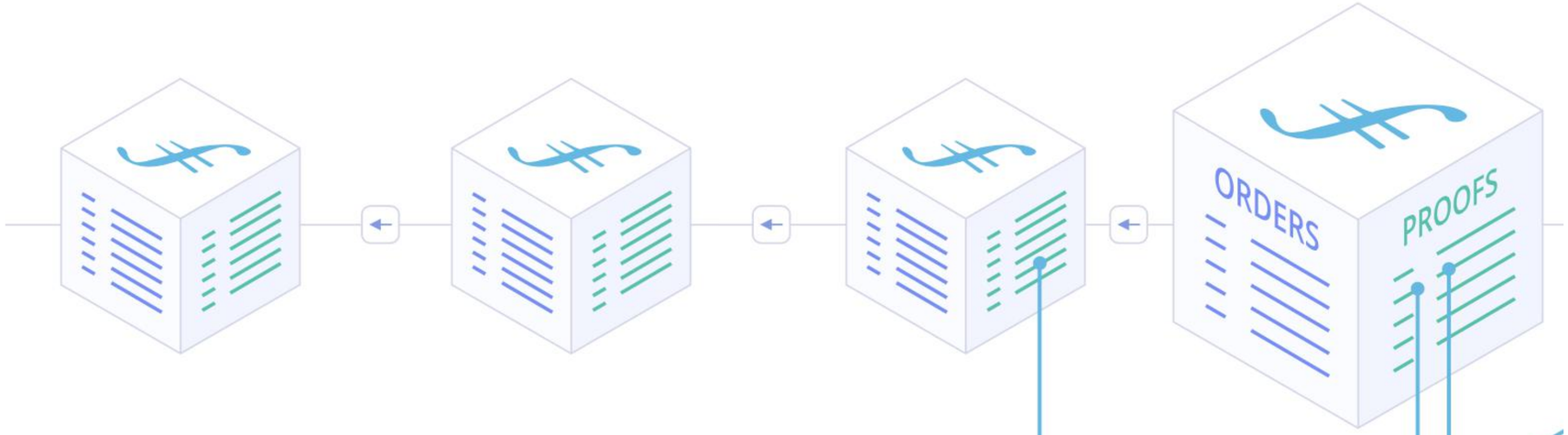


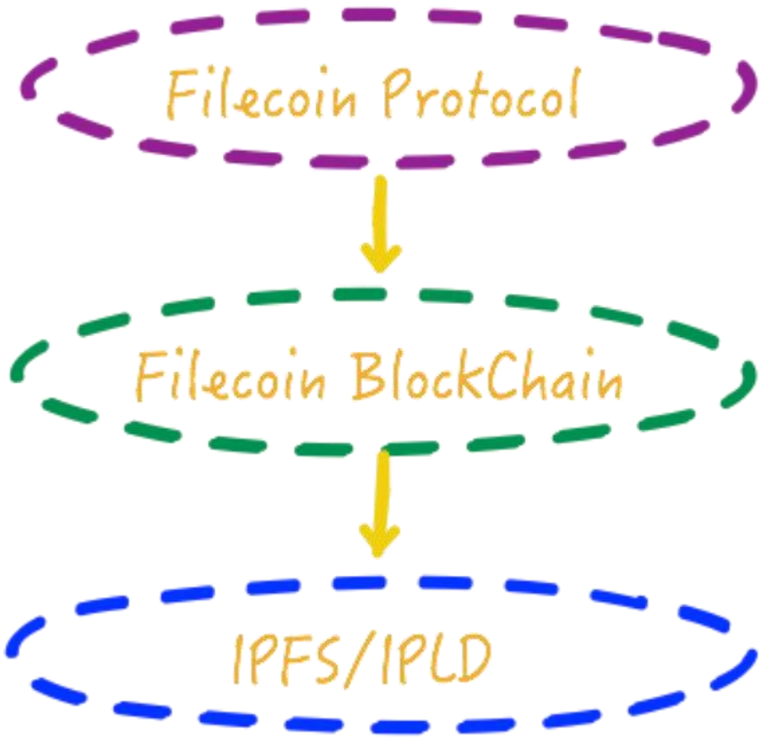
Filecoin区块链/存储协议介绍

Star Li
2019.05.19

内容大纲

- Filecoin框架
- Filecoin存储协议介绍
- IPFS/IPLD基础介绍
- Filecoin存储证明
- Filecoin区块链介绍
- Filecoin挖矿介绍

