StreamChain白皮书

金快快 龚成
jinakuaikuai@sina.com gongchengra@gmail.com

1 概述

过去二十年来，数字技术已经颠覆了全球娱乐和媒体行业。它转变了内容制作和媒体的发行方式。市场预计在未来几年内将继续增长，目前估计价值超过2万亿美元。这些给艺术家带来机遇的同时，也面临着一些急性问题。鉴于数字内容可以在互联网上轻松复制和免费分发，跟踪数字内容版权和特许权使用费仍然十分复杂。

区块链技术可以帮助解决一些问题，通过连接数字内容原创者(艺术家，音乐家，摄影师等)，发行平台与消费者，更有效地组织点播付费和结算服务。它还可以帮助数字权利在整个行业得到更有效的识别和管理，并向正确的艺术家和内容所有者支付适当的赔偿。区别于传统的发行模式，区块链去掉了所有中介平台，例如在市场上发售的音乐和视频等流媒体获得的收入大部分都会归原创者所有。

除了解决版权和付费结算问题，我们还打算利用区块链和P2P技术解决内容文件的存储问题。所有发行的内容文件都会存储在分布式的各个节点中，通过代币驱动各主机节点，实现了完全去中心化，自治的媒体生态。我们的分布式存储费大概只有传统云储存费用的1/10，这能帮助内容原创者降低了作品的存储成本。

2 StreamChain的智能合约

自从2008年中本聪发布了描述比特币协议的白皮书。比特币就已经成为全球公认的最有价值的数字货币。比特币非常成功，这是因为它提供了访问区块链的机制。然而，区块链是一个非常强大的技术，它能够执行复杂的操作，并不仅仅是比特币。这就是智能合约的想法。智能合约已经成为企业区块链应用的基石，可能成为块链技术的支柱之一。下面我们将探讨智能合约是什么，它的工作原理以及它如何被使用。

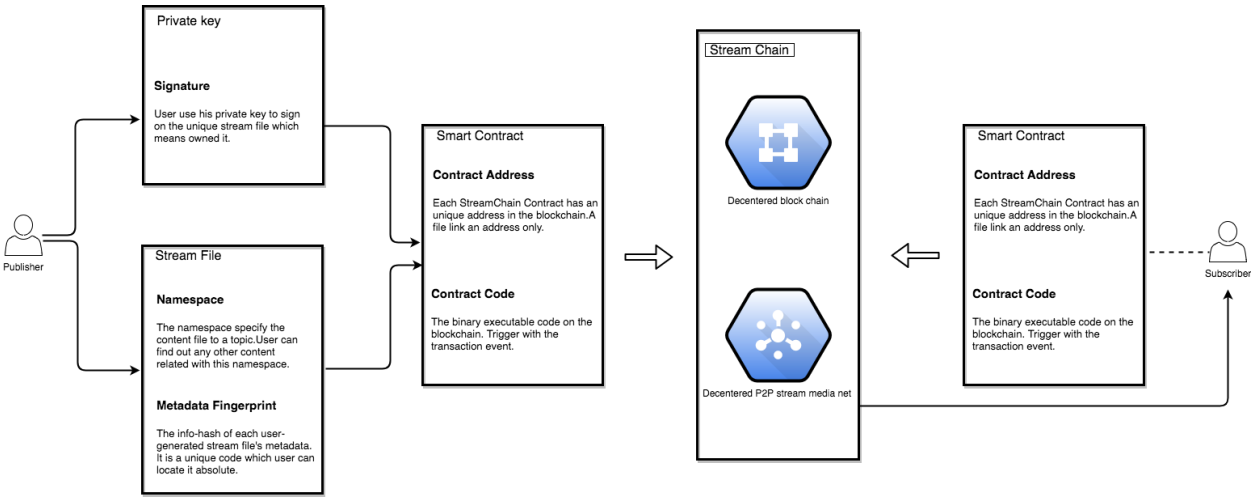
智能合约只是一段计算机程序，合约上的条款可以被预编程，具有自我执行的能力。智能合约的主要目的是让两个匿名方通过互联网进行交易和开展业务，而不需要通过任何中介。
StreamChain将会独立开发自己的区块链系统，为了满足自身的特殊需求，它必须且只能被设计成为一条公有链。和以太坊一样，我们的所有业务都会跑在智能合约之上。StreamChain有三类智能合约：

