



(10) 授权公告号 CN 111433803 B

(45) 授权公告日 2024. 09. 03

(21) 申请号 201880077922.9

(22) 申请日 2018.11.28

(65) 同一申请的已公布的文献号
申请公布号 CN 111433803 A

(43) 申请公布日 2020.07.17

(30) 优先权数据
17425121.5 2017.12.01 EP
102017000145294 2017.12.15 IT

(85) PCT国际申请进入国家阶段日
2020.06.01

(86) PCT国际申请的申请数据
PCT/EP2018/082832 2018.11.28

(87) PCT国际申请的公布数据
W02019/106006 EN 2019.06.06

(73) 专利权人 快特网络有限公司
地址 英国伦敦

(72) 发明人 吉尔伯特·威尔第 科林·帕特森
盖塔诺·蒙代利 保罗·塔斯卡

(74) 专利代理机构 北京三高永信知识产权代理
有限责任公司 11138
专利代理师 关越

(51) Int.Cl.
G06Q 30/06 (2006.01)
G06Q 20/38 (2006.01)

(56) 对比文件
US 2017213209 A1, 2017.07.27
US 2023334036 A1, 2023.10.19
CN 108282474 A, 2018.07.13
CN 116233125 A, 2023.06.06
US 11308448 B1, 2022.04.19
曾诗钦; 霍如; 黄韬; 刘江; 汪硕; 冯伟. 区块链技术研究综述: 原理、进展与应用. 通信学报. 2020, (第01期), 全文.
曾诗钦; 霍如; 黄韬; 刘江; 汪硕; 冯伟. 区块链技术研究综述: 原理、进展与应用. 通信学报. 2020, (第01期), 全文.

审查员 杨璐

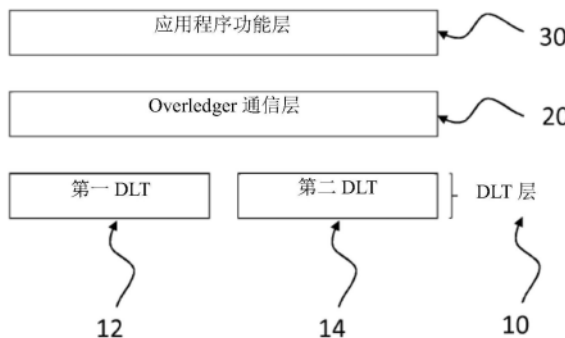
权利要求书2页 说明书19页 附图11页

(54) 发明名称

区块链通信和排序

(57) 摘要

本公开提供了一种由计算机实现的用于维护出现在一个或多个区块链上的交易顺序记录的方法, 所述方法包括: 识别多笔交易, 其中, 所述多笔交易中的每笔交易都记录在所述一个或多个区块链中的任一区块链上; 以及将所述多笔交易的记录存储在数据存储中, 其中, 所述记录指示所述多笔交易的相对顺序。



1. 一种由计算机实现的用于维护每个出现在多个区块链中的任一区块链上的交易顺序记录的方法,所述方法包括:

识别多笔交易,其中所述多笔交易中的至少一笔交易被记录在所述多个区块链中的第一区块链上,并且所述多笔交易中的至少另一笔交易被记录在所述多个区块链中的第二区块链上;

将所述多笔交易的记录存储在数据存储中,其中,所述记录指示所述多笔交易的相对顺序,所述多笔交易的记录包括多个连续的验证集,其中,所述多个连续的验证集中的至少一些验证集包括与所述多笔交易中的一笔或多笔交易各自相对应的一个或多个交易标识符;

在所述多个区块链中的至少一个区块链上,存储所述多个连续的验证集中的最新验证集的记录;

检查所述多个连续的验证集中的先前验证集的记录是否存在于所述多个区块链中的所述至少一个区块链上,以及

如果所述先前验证集的记录不存在于所述多个区块链中的所述至少一个区块链上:
则将所述先前验证集的记录重放至所述多个区块链中的所述至少一个区块链上。

2. 根据权利要求1所述的方法,其中,所述识别多笔交易包括:

读取添加到所述多个区块链中的每个区块链中每个新区块的内容。

3. 根据权利要求2所述的方法,其中,所述识别多笔交易还包括:

将每个新区块中的交易与相关性准则进行比较,其中,识别出的多笔交易包括符合所述相关性准则的交易。

4. 根据权利要求2或3所述的方法,其中,所述将所述多笔交易的记录存储在数据存储中包括:每当在新区块中识别出交易时,将识别出的交易的记录存储在所述数据存储中,其中,所述多笔交易的所述相对顺序为所述多笔交易中每笔交易被识别出的相对顺序。

5. 根据权利要求1所述的方法,还包括:

如果所述先前验证集的记录不存在于所述多个区块链中的所述至少一个区块链上:

则检查所述先前验证集中识别出的每笔交易是否存在于所述多个区块链的对应区块链上,以及

如果所述先前验证集中识别出的一笔或多笔交易不存在于所述或多个区块链的相应区块链上:

则将所述一笔或多笔交易重放至其相应的区块链上。

6. 根据权利要求1所述的方法,还包括:

检查所述最新验证集中识别出的每笔交易是否存在于所述多个区块链中的所述至少一个区块链上,以及

如果所述最新验证集中识别出的一笔或多笔交易不存在于所述多个区块链的相应区块链上:

则将所述一笔或多笔交易重放至其相应的区块链上。

7. 根据权利要求1所述的方法,其中,所述最新验证集的记录包括验证集标识符,所述验证集标识符至少部分地基于所述最新验证集的内容来确定。

8. 根据权利要求7所述的方法,其中,所述最新验证集的所述验证集标识符包括所述最

新验证集的内容的哈希。

9. 根据权利要求1所述的方法, 其中, 所述多笔交易的记录包括多个交易标识符, 每个所述交易标识符至少部分地基于相应交易的内容来确定。

10. 一种由计算机实现的用于维护每个出现在多个区块链中的任一区块链上的交易顺序记录的系统, 包括:

一个或多个处理器; 以及

存储有软件程序的存储器, 其中, 当被所述一个或多个处理器执行时, 所述软件程序使所述一个或多个处理器执行根据前述权利要求中任一项所述的方法。

11. 一种计算机程序产品, 包括计算机程序, 当所述计算机程序被一个或多个处理器执行时, 使所述一个或多个处理器执行根据权利要求1至9中任一项所述的方法。

12. 一种基于软件开发工具包SDK的系统, 包括一组用于开发根据权利要求11所述的计算机程序产品的软件开发工具。

区块链通信和排序

技术领域

[0001] 本公开涉及用于维护出现在一个或多个区块链上的交易顺序记录的方法、系统、计算机程序及软件开发工具包 (SDK), 特别涉及与区块链通信相关的方法、系统、计算机程序及软件开发工具包。

背景技术

[0002] 区块链技术具有广阔的应用前景, 其提供了一种无需中心数据认证机构即可安全可靠地存储数据的方式。因此, 在诸多领域, 区块链技术具备了突出的技术优势, 包括, 例如, 提高数据透明度, 不依赖特定存储实体实现更稳健的数据存储, 增强数据安全性以及提高防欺诈能力, 并且已经在诸如电能分配和点对点云存储等多个领域中应用。

[0003] 区块链是一种去中心化的分布式数字账本, 它可以存储数据的有效不可变的记录。去中心化是因为其不一定需要中央机构来向区块链添加数据或维护区块链的完整性。可以通过向参与区块链的节点的网络广播“交易”的实体将数据添加到区块链, 其中, 交易包括要添加到区块链的数据和符合区块链协议的加密元素。这样, 参与区块链的每个节点都可以检查交易的有效性, 如果通过检查, 则将交易添加到节点当前参与的新区块中。

[0004] 每个节点还可以将已广播到节点网络的其他新交易添加到节点当前参与的新区块中。之后, 节点会将其新区块公布给网络中的其他节点, 该新区块包括各种新交易以及将该新区块与区块链中先前区块相关联的加密数据。通过这种方式, 将连续的区块以加密的方式进行关联, 如果在之后的任何时间更改了区块链中的一个区块 (例如, 某人尝试欺诈性地更改存储在区块链中的数据), 则会很容易地被检测到。

[0005] 在一个节点公布了一个新区块之后, 网络中的其他节点可以检查该新区块的内容, 如果新区块被接受, 则可以开始处理其下一个新区块, 该下一个新块将再次加密地关联到其前一个区块。因此, 可以看出, 每次公布并接受新区块时, 区块链都会增长, 每个新接受的区块都将加密地关联到其先前区块。这种节点检查已公布的新区块并接受其加入到区块链中的过程通常称为“共识”机制。

[0006] 人们已经认识到, 区块链技术可以得到广泛的利用, 并且可以在数据的有效不可变的记录有用的任何情况下使用。这导致当前存在的不同区块链的数量迅速增加, 每种区块链都针对自己的特定用例进行了量身定制, 并且随着技术获得进一步的推动和认可, 预计不同的区块链的数量将持续增加。

发明内容

[0007] 在本公开的第一方面, 提供了一种由计算机实现的用于维护出现在一个或多个区块链 (DLT) 上的交易顺序的记录的方法。该方法包括: 识别多笔交易, 其中多笔交易中的每笔交易都包含在一个或多个区块链中的任一区块链上; 以及将多笔交易的记录存储在数据存储中, 其中, 该记录指示多笔交易的相对顺序。

[0008] 通过维护交易顺序的记录, 可以缓解与无序消息相关联的安全性和稳定性漏洞,

从而提高利用一个或多个区块链的系统的稳定性和稳定性。鉴于多个跨区块链的交易排序实现困难且容易导致安全和欺诈问题,当记录跨两个或两个以上(多个)区块链上的交易顺序时,这种技术优势尤其明显。

[0009] 识别多笔交易可以包括:读取被添加到一个或多个区块链中的每个区块链中每个新区块的内容。

[0010] 识别多笔交易还可以包括:将每个新区块中的交易与相关性准则进行比较,其中,识别出的多笔交易包括符合该相关性准则的交易。通过与相关性准则进行比较,仅需要对数据存储中的相关交易进行识别,这可以提高后续查看该数据存储的速度并减小所需数据存储容量。

[0011] 优选地,将多笔交易的记录存储在数据存储中包括:每当在新区块中识别出交易时,将识别出的交易的记录存储在数据存储中,其中,多笔交易的相对顺序为多笔交易中每笔交易被识别出的相对顺序。

[0012] 优选地,多笔交易的记录包括:多个连续的验证集,其中,多个连续的验证集中的至少一些验证集包括与多笔交易中的相应一笔或多笔交易各自相对应的一个或多个交易标识符。

[0013] 该方法还可以包括:在一个或多个区块链中的至少一个区块链上,存储多个连续的验证集中的最新验证集的记录。这样,可以增强记录的完整性和不可否认性,从而提高了该方法的安全性和稳定性。

[0014] 该方法还可以包括:检查多个验证集中的先前验证集的记录是否存在于一个或多个区块链中的至少一个区块链上;以及如果先前验证集的记录不存在于一个或多个区块链中的至少一个区块链上,则将先前验证集的记录重放到一个或多个区块链中的至少一个区块链上。这样,可以快速识别区块链中的早期分叉(Earlier Forks),同时仍维护记录的交易顺序的可靠性。

[0015] 在一个或多个区块链中的至少一个区块链上,存储最新验证集的记录之前,可以将先前验证集的记录重放到一个或多个区块链中的至少一个区块链上。

[0016] 该方法还可以包括:如果先前验证集的记录不存在于一个或多个区块链中的至少一个区块链上,则检查在先前验证集中识别出的每笔交易是否存在于一个或多个区块链的对应的区块链上;以及如果先前验证集中识别出的一笔或多笔交易不存在于一个或多个区块链的对应区块链上,则将一笔或多笔交易重放至其对应的区块链上。

[0017] 该方法还可以包括:检查在最新验证集中识别出的每笔交易是否存在于一个或多个区块链中的至少一个区块链上;以及如果在最新验证集中识别出的一笔或多笔交易不存在于一个或多个区块链的对应区块链上,则将一笔或多笔交易重放至其对应的区块链上。