Filecoin整体框架



Filecoin Protocol

在Filecoin区块链层次之上的，存储相关的协议。

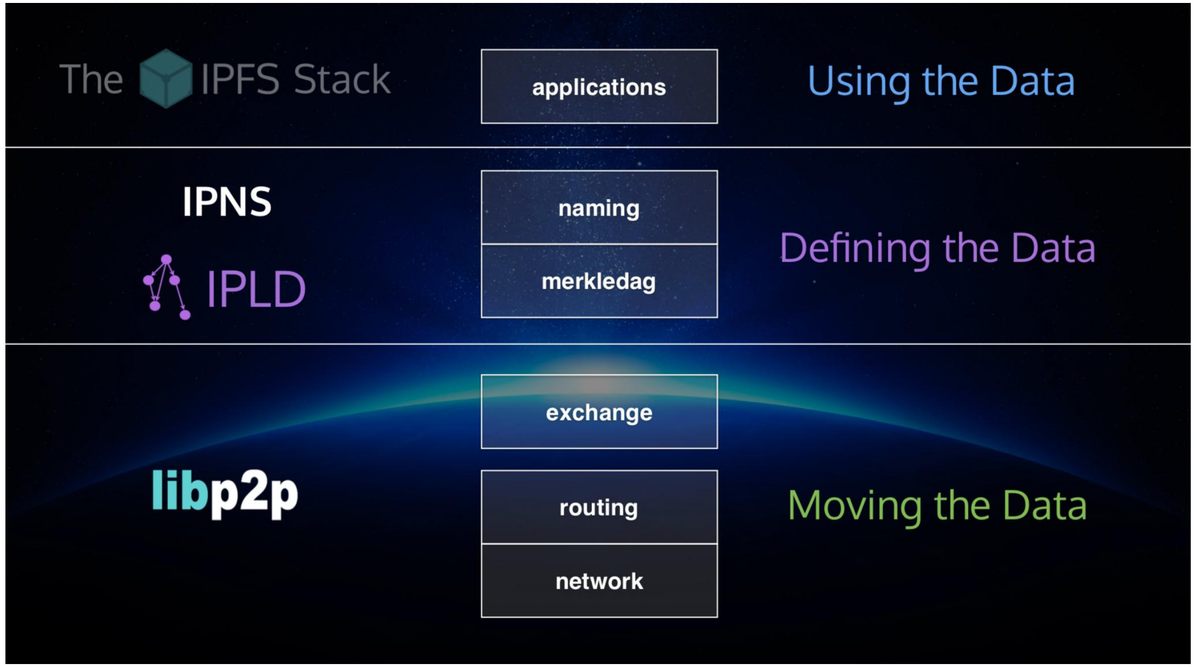
Filecoin BlockChain

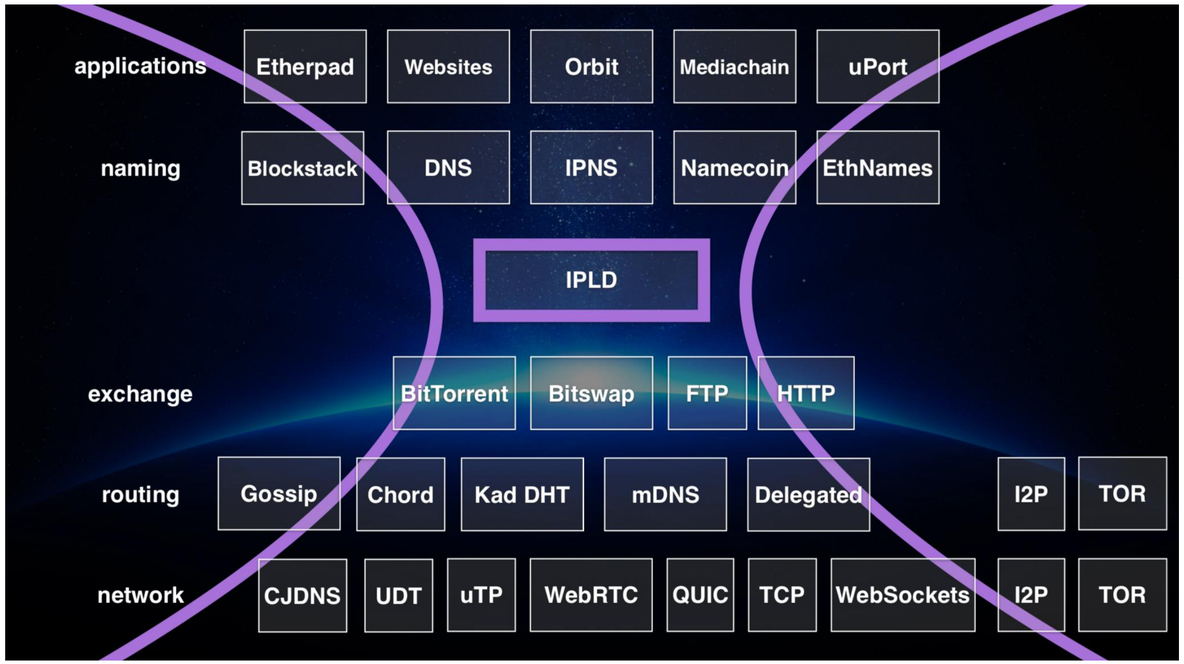
Filecoin系统的核心，也是Filecoin的分布式账本。

IPFS/IPLD

分布式的文件系统，下一代的“Web”协议。数据内容的地址由数据内容的Hash生成（地址即可验证数据内容）。

IPFS/IPLD





IPLD （Thin Waist 瘦腰）协议

IPFS/IPLD

IPFS

内容和地址的映射。

IPLD（IP Linked Data）

在IPFS之上的一种数据模型接口，实现内容的互联互通（地址带有类型属性）。

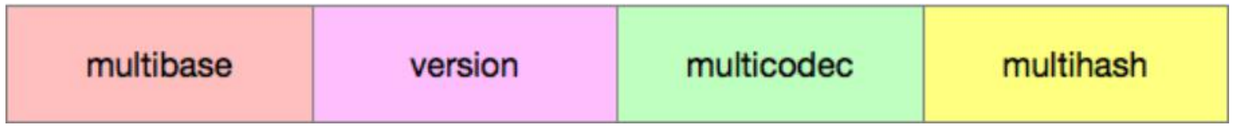
IPLD is a single namespace for all hash-inspired protocols. Through IPLD, links can be traversed across protocols, allowing you explore data regardless of the underlying protocol.

Cid（*typed* content address）

self-describing content-addressed identifier

内容地址描述标记，由MultiHash, MultiCodec, MultiBase等组成。

IPFS/IPLD - Cidv1



ip4	multiaddr	0x04
tcp	multiaddr	0x06
sha1	multihash	0x11
sha2-256	multihash	0x12
sha2-512	multihash	0x13
sha3-512	multihash	0x14
sha3-384	multihash	0x15
sha3-256	multihash	0x16
sha3-224	multihash	0x17
shake-128	multihash	0x18
shake-256	multihash	0x19

