|  |  |  |
| --- | --- | --- |
|  | |  |
|  | |  |
|  | |  |
| IPFS网络研究报告 | | |
|  |  | |
|  | | |

内容

[1 项目背景 3](#_Toc529305135)

[1.1 项目背景 3](#_Toc529305136)

[1.2 IPFS项目介绍 3](#_Toc529305137)

[2 IPFS实验全球网络和技术分析 4](#_Toc529305138)

[2.1 IPFS实验全球网络拓扑 4](#_Toc529305139)

[2.2 IPFS模块分析 5](#_Toc529305140)

[2.2.1 IPNS 5](#_Toc529305141)

[2.2.2 IPLD 5](#_Toc529305142)

[2.2.3 Libp2p 6](#_Toc529305143)

[2.3 IPFS关键技术分析及实验 6](#_Toc529305144)

[2.3.1 Kademlia DHT 6](#_Toc529305145)

[2.3.2 Multihash路由 7](#_Toc529305146)

[2.3.3 DSHT 8](#_Toc529305147)

[2.3.4 Merkle-DAG 9](#_Toc529305148)

[2.3.5 分布式PubSub 10](#_Toc529305149)

[2.4 IPFS应用分析 10](#_Toc529305150)

[2.4.1 下一代搜索：IPFS Search 10](#_Toc529305151)

[2.4.2 去中心化社交：AKASHA 11](#_Toc529305152)

[2.4.3 去中心化游戏：Decentraland 11](#_Toc529305153)

[2.4.4 去中心化计算平台：Computes 12](#_Toc529305154)

[3 IPFS对未来极简网络架构和基础设施的启示 13](#_Toc529305155)

[3.1 IPFS与HON 13](#_Toc529305156)

[3.2 开放无极限网络 13](#_Toc529305157)

# 项目背景

## IPFS项目介绍

IPFS的全称是Inter Planetary File System，星际文件系统。项目起源于2014年Y-Combinator的创业项目比赛，IPFS的创始人Juan Benet从该比赛中获得了足够的资金支持，从而开始实施他的创意。IPFS是基于前人的智慧，包括Bittorrent、SFS、Git，进行有机的组合，实施了大量工程上的实践和创新，并且持续的发展和迭代，社区也吸引了遍布全球的极客的全心投入。

IPFS的初始目标：

* 全星际唯一的文件系统
* 下一代的Web协议

**通过将网络和存储进行整合，全球形成一个文件系统目录，**在网络文件系统层面上，IPFS的理念达到了极致；基于网络文件系统和去中心化，构建去中心化的下一代Web，使用IPFS协议替换HTTP协议，解决当前Web存在的冗余存储、内容不稳定、分发性能低的缺点。

IPFS目前仍然处于快速发展阶段，文件系统还不支持修改文件，Web层面还缺少可行的去中心化DNS。

IPFS的关键技术包括：**DHT分布式HASH表，Merkle-DAG，Multihash路由**等等。

# IPFS实验全球网络和技术分析

## IPFS实验全球网络拓扑

我们基于阿里云、腾讯云、AWS、Azure、华为云搭建了一个IPFS实验全球网络，分别采用构建私网，以及加入公网的方式，对IPFS的原理进行了分析，并尝试做多种创新，在这个过程中进一步加深了对IPFS网络的理解。



## IPFS模块分析

IPFS包括三大模块：IPNS、IPLD和Libp2p。

### IPNS

IPNS的全称是Inter Planet Name Space星际命名空间。

IPNS是IPFS的命名空间。IPNS基于自我认证文件系统，SFS （Self-Certifying File System）的命名管理和认证，详见David Mazières在1998和2000年发表的同名学术文章。

SFS的“自我认证”原理是，向客户端提供的数据通过文件名（由服务器签名）来进行认证。 因此，可以使用本地存储的透明性安全地访问远程内容。

IPNS是一种使用公钥密码（Public Key Cryptography）体系来自我认证由网络用户发布的对象（object）的SFS。 IPFS上的所有对象都可以唯一标识，而且这也可以扩展到节点。

a) 网络上的每个节点都有一组公钥，私钥和节点ID，它们是公钥的散列。

b) 节点可以使用他们的私钥'签名'他们发布的任何数据对象，并且可以使用发件人的公钥来验证这些数据的真实性。

c) 因此，可以通过用公钥密码（public key）来保护内容免受未经授权的更改。

### IPLD

IPLD，星际际链接数据，是IPFS中的协议层，基于Merkle-DAG，Merkle-DAG是一个哈希的有向无环图，Merkle树的概念以1979年获得专利的拉尔夫·梅克尔命名。