[0018] 在一个或多个区块链中的至少一个区块链上,存储最新验证集的记录之前,可以将一笔或多笔交易重放到其相应的区块链上。

[0019] 最新验证集的记录可以包括验证集标识符,该验证集标识符至少部分地基于最新验证集的内容来确定。

[0020] 最新验证集的验证集标识符可以包括:最新验证集的内容的哈希。

[0021] 存储最新验证集的记录可以:包括将交易广播到一个或多个区块链中的至少一个区块链,以包括在至少一个区块链中,其中,广播交易包括最新验证集的唯一标识符。

[0022] 多笔交易的记录可以包括:多个交易标识符,每个交易标识符至少部分地基于其交易的内容来确定。

[0023] 多笔交易标识符可以包括多笔交易的内容的多个哈希。

[0024] 多笔交易的记录可以包括多笔交易中的至少一些交易的内容。

[0025] 在本公开的第二方面,提供了一种系统,包括:一个或多个处理器;以及

[0026] 一种存储有软件程序的存储器。当被该一个或多个处理器执行时,该软件程序使该一个或多个处理器执行根据第一方面所述的方法。

[0027] 在本公开的第三方面,提供了一种或多种计算机程序。当被一个或多个处理器执行时,该计算机程序使该一个或多个处理器执行根据第一方面所述的方法。

[0028] 在本公开的第四方面,提供了一种软件开发工具包(SDK)。该软件开发工具包包括一组用于开发根据第三方面所述的一种或多种计算机程序的软件开发工具。

[0029] 在本公开的第五方面,提供了一种用于提供应用程序功能层和多个不同的区块链之间互操作性(Interoperability)的方法,其中,应用程序功能层包括利用多个不同的区块链中的一个或多个区块链的功能操作。该方法包括区块链通信层:从应用程序功能层接收区块链请求,其中,该区块链请求涉及要在该多个不同区块链中的一个或多个区块链上执行的区块链操作;以及根据一个或多个区块链的协议与一个或多个区块链进行接口连接以执行该区块链操作。因此,应用程序功能层可以与区块链协议分离,从而使应用程序可以更直接地跨区块链操作和在区块链之间切换,从而可以更快地缓解特定区块链中的任何安全性或稳定性威胁。

[0030] 区块链请求可以包括读取一个或多个区块链上的数据的请求。从一个或多个区块链读取数据的请求可以包括数据标识符,该数据标识符指示要从一个或多个区块链读取的数据。

[0031] 与一个或多个区块链进行接口连接可以包括读取一个或多个区块链上的所请求的数据,并且其中,该方法还包括区块链通信层:将从一个或多个区块链读取的数据传送到应用程序功能层。

[0032] 区块链请求可以包括将数据写入一个或多个区块链的请求。

[0033] 写入数据的请求可以包括要被写入至一个或多个区块链的数据。

[0034] 区块链请求可以包括在多个区块链中的第一区块链和多个区块链中的第二区块链之间执行跨账本交易(Cross Ledger Transaction,XLT)的请求,并且其中执行XLT的请求包括XLT数据标识符,该XLT数据标识符指示通过XLT从第一个区块链上传输到第二个区块链的数据。

[0035] 可以根据由区块链编程接口(BPI)限定的方法来执行区块链通信层与应用程序功能层之间的通信。

[0036] 在本公开的第六方面,提供了一种系统。该系统包括:一个或多个处理器;以及一种存储有软件程序的存储器。当被一个或多个处理器执行时,该软件程序使该一个或多个处理器执行根据第五方面所述的方法。

[0037] 在本公开的第七方面,提供了一种或多种计算机程序。当被一个或多个处理器执行时,该计算机程序使该一个或多个处理器执行根据第五方面所述的方法。

[0038] 在本公开的第八方面,提供了一种软件开发工具包(SDK)。该软件开发工具包包括

一组用于开发根据第七方面所述的一种或多种计算机程序的软件开发工具。

[0039] 在本公开的第九方面,提供了一种用于在第一区块链和第二区块链之间执行跨账本交易(XLT)的方法。该方法包括:在第一区块链上识别与XLT相关的拟议中的XLT交易;将拟议中的XLT交易的记录存储在数据存储中的交易记录中;在第二区块链上识别一个就绪的XLT交易;将该就绪的XLT交易的记录存储在数据存储中的交易记录中,其中,该记录指示拟议中的XLT交易和就绪的XLT交易的相对顺序;确定拟议中的XLT交易和就绪的XLT交易是否符合XLT准则;如果拟议中的XLT交易和就绪的XLT交易符合XLT准则,则将提交XLT交易广播到第一区块链和第二区块链以在第一区块链和第二区块链之间执行XLT,其中,XLT准则包括就绪的XLT交易对应于拟议中的XLT交易的第一准则,以及交易记录指示在就绪的XLT交易之前的拟议中的XLT交易的第二准则。因此,即使在一个或多个区块链上有分叉的情况下,也可以可靠地维护和理解跨两个或多个区块链的交易顺序,从而提高跨账本交易的安全性和可靠性,并降低欺诈风险。

[0040] 该方法还可以包括:将一笔或多笔交易广播到第一区块链和/或第二区块链,其中,一笔或多笔交易包括各自的一个或多个交易标识符,每个交易标识符至少部分地基于交易记录的至少一部分内容确定,一个或多个交易标识符共同指示拟议中的XLT交易和就绪的XLT交易的记录。

[0041] 一个或多个交易标识符可以包括交易记录的至少一部分内容的一个或多个哈希。

[0042] XLT准则可以包括第三标准,即一笔或多笔交易包括在第一区块链和/或第二区块链上,从而一个或多个交易标识符被存储在第一区块链和/或第二区块链上。

[0043] 拟议中的XLT交易的记录可以包括拟议中的XLT交易的内容的哈希,并且就绪的XLT交易的记录包括就绪的XLT交易的内容的哈希。

[0044] 在本公开的第十方面,提供了一种系统。该系统包括:一个或多个处理器;以及一种存储有软件程序的存储器。当被该一个或多个处理器执行时,该软件程序使该一个或多个处理器执行根据第九方面所述的方法。

[0045] 在本公开的第十一方面,提供了一种或多种计算机程序。当被一个或多个处理器执行时,该计算机程序使该一个或多个处理器执行根据第九方面所述的方法。

[0046] 在本公开的第十二方面,提供了一种软件开发工具包(SDK),该软件开发工具包包括一组用于开发第十一方面所述的一种或多种计算机程序的软件开发工具。

[0047] 在本公开的第十三方面,提供了一种用于维护在两个或更多个区块链上执行的交易的相对顺序的记录的方法。这样,可以可靠地对不同区块链上的交易顺序进行排序,从而能够提高安全性和可靠性,尤其是在区块链分叉和/或XLT的情况下。

[0048] 在本公开的第十四方面,提供了一种系统,该系统包括:一个或多个处理器;以及一种存储有软件程序的存储器,其中,该软件程序,当其被该一个或多个处理器执行时,使该一个或多个处理器执行根据第十三方面所述的方法。

[0049] 在本公开的第十五方面中,提供了一种或多种计算机程序。当被一个或多个处理器执行时,该计算机程序使该一个或多个处理器执行根据第十三方面所述的方法。

[0050] 在本公开的第十六方面,提供了一种软件开发工具包(SDK)。该软件开发工具包包括:一组用于开发根据第十五方面所述的一种或多种计算机程序的软件开发工具。

附图说明

- [0051] 参考以下附图,仅以示例的方式描述了本公开的各方面,其中:
- [0052] 图1示出了根据本公开的一个方面的Overledger通信层的示例表示;
- [0053] 图2示出了利用图1的Overledger通信层的系统的示例表示;
- [0054] 图3示出了利用图1的Overledger通信层的另一系统的示例表示;
- [0055] 图4示出了使用图3的系统执行的示例方法的步骤的表示;
- [0056] 图5示出了利用图1的Overledger通信层的又一系统的示例表示;
- [0057] 图6示出了区块链中的分叉的示例表示;
- [0058] 图7示出了根据本公开的一个方面的用于维护出现在一个或多个区块链上的交易顺序的记录的方法的示例表示;
- [0059] 图8示出了在其上执行图7的方法的两个区块链的示例表示;
- [0060] 图9示出了在其上执行图7的方法的两个区块链的另一示例表示;
- [0061] 图10示出了根据本公开的一个方面的在其上执行跨账本交易的两个区块链的示例表示;
- [0062] 图11示出了根据本公开的一个方面的另一系统的示例表示;以及
- [0063] 图12示出了由图11的系统执行的方法的示例表示。

具体实施方式

[0064] 发明人已经认识到,随着区块链技术的成长和发展,对于不同的实体来说,安全、稳定且可靠地利用区块链变得日益困难。例如,当前,如果实体希望开发使用区块链技术的新的软件应用程序,则必须启动自己的新区块链(这极耗时且复杂,需要建立全新的区块链协议)或利用现有的区块链技术,例如以太坊。

[0065] 发明人已经认识到,为了利用现有的区块链技术,实体必须开发其软件应用程序以完全符合该区块链的协议,例如使用由协议规定的加密技术、交易格式和通信信道。因此,在软件开发的早期,就需要选择使用哪种区块链技术,该技术选择可能要在完全了解软件应用程序的要求和用途之前做出。

[0066] 每个区块链都有一个为满足该区块链的特定需求而制定的特定的协议。在不引起区块链硬分叉的情况下,不能更改区块链协议,这意味着区块链不能很好地适应需求的变化。因此,如果实体后来认为自己选择的区块链技术不再满足他们的需求(例如,因为所选的区块链技术变得不稳定或不安全,或者因为软件应用程序的要求已发生变化,这种变化使不同的区块链技术是有利的,或者因为更新的区块链优于最初选择的但现在已经过时的区块链技术),为了开始使用替代的区块链技术,必须修改其软件以符合替代的区块链技术的协议。实际上,即使可能,这也会非常困难,因为该软件将建立在由最初选择的区块链技术的协议所规定的基础上。因此,从一开始就选择正确的区块链技术可能非常重要,但是由于区块链技术快速发展的性质,在实践中可能很难实现。

[0067] 此外,由于区块链协议之间的差异,先前存储在区块链上的任何数据都可能无法直接传送到替代的区块链上。

[0068] 鉴于发明人已经认识到的这些长期困难,发明人已经设计出并且在本公开中提供了一种引入新的通信层以将应用程序的功能性与所利用的区块链分开的技术方案。通过将

应用程序功能性与在其上运行该应用程序的区块链的各种需求分离,可以改善应用程序与区块链之间的互操作性。从下面更详细描述的本公开的各方面可以理解,这可能意味着将来区块链技术的变化可以更有效、更灵活地响应。这可能有助于应对安全威胁(例如,如果一种区块链技术被认为有安全漏洞,则应用程序可以更直接地切换到更安全的区块链技术),从而提高安全性和可靠性(因为应用程序可以更容易地跨多个区块链运行,跨区块链存储副本数据(Duplicate data)和/或在区块链之间传输数据)和/或提高效率和速度(例如,在更新的区块链技术提高了交易处理速度的情况下,应用程序可能会更直接地切换到该更新的区块链技术)。

[0069] 在以下公开中,术语“区块链”、“区块链技术”和“分布式账本技术(Distributed Ledger Technology, DLT)”可互换使用。此外,在以下所有公开中,术语“交易(Transaction)”的使用与区块链相关。本领域技术人员可以理解的是,“交易”是将某种形式的信息放入区块链的手段。例如,实体可以广播要放入区块链上的“交易”,该交易包括要放入区块链上的信息以及为了使该交易被接受到区块链上所需的任何其他数据项(例如公钥、签名、哈希(Hash)等)。要放入区块链上的信息可以是任何种类的信息(例如,实体希望存储在区块链上以供其以后参考的一条数据)。

[0070] 图1示出了引入位于应用程序功能层30与DLT层10之间的“Overledger通信层”20的可视化示例。在本公开中术语“Overledger通信层”用来表示将DLT的协议需求与利用该DLT的应用程序的功能操作(有时称为“业务逻辑”)分离的通信层。

[0071] 在图1中,两个DLT——第一DLT 12和第二DLT 14——表示为DLT层10的一部分,尽管应当理解,Overledger通信层20可能能够与任何数量的不同DLT进行接口连接,例如以太坊、Corda、Openchain、Hyperledger等。每个DLT都有自己的特定协议用于区块链操作,例如从区块链读取数据,将数据写入区块链和/或将数据所有权从一个实体转移到同一区块链上的另一个实体,或到不同的区块链(不同的区块链可以允许要在区块链上执行的读取、写入和转移操作中的任何一项或多项操作,具体取决于区块链已被配置为要执行的目的)。

[0072] Overledger通信层20包括可在第一DLT 12和第二DLT 14上执行区块链操作的不同方法。例如,第一DLT 12可以具有用于将新交易写入其区块链的特定协议。协议定义了每个项目的允许格式(例如长度和/或对象类型),必须用于生成哈希和签名的加密技术(例如SHA-256)以及交易中每个项目出现的顺序。该协议还可以定义如何从区块链中读取信息,例如,识别区块链上每笔交易和每个区块中保存的信息、信息的格式和顺序,以及用于生成哈希、签名和工作量证明的加密技术(以便在读取区块链时可以对其进行验证)。相比之下,第二DLT 14可能具有完全不同的协议,该协议允许执行不同的区块链操作,例如只读操作(例如,因为它是任何人都可以读取但仅授权的实体可以写入的封闭的区块链),并且每个区块可以包括与第一DLT 12不同的顺序和/或使用不同的加密技术的不同信息。Overledger通信层20将包括使得能够在第一DLT 12和第二DLT 14中的每一个上执行各种允许的操作的方法。

[0073] 图1中表示的应用程序功能层30包括利用一个或多个DLT的应用程序的功能操作(有时称为“业务逻辑”)。当应用程序的应用程序功能层30希望执行区块链操作时,它将相关信息(区块链请求)传递到Overledger通信层20(例如,包括其希望写入DLT的数据,或希望从DLT读取的信息的性质),然后根据相关DLT的协议与相关DLT进行接口连接,以执行该

