The Metanet

A Blockchain-based Internet

Technical summary

Copyright

翻译版权

翻译: 高喆宇
校对: 张林杰

2019 - 06 - 21

目录

1 简介

我们为分布式点对点互联网提出了一种新的协议Metanet, 它使用底层区块链将数据存储在有向图结构中，使数据可以通过浏览器方便地搜索和访问。

该协议不仅旨在重用互联网的大部分功能，还可以利用区块链技术相关的许多其他优势。 这是一个layer-2 解决方案, 而且不需要对Bitcoin协议或底层区块链的共识规则进行任何更改。

Metanet协议可以为数据和内容创建一个有向图，这些图不一定与支付相关。 在本规范中，任何类因特网或其他内容数据都将被视为Metanet的一部分，因为这些数据及其相关用途已经超出了区块链的简单支付场景。

总而言之，我们提议将结构化的交易组合理解为Metanet有向图的顶点，并将数字签名做为它们之间的(边)链接。 这允许任何人使用链上数据和标准化协议在链下重建Metanet有向图。

该图可以使链上数据被类因特网P2P网络节点有效地关联, 处理并分发. 同时以PoW挖矿机制保证信用是去中心化且高透明度的, 数据是完整的，真实和有效的.

2 Metanet 协议

Metanet是一种layer-2协议，其设计使得任何数据都可以不可篡改地存储在区块链中并且以灵活的方式被搜索和访问。 这主要是将Metanet数据通过公钥地址管理并以有向图的构造来实现的。

Metanet协议中涉及的数据可大致分为：

I. Metanet**标记** - 在每个交易中使用的4字节伞形前缀。此标记表示交易将被视为Metanet有向图的一部分。 所有其他相关的子协议都可以包含在Metanet标记中。

II. Metanet**属性** - 与内容数据相关的任何关联且有效的元数据。 这可以包括但不限于内容数据的索引，许可和编码信息。

III. Metanet**内容** - 有价值的载荷数据, 这可以包括但不限于文件，网页，链接. 且数据格式可以根据需要为原始，压缩或加密的。

具体关于插入,结构化内容及属性数据的方法可以是灵活的并且适配到不同的子协议及使用场景中, 但需要注意的是, 通常的惯例是所有的数据插入方法均符合metanet协议大纲关于有向图的顶点和边结构的定义.

2.1 顶点 和 边 结构

我们现在将详细介绍用于构造Metanet交易的协议，该协议允许寻址，权限和版本控制。而这种分布式点对点网络的结构类似于现有互联网。

这种有向图结构的目的是

（i）将内容通过不同的交易关联;
（ii）允许用户使用人类可读的关键字搜索来查找内容;
（iii）在区块链中构建类似云端的服务器虚拟机。

我们的方法是将与Metanet相关的数据构建为有向图。Metanet有向图的顶点和边对应于

顶点 - 与Metanet协议关联的交易。

顶点存储了内容数据。 一个有向图顶点是通过包含 的OP_RETURN操作码来创建的。

每个有向图顶点及OP_RETURN操作码对应了一个公钥 P_{node}, 同时公钥和交易ID唯一指向一个Metanet顶点索引 $ID_{node} := H(P_{node} \,||\, TxID_{node})$.

边(有向边) - 一种子顶点与父顶点的关联.

一个有向边是通过签名 $Sig\ P_{parent}$ 出现在 metanet交易的input, 所以只有一个父顶点才能创建有向边, 所有顶点最多只能有一个父顶点, 而一个父顶点可以有任意数量的子顶点[1].

需要强调的是, 在任意两个顶点间创建一个有向边就是创建了一个逻辑链接, 这并不一定代表一个顶点花费了余额到另一个顶点, 这使得创建有向边在有效期内保持了最大的灵活度.

有效的Metanet有向图顶点（带有父顶点）由以下格式的交易给出：

$TxID_{node}$	
Inputs	Outputs
< $Sig\ P_{parent}$ > < P_{parent} >	OP_RETURN <Metanet Flag> < P_{node} > < $TxID_{parent}$ >

图 1. 一个 Metanet 顶点交易.