- 发行合约
- 交易合约
- 存储合约

每种合约满足了不同的业务需求，例如发行合约保护了内容原创者的版权，存储合约维护着租用者和主机存储之间的关系。这些合约ded输入和输出的结果都会被矿工打包进区块，同时每个节点都会验证合约结果的正确性。

3 发行与版权

StreamChain利用了区块链不可篡改的特点来保护内容发行者的版权。用户只需要拥有一个私钥就能通过智能合约来发行自己的作品，而且这一过程是完全匿名的。区块链在本质上，是一个共享、可信的公共总账，任何人都可以对它进行核查，但不存在一个单一的用户可以对它进行控制。简单地说，它是一台制造信任的机器。以下将会讨论StreamChain是如何利用区块链来保护内容发行者的版权，它的原理和实现。
StreamChain发行媒体的流程图:



3.1 数字媒体的指纹与签名

在计算机系统中,常用MD5和SHA是用来计算和校验文件报文摘要的工具程序。将一个文件的内容经过计算与校验和分布到一个128位的散列值上。该散列值可以作为该文件的数字指纹使用。两个不同的文件不可能得到同样的散列结果。因此，一旦文件被改动，就可检测出来。 例如在linux中，产生某个md5sum的命令如下：

```
md5sum filename > finame.md5
```

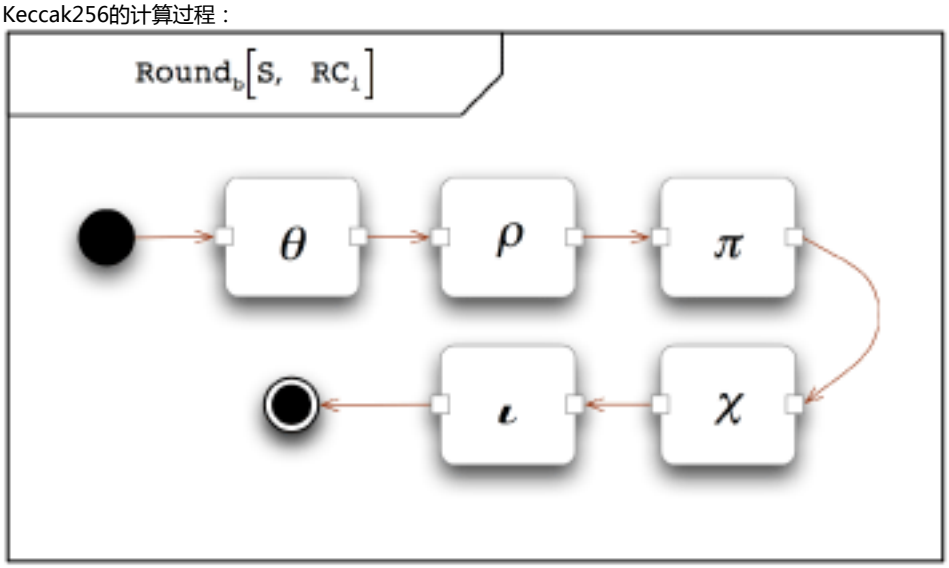
在其他P2P对等网络系统中，常用文件头部的一些特征信息作为散列的内容，例如作者，歌名，年份等。在开源的情况下，这种方案也许是不错的选择。但这种校验结果往往局限在某个对等网络中，在跨系统中存在互相不兼容的现象。

Keccak256加密算法

StreamChain和以太坊一样，采用了更加安全高效的Keccak256散列算法来生成数字内容的指纹。以下是以太坊源码中采用Keccak256来加密字符串。

```
hash := crypto.Keccak256Hash([]byte("helloworld"))
```

经过多年的测试和分析，美国政府选择了Keccak算法作为SHA3的加密标准。Keccak拥有良好的加密性能以及抗解密能力。它采用了创新的的“海绵引擎”散列消息文本。它是快速的，在英特尔酷睿2处理器下的平均速度为12.5周期每字节。它设计简单，方便硬件实现。 Keccak使用了24个变换循环来缩减消息文本为散列值。每个循环连续调用了五个模块。
Keccak256的计算过程：



