时间结绳：一种事务定序共识系统

李宽
kuan@mail.nwpu.edu.cn
2021 年 5 月 16 日

摘要

本文提供一种称作“时间结绳”的事务定序共识系统，具备以下特征：1.尽量还原交易进入网络的时序；2.交易时序无法被节点操纵；3.真正去中心化的时间机制；4.无限产生 NFT；5.真正的跨链机制；6.应用与平台解耦；7.动态的应用分片；8.采用拟态防御安全技术。

其中，事务定序共识方法包括：节点不间断地构造连续的前向依赖数据区块，将接收到的事务立即引入当前数据区块中，形成事务到达本节点时序证据；节点之间同步所述证据，各自还原出对于事务进入网络的时序判断，并经共识处理形成事务进入网络的时序共识；各节点处理共识事务完成对系统状态的一致性变更；进一步地，定义了一种共识时间。

时间结绳系统可以实现事务公平定序共识，解决现有技术中因信息损失而造成的事务定序能力不足，因竞选主节点而造成不公平竞争和浪费，因主节点主观操纵而造成的事务定序失公平，以及由上述问题衍生的问题。

本文将对时间结绳的应用场景、理论模型、解决方案、激励机制等内容做出阐述。

背景

人类文明的本质是将自然能量转化为人类智慧，再运用智慧提高能量转化能力。中国历法和易学建立了天体运动与自然能量动态分布的映射表，实现了可预测的能量采集；进一步地，电力屏蔽了不同能量形式的异构性，实现了稳定的能量供应，以电力为基础的人工载体屏蔽了自然载体带给人类文明的不确定性。于是，人类加速从碳基文明进入硅基文明。当人类涌入硅基文明之后，稀缺性开始凸显，事务因果不可忽略，公平竞序就成了必然要求。

在任何文明中，主体对稀缺资源的公平竞争是永恒的话题。公平的事务定序共识机制尤为重要。然而，现有技术中，事务定序受到中心化机构或“去中心化”网络权威节点的操纵，进而有大量能源被浪费在对这种操纵权力的争夺上。在一些中心化证券系统中，交易被篡改时序和选择性执行是公开的秘密；在一些 POW 区块链系统中，主节点根据交易手续费高低来选择交易入块；在一些 DeFi 系统中，矿工通过操纵区块中的交易顺序而在抢跑交易中获利。对于习惯使用强权和金元来获取优先权的少数人来说，上述平台是“公平的”，但这其