区块链操作。然后,Overledger通信层20可以将任何相关结果返回给应用程序功能层30。从本公开的稍后描述的更具体的示例可以更充分地理解该操作。

[0074] 因此,各种不同的实体可以使用在Overledger通信层20中进行的方法,可以将这些方法例如封装在软件开发工具包(Software Development Kit,SDK)中,此后将其称为Overledger SDK。可以理解的是,在Overledger通信层20中进行的方法可以以任何其他适当的等效方式来封装。通过获取Overledger SDK,开发者可以更直接地选择他们希望使用哪个(或哪些)DLT用于正在开发的特定软件应用程序,并拥有与他们所选的通过Overledger SDK所提供的DLT进行交互所需的方法和工具。

[0075] 图2示出了系统200的示例表示,该系统200包括SDK提供者210、应用程序开发者220、开发的应用程序230、第一DLT网络240和第二DLT网络250。SDK提供者210可以负责构建和维护Overledger SDK,在这个示例中,SDK提供者210包括用于至少与第一DLT 12和第二DLT 14进行接口连接(交互)的方法,尽管其可以包括用于与任意数量DLT的交互方法(即,用于多个DLT)。当应用程序开发者220希望开发利用第一DLT 12和/或第二DLT 14的应用程序时,其可以从SDK提供者210获得Overledger SDK(例如,通过从SDK提供者210或从任何其他Overledger SDK托管实体处下载)。然后,他们可以利用Overledger SDK开发应用程序230,该应用程序230然后可以根据Overledger SDK中定义的方法与第一DLT 12和/或第二DLT 14进行交互。在图2中,第一DLT网络240表示用于第一DLT 12的参与节点的网络,第二DLT网络250表示用于第二DLT 14的参与节点的网络。在图2中所示的特定示例中,应用程序230被示为能够与第一DLT网络240和第二DLT网络250两者进行交互,尽管可以理解的是,应用程序开发者可以替代地利用Overledger SDK将应用程序230配置为仅与第一DLT网络240或仅与第二DLT网络250进行交互。

[0076] 在一个特定方面,作为开发应用程序230的一部分,应用程序开发者220可以开发编程接口以处理应用程序230的Overledger通信层20和应用程序功能层30之间的通信。此后将该接口称为区块链编程接口(Blockchain Programming Interface,BPI),该接口被配置为帮助简化应用程序功能层30和Overledger通信层20的之间的交互。例如,应用程序230的Overledger通信层20可以根据在Overledger SDK中定义的方法进行操作,以便处理到第一DLT网络240和/或第二DLT网络250的通信,和来自第一DLT网络240和/或第二DLT网络250的通信。应用程序开发者220可以开发BPI以定义特定类型的信息,该信息可以由应用程序230的应用程序功能层30在与要在一个或多个区块上执行的区块链操作有关的区块链请求中传递给Overledger通信层20,和/或该特定类型的信息可以从Overledger通信层20接收。例如,其可以提供“写入第一DLT 12”命令,该命令可以指定要写入第一DLT 12的数据的任何相关参数(例如,最大长度、对象类型等)和/或“从第二DLT 14读取”命令,该命令可以指定从第二DLT 14读取信息所需的任何相关数据(例如,如果需要在第二DLT 14上读取已通过特定实体写入第二DLT 14的数据,“从第二DLT 14读取”命令可以指定其需要该特定实体的区块链标识符(例如其公钥),Overledger通信层20通过使用该标识符可以在第二DLT 14的内容中搜索包括该区块链标识符的交易)。因此,可以看出,应用程序230的应用程序功能层30可以利用由BPI提供的各种命令,从而使Overledger通信层20可以与Overledger SDK所支持的任何DLT进行接口连接。

[0077] 因此,借助于根据Overledger SDK操作的Overledger通信层20,应用程序230的应

用程序功能层30与第一DLT 12和第二DLT 14的特定DLT层10协议需求分离。为了与第一DLT 12和/或第二DLT 14进行交互,应用程序功能层30仅需要以由BPI定义的方式向Overledger通信层20发送和接收数据。因此,如果第一DLT 12或第二DLT 14将来要更改其协议,则可以通过Overledger通信层20借由对Overledger SDK进行更改来处理。应用程序230的应用程序功能层30可以维持不变,因为它仍然会如BPI中定义的向Overledger通信层20发送数据,并从中接收数据,或者,如果DLT协议更改为BPI必须以某种方式更改(例如,如果更改了能够被写入DLT的对象类型,则BPI也可能更改,这可能要求应用程序功能层30提供不同的对象类型的数据),则至多只有最小限度的更改。然而,显然,即使在可能需要对应用程序功能层30进行改变的情况下,这也将是最小的改变,并且应用程序功能层30的核心方面可以保持不变。同样,如果应用程序开发者220决定开始使用其他DLT(例如,因为最初选择的DLT现在不稳定或存在安全漏洞,或者因为已经启动了具有改进的速度和/或效率的新DLT),情况也相同。如果Overledger SDK支持该不同的DLT,则应用程序开发者220可以简单地更新BPI,以提供用于与该不同的DLT交互的命令,然后应用程序230的应用程序功能层30可以利用该命令。再次,可以看到,应用程序230的应用程序功能层30所需的更改可以只是最小限度或为零,因为应用程序功能层30不是被开发为与一个特定DLT和DLT所需要的特定协议进行交互——反而,该应用程序功能层30被开发为来执行应用程序的功能需求,并在需要某种形式的区块链交互的任何时候利用BPI。

[0078] 通过考虑应用程序开发者220开发供其他“客户端”应用程序开发者利用的服务提供的场景,可以进一步理解使用Overledger通信层20将应用程序功能层30与DLT层10分离的好处。该服务提供可以与其他应用程序开发者可能希望使用的任何类型的区块链服务有关,例如,可实现将特定类型的信息写入区块链的服务(稍后将在“消息传送方面”一节中进行详细介绍)和/或可实现跨账本交易的服务(稍后将在“跨账本交易方面”一节中进行详细说明)和/或可实现从DLT中读取特定类型信息的服务等。

[0079] 图3示出了示例系统300,以说明可由“客户端”应用程序开发者使用的服务提供。系统300包括SDK提供者210(先前描述),服务方案开发者310、客户端应用程序开发者320、客户端应用程序330、第一DLT网络240(先前描述)和第二DLT网络250(先前描述)。

[0080] 图4显示了示例的方法步骤,这些示例的方法步骤例示了服务方案开发者可能如何利用Overledger SDK。

[0081] 在步骤S410中,服务方案开发者310可以从SDK提供者210获得Overledger SDK(例如,通过从SDK提供者210或从任何其Overledger SDK托管实体处下载)。

[0082] 在步骤S420中,服务方案开发者310可以为正在提供的服务开发BPI并将其适当地封装(例如,在BPI配置文件中,尽管可以替代地使用任何其他合适的替代方案)。BPI将指定可以由客户端应用程序330的应用程序功能层30使用来与一个或多个DLT进行交互的功能、命令等,如前所述。

[0083] 在步骤S430中,客户端应用程序开发者320可以获得Overledger SDK和BPI配置文件(例如,通过从服务方案开发者310或从任何其他合适的托管实体处下载)。

[0084] 在步骤S440中,客户端应用程序开发者320开发其客户端应用程序330的应用程序功能层30。在应用程序功能层30需要以任何方式与DLT进行交互的情况下,应用程序功能层30可以简单适当地利用BPI中的命令或功能,以便将区块链请求传送至Overledger通信层

20,该区块链请求与要在一个或多个区块链上执行的区块链操作相关。因此,可以看出,由服务方案开发者310提供的Overledger SDK和BPI将应用程序功能层30与DLT层10分离。因此,可以将Overledger SDK视为执行Overledger通信层20的功能,并且BPI定义了应用程序功能层30如何与Overledger通信层20进行接口连接,以执行由服务方案开发者的特定服务。

[0085] 因此,如果客户端应用程序开发者320在以后决定希望更改其用于客户端应用程序330的DLT(例如,客户端应用程序330的应用程序功能层30最初可能已配置为将BPI功能和命令用于与第一DLT 12进行交互,但是后来反而决定要使用第二DLT 14),仅需要更改应用程序功能层30使用的特定BPI功能和命令(假设Overledger SDK和BPI支持第一DLT 12和第二DLT 14)。同样,如果Overledger SDK所支持的任何DLT的协议有某些更改,可以更Overledger SDK,并且如有必要,服务方案开发者310可以因此更改BPI。结果,客户端应用程序开发者最多仅需要更改客户端应用程序330的应用程序功能层30使用的特定BPI功能和命令,而不必更改客户端应用程序的应用程序功能层30的任何核心方面。

[0086] 可选地,在服务方案开发者310已经创建了BPI配置文件之后,可以使用标准的公私密钥加密技术对其进行签名,以便客户端应用程序开发者320可以验证其获得的BPI配置文件是有效的。

[0087] Overledger SDK可以包括能使特定类型的DLT交互发生的许多不同的功能。这些特定功能中的两个——“消息传递方面”和“跨账本交易”方面——会在下面更详细地描述。

[0088] 可以理解的是,尽管如上将应用程序功能层30和Overledger通信层20都描述为“层”,但是其可以可替代地被视为软件模块或软件包。例如,它们可以是应用程序230或客户端应用程序330内的两个模块。或者,它们每个都可以位于不同的应用程序内,并可能在不同的电子设备上运行。例如,应用程序功能层30可以是在客户端电子设备上运行的软件模块或软件包,并且Overledger通信层20可以是在服务方案电子设备上运行的不同的软件模块或软件包,其中,这两个层通过任何合适的通信接口按照BPI进行通信。此外,Overledger通信层20可以被视为被配置为能够在任何一个或多个不同的DLT上执行特定的区块链操作的软件模块或软件包,其中,服务方案开发者310允许客户端应用程序开发者下载Overledger通信层20并通过BPI使用Overledger通信层20,或者通过BPI远程利用该Overledger通信层20。

[0089] 消息传递方面

[0090] 发明人已经认识到,在不一定旨在存储这些信息的区块链上维护不可变的信息记录通常可能是有用的。在下文中,使用了术语“消息”,但是应该理解,该“消息”涵盖了希望使用区块链保留其记录的任何类型的信息。

[0091] 通过使用区块链来保留消息的记录,可以保留消息的有效不可变的记录,以便消息的内容将来可以验证。区块链可以接受每个区块所包含的交易中的某些类型的信息,但是这些信息的类型通常是很规范的。例如,已经为其他特定目的开发了其他区块链,并且将来可能还会为更进一步的特定目的开发更多的区块链。在开发Overledger通信层20的过程中,发明人意识到,能够将某些类型的信息存储在不是为此目的而设计的区块链上,这可能是非常有用的。一个特定的示例用途是,如稍后所述的,作为维护一个或多个区块链上的交易顺序的记录的一部分,尽管还有许多其他潜在的用途和好处。

[0092] 在意识到将消息存储在并非为此目的而设计的区块链上的潜在好处后,发明人研究了现有区块链架构内的交易的细节。已经认识到,许多区块链架构包括在交易内的一个字段(field),可以在其中添加非必需的数据。例如,以太坊有一个用于脚本字节码(Script bytecode)的字段,而Ripple有一个“备忘录(Memos)”字段。这些字段通常可以称为“非必需数据字段”。但是,这些字段中的每一个在大小上相对有限,并且因此在用途上相对有限。

[0093] 发明人确定的解决方案是使用这样的字段来存储至少部分地基于消息并且唯一指示消息的值(例如消息的哈希或加密),然后将消息存储在其他位置(链下)。唯一指示消息的值可以充当指向链下存储的完整消息的指针(例如,哈希指针)。通过在区块链上存储唯一表示消息的值,可以随时通过简单地基于链下存储的消息生成测试值并将其与存储在区块链上的值进行核对来检查链下存储的消息的完整性。由于测试值的生成方式与存储在区块链上的值的生成方式相同(例如,使用相同的哈希方法),如果测试值与存储在区块链上的值匹配,则链下存储的消息具有完整性(即,未被更改)。如果它们不匹配,则链下存储的消息已被以某种方式篡改。这是因为区块链的内容是有效不可变的,因此无法以任何方式更改存储在区块链上的值。

[0094] 技术人员将理解,可以以任何合适的方式来生成唯一指示消息的值。例如,可以通过使用诸如SHA-2(例如SHA-256、SHA-512等)或SHA-3等哈希算法对消息的内容进行哈希处理来生成该值,以使该唯一指示消息的值是一个哈希,其在区块链中充当哈希指针。可替代地,可以通过使用任何合适的加密算法对消息的内容进行加密来生成该值。可以基于要存储该值的特定区块链的要求和/或需要满足的任何特定密码需求来选择可以生成唯一指示消息的值的特定方式(例如,由算法提供的完整性级别等)。例如,要在其中存储值的交易字段可能具有特定的最大长度,该长度排除了某些哈希算法,因为其产生的哈希可能太长。Overledger SDK可以包括适用于它支持的DLT的特定哈希和/或加密算法,使得应用程序功能层可以简单地将消息传递到Overledger通信层20,从而该Overledger通信层20可以适当地处理该消息,以便将消息的记录存储在DLT上。然后,应用程序可以将消息的副本存储在链下其他位置(例如,该应用程序正在运行的设备上的本地存储器中,或在例如远程数据库或云存储等远程存储器中)。

