GopherChina2018

Golang打造下一代互联网

IPFS全解析

IPFS

IPFS和目前WEB现状对比

	WEB	IPFS
存储	存在大量重复内容	切分为Chunk可以去重
带宽	需要带宽资源多	需要带宽资源少
历史资源	很容易丢失	可以保留多版本
开放程度	中心化，掌握在少数机构手中	去中心化，更开放

- Distributed Hash Tables (S/Kademlia DHT)
- Block Exchanges (BitSwap)
- Version Control Systems (git)
- Self-Certified Filesystems (SFS)

IPFS DESIGN

- Identities
- Network
- Routing
- Exchange
- Objects
- Files
- Naming

Identities

为了应对网络中的Sybil attack，IPFS的node在创建NodeID时，采用了S/Kademlia 论文中提出的基于加密工作量的方式增加创建NodeID的难度，来增加攻击的成本

```
difficulty = <integer parameter>
n = Node{}
do {
  n.PubKey, n.PrivKey = PKI.genKeyPair()
  n.NodeId = hash(n.PubKey)
  p = count_preceding_zero_bits(hash(n.NodeId))
} while (p < difficulty)
```

Network

IPFS使用了多种技术来促使整个网络高效可靠通信，其中：

- 传输层: IPFS可以使用任何传输层协议，最好符合WebRTC DataChannels或者uTP。
- 可靠性: IPFS所依赖的底层网络不能保证的可靠性的话，便使用uTP或者SCTP 来保证可靠性。
- 连通性: IPFS同样使用ICE NAT遍历技术。
- 完整性: 提供使用哈希校验和来检验消息的完整性。
- 确定性: 提供使用发送者公钥的HMAC来核查消息的真实性。

Network

IPFS可以使用任何网络，包括 overlay 网络。

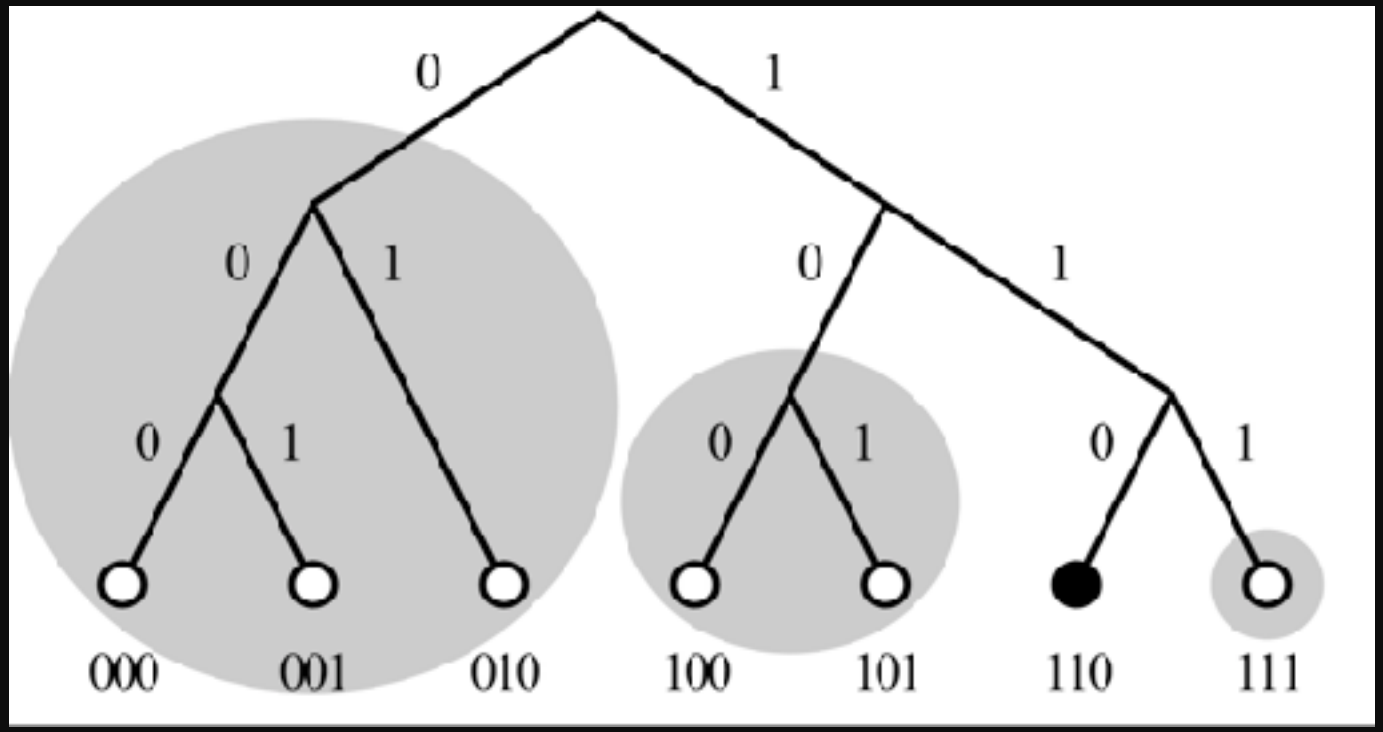
IPFS Peer Address 使用 multiaddr 格式。

```
# an SCTP/IPv4 connection
/ip4/10.20.30.40/sctp/1234/

# an SCTP/IPv4 connection proxied over TCP/IPv4
/ip4/5.6.7.8/tcp/5678/ip4/1.2.3.4/sctp/1234/
```