Filecoin BlockChain - 基本术语

Message

Filecoin网络中的交易。

Block

Filecoin区块链中的交易打包形成一个Block。

Actor

Filecoin网络中的Actor可以类比以太坊网络中的账户（一般账户或者智能合约账户）。

TipSet

一个TipSet，就是多个区块信息的集合，这些区块拥有同一个父亲区块。

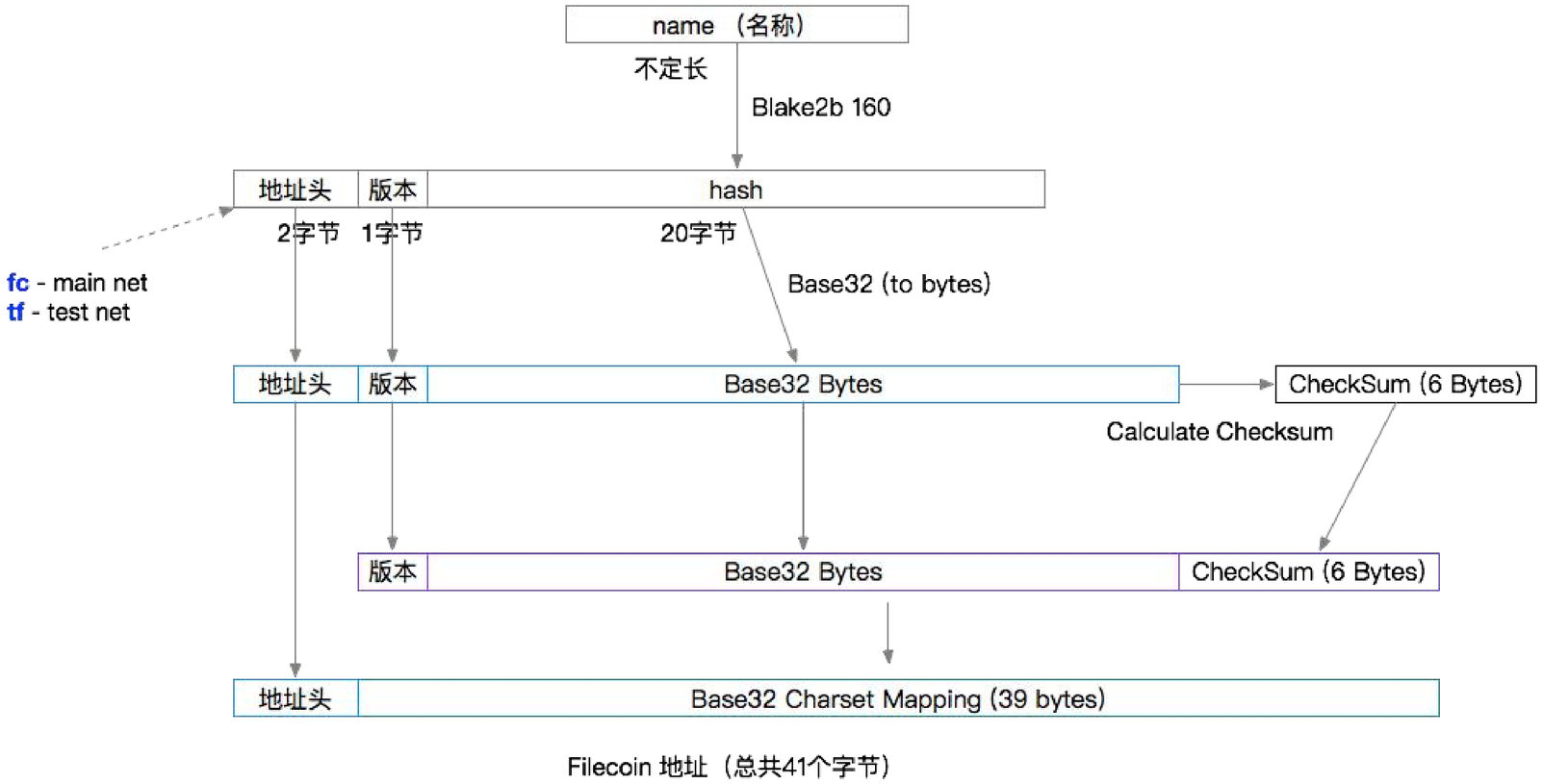
AttoFIL

Filecoin网络使用的代币的最小单位
（1 AttoFIL = 10^(-18) FIL）。

Gas费用

执行Actor中的函数需要消耗Gas费用 (Gas Limit * Gas Price)。不是由指令的消耗决定，而是由函数逻辑决定。

Filecoin BlockChain - 地址生成逻辑



Filecoin 地址（总共41个字节）

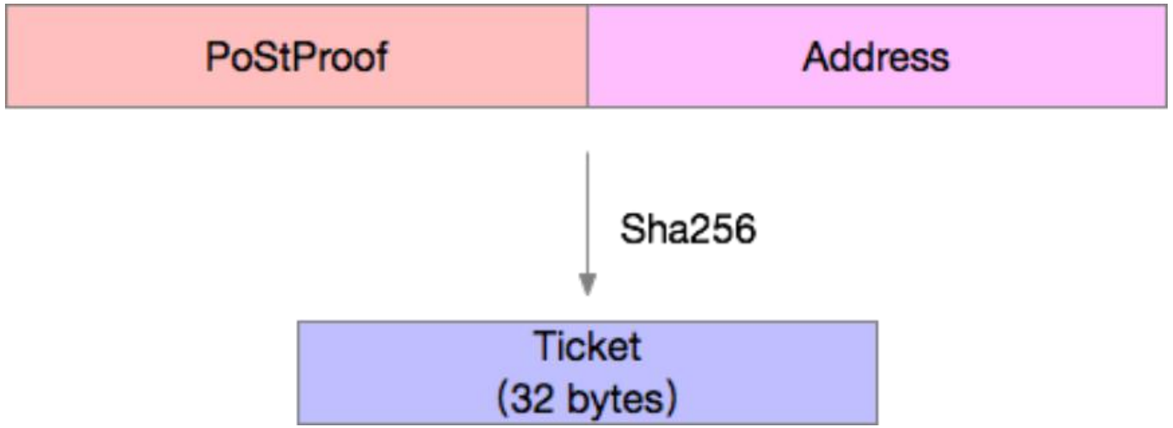
fcqxvnl37zdv8clc26j6r43zn8md7tc2mrfx77vru

filecoin - 铸币地址

storage - 交易市场Actor地址

payments - 支付Actor地址

共识机制（EC）- Ticket



EC - Expected Consensus

每30秒生成一次Ticket

每一轮的Ticket是通过前一轮的区块的Proof以及节点的地址的Hash计算的结果。

目前用固定的30秒作为Ticket生成（区块生成）时间，后期可能通过PoSt的算法执行时间确定。

共识机制（EC）- Leader选举

Ticket: 0.158
0x2872B020C49BA5E353F7CED916872B020C49BA5E353F7CED916872B020C49BA5

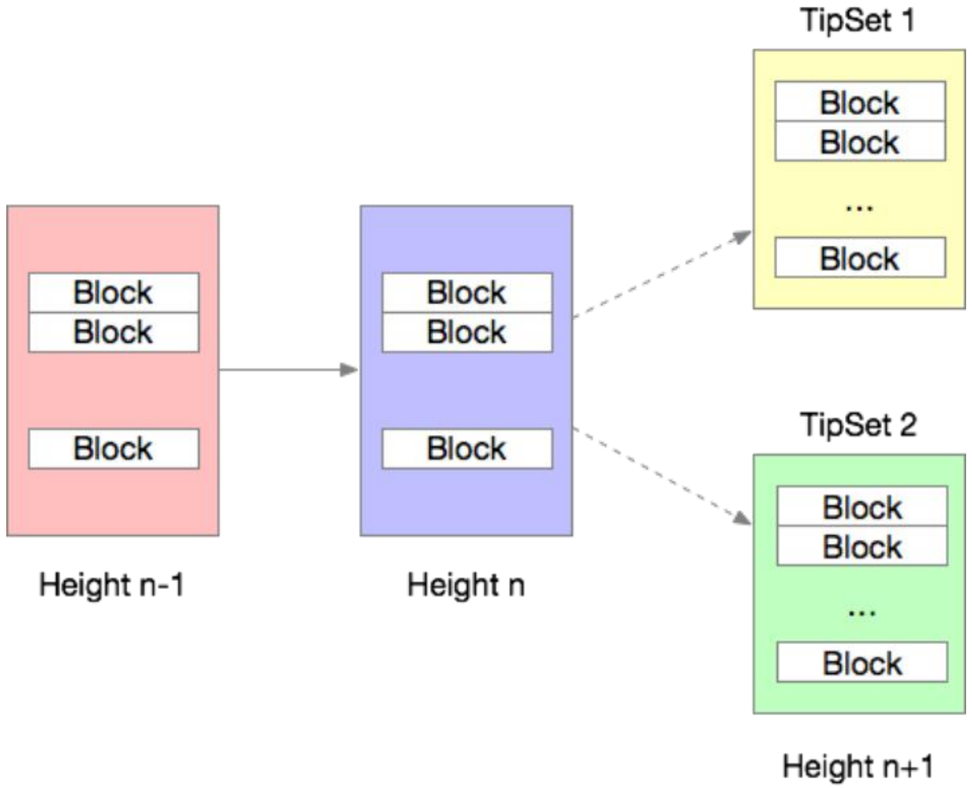
矿工确认的存储大小： 100M
总的确认的存储大小：1000M
矿工确认的存储率：100/1000 = 0.1