[1] 在图论的语言中，每个顶点的indegree最多为1，每个顶点的outdegree是任意的。

如下交易包含指定顶点及其父顶点索引所需的所有信息

$$ID_{node} = H(P_{node} \,||\, TxID_{node}) \,,\; ID_{parent} = H(P_{parent} \,||\, TxID_{parent}) \,.$$

此外，由于父顶点的签名是必需的，因此只有父顶点可以为子节点创建有向边。 这是Metanet协议的基本设计元素，其目的是为了服务更复杂的内容寻址及访问权限控制的需求。

如果 <$TxID_{parent}$> 字段并没有在一个有向图顶点中出现, 或者其没有指向一个有效的metanet交易, 那么该顶点会被认为是一个孤点.

额外的顶点属性也可能会被加入, 它们可以是标记(flags), 命名(names) 及 关键词(keywords). 我们会在下面的小节中讨论这些属性.

我们已经看到了如何将顶点的索引分解为

a）公钥 P_{node} ，我们将其解释为顶点的地址;
b）交易ID $TxID_{node}$ ，我们将其解释为顶点的版本。

两个有趣的功能出现了。

1. **版本控制** - 如果有两个顶点具有相同的公钥，那么我们将拥有最大PoW工作量的交易ID的顶点解释为该顶点的最新版本。

   如果顶点并不都在同一个区块(block), 那么这个功能可以通过区块高度来检查, 如果顶点都在同一个区块中, 那么这个功能可以通过拓扑交易排序(TTOR)来检查.

2. **权限管理** - 子顶点只有在公钥P_{node}的所有者签署了交易输入时才能被创建， 因此P_{node}不仅表示顶点的地址，还表示创建子顶点的权限。 这类似于标准比特币交易 - 公钥不仅是地址，还包括与该地址相关的权限。

   请注意，由于父顶点的签名出现在UXTO解锁脚本中，因此在交易被网络接受时，将通过矿工标准验证程序对其进行验证。 这意味着比特币网络本身也被用于验证创建子顶点的权限。

值得注意的是，标准互联网协议（IP）地址在某个时间点仅在网络内是唯一的。 另一方面，Metanet 中有向图顶点的索引始终是唯一的，并且没有隔离网络的概念，这允许数据被永久地锚定到单个对象 ID_{node}。

Metanet有向图顶点和边结构允许我们将其可视化为图形（下页）。 并且可以使任何有访问相关链上数据权限的人在链下重建Metanet有向图。

.