Route

Kademlia DHT K Bucket



Route

Kademlia DHT

- log2(N)寻址性能，一千万节点最多20跳就可以查到。
- Message比较精简
- 安全，可以抵挡多种攻击，比如拒绝服务攻击。

Coral DHT

- 记录节点中保存的数据HASH，更高查询性能
- 支持Region，优先就近查找。

S/Kademlia DHT

- PoW生成NodeID，消息需要私钥签名
- 在一半恶意节点的网络中保证85%的正确率

Exchange

* BitSwap
  交换 have_list 和 want_list
  获取数据产生债务
  发送数据偿还债务
  节点之间对账防止恶意节点

Exchange

负债率公式

$$r = \frac{\texttt{bytes\_sent}}{\texttt{bytes\_recv} + 1}$$

发送率

$$P\Big(\ send \mid r\ \Big) = 1 - \frac{1}{1 + exp(6 - 3r)}$$

当节点的负债率超过已建立信用额度的两倍时，发送率迅速降低。

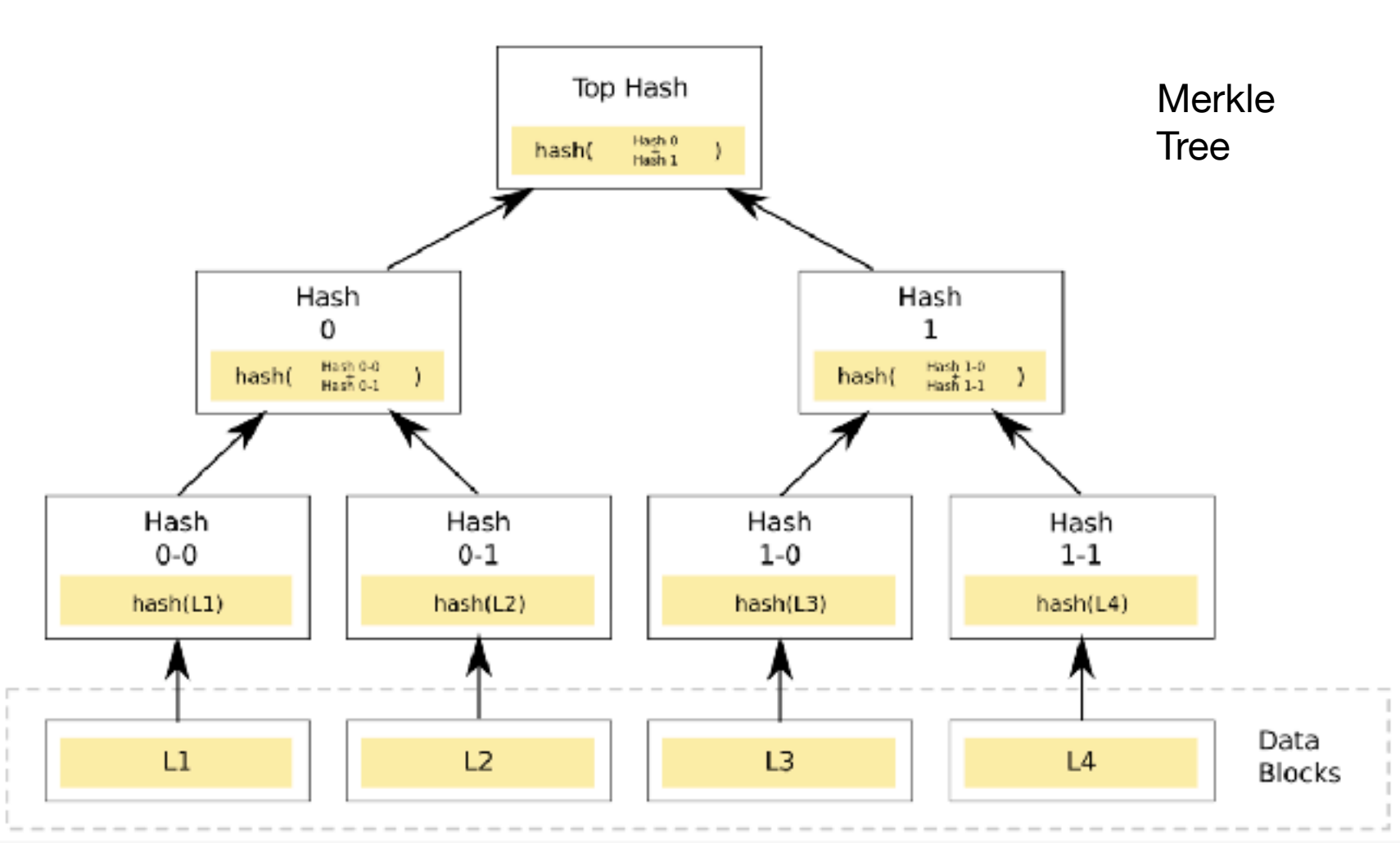
Exchange

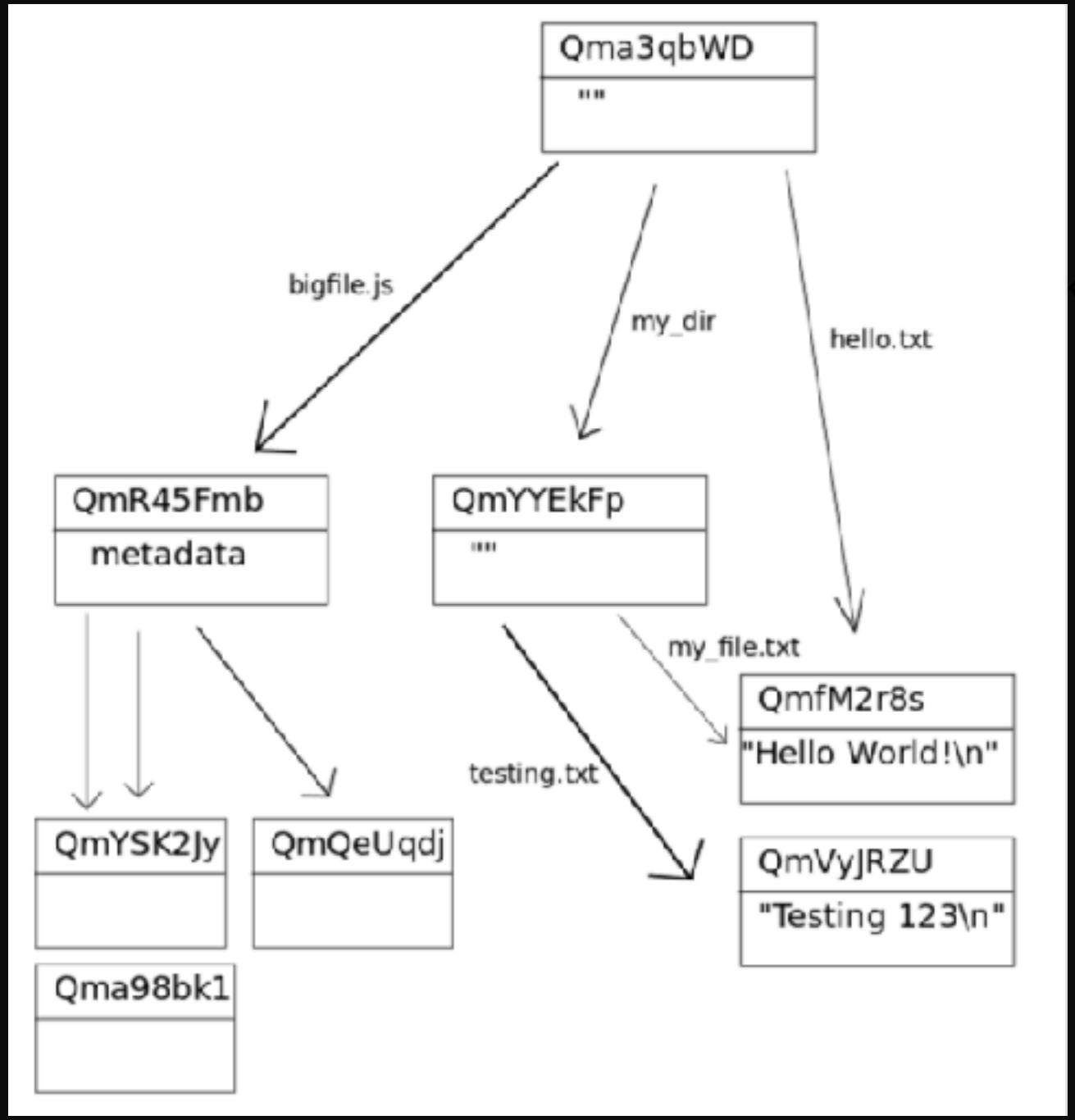
网络更高效

防止自私节点

防止攻击

File System


Top Hash
hash(Hash 0 + Hash 1)
Merkle Tree
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