0.158 > 0.1

该轮此矿工不是Leader。

在某一轮，Leader的个数有可能是0，1，或者更大。

共识机制（EC）- 确认主链

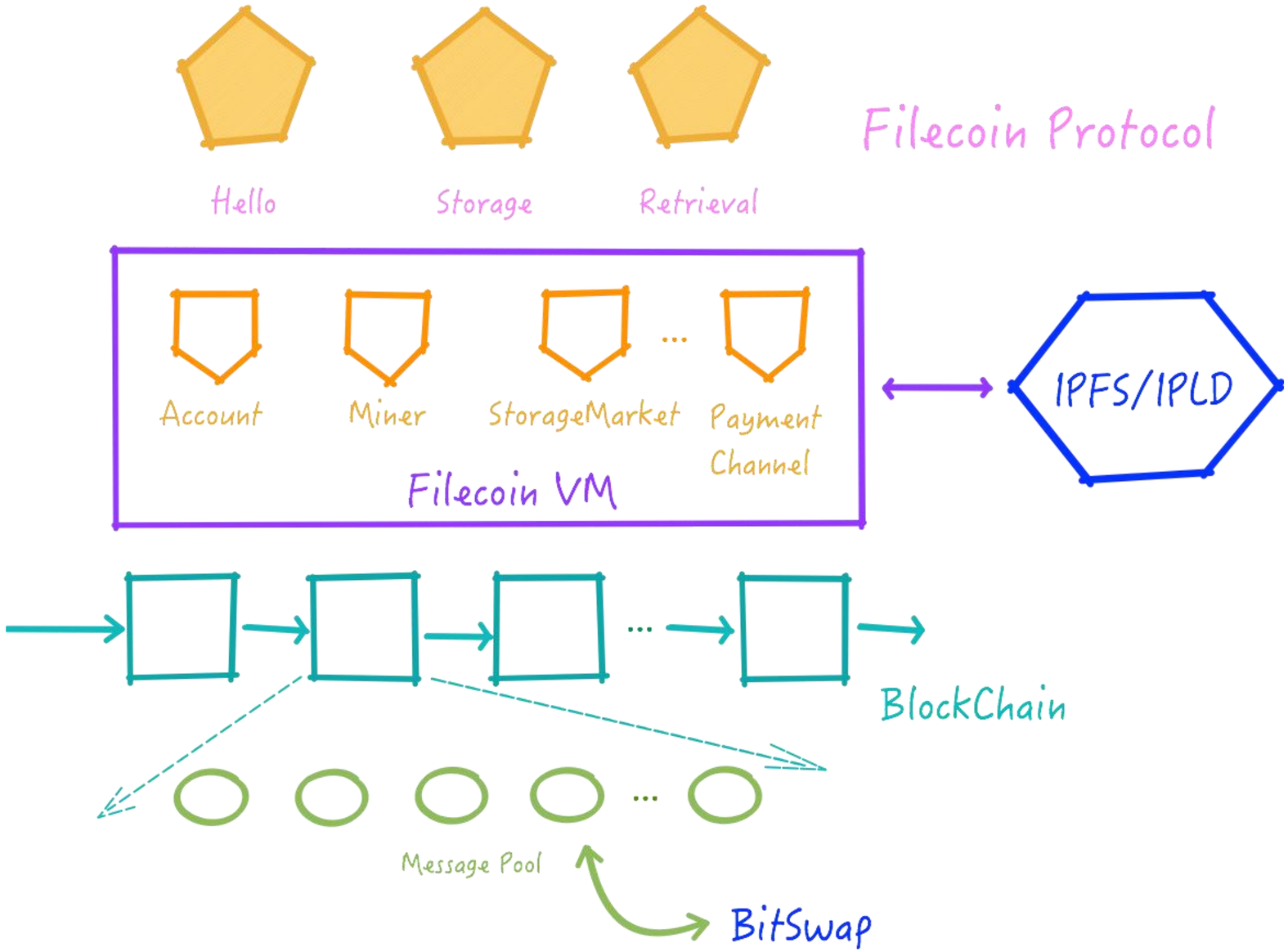


计算TipSet的Weight的计算确定“主链”

