中，每个对等者都充当跟踪器。Azureus 是第一个通过分布式哈希表（DHT）方法实现这种系统的 BitTorrent 客户端。

BitTorrent 本身并不为用户提供匿名性和安全性。可以从跟踪器获取群体中所有当前和可能以前的参与者的 IP 地址。这可能会将不安全系统的用户暴露给攻击。在极少数情况下，如果用户未经版权所有者许可发布文件，则可能会使用户面临被起诉的风险。但是，有办法促进匿名；例如，OneSwarm 项目在最初的 BitTorrent 协议之上分层保护隐私保护共享机制。中等程度的匿名性，足以让从互联网服务供应商给用户的麻烦。首先将 torrent 文件下载到公司的服务器，然后直接下载给用户。使用 i2p 等服务可以高度匿名下载。Tor 并未提供 BitTorrent 的匿名性，并且出于性能原因也阻止了它的使用（通过阻止这种类型的连接）。与 Tor 不同，i2p 被设计为与 BitTorrent 一起使用，但是，使用 i2p 时，只能从 i2p 网络内下载种子。这对于试图避免来自 ISP 的版权投诉，维护隐私或避免审查制度的用户很有用。与公众追踪系统相比，私人追踪系统为用户提供了更高的隐私程度，但却有单一集中故障点的缺点。

4.3.4 当前 BitTorrent 的应用

暴雪娱乐公司使用 BitTorrent（通过一个名为“暴雪下载器”的专有客户端，与暴雪“BattleNet”网络相关）为 Diablo III，StarCraft II 和魔兽世纪分发内容和补丁，包括游戏本身。

战争游戏使用 BitTorrent 的在其流行的游戏世界坦克，战舰世界和战机世界来分发游戏更新。

许多软件游戏，特别是那些由于带宽限制，非常频繁的下载以及不可预知的网络流量变化而导致难以托管的软件游戏，将分发一个专门的，剥离下来的具有足够功能的 BitTorrent 客户端，以从另一个游戏下载游戏运行客户端和主服务器（如果没有足够的对等端可用，则维护它）。

许多主要的开源和免费软件项目鼓励 BitTorrent 以及他们产品的传统下载（通过 HTTP，FTP 等）来提高可用性并减少自己服务器上的负载，特别是在处理大型文件时。

英国政府使用 BitTorrent 分发有关英国公民税金如何花费的细节。Facebook 使用 BitTorrent 将更新分发到 Facebook 服务器。Twitter 使用 BitTorrent 将更新分发给 Twitter 服务器。

4.3.5 基于 BitTorrent 构建的技术

1. 分布式跟踪器

   通过称为“分布式数据库”的系统引入了对“无跟踪者”种子的支持。该系统是一个分布式散列表，它允许客户端使用没有工作的 BitTorrent 跟踪器的种子。

2. 分散关键字搜索

   即使使用分布式追踪器，第三方仍然需要找到特定的 torrent。这通常以内容所有者网站的超链接形式或通过索引网站的形式完成。Tribler BitTorrent 客户端是第一个集成分散的搜索功能的客户端。借助 Tribler，用户可以找到托管在其他同伴中的托管文件，而不是集中在索引站点上。它使用 Gossip 协议为 BitTorrent 协议增加了这种能力。经过十几次下载后，Tribler 软件可以粗略估计用户的下载喜好并推荐其他内容。

4.4 Peer-to-Peer

4.4.1 基本概念

点对点（P2P）是分布式应用程序体系结构，用于在对等体之间分割任务或工作。每个结点享有同样特权，在应用程序中是等同的参与者。据说它们形成了一个点对点的节点网络。P2P 网络是围绕同等节点的概念设计的，每个节点同时作为网络上其他节点的“客户”和“服务器”。这种网络安排模式不同于客户端-服务器模式——其中通信往往来自中央服务器。使用客户端 - 服务器模型的文件传输的典型示例是文件传输协议（FTP）服务，其中客户端和服务器程序是独特的：客户端启动传输，服务器满足这些请求。

4.4.2 网络结构的实现

最常见的结构化 P2P 网络类型实现了分布式散列表（DHT），其中使用一致散列的变体来将每个文件的所有权分配给特定对等体。这使同伴能够使用哈希表在网络上搜索资源：即秘钥，值。对存储在 DHT 中，并且任何参与节点可以有效地检索与给定密钥相关联的值。

4.4.3 混合模型(P2SP)

混合模型是 P2P 和客户 - 服务器模型的组合。一个普通的混合模式有一个中央服务器运作，可以帮助同伴找到对方。Spotify 是混合模式的一个例子。混合模型样式繁多，所有这些模型在结构化服务器/客户端网络提供的集中功能与纯对等非结构化网络提供的节点相等之间进行权衡。目前，混合模型与单纯的非结构化网络或纯粹结构化网络相比具有更好的性能，因为某些功能（如搜索）确实需要集中功能，但受益于非结构化网络提供的节点的分散聚合。