实侵害了主流群体参与文明的时间权利，最终伤害的是平台的公信力和吸引力。

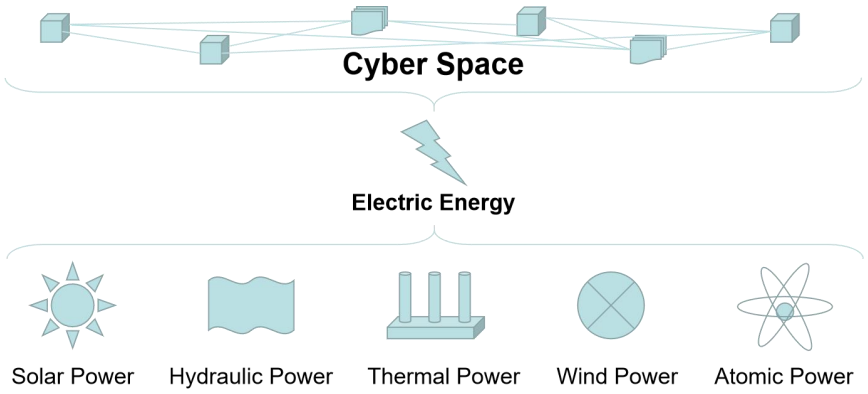

图 1 电能屏蔽了能源异质性，为赛博空间提供统一可控的能量

时间结绳，从华夏文明古老的结绳记事中获得灵感，具备以下特征：

1.共识尽量还原事务进入网络的时序。在跑步比赛中，用撞线的时序给运动员排名是最自然最公平的，同理，将事务进入网络的时序作为定序目标是最为客观公正的；

2.事务时序无法被节点操纵。节点不间断地构造到达绳来记录事务到达本节点的时序，分析各个节点的到达绳来还原出事务进入网络的时序，作恶时间窗口被最大化压缩；

3.真正去中心化的时间机制。随机变换主节点不是网络去中心化的终点，真正的去中心化是指网络内生时间的推进权在任何时段不被某个节点垄断；

4.无限产生 NFT。时间结绳项目本身不产生同质化 Token，但是共识绳的每一个绳结都是一个 NFT，共识绳可以无限延长，NFT 可以无限供应。

5.真正的跨链机制。时间结绳可以给其他链的区块和交易定序；

6.应用与平台解耦。时间结绳的平台层只做事务的定序共识，而数据的安全隐私、事务的解析执行交由应用层自定义，共识应用可以与事务定序共识逻辑分离部署；

7.动态的应用分片。节点可以选择性部署应用，应用也可以选择性找节点部署；

8.采用拟态防御技术。拟态防御技术通过动态异构冗余重组，产生时空不一致的测不准效应，反向建立对攻击者的信息优势，彻底解决网络攻击问题。

本文将对时间结绳的应用场景、理论模型、解决方案、激励机制等内容做出阐述。

应用场景

一句话，时间结绳适合竞序场景。通俗的说，凡是需要“公平地抢”的场景都可以用时间结绳。当然，时间结绳并不是要求任何场景都只能先到先得，但是能为先到先得提供依据，至于权利让渡和规则叠加则交由应用层自主实现。

股票交易。例如：当预期成交价相同时谁能优先成交，尤其是跌停板时谁能优先逃跑。

额度认购。例如：抢红包、游戏抢装备、抢门票/车票、碳排放权交易、电商抢购。

艺术品竞拍。在主持人喊“成交”之前举牌，这也是一种竞序。时间结绳能实现资产的全球去中心化自动竞拍，竞拍者不必担心自己的举牌被中心化系统刻意忽视。

知识产权申报。当发明方案形成后就可以立即将 Hash 值写入时间结绳。

资产确权。对于存证类场景，同样适用。

元宇宙时间。元宇宙是人类梦想的最大化摆脱物理束缚的文明形式，时间结绳可以作为元宇宙中记录文明编年史的基础设施，并提供一种全网共识的时间。

POW 区块定序。在串行写入的区块链方案中，区块上链也是一种稀缺机会。一旦有了时间结绳，“最长链原则”就立刻失去了意义，每个区块都可以在时间结绳上定序，其他节点一旦发现本轮有区块被打结到时间结绳，就可以立即停止本轮挖矿，自私挖矿就变得愚蠢，节点不再需要竞争最长链，也不再需要连续后继 6 个区块才能确认本区块的交易。

跨链交易。每个公链的本质是一个去中心化的时间系统，实现跨链交易的本质是引入第三方时钟来协调两个不同时间系统。时间结绳提供了这种能力。

研究动机

在人类文明中，一切抽象为主体、事务和状态。

事务是变更状态的意志载体，状态是事务执行的现实结果。状态是发起新事务的依据，事务是产生新状态的动因。主体发起事务，获取状态，如此循环。

只有被执行的事务才能改变状态，不同的事务执行顺序会产生不同的状态变化轨迹。一个状态被改变的机会是一种一次性资源。当一个状态诞生后，该状态被改变的机会窗口就打开，当该状态被改变到新状态后，原状态被改变的机会窗口就关闭，也就是当一个事务将原状态改变到新状态后，就不可能有另一个事务基于原状态进行另一次改变。于是，事务之间存在因果顺序，一个事务只能因应前一个事务造成的状态去创造新的状态。事务的因果顺序决定事务的执行顺序，也就决定了状态变化轨迹。