Weight = ParentWeight + ECV + ECPrM * ratio

ECV设置为10， ECPrM设置为100，ratio是当前节点的存储率

Filecoin区块链整体框架



预置三个协议：

Hello - 同步区块（TipSet）信息

Storage - 存储撮合协议

Retrieval - 数据获取协议

预置四个Actor：

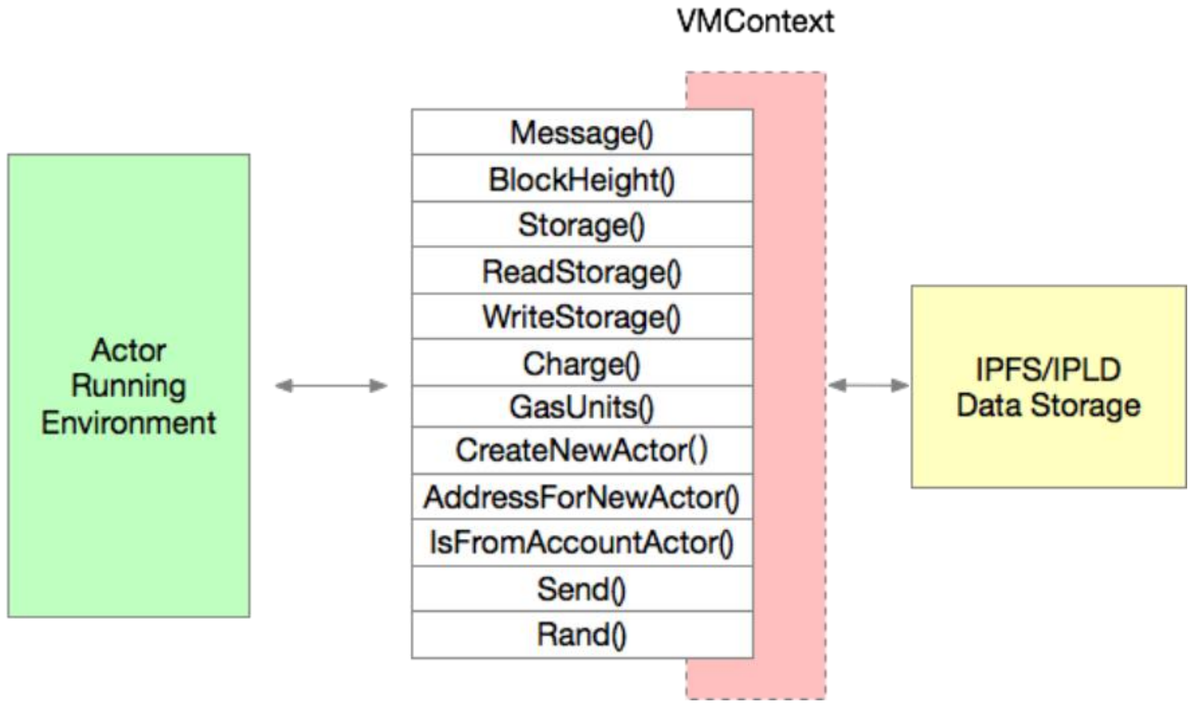
Account - 普通账户信息

Miner - 存储竞价，存储证明等等

StorageMarket - 矿工信息，存储容量等等

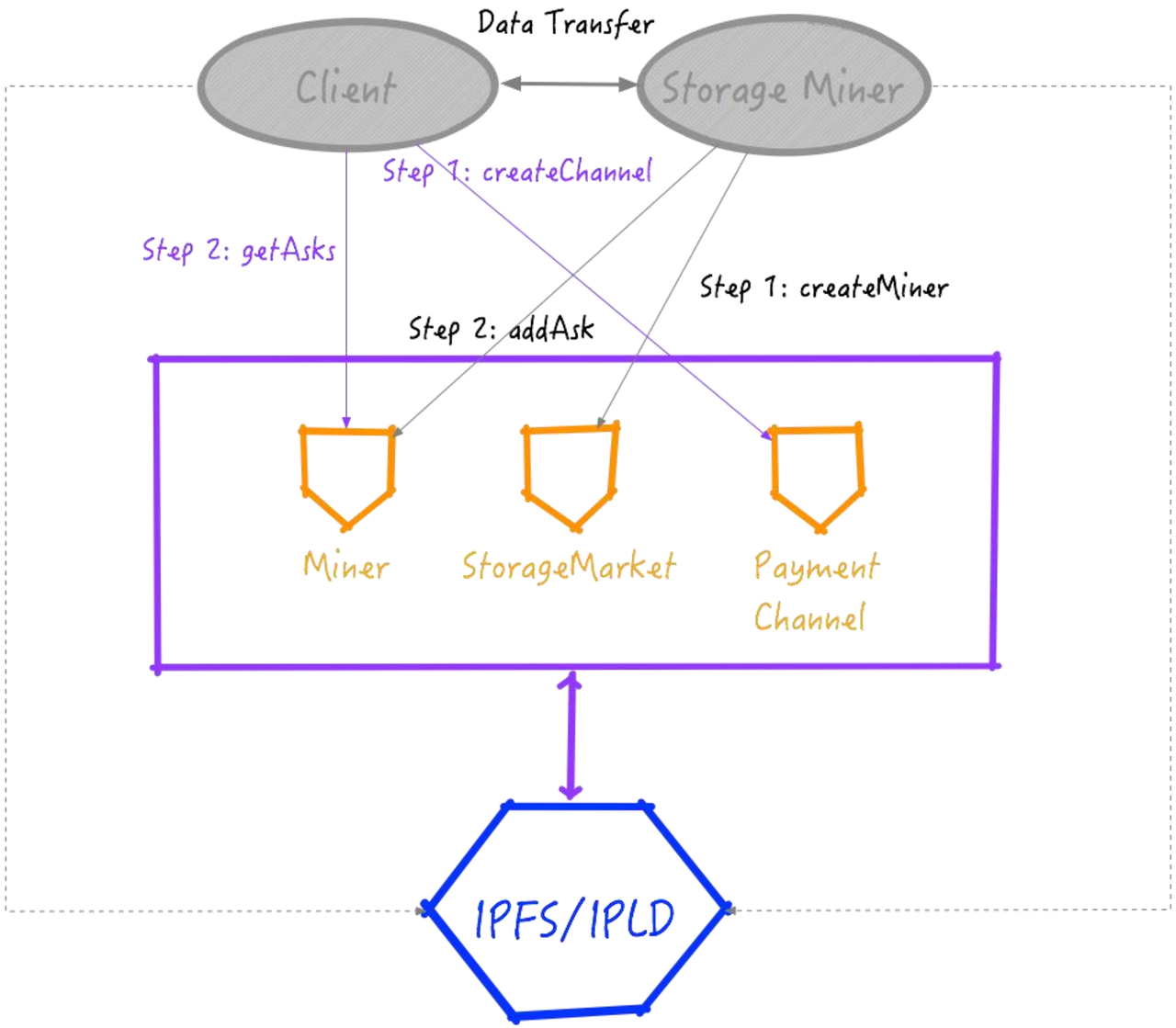
Payment Channel - 支付通道信息

Filecoin虚拟机



Message函数 - 当前交易Message的信息
BlockHeight - 当前区块高度信息
Stoage/ReadStorage/WriteStorage - 存储访问
Charge - 油费耗费的调用
CreateNewActor/AddressForNewActor/
IsFromAcccountActor - Actor地址的创建以及基本查询功能
Rand - 随机数生成
Send - 调用其他Actor函数