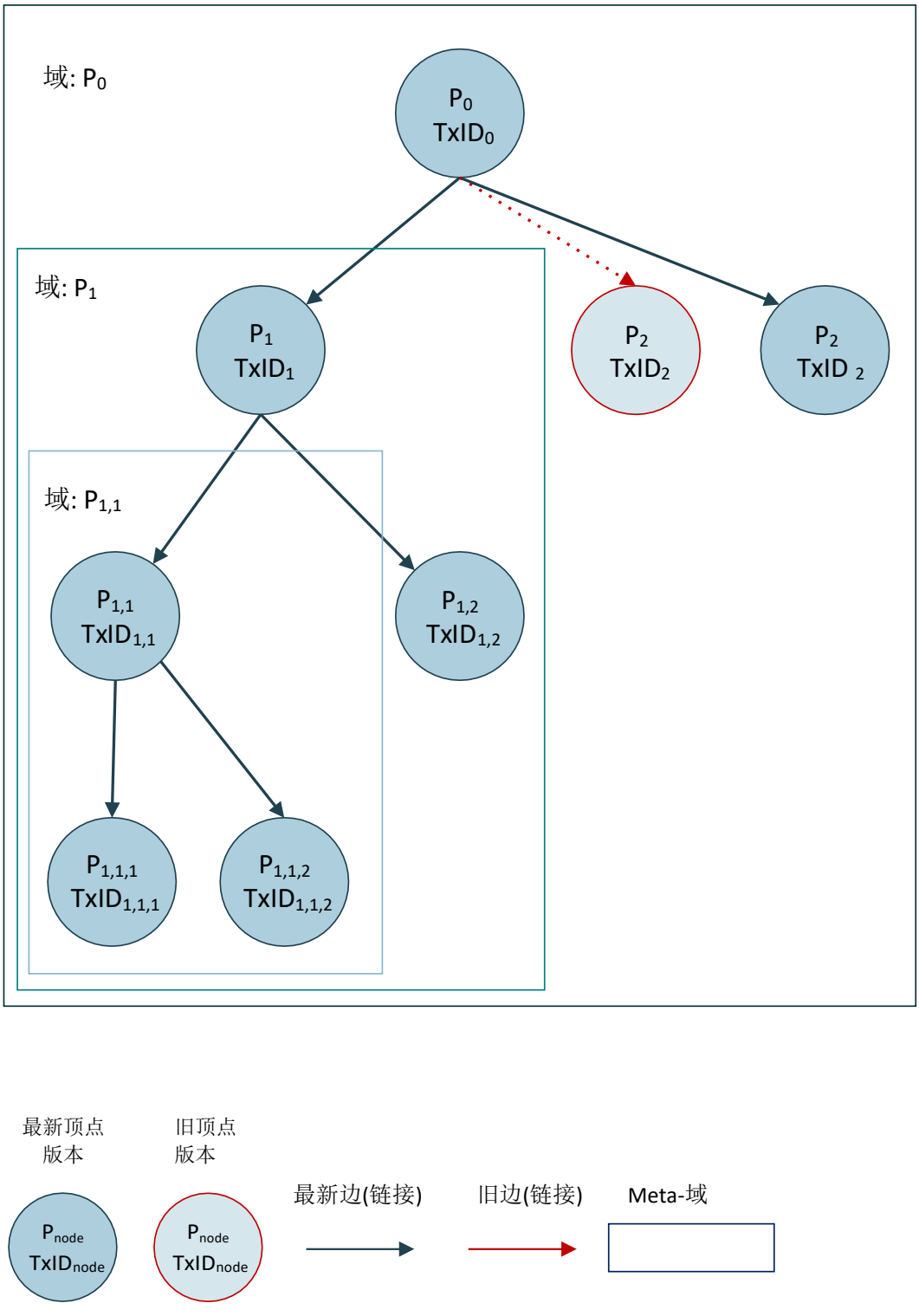

图 2. Metanet 有向图结构 .

2.2 域, 命名 及 定位

有向图的层次结构允许类似域名的结构出现。我们将孤立顶点(孤点)解释为顶级域名，将孤点的子顶点解释为子域名，将孤点的孙顶点解释为子子域名等，将无子的顶点解释为端点。见上面的图2。Metanet中的每个顶级域名可以被认为是树，其中根是孤点，叶子是无子顶点。Metanet本身则是一个树的全集合，从而形成了一个有向图。

Metanet协议没有规定任何顶点包含内容数据，但叶（无子）顶点表示数据树有向路径的末端，因此通常用于存储内容数据。 但内容任然可以存储在树中的任何顶点上。 协议相关的标记(flags)（作为属性包含在顶点中）可用于指定顶点在数据树中的角色（磁盘空间，文件夹，文件或权限变更）。

还记得互联网使用域名系统（DNS）将人类可读的名称与互联网协议（IP）地址相关联。DNS在某种意义上是分散的，尽管在实践中它由少数主要参与者控制，例如政府和大公司。 根据您的DNS提供商，相同的名称可能会将您带到不同的地址(译者注:例如GFW防火墙)。 将简短的人类可读名称映射到计算机可任意生成的数字时，这个问题是不可避免的。

让我们假设存在一个等价的分布式系统，它将人类可读的顶级域名映射到根顶点的分散索引 ID_{root}。换句话说，存在1-1对应函数κ，其将人类可读的名称映射到Metanet根节点索引, 例如

$$\kappa('bobsblog') = ID_{bobsblog} \, (= H(P_{bobsblog} \, || TxID_{bobsblog})) \, .$$

左侧的输入是人类可读的单词，而右侧的输出是散列摘要，通常是256位的数据结构。 请注意，$P_{bobsblog}$ 和 $TxID_{bobsblog}$ 通常也不是人类可读的。 在标准IP协议中，这将是一个将 *www.bobsblog.com* 映射到网络中相应域的IP地址的表。

该Metanet映射表应确保与互联网DNS域名映射表的向后兼容性, 但Metanet有向图结构的命名和寻址方案并不局限于该映射表。

映射函数可以包括现有 IPFS 所采用的 DNSLink[2] 或 OpenNIC 服务[2]，但一般来说这些服务为了方便提供 1-1的映射 牺牲了一些去中心化要素[3]。

命名系统的目的是让用户能够通过好记的单词（例如公司名称）而不是哈希摘要来识别Metanet中的顶级域名。 这也将使搜索域名更快，因为搜索的是关键字而不是哈希摘要。

[2] 该映射可以作为DNS的一部分存储在现有的TXT记录中。 这类似于行星际文件系统(IPFS) [1]中的DNSLink.