意志未必能改变现实，确定了因果顺序的事务未必被执行。事务被执行需要满足两个条件，一是所基于的状态未被其他事务改变，二是事务的预期执行结果满足合理性。当两个事务相互冲突（希望基于相同的状态做出改变）时，就需要确定事务的因果顺序，前位事务被执行，后位事务被舍弃，例如同一块钱不能被花两次（特指“双花”）；当两个事务不预设所基于的状态时，后位事务因为要基于未知的新状态而可能丧失预期执行结果的合理性，从而

不被执行，例如从只有一块钱的钱包不可能两次取出一块钱。任何主体都不希望自己的意志被舍弃或压制，因此，事务对因果顺序存在竞争需求，也就对事务定序机制提出要求。

理想的事务定序机制应当客观、公平、安全、稳定、高效。所谓客观，是指事务因果顺序应当基于客观证据而不被主观操纵；公平，是指不应当让事务在定序过程中获得特权；安全，是指已经被确定的事务因果顺序不应当被破坏；稳定，是指主体对事务的因果顺序有稳定的预期；高效，是指事务从发起到完成定序的延迟不应过长。

主体的感知能力有限，不可能在第一时间知道世界上所有新发生的事。并且显然地，主体对事务定序的前提是获知待定序的事务，各主体对事务因果顺序达成共识的前提是至少形成了一个可理解和可验证的事务定序提案，而这个前提是存在一种公允的事务定序方法。

人类创造了一种公允的事务定序方法，将一个或一组可持续、可预测、不可逆、公共可见、变化足够频繁、不受人类控制的状态变化轨迹作为参考系，（即时间定义），设计一种计数系统对参考系的状态进行编号，（即时间解释），当参考系产生新状态时就要将当前编号更换为新状态对应的编号，（即时间推进），将事务分别与该参考系的状态编号建立映射，（即计时或盖时间戳），并以一种不易失和可举证的方式记录这些映射，（即时序举证），再以状态的先后顺序确定事务因果顺序，（即定序，时序确定）——这就是时间机制。

在物理空间中，人类将天文现象（如地球自转、地球公转、月球公转）作为参考系，通过政治博弈完成了人类时间的全球统一标准，由中心机构完成时间解释（即什么样的天体运动状态对应什么时刻）和时间发布（即授时），从而建立了一种中心化的人类时间机制。目前人类还不能明显操纵天体运动，因此人类暂时不掌握物理世界的时间推进权。

在共识网络中，数据在不同节点中存有副本，各节点接收并执行事务来变更数据副本的状态。为了使数据副本有一致的同步的状态变化轨迹，节点需要就事务因果顺序达成共识，并按照所共识的事务因果顺序来确定事务执行顺序。

然而，网络空间中并没有物理空间的天文现象作为参考系，为摆脱对物理空间的依赖，就需要将且仅将网络内部活动（如计算、通信）所形成的状态变化轨迹作为参考系。网络空间是人造的，网络内部活动也是可以主观控制的，为了避免网络时间的解释、推进、发布权力被单一节点垄断或形成特殊优势，就需要建立不被少数节点操控的时间机制。

相关工作

事务定序共识是指对一组事务进行排序并就所述排序达成共识，是区块链技术的核心。区块链系统的本质是去中心化的网络时钟，各节点对时间的解释和推进规则有统一的理解，

但时间推进权不由单一节点垄断，也无法预测由哪个节点完成下一次时间推进。每次时间推进都依赖上一个时间戳和可能的一组已接收但未绑定时间戳的事务而产生一个新的时间戳。节点对时间戳的依赖关系和事务与时间戳的绑定关系达成共识，并以此确定事务的因果顺序，即，与同一个时间戳绑定的事务为并发，并晚于上一个时间戳所绑定的事务，以此类推所有时间戳的事务。区块链就是一种存储时间戳的依赖关系和事务与时间戳绑定关系数据，并使之在一定程度上不可逆且不可篡改地增量存储的技术。