[0095] 如前所述,上述消息传递方面可能有许多潜在用途。下面参考图5描述一种特定的、非限制性的示例用例。

[0096] 图5示出了系统500,包括服务方案开发者310、客户端应用程序开发者320,客户端应用程序后端510、第一客户端应用程序520、第二客户端应用程序530和第一DLT网络240。服务方案开发者310可以为消息服务创建BPI,该消息服务方案用于通过使用上述消息传送方面在一个或多个DLT上记录消息的BPI功能。客户端应用程序开发者320然后可以使用服务方案开发者310的Overledger SDK和BPI配置文件开发消息传送应用程序,该应用程序允许使用该程序的用户彼此安全发送消息。第一客户端应用程序520和第二客户端应用程序530可以分别安装在第一用户和第二用户的电子设备上,例如在移动电子设备(诸如智能手机、平板电脑或笔记本电脑)上或在台式计算机上运行的相同的客户端应用程序。第一客户端应用程序520和第二客户端应用程序530可以作为设备上的软件安装或作为基于网络浏览器的应用程序等在第一用户和第二用户的电子设备上运行。

[0097] 如果第一用户希望将消息传达给第二用户,其可以将该消息和第二用户的标识

(该标识可以采用任何合适的形式,例如该客户端应用程序的第二用户的用户名)输入至第一客户端应用程序520。客户端应用程序的应用程序功能层30可以配置为将消息传递到第一客户端应用程序520的Overledger通信层20,以通过使用由服务方案开发者310生成的BPI中的适当功能/命令将该消息添加到第一DLT 12中。然后,Overledger通信层20可以根据用于与第一DLT 12交互的SDK方法对消息进行哈希处理,为包括该消息的哈希的第一DLT 12构造适当的交易,并将该交易传送给第一DLT网络240。因此,该消息的哈希将包括在第一DLT 12上。第一客户端应用程序520的Overledger通信层20还可以可选地被配置为当确定已经将交易成功地添加到第一DLT 12时,给应用程序功能层30返回一个通知。整个消息的副本和该消息的哈希也可以传送到客户端应用程序后端510(其可以是例如由客户端应用程序开发者320提供的服务器、数据库或基于云的存储系统,以接收、存储和检索消息)例如,通过应用程序功能层30从Overledger通信层20接收消息的哈希,然后将其与消息一起传送到客户端应用程序后端510。可选地可以使用任何适当的加密技术来确保第一客户端应用程序520与客户端应用程序后端510之间的通信。

[0098] 然后,第二用户可能会意识到已收到消息。这种情况可能以许多不同的方式发生,例如第二用户标识符可能已经是第二用户的DLT公钥或DLT钱包,或者可能已被第一客户端应用程序520用于获取第二用户的DLT公钥或DLT钱包,并且第一DLT 12上的交易可以将第二用户的公共密钥或DLT钱包标识为交易的接收者。在这种情况下,第二客户端应用程序530可以注意到第一DLT 12上的新交易标识了第二用户(如果第二客户端应用程序530配置为监视添加到第一DLT 12上的每个新区块的内容以寻找标识第二用户DLT公钥或DLT钱包的任何新交易)。或者,第三方服务可以执行此监视过程并通知第二客户端应用程序530。然后,第二客户端应用程序530的应用程序功能层30可以使用适当的BPI“读取”功能从第一DLT 12获得消息的哈希。

[0099] 一旦第二客户端应用程序530具有消息哈希的副本,则应用程序功能层30可以将其与某种形式的认证一起传送给客户端应用程序后端510,以便向客户端应用程序后端510认证该第二用户(例如,使用第二客户端应用程序530和客户端应用程序后端510已经建立的任何形式的认证过程,或通过提供由第二用户的DLT私钥所签名的数字签名,客户端应用程序后端510可以通过使用与存储在第一DLT网络240上的消息哈希相关联的公钥来检查)。如果通过了认证,则可以将与消息哈希相对应的完整消息返回给第二客户端应用程序530。然后,第二客户端应用程序530可以通过使返回的消息经过与第一客户端应用程序520所使用的哈希算法相同的哈希算法(例如,通过使用适当的“验证消息”BPI功能,或以服务方案开发者310和/或客户端应用程序开发者320认为合适的任何其它方式)来创建测试哈希,然后将该测试哈希与从第一DLT 12检索到的消息哈希进行比较,以验证该返回的消息的完整性。

[0100] 此特定示例过程的许多好处可能是立即显而易见的。首先,可以将消息的内容从第一用户安全地传递到第二用户,而无需执行复杂且通常难以可靠实现的技术,例如加密电子邮件或需要在两位用户之间达成一致的密码的加密文档。此外,第二用户可以确保客户端应用程序后端510没有以任何方式篡改消息,因为否则测试哈希将与存储在第一DLT 12上的消息哈希不匹配。此外,如果在将来的任何时候第一用户和第二用户之间存在关于消息内容的某种形式的争议,则完全可以使用存储在第一DLT 12上的消息哈希来解决该争

议。

[0101] 第二用户同样可以类似的方式回复消息。

[0102] 可以理解的是,这仅是本公开的消息传递方面的一个特定示例,而非全部示例。此外,上述特定用例的许多替代实施方式是可能的。例如,可替代地,可以由服务方案开发者310来提供应用程序后端,并且BPI可以被简单地配置为接收消息,并且Overledger通信层20负责将消息哈希放到第一DLT 12上,并且还将消息和消息哈希传递到客户端应用程序后端510。在这种情况下,客户端应用程序的应用程序功能层30可能仅与其他功能有关,例如GUI和客户端应用程序提供的其他功能。此外,应用程序功能层30可以利用BPI提供的功能或命令来将消息哈希放到不止一个而是许多DLT上,从而为该消息哈希提供进一步的冗余度(Redundancy)和备份。

[0103] 交易排序方面

[0104] 可靠的区块链操作的一个重要需求是交易排序。

[0105] 发明人已经认识到,可靠的交易排序对所有类型的区块链都非常重要。例如,可以理解的是,在各种互联网和电信领域和协议中,错误排序的消息传递可能容易遭受安全性和稳定性攻击。例如,TCP/IP数据包的排序和同步可以以各种方式影响系统的安全性,例如消息脱离了正确顺序或消息集不完整,导致系统的安全性在完整性或有效性方面受到损害。因此,发明人已经认识到,例如在根据上述消息传递方面被用来承载消息时,以及当被用于任何其他目的时,维护区块链内正确的交易顺序对于实现基于一个或多个DLT的安全、可靠和防欺诈的系统具有重要意义。

[0106] 发明人已经识别出在DLT中维护正确交易顺序面临的两个挑战:

[0107] 1.如果区块链分叉,则在该分叉的短分支(Shorter prong)上一笔或多笔交易可能会从区块链中丢失。技术人员容易理解DLT产生分叉的可能性,以及在这种情况下DLT的行为特征。但是,值得注意的是,传统上,当交易因分叉而丢失时,之后可以在已经识别到该交易已丢失的情况下,由节点参与的网络将其附加到该分叉的长分支(Longer prong)中。

[0108] 图6示出了区块链中的分叉的示例表示。交易1是区块B1的一部分,区块B1出现在区块链的“分支1”上。但是,由于分叉,区块链在长分支“分支2”上延续,因此交易1实际上已经从区块链中丢失了。交易2是区块B3的一部分,出现在“分支2”上。之后,交易1作为“分支2”上区块B8的一部分被附加到该区块链。因此,交易1在区块链中的出现时间晚于交易2。但是,如果交易1先于交易2很重要(例如,它们与连续的消息有关),这可能会导致使用该区块链的应用程序中的安全性或稳定性问题,或者可能导致欺诈发生。

[0109] 2.如前所述,在某些情况下,最为理想的方式是应用程序使用两个或多个不同的DLT。作为其中的一部分,有时可能需要执行跨账本交易(XLT)以便于将数据(例如,一条消息(如前所述)或存储在DLT上的另一种其他类型的信息)从一个DLT移动到另一个DLT。但是,虽然可以开发每个区块链来维护区块链上不同区块之间的顺序(例如,使用时间戳),但要可靠地辨别出现在不同区块链上的交易之间的顺序可能非常困难,因为每个DLT可能利用不同的时间戳流程和/或其用于时间戳的时钟可能不同步。因此,当利用两个或多于两个(多个)DLT时,确定和维护出现在DLT上的交易顺序对于使用该DLT的应用程序的安全性和稳定性可能非常重要。

[0110] 图7示出了发明人开发的使用“验证集”维护交易顺序的过程的示例方法步骤。图8

示出了两个DLT和验证集的可视化示例,以帮助理解图7的方法步骤。

[0111] 在步骤S710中,监视第一DLT 12和第二DLT 14以识别何时将新的感兴趣的交易包括在被添加到第一DLT 12或第二DLT 14的区块中。当将每个新块添加到DLT时,可以通过读取每个新区块的内容来监视第一DLT 12和第二DLT 14。构成感兴趣交易的相关性准则取决于此方法的特定实现。例如,上面公开的BPI可以配置为使得通过利用BPI放入DLT上的任何交易都包括一些特定的标识符,该标识符将其与通过任何其它方式(例如,通过其他实体而非利用BPI)放入DLT的交易区分开来。特定标识符可以是,例如,用于交易的公钥,或交易内的某些其他类型的信息。因此,相关性准则可以是BPI包括的标识符,以使得包括在使用服务方案开发者310提供的服务的第一DLT 12或第二DLT 14中的任何交易(例如,由第一客户端应用程序520或第二客户端应用程序530添加的交易,或者实际上是由服务方案开发者操作以在必要时将交易放在DLT上的某些服务应用程序)应该符合该相关性准则,并且可以被识别为相关。如果第一DLT 12和/或第二DLT 14是许多不同实体可以利用的通用DLT,则这可能特别有用,但是服务方案开发者310需要识别与其服务有关的交易以确保那些交易的正确排序,从而保障其服务方案的安全性和稳定性。

[0112] 在图8中,交易A、交易C、交易E和交易O都是感兴趣的交易。标记有“交易X”的交易为非感兴趣的交易。

[0113] 当新区块被添加到第一DLT 12或第二DLT 14中并且在步骤S710中被识别出它包括相关交易时,在步骤S720中,可以至少部分地基于交易内容来确定每个识别出的交易的交易标识符(例如,哈希)。然后,将该标识符添加到最新的或“进行中”的验证集中。验证集是相关交易的链下记录的一部分,存储在数据存储中,例如任何合适的存储、数据库、数据阵列、云存储等。交易标识符可以充当指向DLT中的相关交易的指针(例如,哈希指针)。每个新的交易标识符都按照其被检测到的顺序添加到“进行中”的验证集中。这样,识别出的交易的记录指示交易的相对顺序,而不管其是否已出现在第一DLT 12或第二DLT 14中。结果,可以看出,通过监测两个DLT,并且一旦被添加到第一DLT 12或第二DLT 14中就对每个相关交易添加交易标识符,可以跨一个或多个不同的DLT记录交易的相对顺序。技术人员可以认识到的是,可以使用保留相关交易的顺序的任何合适的数据存储技术来存储“进行中”的验证集。一旦新的相关交易已经被添加到“进行中”的验证集中,则该过程可以返回到步骤S710以继续监测第一DLT 12和第二DLT 14。

[0114] 在特定时间,该过程可以进行到步骤S730以开始将“进行中”的验证集(例如,至少部分地基于“进行中”的验证集的内容的哈希或加密)的记录(例如,验证集标识符)存储至第一DLT 12和第二DLT 14的过程。该过程进行到步骤S730的特定时间取决于实现方式,并且可以是例如自从将最近的先前验证集的记录添加到DLT以来已经花费了特定时间来监测该DLT之后,或者在将特定数量的相关交易添加到“进行中”的验证集之后,或者自从将最近的先前验证集的记录被添加到DLT以来已将特定数量的区块添加到第一DLT 12或第二DLT 14中之后等。

[0115] 在步骤S730中,确定在第一DLT 12和第二DLT 14上是否都存在最近的先前验证集的内容的记录(例如,哈希或加密)(例如,检查在两个DLT中都没有导致该记录丢失的分叉)。如果记录确实出现在两个DLT中(图7中的“是”),则该过程进行到步骤S750。如果记录在两个DLT中均未出现(图7中的“否”),则该过程进行到步骤S740。该检查可以通过在两个

DLT中搜索最近的先前验证集的记录来执行。

[0116] 在步骤S740中,可以将丢失的记录重放到丢失了该记录的DLT上(实质上,将记录首先放入DLT上的交易中——例如,使用上面的“消息传递方面”——被重新广播并添加到DLT中)。但是,在重放丢失的记录之前,可以首先确定在最近的先前验证集中识别出的任何交易是否也从其应该出现的相应DLT中丢失了。这可以通过使用存储在交易记录中的最近的验证集中的交易标识符的指针性质(例如,哈希指针)来快速确定。然后,在重放最近的先前验证集的丢失记录之前,可以将任何丢失的交易重放到相关的DLT上。尽管任何重放的交易现在可能会比该丢失的交易本应该在其之前发生的交易(如果没有丢失)在DLT中出现得更晚,验证集也仍将维护正确的交易顺序的记录。因此,仍然可以将丢失的验证集记录以及与之关联的任何丢失的交易重新添加回DLT,而不会丢失发生交易的正确顺序。从下面对图8和9的讨论中可以更充分地理解该步骤的细节。