θ: $a[i][j][k]a[i][j][k]parity(a[0...4][j1][k])parity(a[0...4][j+1][k1])$

ρ: $a[i][j][k]a[i][j][k(t+1)(t+2)/2], where\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}^t \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}^t \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

π: $a[i][j]a[j][3i+j]$

χ: $a[i][j][k]a[i][j][k](\check{n}a[i][j+1][k]\&a[i][j+2][k])$

ι: 把数组中的一个元素与一循环常量异或。

Keccak256加密的hash结果将会作为某个文件在streamChain网络上的唯一标识ID。某个用户节点想要获取某个文件时，必须要带上该该文件的指纹才能找到该文件。

3.2 签名与验证

如何证明A用户拥有B文件内容的版权呢？streamChain将会围绕该问题做一些非常复杂而有效的工作。大多数加密货币都是利用ECDSA椭圆曲线非对称加密算法来实现签名的。例如Alice拥有一把公钥和一把私钥。她可以利用私钥来完成对某一内容的签名，而另外一个人则可以利用该公钥来验证签名。比特币账户的地址是公钥的两次SHA-256加密，在经过Base58编码而成。

```
Checksum = SHA-256(SHA-256(Key hash))
Bitcoin Address = Base58Encode(Checksum)
```

不同于比特币，streamChain拥有两类账户: 外部用户的账户，通过私钥控制；智能合约账户，通过合约代码控制。
区块链节点验证交易签名需要发送方的公钥，公钥并没有标准的发布方法，通常会把公钥增加到交易里。由于比特币和以太坊所用的ECDSA算法有一个很有意思的特性：可以从签名推导出公钥，streamChain的交易格式只包含了签名，验证交易签名前先用算法推导出公钥，然后再进行验证签名。

现在问题很简单了，那么我们只需要一个用户A对文件B的签名就可以证明它拥有B文件的版权。StreamChain把该过程通过智能合约简化了，但验签原理是一样的。我们把用户A和文件B的关系都经过智能合约发布在了区块链上了，而且完全是去中心化的。

3.3 区块链上的版权存储

可以把作者和作品的关系存储在分布式的区块链上，而且只有被验证通过的作者签名的地址才有资格确定和作品的指纹绑定关系。streamChain的每个节点都会把这些存储的过程验证一遍，因此该关系的合法性自然而然的得到了大多数节点的认可。况且通过工作量证明来防止一些意外发生，例如双花这种情形。以上证明有效性的前提是发行作品过程的合约和源码必须是公开的。静态地限制存储的读写权限，并且保证作品指纹的唯一存储性。
简单的存储结构:

```
struct Stream {
        bytes32 name; //数字作品名称
        uint version;  //发行版本号
        bytes32 date; //发行时间
        bytes32 author_name; //拥有着的名称
        address owner; // 数字作品拥有者
    }
mapping(info_hash => Stream) public Streams;// 数字作品的hash-->Stream
```

4 交易合约和交易自动化

StreamChain不仅仅要解决版权的问题，而且要实现交易流程流程自动化。智能合约可以轻松地为我们解决这一问题。StreamChain中有两类账户，它们共用同一个地址空间。外部账户，该类账户被公钥-私钥对控制（人类）。合约账户，该类账户被存储在账户中的代码控制。StreamChain将会在以太坊上创建一个付费的智能合约，并且公开该合约的地址。只要有观众向这个地址发送交易的请求，合约代码就能自动地把费用分发到某个作品的所有利益方，例如作者，平台，广告商等。StreamChain的智能合约拥有一些付费相关高级特性，例如交易，转让，定价，分佣，退款，分期，提现等。而且这些都是透明的，任何人都有权查看某个作品的所有历史交易记录。

版权认可和支付透明这两个条件，可以让StreamChain网络上的所有节点对流媒体的内容分发做到定量可控。P2P流媒体网络本身也是去中心化的，分布式存储的，因此可以天然地跟区块链网络结合。这将会让我们进一步的做到整个流程的可控。

5 去中心化文件存储

类似于Sia和storj，streamChain也会把音乐，视频等流媒体文件分散地存储在各个主机节点上。不同于Sia,我们的文件都是公开发行的，任何人只需要通过发送代币即可下载和浏览。