个性地址

Metanet顶点地址所使用的公钥通常不是人类友好的可读对象. 然而, 创造一个人类可识别的公钥地址是可能的, 个性地址 - P_{vanity} - 包含一个可识别的文本前缀是可以被用户理解的.

一个包含有需求的文本前缀个性地址是

$P_{bobsblog}$: bobsblogHtKNngkdXEeobR76b53LETtpyT

前缀: bobsblog

后缀: HtKNngkdXEeobR76b53LETtpyT

地址 P_{vanity} 和 *TxID* 组合而形成的 ID_{node} 也是有益的, 因为这意味着没有中心化的域名发行方 (*TxID* 是由去中心化的PoW工作量证明来产生的), 且名字也是可以从区块链本身恢复的. 从此互联网不再会有DNS失效问题存在.

因为Metanet域名已经有了一个所有权系统(公钥), 所以不再需要证书去证明所有权, 把区块链用于这种目的也被如namecoin [4]这样的区块链探索过, 而我们仅需使用一条公链来闭环实现该场景所以不再需要额外的区块链.

定位资源

鉴于我们有一个从域名到顶点索引的映射，我们可以构建一个类似于Internet的统一资源定位符（URL）的定位符。 我们将此称为Metanet URL（MURL），并采用格式

MURL = 'mnp:' + '//domain name'+ '/path' + '/file' .

URL的每个组件 - 协议，域名，路径和文件 - 都已映射到MURL的结构，使得对象更加用户直观并且可以与现有的互联网架构集成。

这假定每个节点都有一个与其公钥（地址）相关联的“名称”，该名称在域树中同一层级内是唯一的。此名称始终是给定顶点的MURL的最右侧部分，类似于目录中特定文件的名称。 如果树中同一级别的两个顶点具有相同的名称，则它们将具有相同的公钥，因此将（根据 *TxID* ）获取最新版本。

许多协议可以被开发出来用于定位链上资源，例如已经在使用的'b：//'和'c：//'协议。 这与<Metanet Flag>是所有链上Metanet数据的标记方式相同，所以一个通用资源位置协议 'mnp://' 可通过伞形协议标记的方式将域名链接到顶点标识符 ID_{node}.

这意味着，如果我们能够将入口与 ID_{node} 相关联，那么我们可以构建MURL, 来简单地通过沿着与根 ID_{node} 相关联的Metanet树的路径, 来定位资源。 此概念可以被泛化为使资源定位器支持更复杂的内容寻址或 *TxID* - 寻址, 因为资源定位器可以通过顶点ID解析此类查询。

2.3 Metanet 搜索

我们已经定义了Metanet有向图结构，使得每个顶点都具有唯一索引 ID_{node}，并且还可以赋予其名称属性以便使用MURL定位内容。 为了实现快速搜索功能，我们可以给顶点添加其他关键字属性。

顶点所固有的属性是索引（ ID_{node} ）和父顶点的索引（ ID_{parent} ），每个包含两个字段（ *P*，*TxID* ），可选属性可以包括名称"name"和关键字"kwd"。

顶点属性:

{

index: $H(P_{node} || TxID_{node})$;

index of parent: $H(P_{parent} || TxID_{parent})$; (NULL if orphan)

name: 'bobsblog';

kwd1: 'travel';

kwd2: 'barbados';

⋮

}

一种实用的Metanet搜索方法, 是先使用区块浏览器扫网区块链并发现所有带Metanet标记的交易，然后检查它们是否是有效的Metanet有向图顶点，如果是，则将其索引(index)和关键字(kwds)记录在数据库中。 然后，该数据库可用于有效地搜索具有所需关键字的顶点。 一旦找到了包含所需关键字的顶点的索引(index)，内容就可以从区块浏览器中获取并查看。

作为示例，考虑图3的 P_1 分支，其中对应于公钥 P_0 ，P_1 和 $P_{1,1}$ 的顶点分别表示主页(P_0)，主题页面 (P_1) 和子主题页面($P_{1,1}$)。 这些节点的名称分别为"bobsblog"，"summer"和"caribbean"，其属性如下所示。