Filecoin协议 - 存储撮合协议



createMiner

最少提供10 sector的存储
担保金计算公式：0.001*sector个数

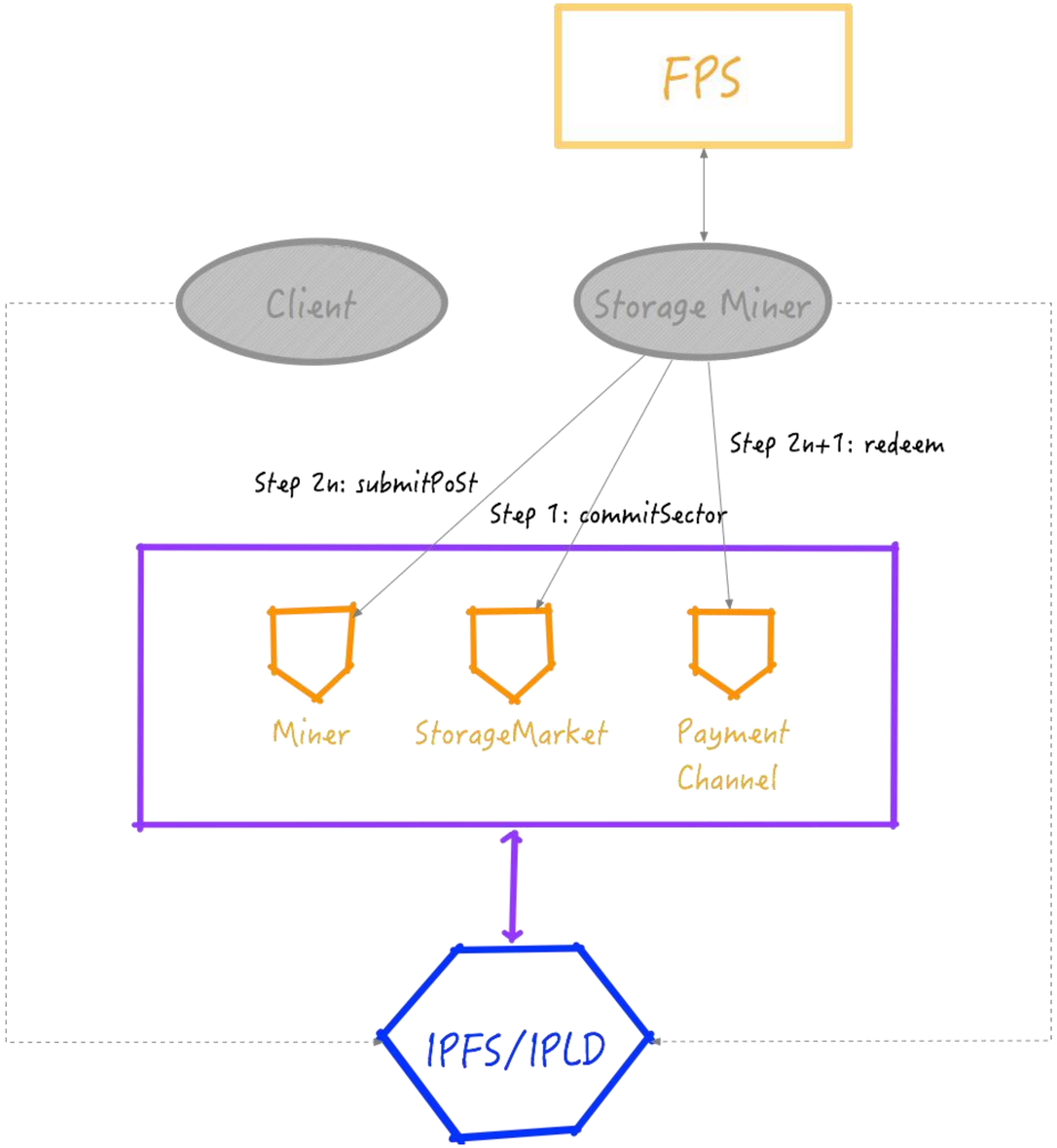
addAsk

存储服务竞价

createChannel

创建支付通道

Filecoin协议 - 存储撮合协议

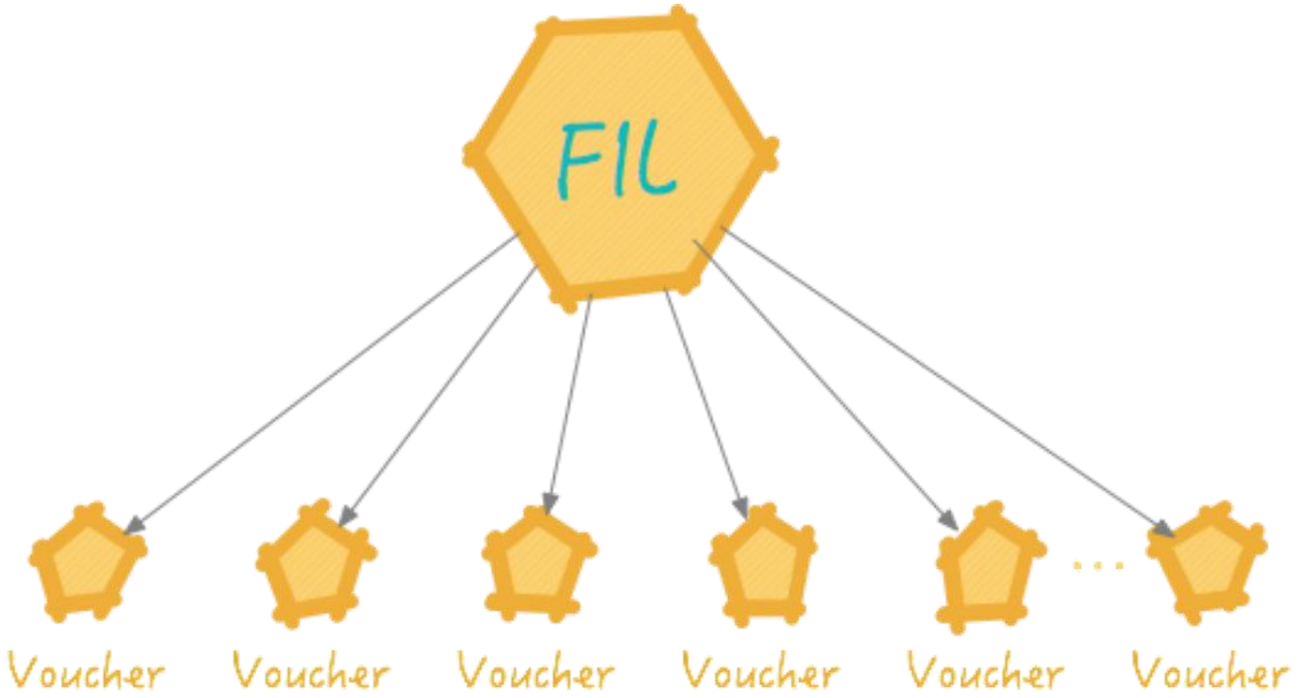


commitSector
Sector的存储证明

submitPoSt
所有Sector的存储证明

redeem
兑现存储费用

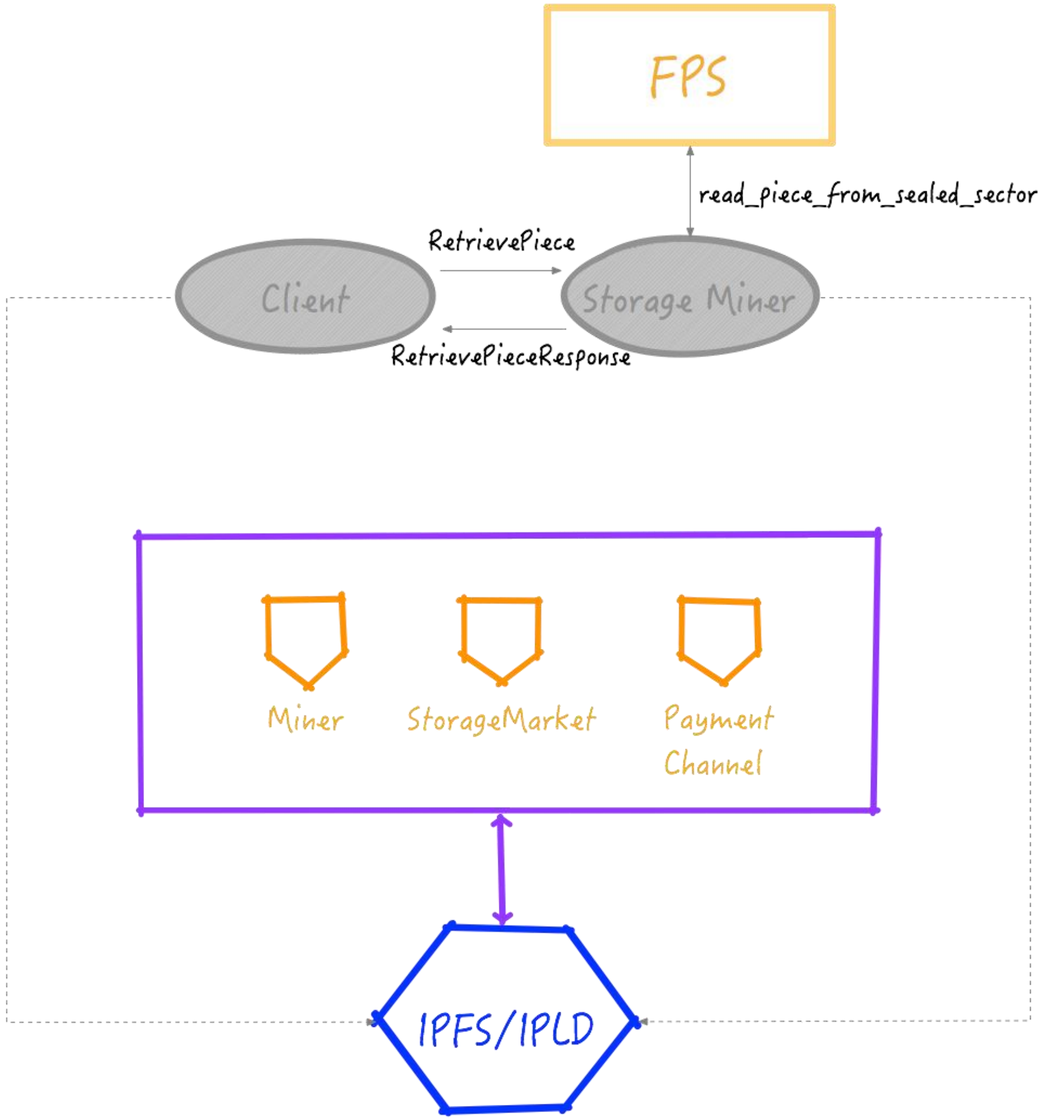
Filecoin协议 - 支付通道



存储费用(AttoFIL)
Size * Duration (block time)