IPLD是所有哈希启发协议的单一命名空间。通过IPLD，可以跨协议遍历链接，无论底层协议如何，都可以探索数据，从而使理论上可以超越IPFS或比特币。

* merkle-link是两个对象之间的链接，它使用目标对象的加密哈希进行内容寻址，并嵌入到源对象中
* IPFS中的Merkle-link支持：加密完整性检查，不可变数据结构
* 取消引用链接时，除非链接路径无效，否则地图本身将被其指向的对象替换
* 要支持复杂的对象关系，链接可以是多哈，在这种情况下，假设它是/ ipfs层次结构中的链接，或者直接指向对象的绝对路径

组件：

* CID：分布式系统的自描述内容寻址标识符。一种在IPLD中定义哈希结构的格式。
* IPLD树：基于JSON和Protobuf描述的数据模型;和路径导航代码，意味着易于互操作，并具有可插入格式的解析器。
* IPLD解析器：它们可以表达到达工具/协议特定数据结构的路径，可以以原生格式交叉链接和遍历它们，类似于格式转换器。

### Libp2p

Libp2p是IPFS中的网络部分，包括Exchange、Routing和Network三部分。

Exchange是基于Bittorrent的Bitswap协议，用于在Peer之间互相交换自己拥有以及感兴趣的内容数据块。

Routing就是DHT部分，是IPFS的核心。

Network是Libp2p的基础模块，包括Swarm、transport和NAT穿越。Swarm负责管理Peer建立，包括bootstrap、discovery。Transport是IPFS的传输模块，集成了多种传输协议，包括TCP、UDP、QUIC等等。

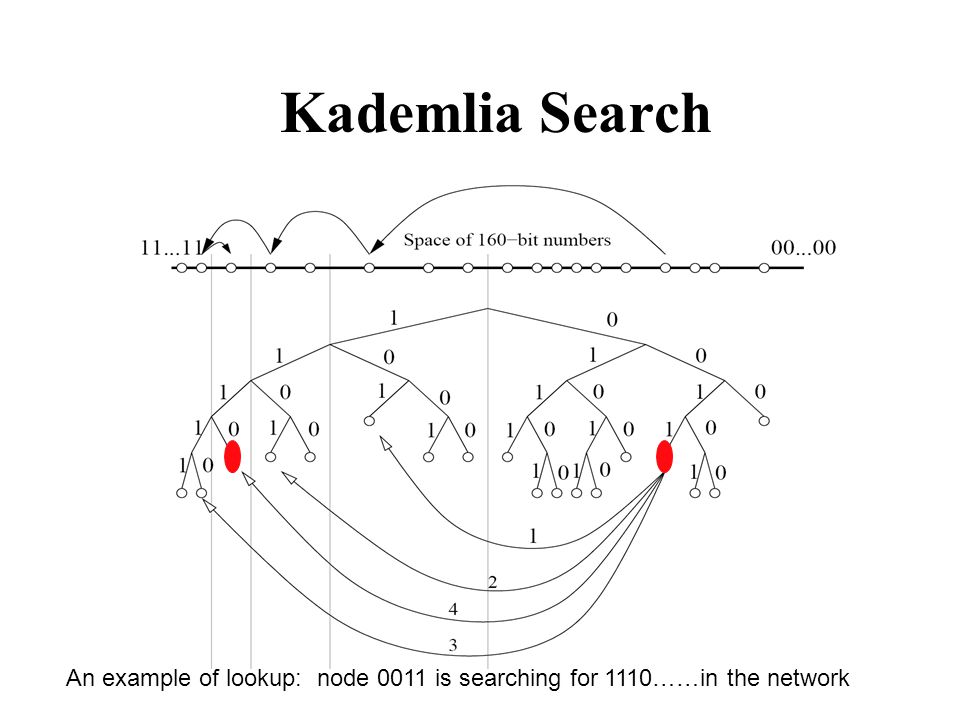
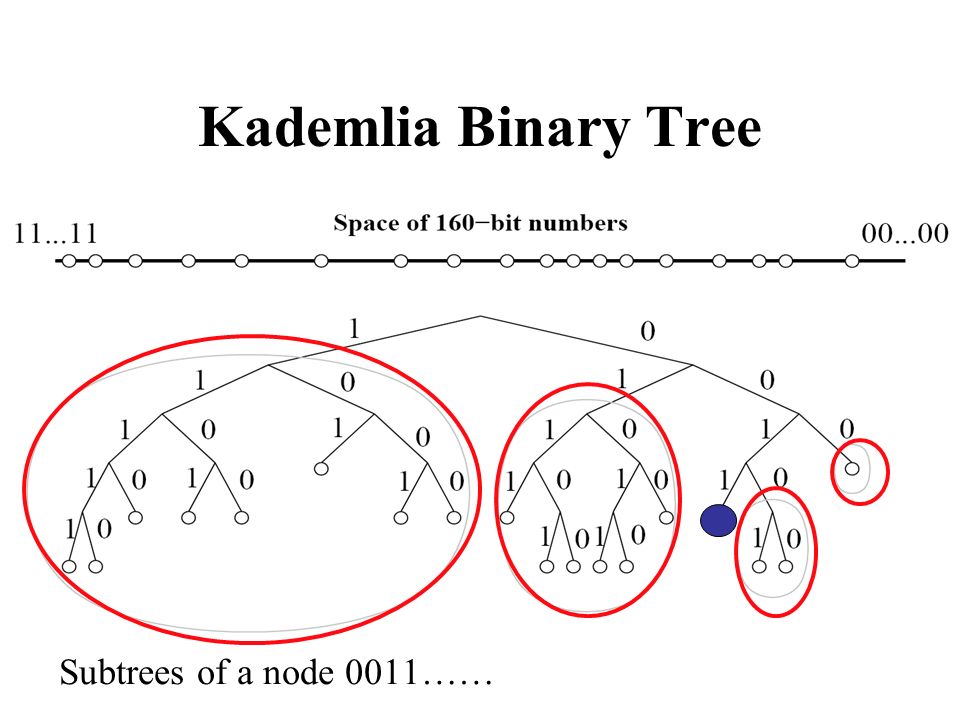
## IPFS关键技术分析及实验