只把数据存储在一台不受信任的主机上，在可用性，带宽和服务质量上得不到很好的保证。因此，我们建议把数据存储在多个主机上。而且多主机并行传输能够加速下载速度。利用纠删码(erasure code)或者随机网络编码等技术可以保证数据的高可用性和传输的无冗余性。

我们利用DHT算法最终确定数据的存储位置。DHT全称叫分布式哈希表(Distributed Hash Table)，是一种分布式存储方法,可由键值来唯一标示的信息按照某种协议被分散地存储在多个节点上，这样也可以有效地避免中心服务器的单一故障而带来的整个网络瘫痪。常用的实现方案有Kademlia，CHORD，Pastry等。其中Kademlia(简称Kad)是应用最广泛的一种。这里我们只讨论Kademlia的一些技术原理和细节。

5.1 Kademlia网络

Kad中所有节点都被当做一颗二叉树的叶子，每一个节点的位置都由其ID值的最短前缀唯一的确定。对于每个节点来说按照离自己的远近区域又可以把这棵树划分为160棵子树，每一个子树和该节点都有一个共同的前缀，共同前缀越少离得越远。如下图示黑色节点拥有3棵子树(灰色圈表示)。Kademlia使用独特的异或距离算法来计算节点间的距离，异或是一种简单的数学计算。对于异或(xor)操作，拥有如下数学性质:

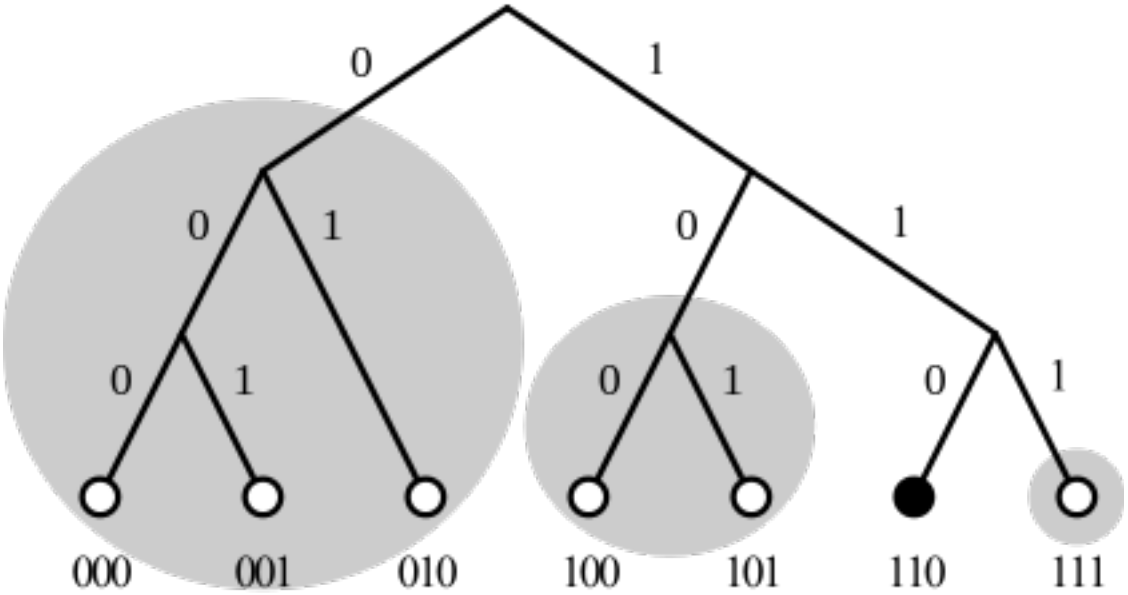
$$d(x,x) = 0$$

$$d(x,y) > 0$$

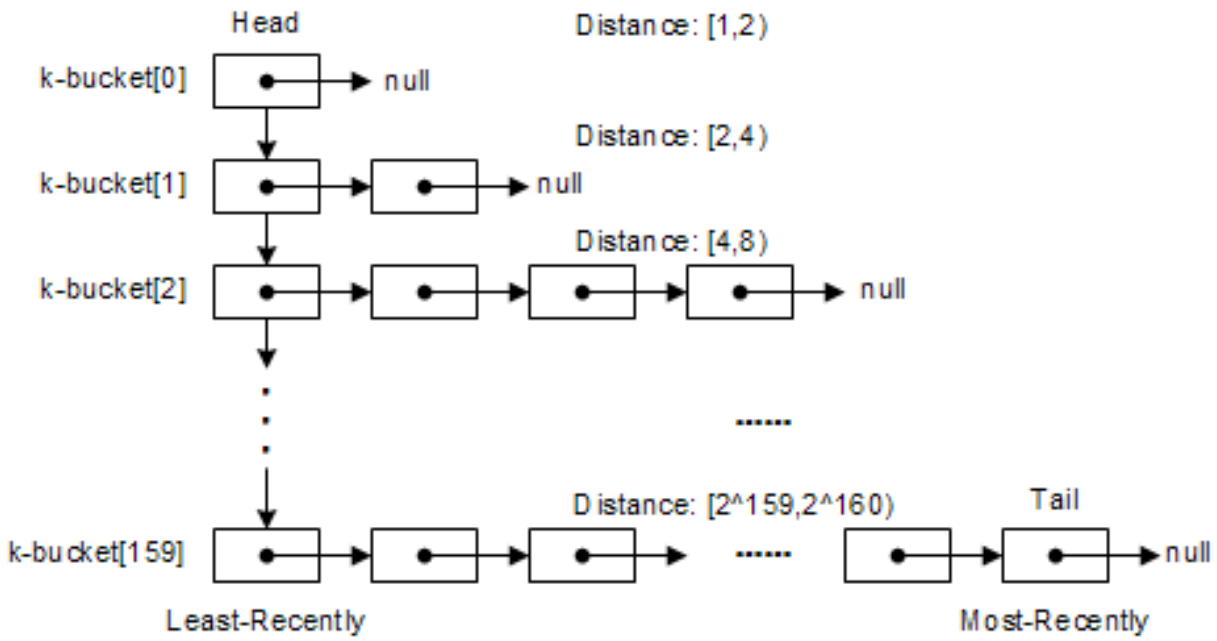
$$d(x,y) = d(y,x)$$

$$d(x,y) + d(y,z) \geq d(x,z)$$

一个典型的Kad网络拓扑结构如下:



和区块链不同，点对点的流媒体网络存储的资源信息比较大，因此在大规模应用的情况下，每个节点都存储所有的资源和节点信息是不可能的。如果需要找到目标节点不是领节点，就需要通过间接的方法找到该节点。Kad的路由表是通过“K桶”的表格构造起来的。 K桶路由表结构图示:



由此可见，每个节点都保存一些和自己距离 $[2^i, 2^{i+1}]$ 内的一些节点的信息，这些信息由(IP address, UDP port, Node ID)构成。K-桶中的条目是按与其他临近节点的状态信息排序的，每当收到一个消息时，就要更新一次K桶。经过证明，对一项有N个节点构成的Kad路由网络，最多只需要经过 $logN$ 次查询，就可以精确定位到目标节点。

Kademlia协议仅定义了四种操作：

- PING: 探测一个节点是否在线
- STORE: 令对方储存一份数据
- FIND NODE: 根据节点ID查找一个节点
- FIND VALUE: 根据键查找一个值(数据)

5.2 默克尔树(Merkle Tree)

StreamChain的音乐，视频等流媒体文件将会分割成大量的很小的数据片段，这些片段会根据“K桶”存储在Kademlia网络的节点上。默克尔树有效的组织所有片段，并且能证明某数据片段是否来自某个原始文件。
Merkle Tree具有以下特性：
+ 它是一棵二叉树，也可以是多叉树，但具有树的所有特点。 + 叶子节点是所有数据的hash值。 + 非叶子节点是所有子节点的组合的hash值。

我们把所有数据片段的hash值存储在Merkle Tree的所有叶子节点上。利用它可以用来组织数据片段，文件完整性证明，以及存储证明等。下节将会介绍我们如何利用它来实现存储证明。

6 存储证明(Proof of Storage)

内容发行者需要消耗代币来让streamChain上各个主机节点存储自己的数据。消耗代币的多少取决于文件的大小和存储时间。但存储的主机只能分期的获得租用者的佣金，以便保障租用者的权益。这些规则都会定义在streamChain的存储智能合约上。每个主机都需要定期的发送自己的存储证明。