[0117] 在步骤S750中,确定在“进行中”的验证集中识别出的所有交易是否仍出现在相关的DLT中(例如,检查没有由于DLT中的一个或两个分叉而丢失)。如果其的确仍然全部存在于相关的DLT中(图7中的“是”),则该过程进行到步骤S770。如果其没有全部出现在相关的DLT中(图7中的“否”),则该过程进行到步骤S760。该检查可以借助于“进行中”的验证集中的转换标识符的指针性质来执行。

[0118] 在步骤S760中,可以将被识别为当前丢失的交易重放到其应出现的相应DLT上(实质上,重新广播并添加到该DLT中)。尽管任何重放的交易现在可能会比该丢失的交易本应该在其之前发生的交易(如果没有丢失)在DLT中出现得更晚,“进行中”的验证集也仍将维护正确的交易顺序的记录。因此,仍然可以将丢失的交易添加回DLT,而不会丢失发生交易的正确顺序。

[0119] 在步骤S770中,至少部分地基于“进行中”的验证集的内容来确定“进行中”的验证集的记录(例如,至少部分地基于验证集的内容来确定诸如哈希或加密的验证集标识符)并将其添加到第一DLT 12和第二DLT 14中。可以通过利用上文“消息传递方面”中描述的技术将记录添加到两个DLT中。因此,可以看出,通过使用上述“消息传递方面”,可以将“进行中”的验证集的内容的记录添加到两个DLT中,从而提供该验证集的内容的有效不可变的记录,通过使用该有效不可变的记录,就可以实现相关交易的记录的数据完整性,从而提高安全性和可靠性。然后,该过程可以返回到步骤S710,以继续监测DLT,并开始处理下一个“进行中”的验证集。

[0120] 返回图8,可以看到存储在链下数据存储(可以是任何合适类型的数据存储、数据库、云存储或任何其他类型的存储器)中的相关交易840的记录,包括第(n-1)验证集810、第n验证集820和第(n+1)验证集830。第(n-1)验证集810包括交易Y的哈希和交易Z的哈希,它们是第一DLT 12或第二DLT 14上的早期的交易,并且为简单起见,在图8中未另外表示。第n验证集820包括交易A的哈希822'、交易O的哈希824'和交易C的哈希826'。交易A822和交易C826出现在第一DLT 12中,并且交易O824出现在第二DLT 14中。它们出现在第n验证集820中的顺序表示在交易A822之后,但在交易C826之前检测到交易O824。第(n+1)验证集830包括交易E的哈希832',并且在该示例中可以被视为“进行中”的验证集。

[0121] 可以看出,按照上述步骤S770,第(n-1)验证集的哈希810'和第n验证集的哈希820'都已预先被添加到两个DLT中。当第n验证集820是“进行中”的验证集时,可以理解的

是,当来到上述步骤S730时,第(n-1)验证集的哈希810'存在于两个DLT中,因此该过程可以直接进行至步骤S750。同样地,在步骤S750中,交易A822、交易O 824和交易C 826仍然都出现在其相关DLT中,因此该过程随后可以直接进行到步骤S770,以将第n验证集的哈希820'添加到两个DLT。

[0122] 作为该过程的结果,可以看到,借助于交易840的记录中的验证集,保留了出现在两个区块链中的交易顺序的链下记录。该链下记录指示了其所关联的交易的相对顺序。通过还使用上述“消息传递方面”在两个DLT上存储每个验证集的哈希指针,可以在DLT上保留每个验证集的记录,这在维护验证集的链下记录的数据完整性和安全性方面以及在第一DLT 12和/或第二DLT 14中发生分叉的情况下特别有用,如下所述。

[0123] 图9示出了两个DLT和验证集的另一可视化示例,以帮助理解图7的方法步骤,特别是步骤S730至S760。尽管图9与图8相似,但是图9包括第一DLT 12中的一个分叉的表示,这导致两笔交易和一个验证集的记录已从第一DLT 12中丢失。

[0124] 在图9中,第一DLT 12在包含交易C 826之前分叉。交易C 826,第n验证集的哈希820'和交易E 832全部出现在该分叉的“分支1”上。但是,在此示例中,是分叉的“分支2”随着第一DLT 12向前移动,并且“分支1”的内容实际上丢失了(至少是暂时丢失)。在将第n验证集的哈希820'添加到DLT之后,就开始处理第(n+1)验证集930,可以看出,第(n+1)验证集930按顺序包括交易E的哈希832'和交易G的哈希934'。在图9中标识的点P处,图7的过程进行到步骤S730,此时,标识第n验证集的哈希820'和交易C 826不再出现在第一DLT 12中(因为它们都出现在“分支1”中,这是已丢失的分叉的分支)。结果,在步骤S740中,交易C被重放(在图9中由附图标记926标识,尽管可以理解的是原始的交易C 826和重放的交易C 926的内容将是相同的),然后第n验证集的哈希820'被重放(在图9中由附图标记920'标识,尽管可以理解的是,第n验证集原始的哈希820'将与重放的验证集的哈希920'相同)。随后,在步骤S750中,确定交易E 832(在“进行中”的第(n+1)验证集930中识别)不再出现在第一DLT 12中,因为它被包括在“分支1”中。因此,在步骤S760中,交易E被重放(在图9中由附图标记932标识,尽管可以理解的是,原始的交易E 832和重放的交易E 932的内容将是相同的)。之后,在步骤S770中将验证第(n+1)验证集的哈希930'添加到两个DLT。因此,可以看出,即使交易C 826、第n验证集的哈希820'和交易E832都从第一DLT 12丢失了,也仍可以在交易G 934(最初发生在交易C 826和交易E 832之后)之后被加回到第一DLT 12上,而不会丢失交易真实顺序的可靠记录,从而确保了系统的安全性和稳定性。因此,尽管在DLT中包含验证集的哈希是可选的(或者,可以简单地保留交易的链下记录,而无需在DLT中存储验证集的任何哈希),但可以理解的是,在DLT上存储每个验证集的记录,不仅确实通过可实现链下验证集的数据完整性来提高安全性和可靠性,而且还可实现快速识别出分叉,并可以将丢失的记录和/或交易添加回相关的DLT。

[0125] 可以理解的是,尽管图9中所示的每个区块仅包括单笔交易,但是实际上每个区块可能将包括许多笔交易。此外,尽管以上描述了使用两个DLT的过程,但是可以理解的是,其可以应用于单个DLT或三个或更多不同的DLT。此外,尽管在上文中描述了验证集和交易的“哈希”,但是可以理解的是,可替代地使用至少部分地基于验证集的内容确定的任何合适的验证集标识符和至少部分地基于验证集的内容的任何合适的交易标识符。

[0126] 此外,尽管在上述过程中,将验证集的哈希添加到两个DLT中,替代地也可以将其

仅添加到DLT中的一个DLT中。即便如此,优选的是将验证集的哈希包含在正在其上运行该过程的两个或多于两个DLT上,因为即使在发生分叉的情况下,在任何时候都应有至少一个最近的先前验证集的记录(两个DLT同时独立地分叉,导致它们都失去最近的先前验证集的哈希是极其不可能的)。这可能对数据完整性以及由此带来的安全性很有用。

[0127] 此外,作为在验证集中存储相关交易的哈希的补充或替代,可以存储相关交易的完整副本。然而,出于数据存储效率的考虑,优选的是存储相关交易的哈希。在一个示例实现中,可以存储每个相关交易的哈希,并且可以存储最近相关交易的完整副本(例如,最近的一个、两个、三个、四个验证集中的交易),这对于重放丢失的交易可能是有用的。通过仅存储最近的相关交易,数据存储需求仍然可以最小化。

[0128] 此外,识别和重放丢失的交易是可选的,尽管优选的是执行这些步骤以确保DLT尽可能是最新的,并且确保DLT上的交易排序尽可能与如在交易840中所识别的实际交易排序相对应。

[0129] 跨账本交易方面

[0130] 为进一步帮助理解上述交易排序方面的重要性,现在将描述通过利用交易排序方面的跨账本交易(XLT)过程示例。

[0131] 在该示例中,最为理想的是将出现在第一DLT 12中的交易的信息传送到第二DLT 14。该信息可以是任何事物,例如哈希消息(如上文“消息传递方面”所述)或任何其他数据。交易可能是将信息从第一所有者转移到第二所有者,或者可能根本没有所有权转移(例如,实体可能只是希望将信息存储在不同的DLT中并有效地将其从先前的DLT中删除)。

[0132] 图10示出了两个DLT的可视化示例,以及构成链下存储的交易1040的记录的相应验证集,其中将执行XLT。交易S1012包括要从第一DLT 12传送到第二DLT 14的信息。第(n-1)验证集1010包括交易S的哈希1012'。可以使用两阶段提交(Two-phase commit)过程来执行该XLT。本领域技术人员应当充分理解两阶段提交过程,但是为了完整起见,下文已对其进行简要描述。首先将用于将交易S中的信息从第一DLT 12传输到第二DLT 14的拟议添加到第一DLT 12(在图10中由“拟议中的XLT(S) 1022”标识)。这有效地锁定了交易S1012,使得它不能在任何其他交易中使用。预期的接收者(在某些情况下可能与要转移的信息的当前所有者是同一实体,如前所述)可以随后在第二DLT 14中指示其准备就绪来接受传送(由“就绪的XLT(S) 1026”标识)。替代地,在一些DLT协议中,可以通过在第二DLT 14中重放“拟议中的XLT(S) 1022”来指示就绪。一旦已经在第二DLT 14上指示就绪,就可以通过添加到两个DLT的提交交易(在图10中由“提交XLT(S) 1032”和“提交XLT(S) 1034”标识)来执行跨账本交易。如技术人员可以理解的,该提交有效地从第一DLT 12移除了交易的项目并将其放入第二DLT 14。

[0133] 为了能够在XLT发生之前安全可靠地执行XLT,而没有欺诈或系统故障的风险,有必要确保在提交交易被放入这两个DLT之前第二DLT 14上出现就绪的指示。此外,优选的是对于就绪的指示总是跟随该拟议,以便可以适当地对两者进行匹配和认证。通过这种方式,可以确保准备就绪的指示实际上并不涉及其他类似但不同的交易(例如,除了在第一DLT 12上的不同的早期交易的内容之外,涉及同一当事方之间的XLT拟议)。还可以确保准备就绪的指示不会欺诈性地更改XLT的某些细节。

[0134] 如前所述,以前在不同DLT上确定正确的交易顺序非常困难,尤其是在一个或多个

DLT中存在分叉的情况下。因此,很难安全、直接、可靠地执行XLT。但是,通过利用上述交易排序方面,即使在一个或多个DLT中存在分叉的情况下,也可以记录相关交易的正确顺序。

[0135] 例如,交易排序方面可用于确保满足XLT准则,例如与就绪的XLT(S) 1026对应的拟议中的XLT(S) 1022(以及可选地,两者仍分别出现在第一DLT 12和第二DLT 14)和拟议中的XLT(S) 1022先于就绪的XLT(S) (即使在发生分叉的情况下)。在这方面,仅在符合XLT准则之后,才可以将提交XLT(S) 1032交易添加到DLT。可选地,XLT准则还可包括以下准则:在一个或两个DLT上都存在一个验证集的哈希,该验证集用于标识拟议中的XLT(S) 1022和就绪的XLT(S) 1026交易,这将确保拟议中的XLT(S) 1022和就绪的XLT(S) 1026交易都存在于各自的DLT中。可以理解的是,尽管在该示例中,拟议中的XLT(S)的哈希1022'和就绪的XLT(S)的哈希1026'出现在同一验证集中,但是它们可以替代地出现在不同的验证集中。如果拟议中的XLT(S)的哈希1022'出现在就绪的XLT(S)的哈希1026'出现的验证集之前的验证集中,则可以确认正确的顺序。如果拟议中的XLT(S)的哈希1022'出现在就绪的XLT(S)的哈希1026'出现的验证集之后的验证集中,则可以确认交易顺序错误,因此XLT不应发生。

[0136] 从图10中可以看出,第n验证集1020包括拟议中的XLT(S)的哈希1022',并且之后是就绪的XLT(S)的哈希1026'(第n验证集1020还包括被认为相关,但与此处讨论的XLT无关的其他交易的哈希)。实际上,构建验证集的实体可能会在DLT中识别许多相关交易,并在验证集中对其进行识别。这些相关交易中的某些交易可能彼此相关,例如与特定XLT相关的交易,而其他交易则不会以任何方式彼此相关,例如与不同的XLT相关,或者该实体认为具有相关性的任何其他类型的DLT交易。第n验证集的哈希1020'出现在两个DLT中,随后是第一DLT 12上的提交XLT(S) 1032和第二DLT 14上的提交XLT(S) 1034(随后在下一个第(n+1)验证集1030中被识别)。