### Kademlia DHT

DHT是分布式HASH表，在分布式存储、分布式网络中一直在应用。DHT中比较知名的算法包括Chord、Kademlia、Pastry等等。Kademlia因为算法定义的清晰，被大量实现采纳，IPFS也是基于Kademlia算法。

对于Kademlia，文件的HASH和节点ID的HASH是采用相同的算法，重叠的空间；因为不一定存在与文件HASH完全相同的节点ID，所以，在KAD网络里，会寻找与文件HASH距离最近的节点ID来存储该文件HASH的路由信息。此处的距离是指在KAD的二叉树上逻辑距离最近，通过异或操作可以快速的获取距离情况。对于查找该路由的节点，也是寻找距离文件HASH最近的节点ID进行查找，也就是说注册路由和查找路由都是采用相同的算法。

但是，KAD算法只考虑逻辑距离，不考虑物理距离，在现实应用中会大大增加查询的时间开销，因此，如何基于物理距离进行KAD算法优化，是我们寻找创新的一个方向。



### Multihash路由

HASH地址的优点是，去中心化自生成地址。这与当前IP地址采用集中分配的方式完全不同，具备极大的开放性和灵活性。只要保证自生成的HASH地址不重叠即可。

Multihash实际上是一种定义，通过TLV的方式对“HASH生成算法标识 + HASH长度 + HASH”进行定义，来实现系统对不同HASH算法支持的可扩展性。例如，Kad经典算法采用160bit长度，即“Kademlia + 160 + QmYKZbwYeut9LRQg721i8BHL9skhyz8TsraU41Lo6VLzc5”。相当于未来无论出现什么新的HASH类型，都可以简单的通过Multihash来扩展。

但是，在一个系统内支持多种变长HASH的DHT路由将面临两个挑战：

* 变长地址路由如何实现路由存储的负载分担，避免负载不均带来的个别节点压力过大；
* Kademlia是采用定长（160 bit）方式进行异或匹配，变长地址路由如何进行对应的异或匹配。