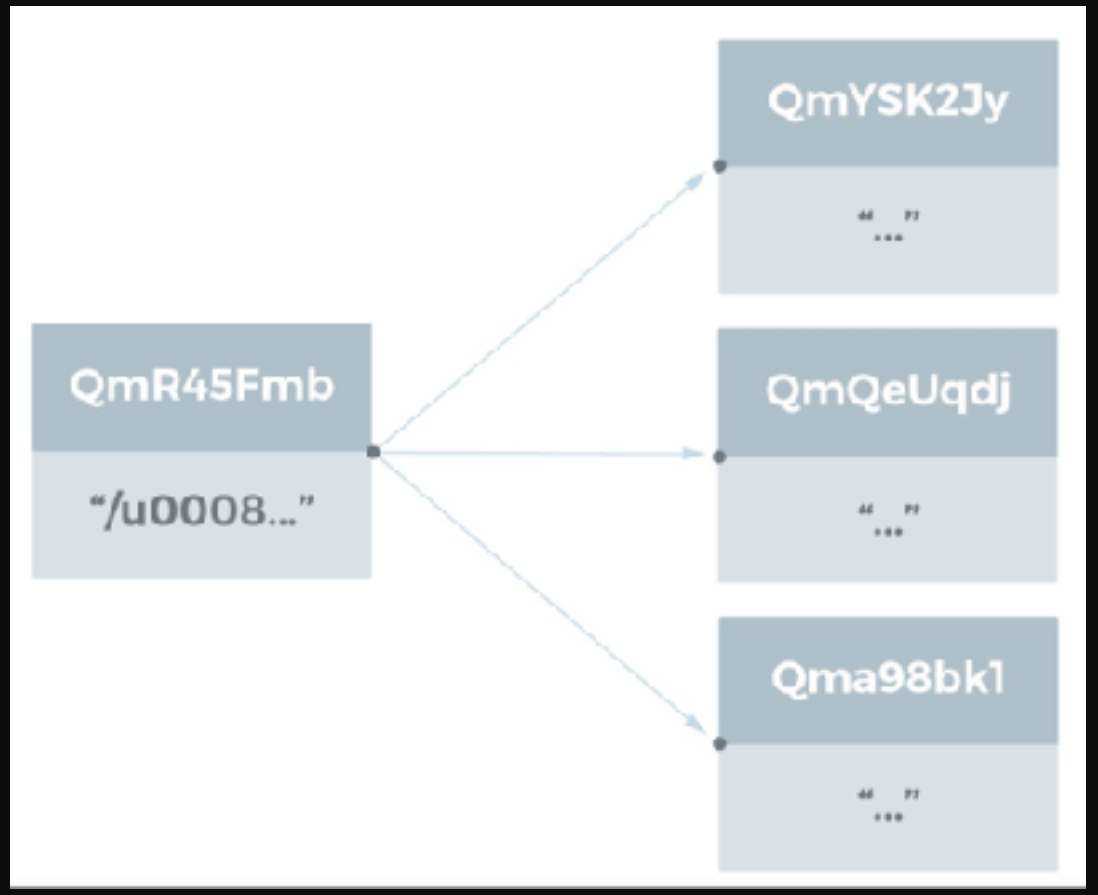

Qma3qbWD
""
bigfile.js
my_dir
hello.txt
QmR45Fmb
metadata
QmYYEkFp
""
my_file.txt
QmfM2r8s
"Hello World!\n"
testing.txt
QmYSK2Jy
QmQeUqdj
QmVyJRZU
"Testing 123\n"
Qma98bk1