[0137] 可以以许多不同的方式来实现这种类型的XLT。图11示出了一种特定实现的示例,其包括服务方案开发者310,其在该特定示例中已经开发了用于执行XLT的服务应用程序(或软件包/模块) 1110,并且还向客户端应用程序开发者320提供了BPI。因此可以开发其特定的XLT客户端应用程序(替代地,服务方案开发者310可以为自己创建客户端应用程序,在这种情况下,可以不向任何外部开发者提供BPI)。第一客户端应用程序1120和第二客户端应用程序1130是第一实体1140和第二实体1150分别已经从客户端应用程序开发者320获得的XLT客户端应用程序的副本。在该示例中的客户端应用可以有效地实现应用程序功能层30,并且服务应用程序1110可以有效地实现Overledger通信层20。

[0138] 图12示出了图11所示示例系统如何执行XLT的示例步骤。

[0139] 在步骤S1210中,服务应用程序1110从第一客户端应用程序1120接收请求以执行XLT,以将出现在第一DLT 12中的信息项传送到第二DLT 14,以将所有权传送到第二实体1150。可以理解的是,该请求可以包括如BPI中所定义的用于执行XLT的任何必要信息。

[0140] 在步骤S1220中,服务应用程序1110可以将拟议中的XLT交易广播到第一DLT网络240,以添加到第一DLT 12。

[0141] 在步骤S1230中,服务应用程序1110然后可以就绪的XLT交易广播到第二DLT网络250,以添加到第二DLT 14。可选地,服务应用程序1110可以已经将执行XLT的意图传达给第二客户端应用程序1130,然后仅在从第二客户端应用程序1130接收到交易应当发生的确认之后才执行步骤S1220或S1230。

[0142] 一直以来,服务应用程序1110执行上述交易排序方面,并将指示相关交易顺序的相关交易记录存储在数据存储1115中。

[0143] 在步骤S1240中,服务应用程序1110可以检查在所存储的交易记录中识别出的拟议中的XLT交易是在对应的就绪的XLT交易之前发生的,并且可选地,相关验证集的哈希已经出现在一个或两个DLT中。

[0144] 一旦通过了该检查,则在步骤S1250中,服务应用程序1110将提交XLT交易广播到第一DLT网络240和第二DLT网络250,以添加到第一DLT和第二DLT。可选地,服务应用程序1110可以将成功的XLT通知给第一客户端应用程序1120和/或第二客户端应用程序1130(可以在提交XLT交易已经出现在两个DLT上之后,或者在识别出提交XLT交易的验证集的哈希已包含在一个或两个DLT中之后执行)。在未通过检查的情况下,在一段时间之后,该XLT过程可能超时,并且可以向第一客户端应用程序1120和/或第二客户端应用程序1130通知失败。

[0145] 因此,可以看出,通过执行上述排序方面,然后仅在拟议中的XLT交易和相应的就绪的XLT交易已在验证集中以正确的顺序进行验证之后,才将提交交易发布到DLT,服务应用程序1110可以确保安全、稳定地执行XLT。

[0146] 可以理解的是,这仅仅是一个特定的、非限制性的示例。替代地,第一客户端应用程序1120本身可以将拟议中的XLT交易广播到第一DLT网络240,和/或第二客户端应用程序1130本身可以将就绪的XLT交易广播到第二DLT网络250,这取决于由服务方案开发者310提供的特定BPI和/或由客户端应用程序开发者320提供的客户端应用程序的实现。在这种情况下,服务应用程序1110可以简单地执行上述交易排序方面,然后当在相关DLT中识别了拟议中的XLT交易和相应的就绪的XLT交易时,执行步骤S1240和S1250。此外,XLT可能仅涉及一个实体,因为其可能只是在所有权未发生任何变化的情况下将其信息从一个DLT转移到另一个DLT。

[0147] 可以理解的是,Overledger SDK可以包括用于执行交易排序方面以及可选地还包括XLT方面的规则和方法。这样,Overledger SDK提供了工具,服务方案开发者310可以通过这些工具在其选择的Overledger SDK支持的任何DLT上实现利用了交易排序方面(可选的还有XLT方面)的任何服务应用程序/模块。然后,可以创建自己选择的相应BPI,以允许客户端应用程序开发者320利用服务方案的任何合适功能,从而在执行排序方面与DLT进行的交互可以由Overledger通信层20执行,并且可以与客户端应用程序上任何应用程序功能层30的操作分离。因此,客户端应用程序可以利用交易排序方面提供的安全性和稳定性优势,同时仍可以与直接与DLT进行交互分离。

[0148] 结果,可以看出,通过提供一个包括了执行交易排序方面的规则和方法的Overledger SDK,为交易排序方面的技术优势提供了巨大的灵活性和机会,可被各种不同实体以不同的方式利用,同时仍将区块链交互与应用程序功能层30分离。

[0149] 技术人员可以容易地理解的是,可以在不脱离本公开的范围的情况下对本公开的上述方面进行各种改变或修改。

[0150] 尽管图2、3、5和11中箭头所示的所有接口都显示了每个不同实体和模块之间的直接连接,但是可以理解的是,作为这些接口的一部分,可以有任意数量的中间实体或模块,例如通信路由器等。此外,每个接口可以根据任何适当的通信架构和协议在每个实体和模

块之间承载数据/通信。

[0151] 上文所有描述的本公开的方面可以通过软件、硬件或软件和硬件的组合来实现。例如,交易排序方面和/或XLT方面和/或Overledger消息传递层可以由包括计算机可读代码的软件来实现,该软件在任何电子设备(例如,任何标准计算设备,诸如服务器、台式计算机或笔记本电脑等)中的一个或多个处理器(例如一个或多个微处理器或可编程逻辑器件)上执行时,都可以执行上述功能。该软件可以存储在任何合适的计算机可读介质上,例如,非暂时性计算机可读介质,诸如只读存储器、随机存取存储器、CD-ROM、DVD、蓝光、磁带、硬盘驱动器、固态驱动器和光盘驱动器。该计算机可读介质可以分布在与网络耦合的计算机系统上,从而以分布式方式存储和执行计算机可读指令。