为了解决如上两个挑战，采用将变长地址散列映射为定长的HASH地址，通过定长HASH地址的异或匹配来解决。因为不同地址族全部散列到定长的HASH地址，所以HASH地址可能存在大量重复，不过该重复仅仅存在于注册和查询过程，并不影响实际业务地址的唯一性和地址空间。

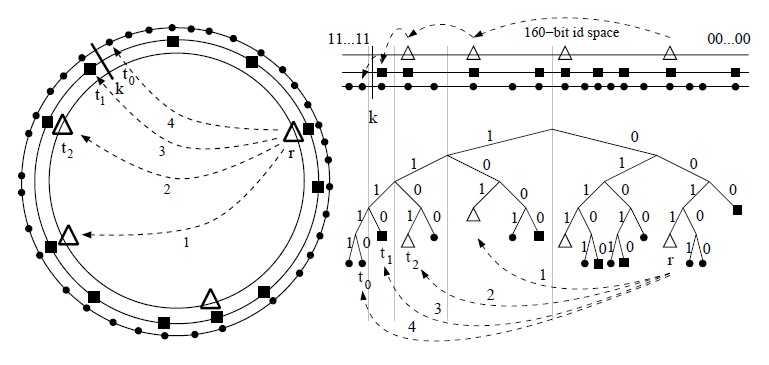
IPFS创始人Juan Benet已经在IETF提交了Multihash的草案，定义Multihash的格式。

### DSHT

DSHT全称是distributed sloppy hash table分布式草率哈希表。我们最初在对KAD算法作分析的时候，认为KAD算法有两个方向可以优化：

* KAD当前完全是基于逻辑距离进行存储，没有基于物理距离进行优化；
* 可以在KAD基础之上，实现区域性的优化，不同区域内都有一个相同HASH ID的路由存储节点，这些相同HASH ID的路由存储节点之间可以构成Cluster，在Cluster内做路由的同步，这样区域内的节点就在区域内进行查询，避免不必要的全球化查询；

后来发现DSHT已经实现了这两个优化，并且IPFS也已经按照DSHT进行了优化。对于优化点1，DSHT是基于RTT进行距离测量。除此之外，DSHT还实现了其他一些优化，例如，在路由存储节点上，可以按照一对多的方式进行存储，也就是一个路由Data，可以对应多个node\_address，而当前KAD是一对一的方式；



### Merkle-DAG

Merkle-DAG技术来源于Github的版本控制，可以实现全网一份数据及按需更新，IPFS在Github的Merkle-Tree基础之上扩展为Merkle-DAG，即有向无环图。

通过基于Merkle DAG的通用数据结构，管理网络中的文件对象，通过DAG，可以实现文件版本的追踪，按需更新。由于建立了文件的版本，因此全网可以只保持一份文件对象（此处不包括冗余）。通过这种方式，大大提高了网络的缓存效率，以及面向文件对象的快速路由。

两个对象间的链接，通过Merkle-link表示，Merkle-link将构成整个DAG图，用来做内容的寻址、版本管理和认证。通过层层叠加的DAG HASH，数据无法被篡改。

与BitTorrent的无层级分块不同，IPFS借用Github版本控制中的Merkle-tree结构，并做了框架上的扩展为Merkle-DAG，采用层次化的数据结构；理论上不同文件之间可以link到一个相同的子数据块，实现文件的最优化存储，以及按需更新。



一个实际的例子：

root@huabei1:/usr/Ipfs\_yzy# ipfs add 11.txt.txt

added QmTyaQsv3apDd8PTcA8aCYFG45UD6RE4ooWfkwjcBeMruM 11.txt.txt

root@huabei1:/usr/Ipfs\_yzy# ipfs object get QmTyaQsv3apDd8PTcA8aCYFG45UD6RE4ooWfkwjcBeMruM

{"Links":[

{"Name":"","Hash":"QmXncpAKwqcGZ2QDNmKXrFhE2H7vZRcjRhR8EXFFXUgMV4","Size":262158},

{"Name":"","Hash":"QmahgL2TsNdUv9KFzriqJLJYUm2WtJ2BCXKtDdzi4cvsju","Size":262158},

{"Name":"","Hash":"QmW7XNMyfuQqSZkZbfKijTA8cVLU8TiNpjNXrryBtpAMYb","Size":262158},