submitPoSt
每20000个区块提交一次。也就是，6天提交一次。

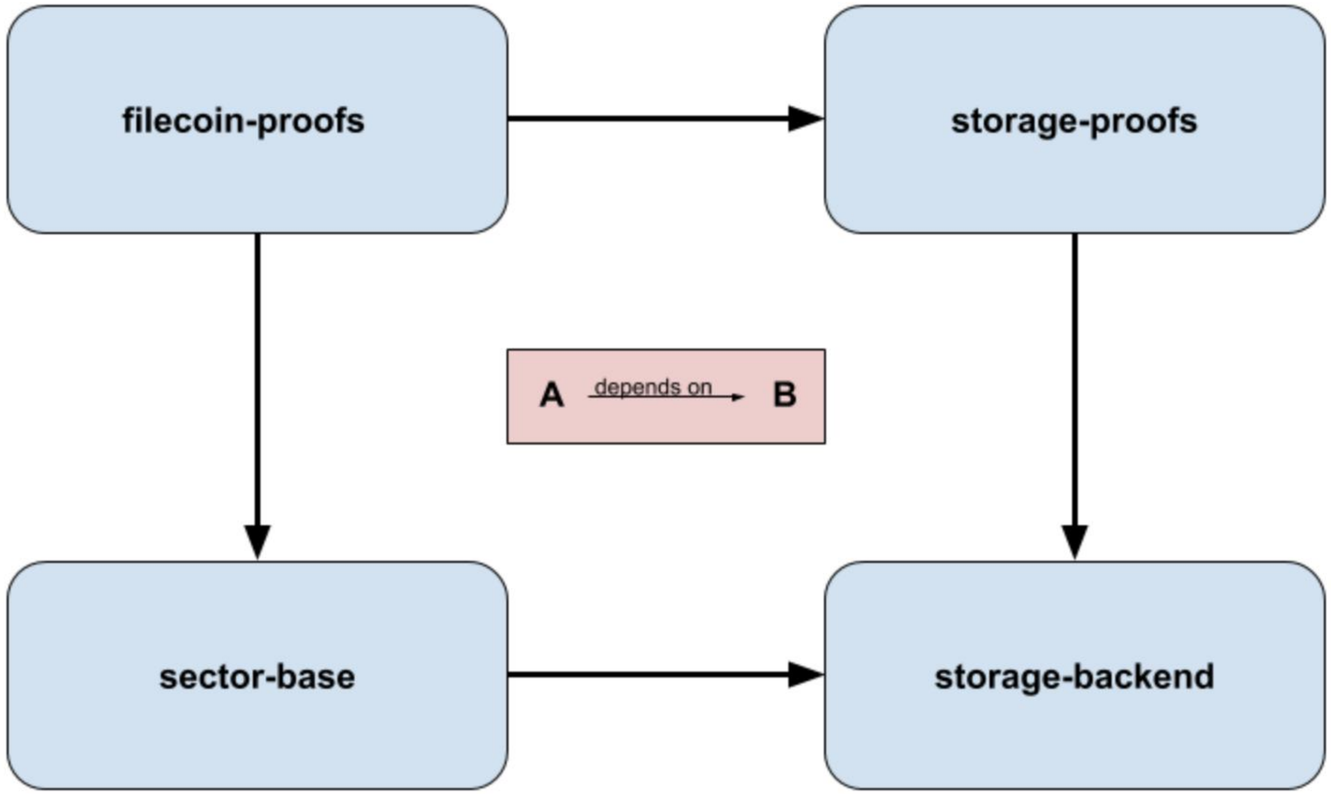
Filecoin协议 - 免费读取协议



Retrieval Protocol

从存储矿工获取存储数据

Filecoin协议 - FPS



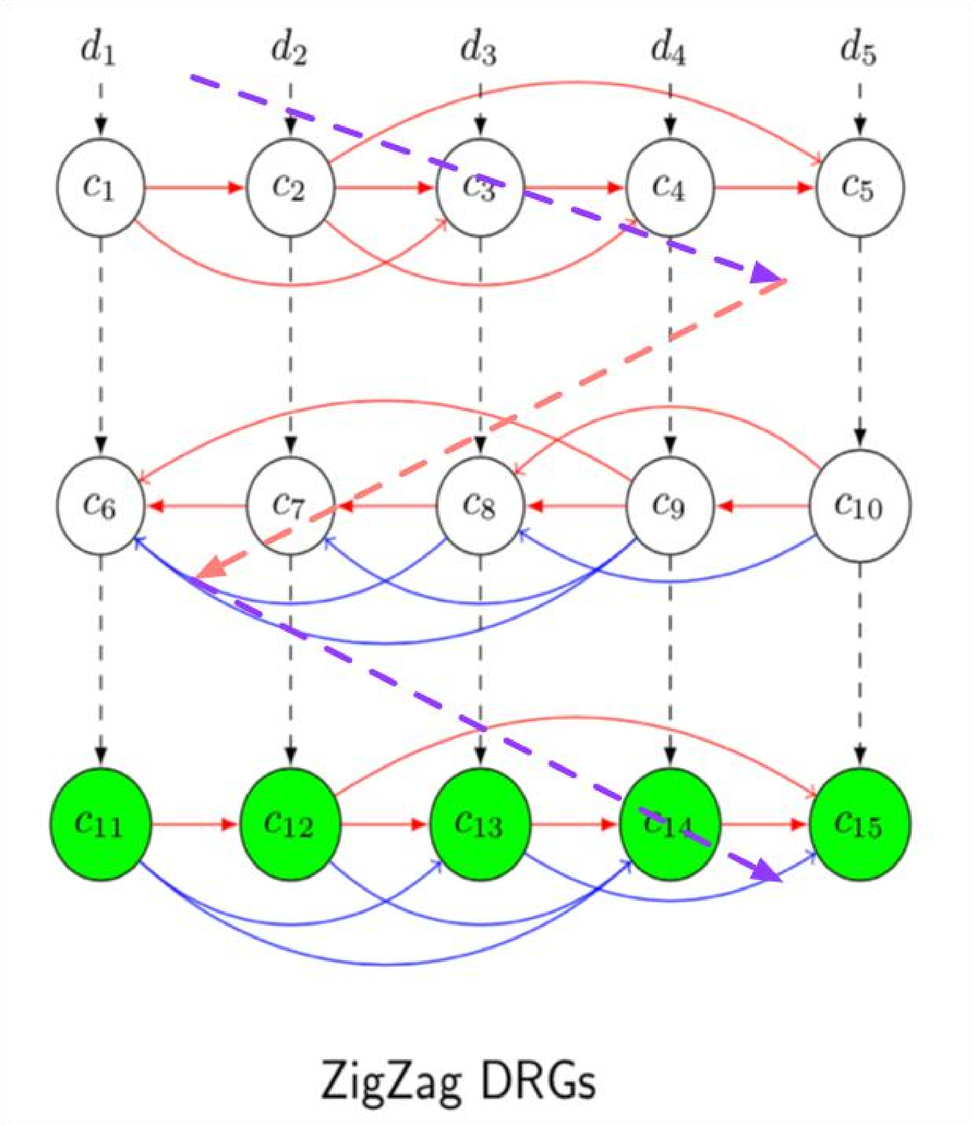
Sector大小

256M
1K

Sector状态

Staging - Sector未写满，也没有超时
Staged - Sector已经写满，或者超时
Sealed - PoRep生成

Filecoin协议 - PoRep



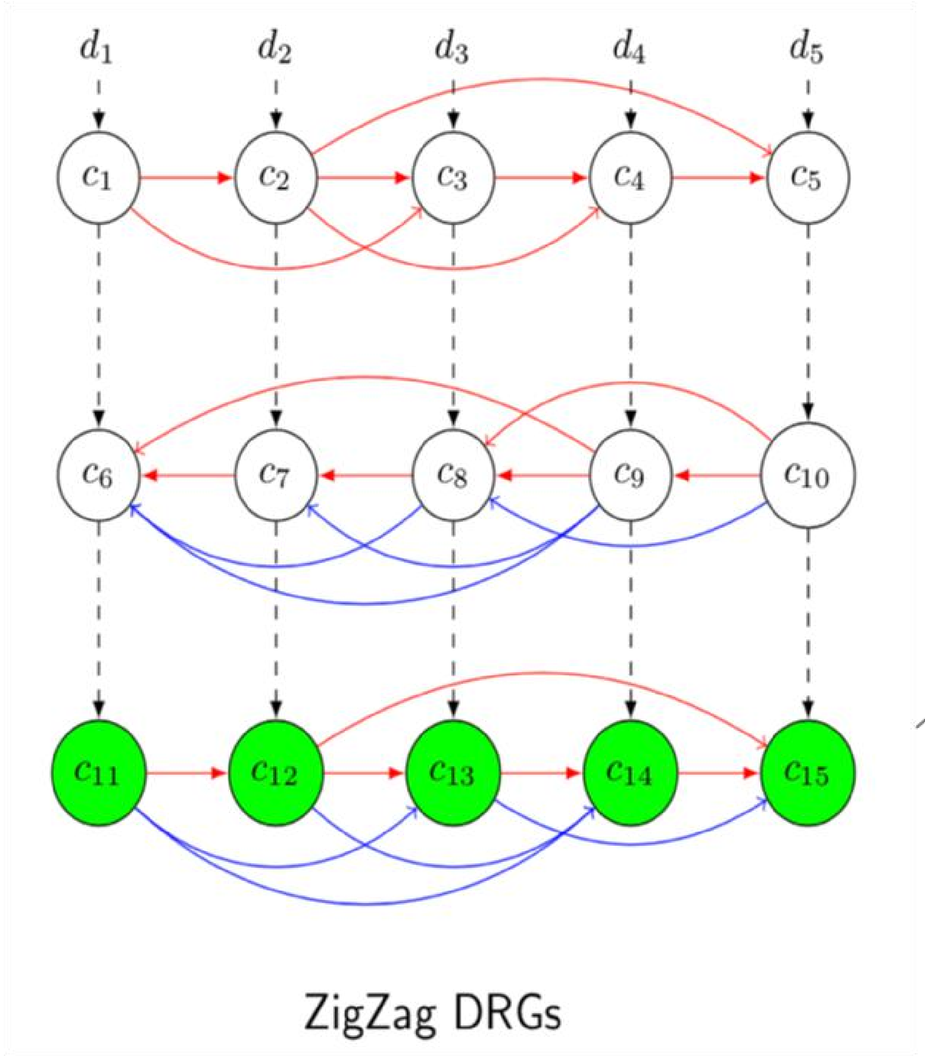
ZigZag DRGs

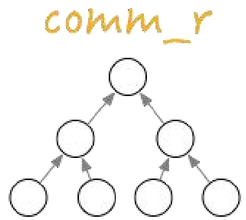
ZigZag-DrgPoRep
Drg - Depth Robust Graphs