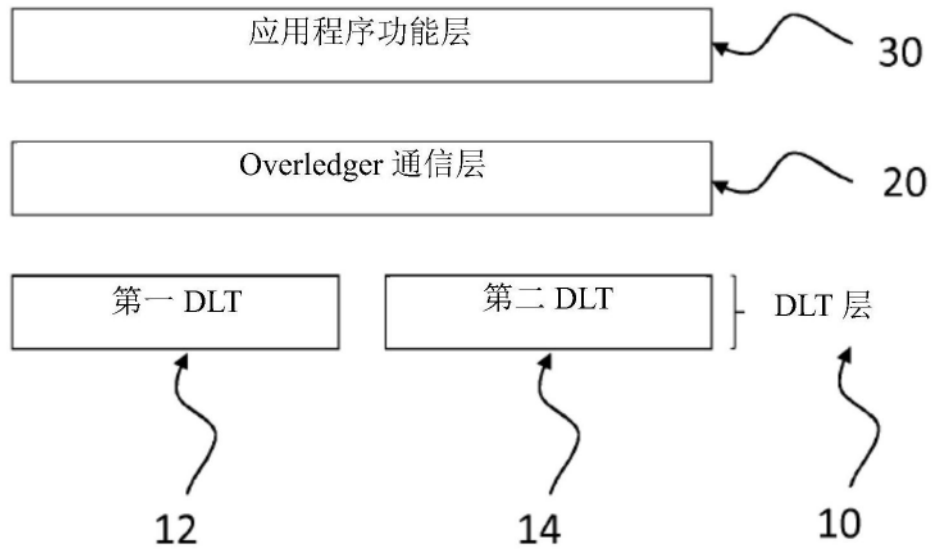


图1

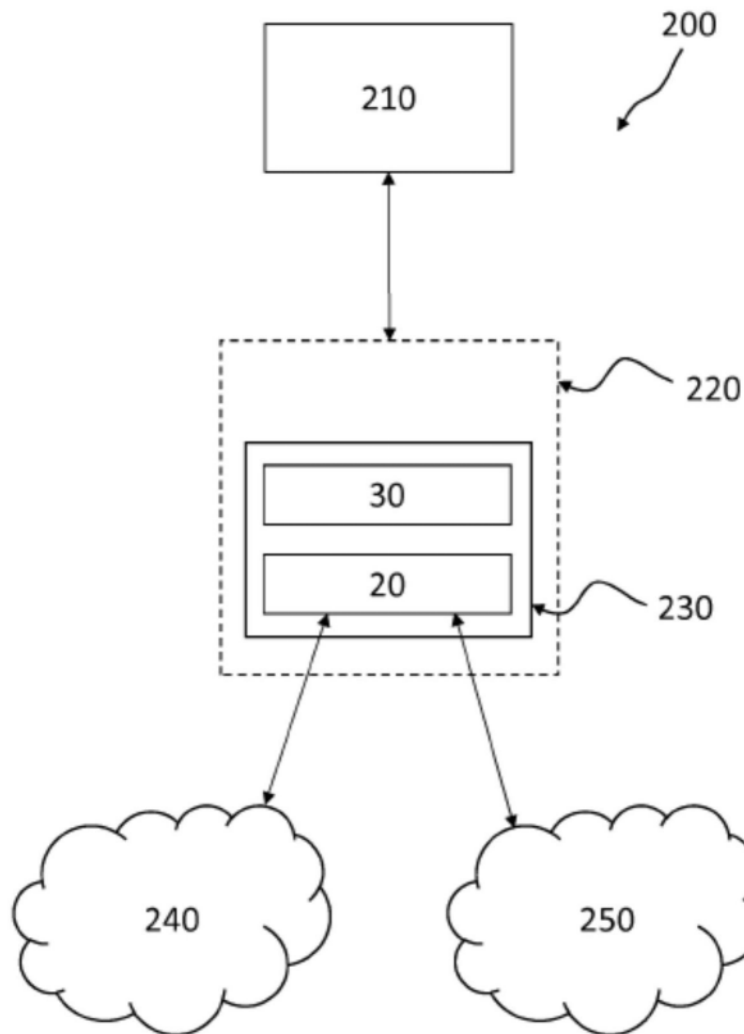


图2

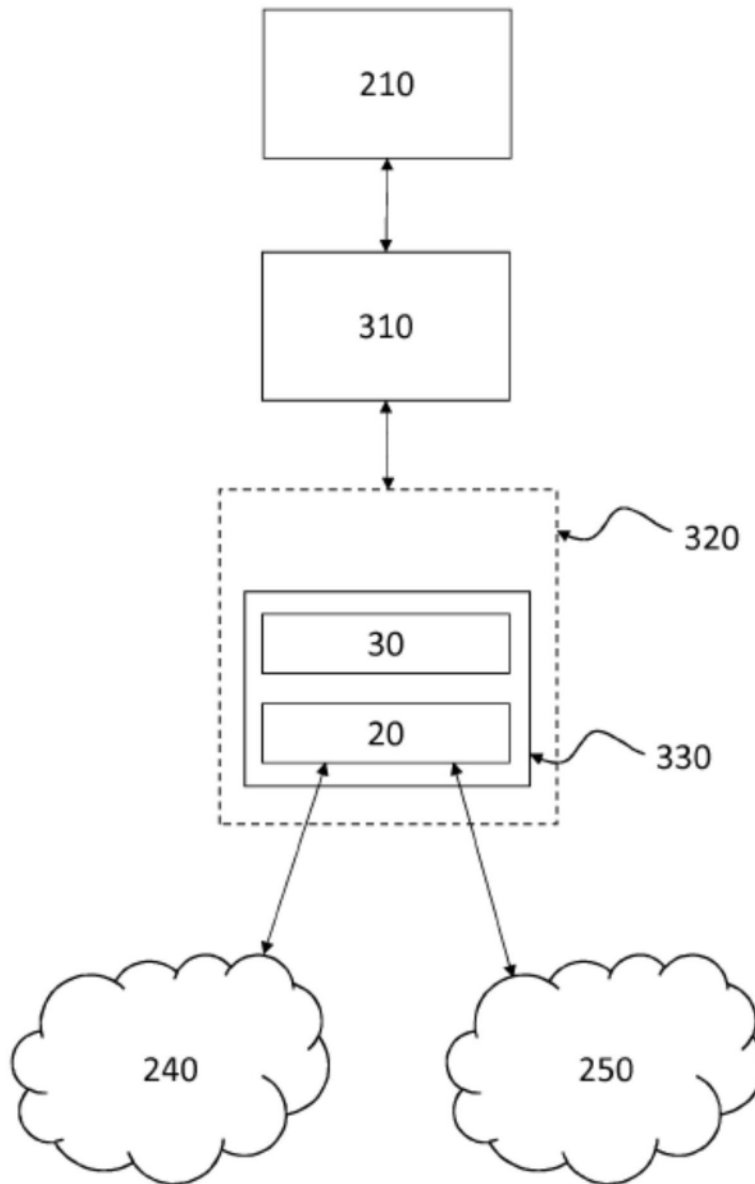


图3

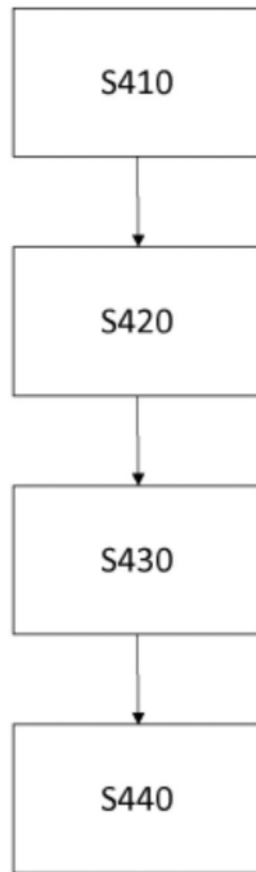


图4

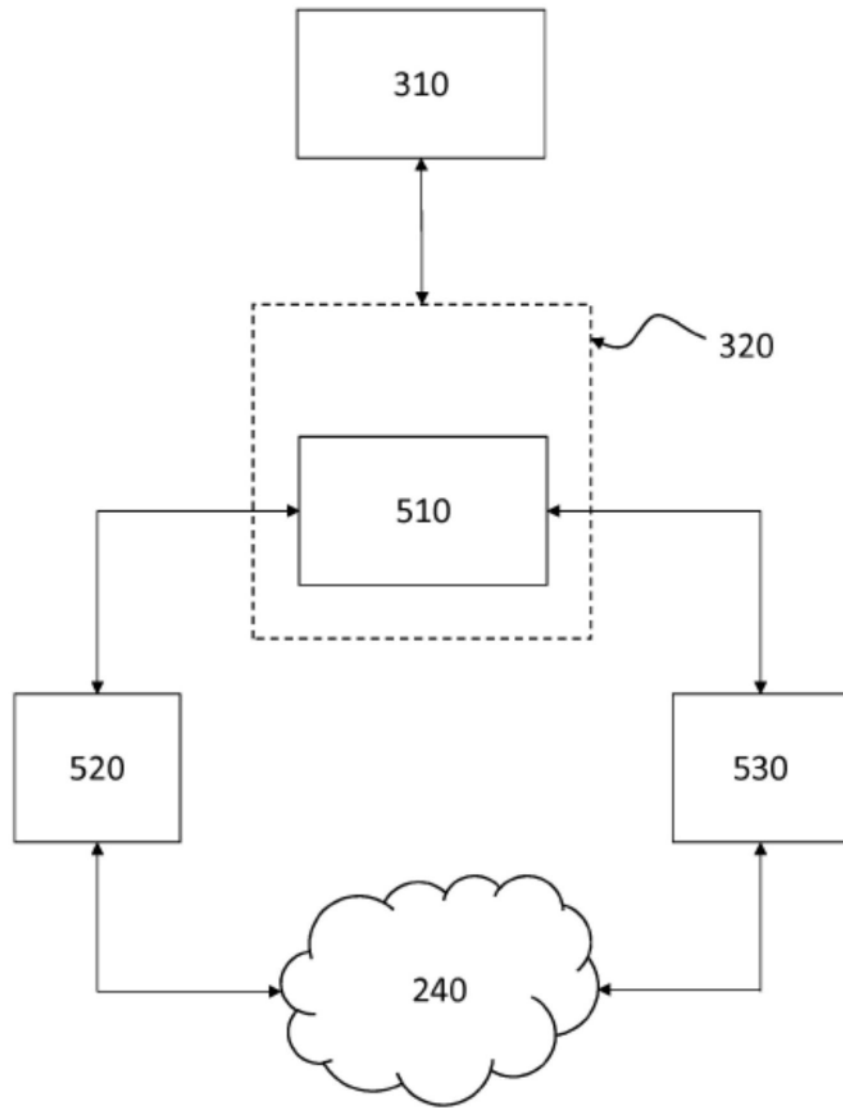


图5

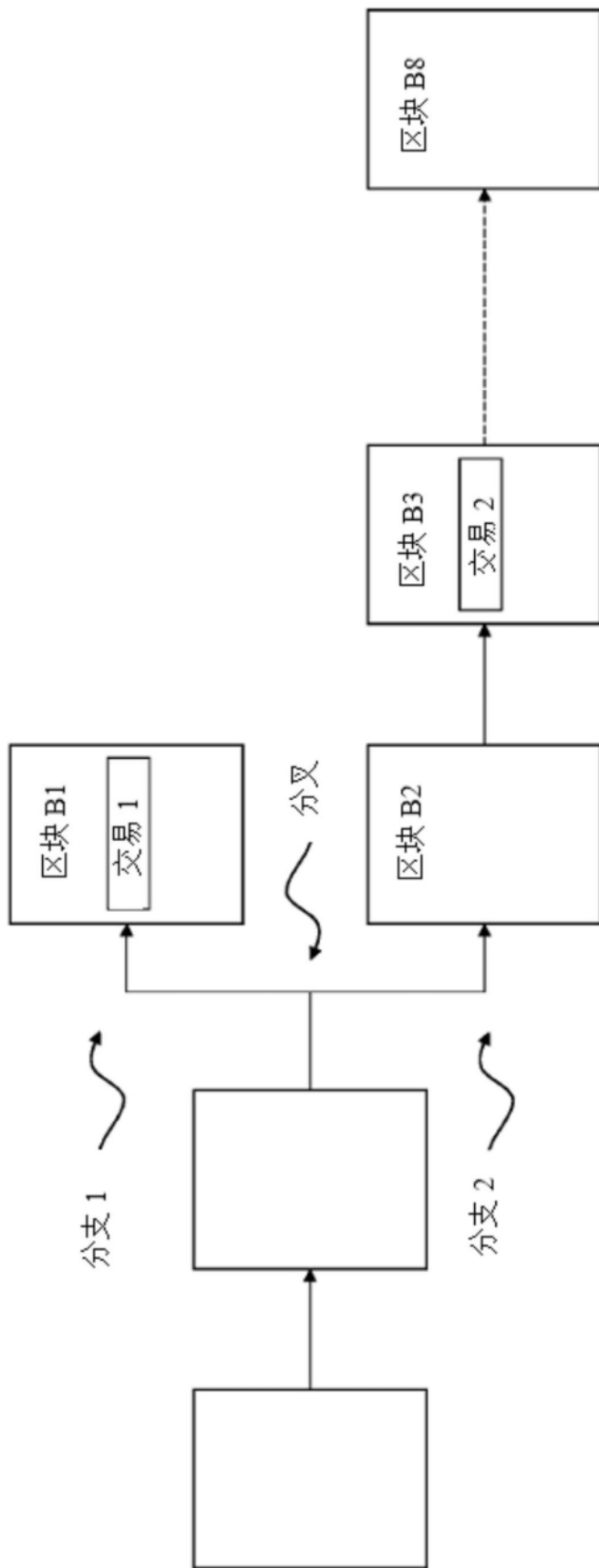


图6