顶点 P_0 主页

MURL: mnp://bobsblog

{

index: $H(P_0 || TxID_0)$;

index of parent: *NULL*

name: 'bobsblog';

kwd1: 'travel';

kwd2: 'barbados';

⋮

}

子顶点 P_1 主题页面

MURL: mnp://bobsblog/summer

{

index: $H(P_1 || TxID_1)$;

index of parent: $H(P_0 || TxID_0)$;

name: 'summer';

kwd1: 'travel';

kwd2: 'barbados';

⋮

}

孙子顶点 $P_{1,1}$ 子主题页面
MURL: mnp://bobsblog/summer/caribbean
{
index: $H(P_{1,1} \| TxID_{1,1})$;
index of parent: $H(P_1 \| TxID_1)$;
name: 'caribbean';
kwd1: 'travel';
kwd2: 'barbados';
⋮
}

在此示例中，叶顶点 $P_{1,1,1}$ ，$P_{1,1,2}$ 和 $P_{1,1,3}$ 分别被赋予名称"海滩"，"夜生活"和"食物"，并单独用于存放博客文章。 完整的域结构显示在下面的示意图上（图3），包括与metanet顶点树中每个节点相对应的MURL搜索路径。

我们注意到，Metanet还可以通过由存储在顶点交易的附加哈希内容摘要属性来兼容CAN (内容可寻址网络)。 这意味着Metanet顶点内容也可以通过内容的哈希摘来进行索引和搜索。

.