6.1 算法

主机只需要提供数据的一个片段(segment)。各个节点只需要比较该数据片段的hash值和默克尔树比较来验证。这些信息足以证明该数据片段来自原始文件。选择的数据片段编号是随机的，而且必须在一个“挑战窗口”内完成。

随机种子的定义如下:

$$H(contractID||H(B_{i-1}))$$

数据的随机片段编号根据该随机种子生成。所有的“挑战窗口”将会充满整个存储的时间(合约上定义的时间)，只有完成了大部分调整的主机才能拿到所有的报酬。如果存储主机不能完成50%的挑战，该主机也会惩罚部分的报酬。通过代币激励这种模式，让P2P网络上的存储节点更好的为所有用户服务，包括内容发行者和消费者。

7 数字媒体的P2P传播与校验

通过以上对Kad网络的讨论可知，所有内容的存储也是分布式的。不仅原创者发行内容需要代币，内容消费者也需要消耗代币来观看内容。 我们如何控制这些节点有效的传播到点播节点呢? 例如 媒体A要分发到用户B。

1. B已经产生了对A的交易。
2. B的行为得到了大多数节点(存储A的节点)的认可。
3. B节点校验A内容的有效性。

因此只需要满足以上三个条件，就可以认为某个媒体内容在StreamChain上的传播是可接受的。StreamChain在区块链上交易的透明性可以满足条件1。而条件2和条件3都会加在StreamChain的传播协议里面。这些步骤的唯一的目标是让节点选择拒绝还是接受这两种行为。这其中所有的验证都运行在StreamChain的三类智能合约上。

一旦校验完成，被交易的流媒体数据会立即地发送给内容订阅者，由于这一传输过程是并行的，因此下载速度会很快。利用纠删码(erasure code)或者随机网络编码等技术可以保证数据的高可用性和传输的无冗余性。StreamChain采用随机网络编码来完成数据的高可用传输。

7. 1 随机网络编码

与传统的数据传输模式不同，为了避免多条线路的数据重复传播，并且保持对不同线路数据组装的一致性，StreamChain采用随机网络编码分发数据技术。这种方式的优点在于不需要事先人为设定数据传输的内容，只需要少量冗余的代价即可实现数据灵活的分割与组合。