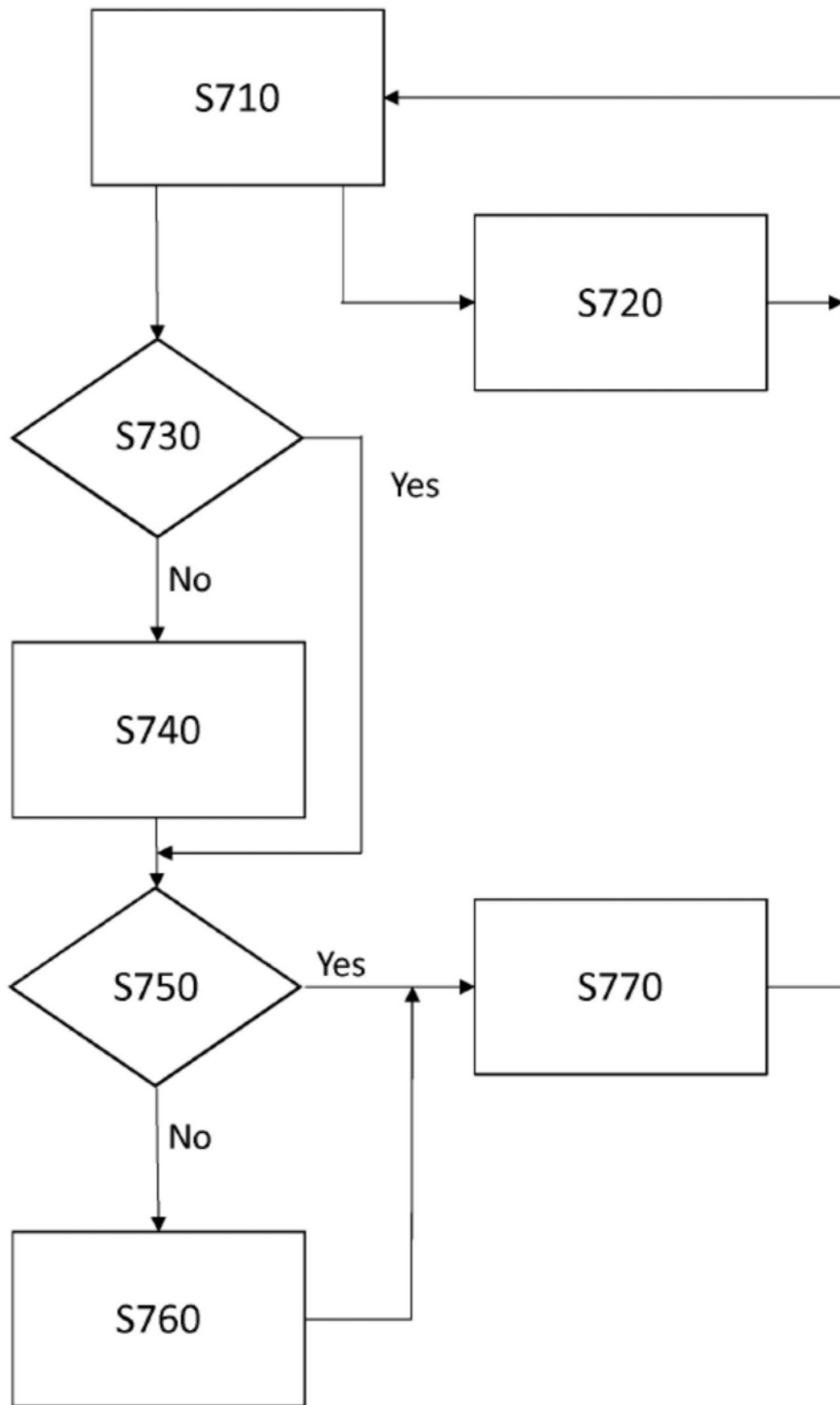


图7

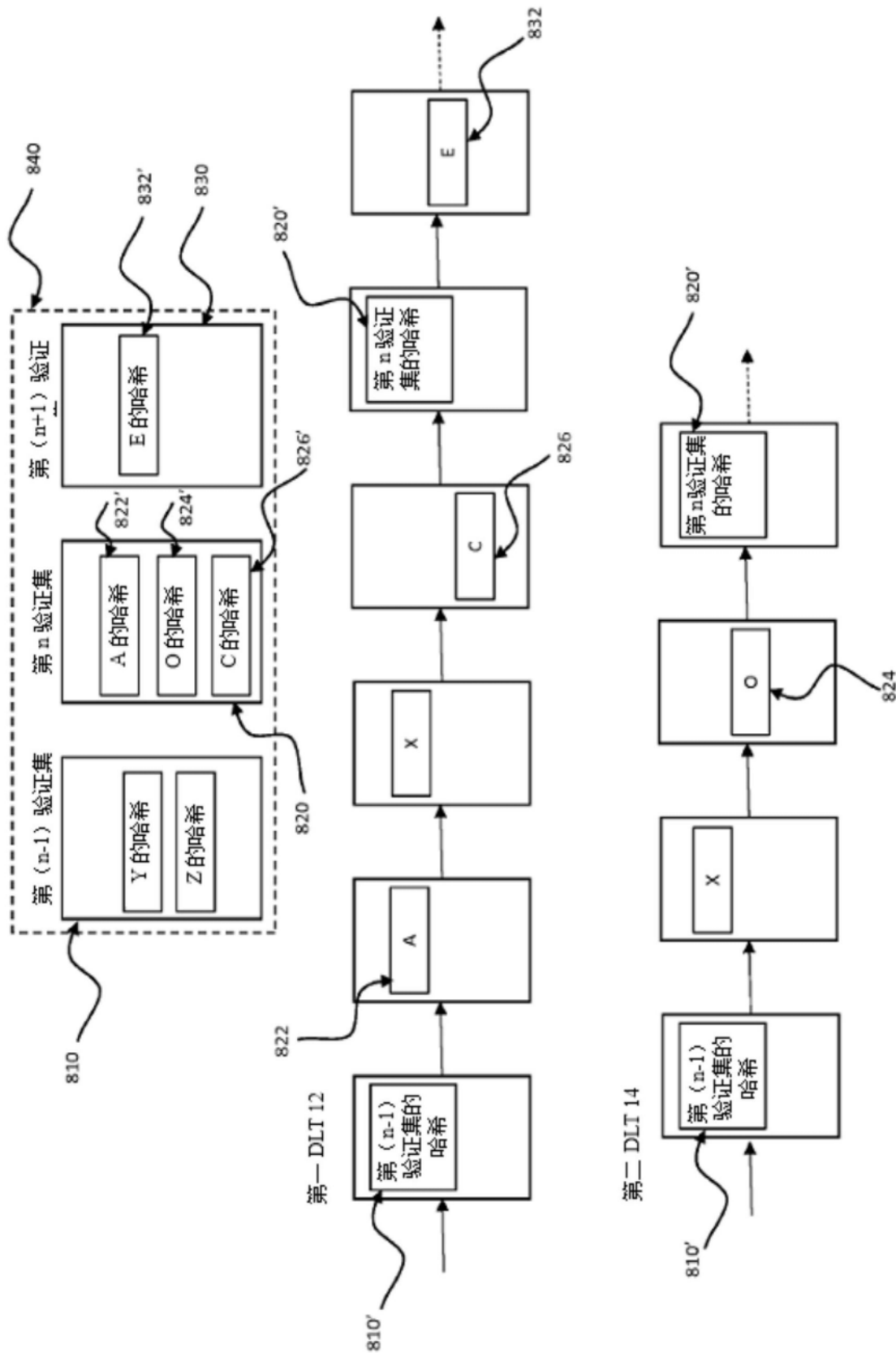


图8

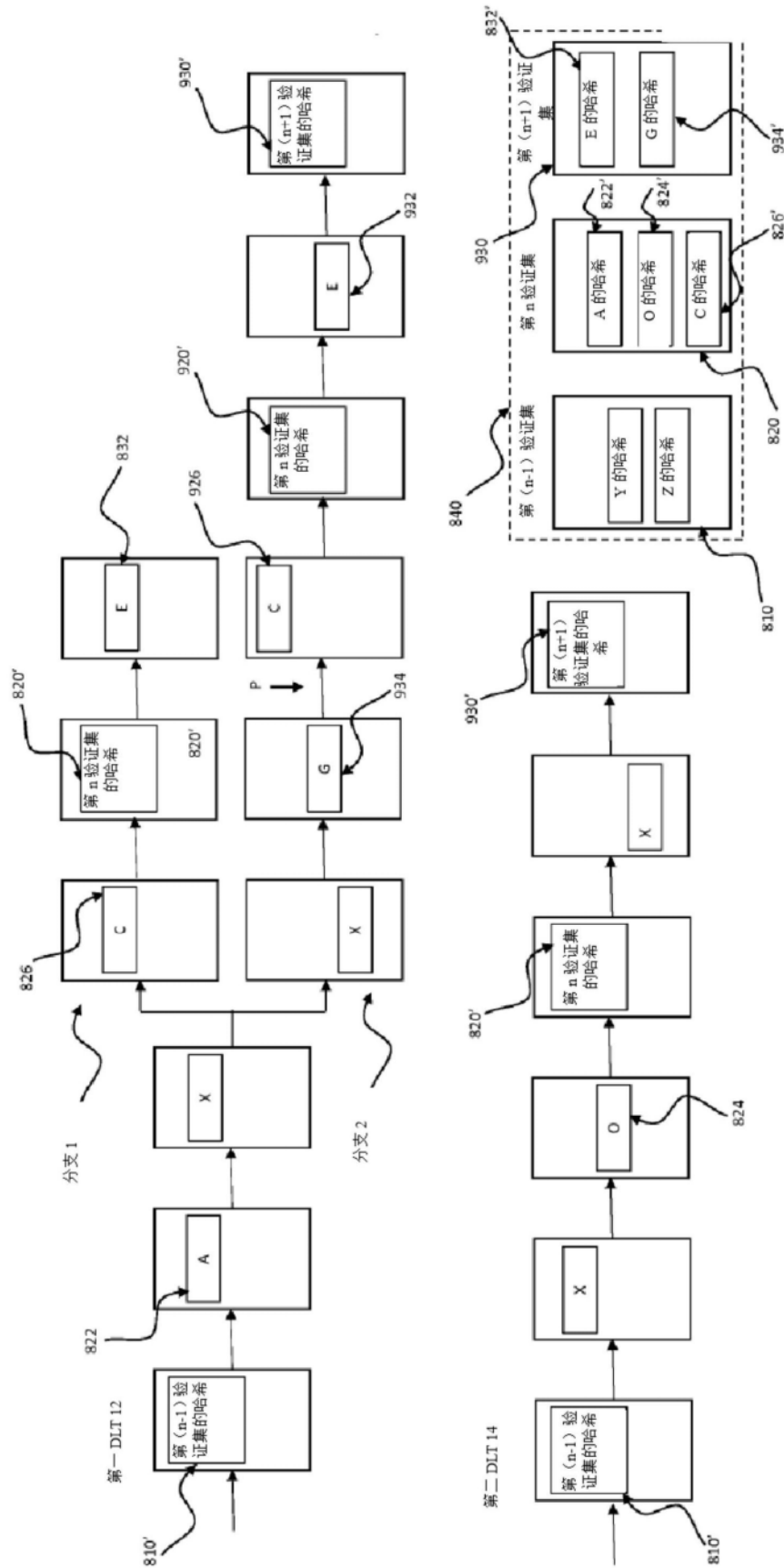


图9

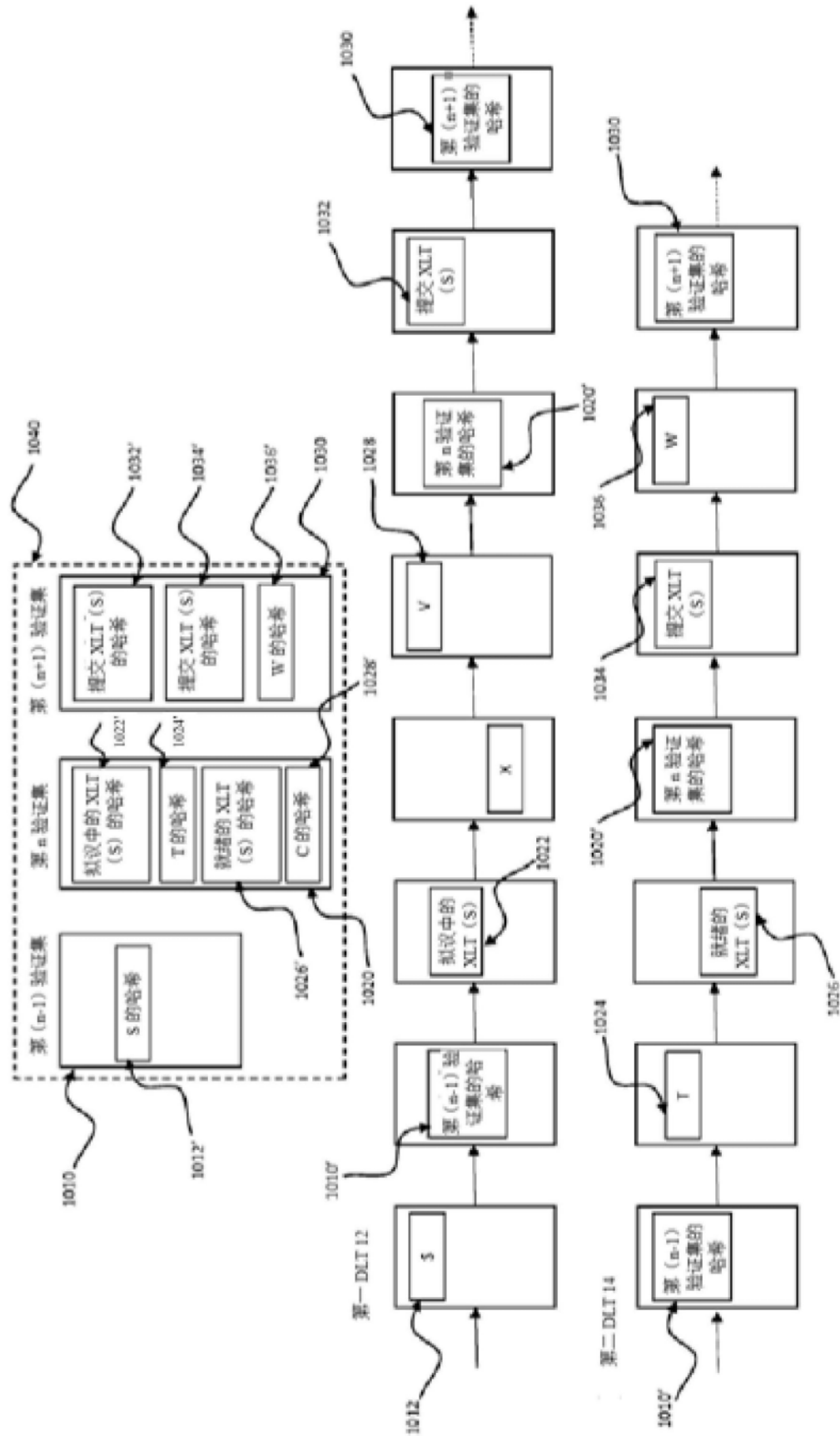


图10

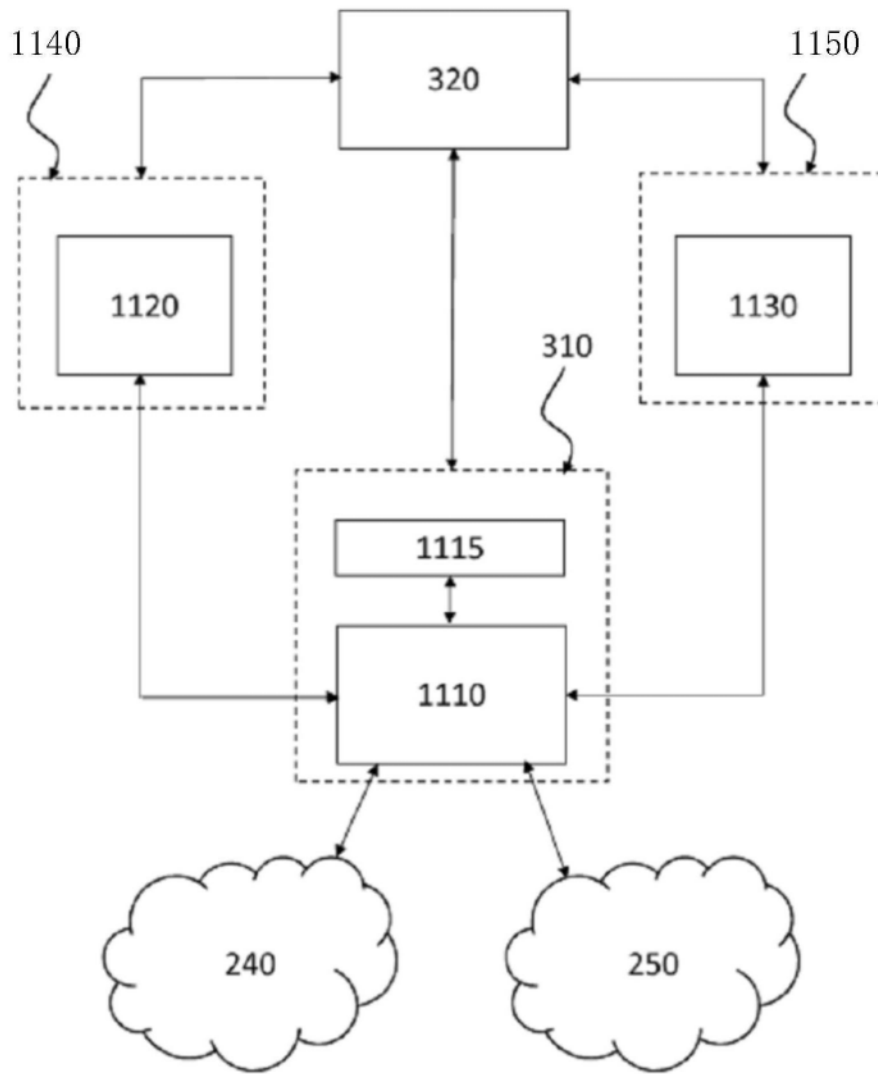


图11

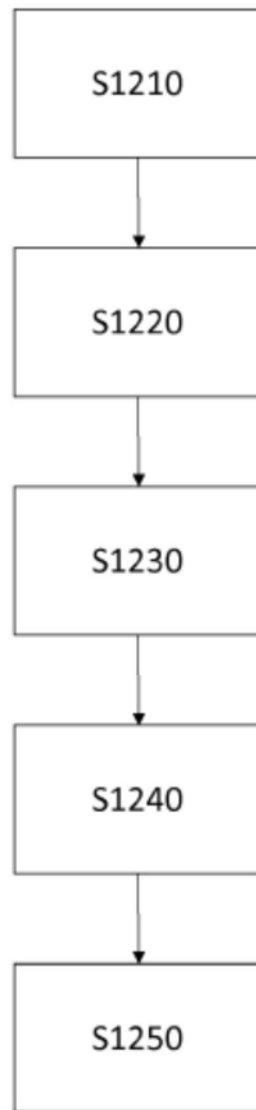


图12