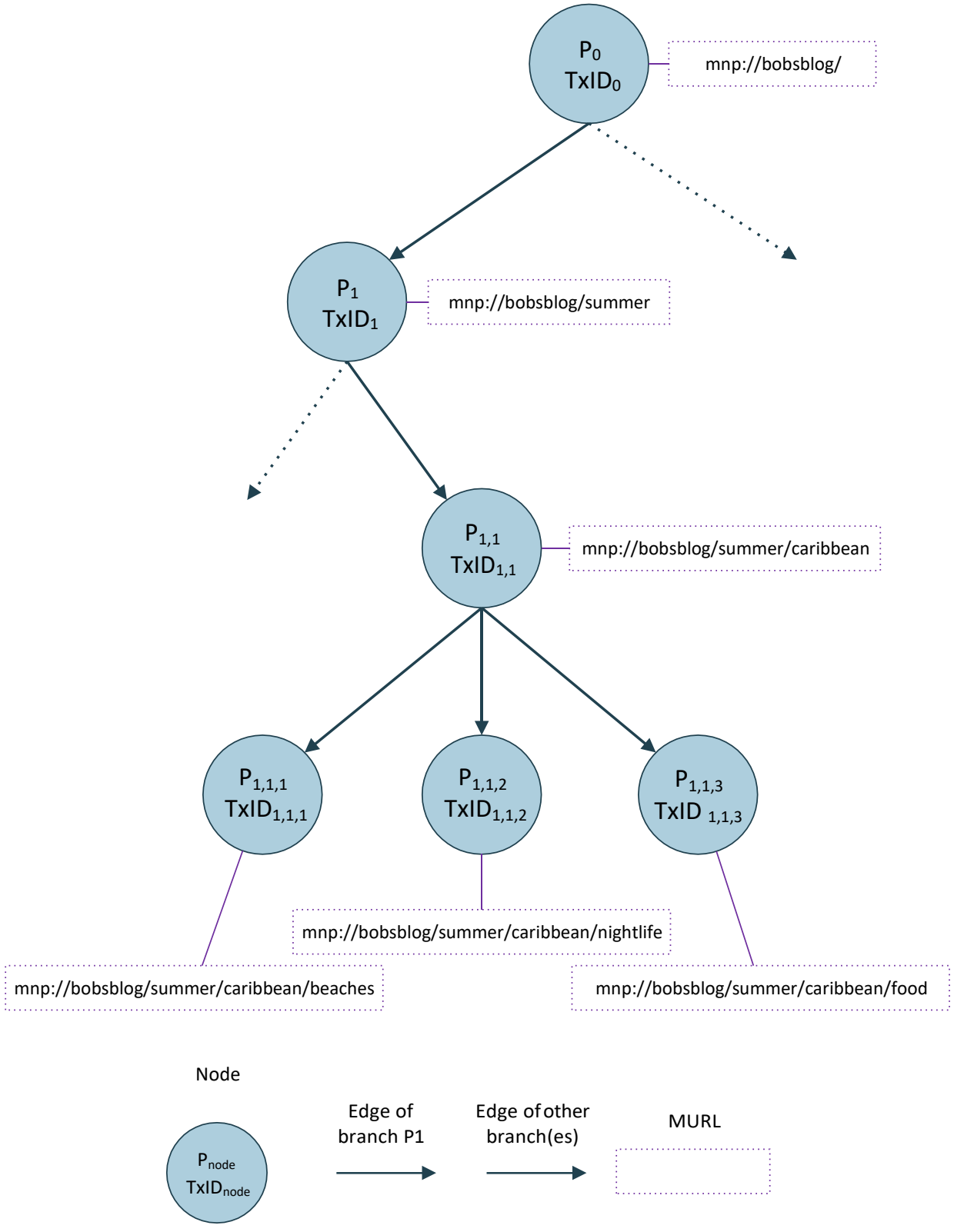

图 3. 一个关于域 'bobsblog' 且包含MURL搜索路径的Metanet树

2.4 数据插入

在2.1节中，我们提到了这样一个情况，即Metanet协议产生了链上内容的有向图结构，可以支持许多不同的插入数据的方法。 例如，在多数情况下，将内容存储在单个交易中是可行的，但在其他情况下，则需要使用Metanet协议通过多组交易将拆分数据链接起来。

此外，可能有用户希望将所有数据上传到OP_RETURN有效负荷中，而其他人可能希望将内容数据上传为可交易的[1]：<Content>OP_DROP。

如之前所述，所有这些方法都是与Metanet协议兼容并支持的, 只要它们都采用了Metanet顶点交易的基本形式。

还记得图1中所示的交易指定了交易中所需的最少信息，以使其被视为Metanet的一部分并且可以根据有向图结构进行理解。 只要该图中所示的元素都存在，则可以使用任何方法将内容数据及其属性嵌入。

我们在这（非详尽地）提供了一些如何以符合Metanet协议的方式嵌入数据的例子。 在下文中，Metanet协议的核心功能会以蓝色突出显示。

.

2.4.1 仅使用 OP_RETURN

下图显示了一个Metanet有向图顶点，其中包含单个OP_RETURN输出中的所有内容数据。

$TxID_{node}$	
Inputs	Outputs
< $Sig\ P_{parent}$ > < P_{parent} >	OP_RETURN < *Metanet Flag* > < P_{node} > < $TxID_{parent}$ > <file_name> <file_type> <file_data>

图 4. 一个使用单一 OP_RETURN的Metanet顶点.

文件名,文件类型作为OP_RETURN输出中的可选属性以及随后的文件数据，被包含在实际有效的负荷中。

任何根据文件类型或用例来编码的相关元数据属性集合和模式，以及文件数据都可以在输出内被分成几个离散数据块。

[1] 译者注: spendable 可以翻译为可花费, 也可以为可交易的, 但在此处, 因为比特币交易的解锁机制, 任何数据在地址间input, output的所有权转移, 必须要拥有input地址的私匙, 这些数据包括但不限于bitcoin货币余额数值, 通证, 授权软件激活码, 用户数据等.

2.4.2 使用 OP_RETURN 及 OP_DROP

下图显示了一个Metanet顶点，其中文件数据已在OP_DROP操作码语句中的两个可交易输出中分割。 如果要嵌入更大的文件，或者仅希望创建较少的输出，则可以使用该插入方法。

<table>
<tr><td colspan="2">TxID_{node}</td></tr>
<tr><td>Inputs</td><td>Outputs</td></tr>
<tr><td>< <i>Sig P_{parent}</i>> <<i>P_{parent}</i> ></td><td>OP_RETURN <<i>Metanet Flag</i>> <<i>P_{node}</i>> <<i>TxID_{parent}</i>>
<sub-protocol identifier>
<file_name> <file_type>
<recombination_scheme></td></tr>
<tr><td></td><td><file_data 1> OP_DROP</td></tr>
<tr><td></td><td><file_data 2> OP_DROP</td></tr>
</table>

图 5. 一个使用 OP_RETURN 与 OP_PUSHDATA的 metanet 顶点.