Vde (Verifiable Delay Encoder)
Sloth algorithm

zk-SNARK
groth16

Filecoin协议 - PoRep





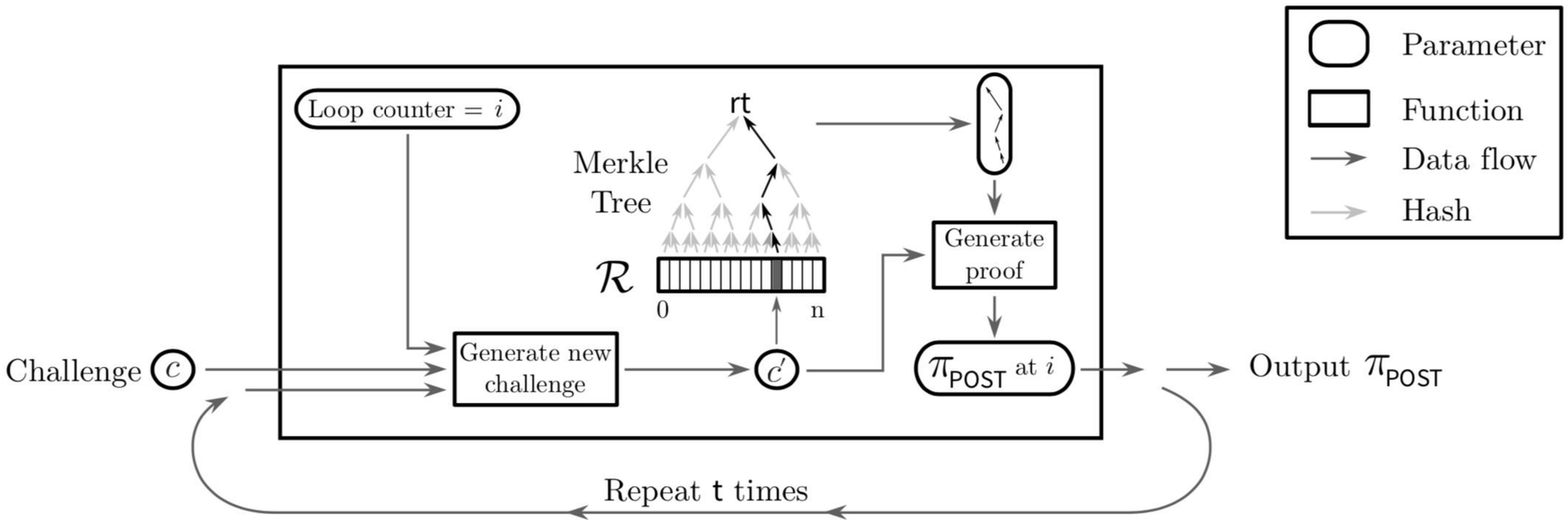
Comm_r
Commitment of Replicate

Comm_r_star
Commitment of all internal Replicates

Comm_d
Commitment of Data

1G数据大约需要50分钟

Filecoin协议 - PoSt



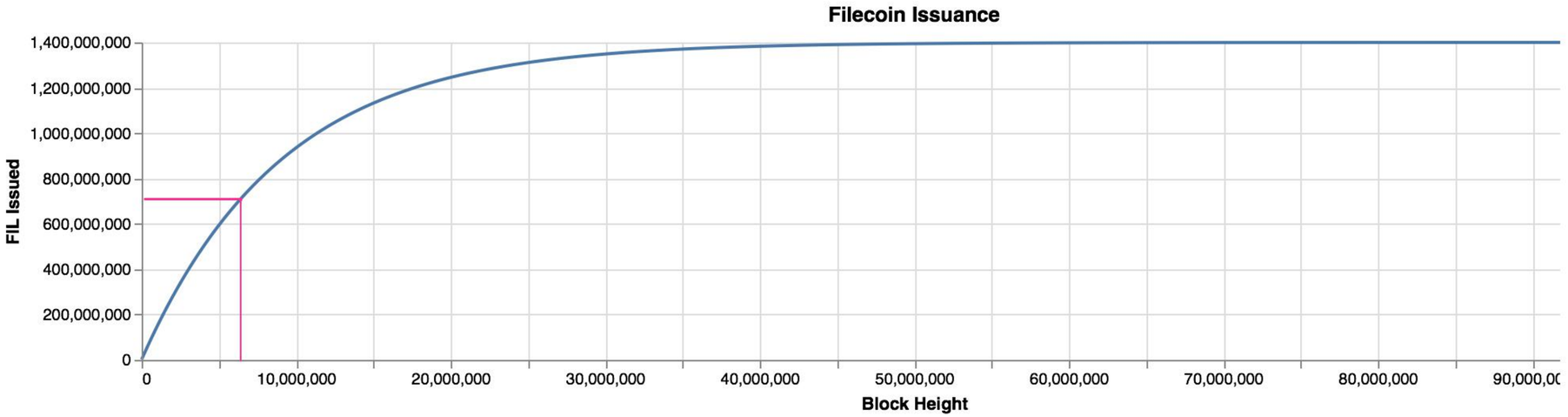
R

All replicates data

Challenge

目前代码是随机生成

Filecoin区块奖励



总的区块奖励代币

1400000000 （14亿）

每6年，奖励减半

6307200

一周内的奖励不变

20160

第一个区块奖励

154.1083052162412

Filecoin奖励的核心因素

奖励 = 区块奖励 + Gas费用

Gas费用相对很少，主要是区块奖励

有效存储率

1）选Leader的可能性大
2）确定为主链的可能性大（Weight高）

Gas费用

调用Actor函数的费用大概为100 AttoFIL。
也就是1个FIL可以实现10^16次调用。

自己存储数据?

目前的代码没有区分是否自己存储数据。

Filecoin代码中的TODO

```
plumbing/msg/waiter.go:// TODO: This implementation will become prohibitively expensive since it
plumbing/msg/waiter.go:         // TODO: this should return an error if a receipt doesn't exist.
plumbing/msg/waiter.go: // TODO: out of bounds receipt index should return an error.
plumbing/msg/waiter.go:// TODO: find a better home for this method
proofs/sectorbuilder/interface.go:      // TODO: Replace this method with something that accepts a piece cid and a
proofs/sectorbuilder/interface.go:      Size uint64  `json:"size"` // TODO: use BytesAmount
proofs/sectorbuilder/testing/interface_test.go:                 // TODO: This should be generates from some standard source of
proofs/sectorbuilder/testing/interface_test.go:                 // TODO: Replace these hard-coded values (in rust-proofs) with an
proofs/sectorbuilder/testing/builder.go:        // TODO: Replace this with proofs.Live plus a sector size (in this case,
proofs/rustverifier.go:         // TODO: change this to the bool statement
protocol/storage/miner.go:// TODO: replace this with a queries to pick reasonable gas price and limits.
protocol/storage/miner.go:      // TODO: Check signature
protocol/storage/miner.go:      // TODO: use some sort of nicer scheduler
protocol/storage/miner.go:              // TODO: handle resumption of deal processing across miner restarts
protocol/storage/miner.go:      // 'Receive' the data, this could also be a truck full of hard drives. (TODO: proper abstraction)
protocol/storage/miner.go:      // TODO: this is not a great way to do this. At least use a session
protocol/storage/miner.go:                      // TODO: signature?
protocol/storage/miner.go:                      // TODO: figure out faults and payments here
protocol/storage/miner.go:      // TODO: real seed generation
protocol/storage/miner.go:              // TODO: proper fault handling
protocol/storage/miner.go:              // TODO: what should happen in this case?
protocol/storage/miner.go:              // TODO: what to do here? not sure this can happen, maybe through reordering?
protocol/storage/miner.go:              // TODO: we are too late, figure out faults and decide if we want to still submit
protocol/storage/miner.go:      // TODO: figure out a more sensible timeout
protocol/storage/miner.go:      // TODO: algorithmically determine appropriate values for these
```

目前Filecoin代码的成熟度不高，很多TODO。

1. Ticket的时间延迟？
2. 存储挑战的随机数生成？
3. PoRep的Layer个数
4. 惩罚机制
5. 油费的设置和计算
6. ...

Q & A

欢迎关注 星想法