4.4.4 分布式存储和搜索

P2P 网络中存在关于数据备份，恢复和可用性相关的优点和缺点。在集中式网络中，系统管理员是控制共享文件可用性的唯一力量。如果管理员决定不再分发文件，他们只需将其从服务器中删除，则用户不再可用。除了让用户无法决定整个社区的分布情况以外，这使整个系统容易受到来自其他方面的威胁和请求的影响。虽然服务器 - 客户端网络能够监视和管理内容可用性，但他们可以在选择托管的内容的可用性方面拥有更大的稳定性。客户在访问稳定的中央网络上共享的过时的内容时不应该遇到麻烦。但是，P2P 网络在共享少见的的文件方面不太可靠。因为在 P2P 网络中共享文件要求网络中至少有一个节点具有所请求的数据，并且该节点必须能够连接到请求数据的节点。这项要求偶尔难以满足，因为用户可能会在任何时候删除或停止共享这些文件。

从这个意义上说，P2P 网络中的用户社区完全负责决定可以获得的内容。不受欢迎的文件最终会消失，并随着更多人停止共享它们而变得不可用。然而，流行的文件将会变得相当容易分发。P2P 网络上的流行文件实际上比中央网络上的文件具有更高的稳定性和可用性。在集中式网络中，服务器和客户端之间的简单连接丢失足以导致失败，但在 P2P 网络中，必须丢失每个节点之间的连接才会导致数据共享失败。

4.4.5 P2S，P2P 和 P2SP 的对比

4.4.5.1 下载资源

P2S 的下载方式是通过下载服务器进行下载，同时下载资源须上传到服务器后，才可进行下载，受到下载服务器的限制，其资源是有限的。 而 P2P 下载是通过种子的方式进行传播，如果有人想把文件提供下载，只要通过软件把文件制作成种子而且发布到页面上就可以了，同时种子体积非常小，便于发布。种子发布后，只要有一个人提供共享，那么其它人就可以通过 BT 软件进行下载。P2SP 可以把所有的 P2P 共享资源与各下载服务器进行整合，所以其下载资源远远大于 P2S 方式，同时 P2SP 采用的多媒体搜索引擎技术还可以把服务器端的同一个文件的各个镜像同时找到，能够实现各个服务器同时下载，这样下载资源便非常丰富。

4.4.5.2 下载速度

P2S 虽然可以实现多线程下载，但由于其下载资源来自单一服务器，这样当下载的人数一多，其下载速度就会变的缓慢，服务器负载加大可能导致崩溃，一旦服务器崩溃或者资源不存在将无法继续下载。而 P2P 则是下载的人数越多，其下载速度就会越快，从而让我们实现飞速下载，但美中不足的是，当下载人数减少时，特别是现在有好多人

下载完成后，就不想再做种子为他人服务，这时下载速度就会急剧下降。如果找不到种子则就无法继续下载。P2SP 则是通过独特的多媒体搜索引擎技术，把服务器端的文件整合到一起，实现同时从多个服务器端下载文件，而不像 P2S 方式那样只是从一个服务器端多线程下载，这样就能用有效的使用其它服务器，这样不但减轻了服务器的压力，还为稳定高速下载提供了保障。如果服务器和镜像资源都无法链接下载，则启用 P2P 的种子资源下载。 P2S 中的 S 指代的是独立的单一服务器；而 P2SP 中的 S 则是多台服务器的一个并合。

4.4.6 P2P 以及 BitTorrent 技术应用实例

4.4.6.1 迅雷下载器

迅雷是基于 P2SP 的一款下载软件，能够大大增强下载速度。迅雷会在后台未告知用户的情况下上传用户的本地文件用于资源管理

4.4.6.2 下载原理

首先用户请求下载某个文件，迅雷客户端会获得该文件的唯一校验值，并在自己的资源服务器上搜索存放统一文件的其他服务器列表，并获得该文件在此服务器上的目录和文件名（如 ftp 或是某个公网用户）。客户端会从不同的地址下载不同的文件块以实现高速下载。在下载完成后，客户端会检验该文件的服务器列表中是否有当前用户点击的下载链接，若没有则会将改地址添加至服务器列表以供后续使用。同时，如果该用户也是公网用户，则迅雷也会将此用户地址添加至文件的服务器列表，之后下载此文件的用户可以从此用户处下载。在校验文件方面迅雷下载器使用了独特的方式，由于保密原因无从知晓，但其拥有强大的抗干扰能力（如文件名的不同）以及判断文件的完整性等。

参考文献

Wiki for BitTorrent
Wiki for Peer-to-peer
wiki for HTTP

P2S、P2P、P2SP 之对比
BitTorrent Sync: First Impressions and Digital Forensic Implications
详解 IPFS 的本质、技术架构以及应用
P2P Mixing and Unlinkable Bitcoin Transactions
IPFS(中文白皮书)

Achain 区块链白皮书

中国银行业白皮书 区块链 一银行业游戏规则的颠覆者

比特币白皮书:一种点对点的电子现金系统

区块链：从入门到精通-微软亚洲研究院

区块链入门教程-阮一峰

https://www.zhihu.com/question/19719790（关于镜像站的主要参考，张成）

https://zhuanlan.zhihu.com/p/32615963
https://www.zhihu.com/question/37439960
https://www.jianshu.com/p/50047ed80aa9
http://www.infoq.com/cn/articles/features-and-design-concept-of-distributed-system
http://blog.51cto.com/xiexiaojun/1855211
https://image.hanspub.org/Html/1-2690253_20184.htm