{"Name":"","Hash":"QmavdKobdfvQ745QXc4oBWQvnzjz9HzdmGy6vQiRYSkWET","Size":262158},

{"Name":"","Hash":"Qme14BUkr961wLv3Y2hHmyW75Gidn2AA1pvUenWWvutnd5","Size":63573}

],"Data":"\u0008\u0002\u0018??C ??\u0010 ??\u0010 ??\u0010 ??\u0010 ??\u0003"}

### 分布式PubSub

分布式PubSub是一个非常有用的特性，基础就是IPFS的分布式路由网络。

Pub/Sub模式对于DHT来讲，仅仅是改变了顺序，例如由消费者首先Sub对HASH的兴趣到网络中，该兴趣根据重HASH和Kad的XOR算法找到最接近的业务路由节点记录该兴趣。当生产者向网络中Pub兴趣的时候，也是采用重HASH和Kad的XOR算法找到相同的业务路由节点发布内容；业务路由节点发现Sub/Pub匹配之后向生产者发送消费者的Sub消息和包括消费者物理位置ID的路由，生产者接收到消息之后，根据消费者物理位置ID，向消费者发送数据。

分布式PubSub相当于在IPFS网络基础之上提供了一定的实时消息机制，这带来了很大的价值：

* 实现了去中心化的消息机制，无需服务器；
* 有IPFS现网做基础设施，现在就可用；

分布式PubSub对于IOT这种海量端节点的场景有比较大的价值，减少了云端的投资，也有一定的性能保障；当然，网络规模越大，性能越好。

在Github上，有一些最新的IOT项目，采用的就是IPFS提供的分布式PubSub功能。

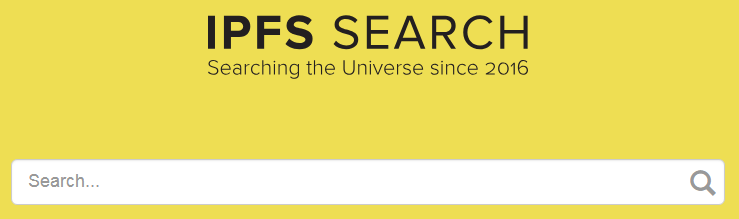
## IPFS应用分析

### 下一代搜索：IPFS Search

IPFS定位是下一代的去中心化Web协议，那下一代的Web协议自然需要对应的搜索引擎。

IPFS Search就是IPFS网络上的一个初级搜索引擎，由荷兰的极客Mathijs开发并上线运行。当前还只支持对文件名的搜索，不支持对文件内容的搜索，另外搜索结果打开的速度很慢也不稳定。

IPFS Search当前对文件名的搜索，是基于另外一个开源项目tika，实现对IPFS文件Metadata的索引。



### 去中心化社交：AKASHA

AKASHA是基于以太坊和IPFS实现的去中心化社交软件，希望通过通过区块链的激励方式，实现“社交即挖矿”。

在AKASHA上的所有操作，包括发布文章，发表评论，都是需要付出Token的，而别人的点赞，作者也可以获得对应的Token。

AKASHA的所有内容都存储在IPFS上，目标就是一个分散的、不可删除的存储。



### 去中心化游戏：Decentraland

Decentraland相当于去中心化的“明日世界”。整个游戏中的世界，人人都可以参与构建，并且可以购买其中的土地，土地的交易记录在区块链之上，而用户构建世界中物理的模型，存储在去中心化的IPFS之上。IPFS网络存储的不可删除性，也保障了用户模型的安全。

因此，Decentraland从三个维度看都是去中心化的：土地交易、世界以及模型存储。

不过Decentraland项目的进展非常慢，到目前为止还看不到任何实质进展。

### 去中心化计算平台：Computes

Computes.io项目是基于IPFS的去中心化计算项目，基于IPFS提供的去中心化网络做通信以及存储。

Comuptes项目的创始人Chris有连续创业成功的经验，以前创立的实时IOT平台Octoblu被Citrix收购。

2018年10月初，Computes.io项目被Magic Leap收购，Magic Leap可能看中的是Computes项目提供的无处不在计算能力，或许可以卸载Magic Leap头显的计算到Computes构建的分散计算平台上，结合无线充电，那未来完全可以省掉Magic Leap挂在腰上的计算+供电盒子。