随机网络编码算法:
编码过程

$$x = \sum_{i=1}^{m} c_i^p \cdot b_i^j$$

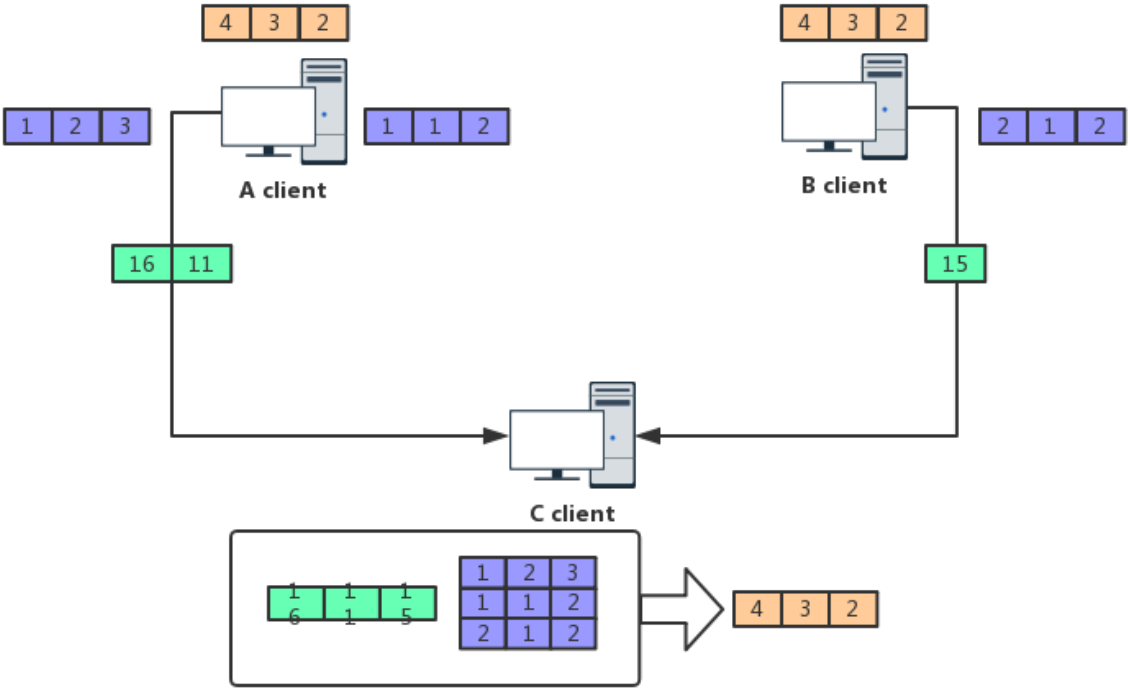
解码过程

$$b = A^{-1}x^T$$

c_i^p: 上游某节点发送给节点p的第i个区块对应的随机系数。
b_i^j: 上游某节点待发送到节点p的第i个区块数据。
A: 节点P收到的上游所有节点发送的随机系数c_i^p组成的矩阵。

例如假设当前时刻欲传输的数据为4,3,2（图示A client和 B client上方的数据）。A和B向C传输时，不会直接传输原始数据。随机生成随机系数([1,2,3] [1,1,2] [2,1,2])，然后将原始数据和随机系数依次相乘并相加得到要传输的数据A。最终C得到了[16,11,15]和三个随机系数。然后C根据求解多元线性方程组的方法反算出原数据。这样传输方式不需要用户节点的各个上游节点事先约定，只要用户接收到足够多的数据即可计算出原始数据。虽然增加了系数的传递，但考虑整个系统需要这样的灵活方便性还是值得的。

随机网络编码图示:



8 Sybil攻击

如果某个恶意的实体模仿很多个身份，他就可以控制整个P2P网络的流动方向。在网络上，伪装成很多个地址是很容易的，例如他可以伪装成多个主机节点来获得更多的文件存储。我们可以通过某种信誉评级机制来抵御Sybil攻击。streamChain需要每个存储主机提供销毁证明，把收益的2%发送给一个公认的不可再使用的地址。而且销毁的越多，信誉评级就越高。每个文件streamChain上至少有3个备份，因此攻击者需要付出更高的代价来控制整个P2P网络的流向。随着streamChain节点数量的增长，要收集那么多代币发动Sybil攻击几乎是不可能完成的任务。攻击难度足以媲美比特币的51%攻击。

9 关闭窗口攻击

假设某个恶意的矿工把一个块的存储证明删除，并以此敲诈存储主机。这就是关闭窗口攻击，认为的把某个挑战窗口给抹掉了，从而让存储主机拿不到报酬。streamChain建议存储主机选用比较大的挑战窗口，让一个挑战窗口的存储证明分散在各个区块中，每个矿工只能拥有部分的存储证明，这样存储主机就能拿到大部分的报酬。存储主机也可以拒绝选择不适合的挑战窗口合约。

10 代币

StreamChain的代币也会遵循ERC20标准,代币合约将提供提供以下几个接口:

- 获得代币总供应量
- 获得账户余额
- 转让代币
- 批准花费代币

这些代币将会用于音乐，视频等媒体的发行费用，存储费用，同时消费者也可以利用代币来下载查阅媒体内容。代币是激励StreamChain自主运行的关键原材料。整个流媒体生态也可以通过这些代币获得更好的良性循环。代币的价格也暗示着整个流媒体生态环境的好坏。

11 网络自适应模型与算法

假设节点个数为n。$p_i(i=1..m)$ 表示第i个固定节点。用 P 表示节点的集合，则

$$\{P=\{p_i\},i=1..n$$

r 表示支撑固定节点所需数据的正常运行的最小流率。用 u_i 表示第i个固定节点的上传带宽。u_s表示源节点负载大小,u_{smin}系统通用流。 对于P

$$u_{smin}\geq r$$

且所有的最小流率之和不能大于所有的上传带宽之和。

$$u_{smin}+\sum_P u_i\geq nr$$

固定节点和移动节点的平均上传带宽

$$\begin{cases}\bar{u}=\dfrac{\sum_{i=1}^{n}u_i}{n}\end{cases}$$

结合以上两公式:

$$u_{smin}+n_f(\bar{u}-r)\geq 0$$

整理得:

$$\begin{cases}u_{smin}\geq r\\ u_{smin}\geq n_f(r-\bar{u})\end{cases}$$

令 $MAX=max[r,n(r-\bar{u})]$
所以

$$u_{smin}\geq MAX$$

令 $u_s=MAX$

用$s_{oi}(i=1,..,n_f)$表示源节点发送给 p_i 的多媒体数据的流率大小，用 s_{ki} 表示 p_k 。用 $tr_i(i=1,..,m)$ 表示 p_i 所接受多媒体数据的总流率。当 $r\geq n(r-\bar{u})$,即:
$r\leq\frac{n\bar{u}}{n-1}$时，s为 p_i发送的数据流率为:
$s_{oi}=\frac{u_i r}{n\bar{u}}$，所以:

$$\sum_P=r\leq u_s$$

因此s有足够的上传带宽实现以上发送。 又由于

$$(n-1)s_{oi}=\frac{(n-1)u_i}{n\bar{u}}r\leq\frac{(n-1)u_i}{n\bar{u}}\cdot\frac{n\bar{u}}{n-1}=u_i$$

因此，p_i^f 有足够的上传带宽把数据发送给其他固定节点。 每个固定节点接受的总流率为

$$tr_i=s_{oi}+\sum_P^{j\neq i}s_{oj}=r$$

所以该条件能满足所有固定节点获取足够的流率，满足媒体正常播放。 当$r<n(r-\bar{u})$时，s为p_i 发送两次流媒体数据， 第一次流率:

$$s_{oi}^1=\frac{u_i}{n-1}$$

第二次为

$$s_{oi}^2=r-\frac{u(P)}{n-1}=r-\frac{n\bar{u}}{n-1}$$

经过计算推导

$$\sum_{P} s_{oi}^{1} + \sum_{P} s_{oi}^{2} = n_f(r - \bar{u}) \leq u_s$$

所以s有足够的上传带宽实现以上发送。又由于

$$(n-1)s_{oi}^{1} = (n-1) \cdot \frac{u_i}{n-1} = u_i$$

p_i有足够的上传打款将所得的 s_{oi}^{1} 包含数据一同样的流率转发给其他节点。总流率为

$$tr_i = s_{oi}^{1} + s_{oi}^{2} + \sum_{P}^{j \neq i} s_{oi}^{1} = r$$

因此 $p_i(i = 1...m)$ 获得足够的流率来保证媒体的正常播放。 综上两种情况，都按照算法设计实现了所有节点的正常播放。

12 StreamChain生态

StreamChain既是音视频等流媒体的发行平台，也是去中心化的公共数据存储平台。所有的自动化交易都运行在区块链的智能合约上。StreamChain将会颠覆整个音乐，短视频等流媒体行业，这只是一个必然的趋势。StreamChain未来将会和一些现有的媒体平台合作，例如QQ音乐，酷狗音乐，秒拍，快手等。可以说StreamChain未来将会成为整个流媒体行业的上游应用。我们也会构建自己的去中心化的媒体客户端，和Steemit一样加入去中心化的点评排名机制，促进原创内容的增长。

参考文献

1. Satoshi Nakamoto. 比特币白皮书[EB/OL]. https://bitcoin.org/bitcoin.pdf，(2008).
2. Wikipedia. SHA-3 wiki[EB/OL]. https://en.wikipedia.org/wiki/SHA-3
3. Ethereum Wiki. 以太坊白皮书[EB/OL]. https://github.com/ethereum/wiki/wiki/White-Paper
4. 廖丹,孙罡,曾帅. P2P流媒体系统关键技术[M]. 北京:国防工业大学出版社. (2014)
5. Wikipedia. Kademlia网络[EB/OL]. https://en.wikipedia.org/wiki/Kademlia
6. Yandong Wen1,Kaipeng Zhang1,Zhifeng Li1,Yu Qiao. Discriminative Feature Learning Approach for Deep Face Recognition. http://ydwen.github.io/papers/WenECCV16.pdf. (2016)
7. Don Johnson,Alfred Menezes,Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf.
8. David Vorick,Luke Champine. Sia: Simple Decentralized Storage. http://www.sia.tech/whitepaper.pdf. (2014)