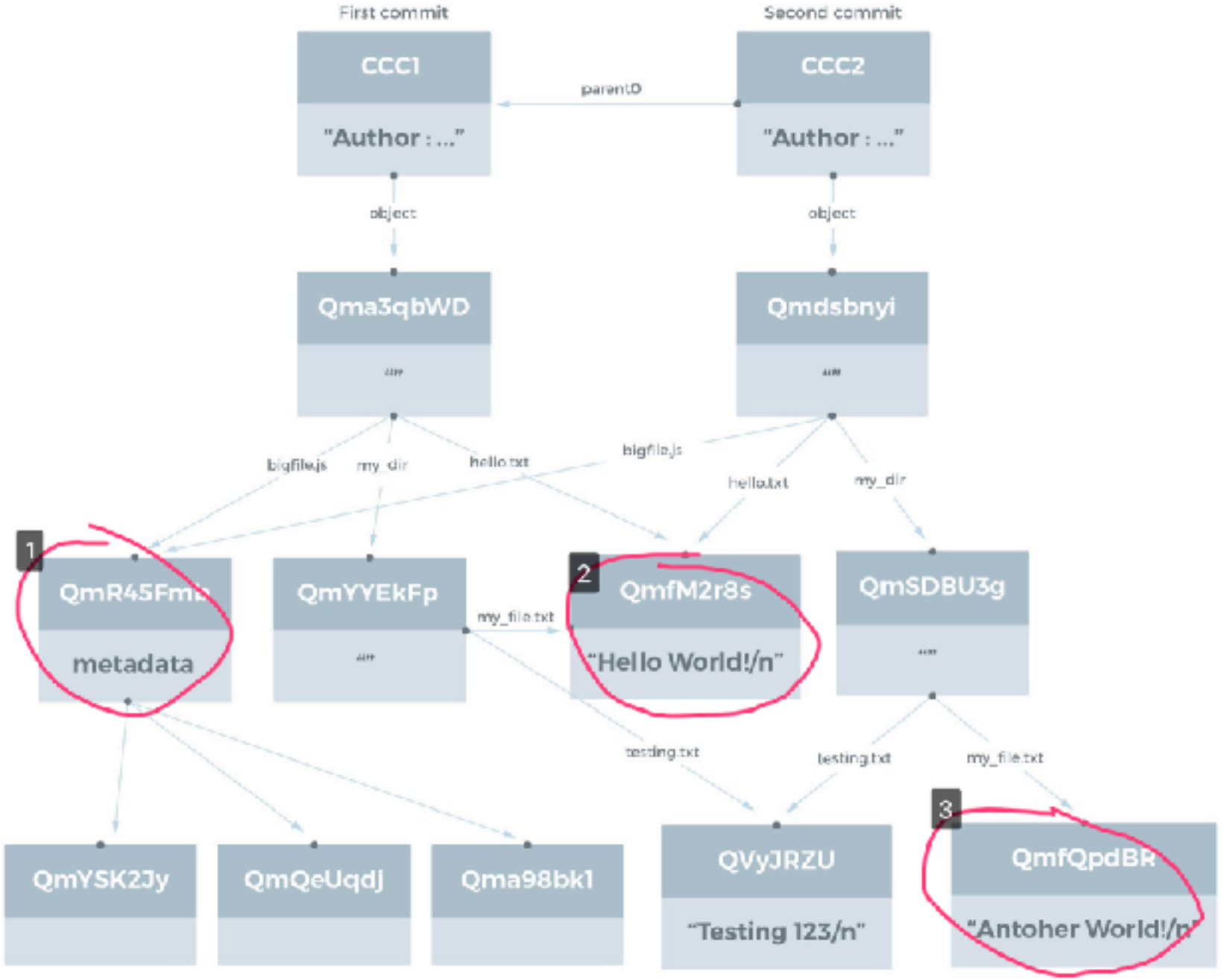
File System

* IPLD (Merkel DAG)
    1. 内容寻址: 所有的内容都通过它的multihash校验和唯一标识，包括links。
    2. 防止篡改。
    3. 重复删除: 所有的对象内容完全相同便会只存储一份。

File System

Small Data < 256K




First commit
CCC1
"Author : ..."
parentD
Second commit
CCC2
"Author : ..."
object
object
Qma3qbWD
""
Qmdsbnyi
""
bigfile.js
my_dir
hello.txt
bigfile.js
hello.txt
my_dir
1
QmR45Fmz
metadata
QmYYEkFp
""
my_file.txt
2
QmfM2r8s
"Hello World!/n"
QmSDBU3g
""
testing.txt
testing.txt
my_file.txt
3
QmYSK2Jy
QmQeUqdJ
Qma98bk1
QVyJRZU
"Testing 123/n"
QmfQpdBR
"Antoher World!/n"

Naming

SFS自命名方案

1. IPFS中: NodeId = hash(node.PubKey)

2. 我们给每一个用户分配了一个可变的命名空间: `/ipns/<NodeId>`

3. 一个用户可以通过附上它的私钥签名发布一个对象到这个路径上, 比如:`/ipns/XLF2ipQ4jD3UdeX5xp1KBgeHRhemUtaA8Vm/`

4. 当别的用户检索这个对象时, 他们可以检查这个签名是否匹配公钥和NodeId。这将验证用户发布对象的真实性，获取可变状态的查询。

Naming

Inter Planetary Name Space（IPNS）

1. 域名访问

```
# this DNS TXT record
ipfs.benet.ai. TXT "ipfs=XLF2ipQ4jD3U ..."
# behaves as symlink
ln -s /ipns/XLF2ipQ4jD3U /ipns/fs.benet.ai
```

2. Proquint可发音方案

```
/ipns/dahih-dolij-sozuk-vosah-luvar-fuluh
```

3. 短地址

```
/ipns/shorten.er/foobar
```

Q & A