在这种情况下，我们默认在OP_RETURN输出中包含文件所有元数据和属性是明智的，因为它们将充当文件的文件头“header”。

但是，如果有一些属性属于<file_data 1>而不是<file_data 2>，那么在可交易的输出中包含额外属性也是合理的。

请注意，我们也包括了一个附加标识符<sub-protocol identifier>来表示使用特定的假想数据插入协议。 我们看到子协议也是符合Metanet协议框架内的，并且可以认为它们是作为Metanet协议伞之上的衍生协议。

此顶点形式的唯一条件仍然是它遵循基本的Metanet有向图顶点交易格式。 这意味着Metanet标记，顶点公钥 P_{node} 和父交易标识符 $TxID_{parent}$ 首先出现在OP_RETURN输出中，并且父顶点的公钥和签名 $Sig\ P_{parent}$ 也出现在输入中。

2.4.3 使用多个交易

虽然也可使用任何先前的数据插入方法，但也可以通过使两笔交易成为Metanet协议下的相关连交易的方法来分割数据。

下图显示了一对子交易 Tx_1 和 Tx_2，且它们都具有相同的父metanet交易，因此它们也是兄弟顶点。

$TxID_{node\ (1)}$	
Input	Output
< $Sig\ P_{parent}$ > < P_{parent} >	OP_RETURN < *Metanet Flag* > < $P_{node(1)}$ > < $TxID_{parent}$ > <sub-protocol identifier> <file_name> <file_type> <data_index 1> <recombination_scheme>
	<file_data 1> OP_DROP

$TxID_{node\ (2)}$	
Input	Output
< $Sig\ P_{parent}$ > < P_{parent} >	OP_RETURN < *Metanet Flag* > < $P_{node(2)}$ > < $TxID_{parent}$ > <sub-protocol identifier> <file_name> <file_type> <data_index 2> <recombination_scheme>
	<file_data 2> OP_DROP

图 6. 一对用于插入分割大文件的Metanet兄弟顶点交易.

这两个交易具有共同的父交易，因此父交易签名将出现在它们各自的输入中。 需要记住的是，该签名并不需要被用于交易父顶点本身的输出，子顶点只需要该签名是父公钥的有效签名即可。

当子交易的输入确实花费了其父交易的输出的情况下，所产生的Metanet有向边(链接)仍然与子交易花费由父公钥 P_{parent} 签署的任意父交易输出的结果完全相同[1]。

[1] 译者注: 在一个有效的比特币交易中, 只要交易输出的值之和 不大于 输出的值之和+矿工费, 即可被认为是有效交易, 这意味着, 一个有效交易可以有很多输出地址的值为0, 这意味着, 输入地址的拥有者可以不给输出地址转任何比特币, 但可以通过OP_RETURN 和 OP_DROP 给他们转metanet信息.

2.5 Metanet 和 互联网

Metanet是一种分布式且基于区块链作为底层技术来构建链上数据内容的网络协议.

下表给出了Metanet协议和Internet协议之间的对比：

Internet	Metanet
网站/文件	顶点
拥有者	公钥 P_{node}
IP 地址 (非唯一)	顶点索引 (唯一) $ID_{node} = H(P_{node} \| \| TxID_{node})$
域名结构	顶点树结构
域名系统	从根顶点名映射到顶点的索引
URL http://www.bobsblog.com/path/file	**MURL** mnp://bobsblog/path/file

表 1. Internet和Metanet协议之间的类比总结.

其原则是确保Metanet至少可以提供与现有互联网相同的功能，同时还带来与其底层区块链技术继承来的优势和新功能。

3 参考

编号	参考
[1]	"IPFS Documentation", *Docs.ipfs.io*, 2019. [Online]. Available: https://docs.ipfs.io/guides/concepts/dnslink/. [Accessed: 22- May- 2019].
[2]	"OpenNIC Project", *Opennic.org*, 2019. [Online]. Available: https://www.opennic.org/. [Accessed: 22- May- 2019].
[3]	"Ten terrible attempts to make the Inter Planetary File System human-friendly", *Hacker Noon*, 2019. [Online]. Available: https://hackernoon.com/ten-terrible-attempts-to-make-the-inter-planetary-file-system-human-friendly-e4e95df0c6fa. [Accessed: 22- May- 2019].
[4]	"Namecoin", *Namecoin.org*, 2019. [Online]. Available: https://namecoin.org/. [Accessed: 22- May- 2019].