现有，区块链技术的事务定序共识方法一般包括“选主、出块、验证、上链”四个步骤。“选主”就是共识网络中各个节点就成为拥有时间推进权的主节点进行竞争或选举；“出块”是指主节点将自己产生的时间戳和一组预先选定的绑定该时间戳的事务打包成区块并传播给其他节点；“验证”和“上链”指的是共识节点验证主节点所出区块，将通过验证的区块写入区块链并执行。选主和出块的本质是选出一个主节点来完成一次时间推进。

然而，节点在接收事务时没有可举证的计时手段，而是把事务无差别地接收进缓冲区，也就丢失了事务到达本节点的时序信息，导致节点在定序时缺乏时序性的客观依据，而不得不利用手续费等非时序性的信息依据，也就给了事务以高额手续费获取定序优先权的机会。主节点时间推进权，包括选择待绑定事务、产生时间戳、决定因果、获得奖励等特权，选主过程可能会引发各节点非理性的竞争，耗费大量资源。节点在选择事务打包区块时存在主观性，往往依据的是使主观利益最大化而不是客观的事务时序证据，节点也无从获得客观证据，这会造成事务定序结果失真，本质上是对事务真实时序的篡改行为。

现有的公平选主机制，只是“民主地选出一位能力低下的独裁者”。选主机制表面上实现了记账权分配的公平性，但对事务定序准确性、时效性和吞吐量的提高没有实质上的帮助，反而因为争夺记账权而造成算力和电力的浪费越来越严重。

出块机制将一组事务与同一个时刻绑定从而认为这组事务同时发生，降低了事务时序的分辨率（好比是认为所有赛跑运动员都同时撞线），也明显与事务的真实时序不符。

区块由单一节点向全网复制就存在区块同步延迟，系统通过设置出块难度来维护稳定的出块间隔时间为保证区块被同步到全网预留充足时间窗口，这也导致网络共识推进和事务的执行缺乏平滑性，这种出块间隔也给了攻击者充分的作恶时间窗口。

区块数据在多轮共识后才会在概率上被确认（例如比特币的交易确认一般要经过六个区块周期，平均每个周期 10 分钟），这导致事务确认延迟较高。这意味着网络状态数据长时间处于不确定性，这种不确定性给主体的决策造成风险。

上述这些问题会导致主体对于所提交的事务被如何定序及何时执行缺乏稳定的预期，对

系统的公平性提出质疑，并承担越来越高的成本，从而不愿意使用这种系统。因此，现有技术局限于对事务定序实时性、公平性和吞吐量要求不高的业务场景。

原理分析

分布式系统的事务定序共识问题在上世纪 70 年代由 Lamport 开始讨论，但是受狭义相对论影响，研究者们把不同的节点理解为不同参考系中的观察者，并且认为不同参考系的观察者对事务的因果顺序有不同的认识，只能通过在不同参考系中传递事务因果顺序来形成共识，于是事务定序共识往往要先选定以哪一个参考系中的观察者所判断的事务因果顺序为准，也就是确定主参考系，对应选主；并将主参考系的观察者所判定的事务因果顺序强制推行到其他参考系，对应出块；作为整个网络对事务因果顺序的共识，对应上链。这种设计存在的问题包括：第一，给了单一参考系主观操控事务因果顺序的能力。事务定序共识方法的“公平性”被妥协为，不能由单一参考系一直垄断事务定序权，而是要设计一种机制来随机变换主参考系，但这并没有改变单一参考系在成为主参考系后对事务因果顺序的主观操纵能力，没有根本上解决公平问题；第二，由某一参考系判定的事务因果顺序本身不具有排他性，只能由后续多轮次的参考系通过继承这种判定来逐渐巩固这种事务因果顺序的不可逆转性，于是事务因果顺序的最终确认必然存在明显的滞后性；第三，单一参考系对一组因果顺序的判定只能从该参考系出发向全网其他参考系传播，并且预留足够的传播时间使所述判定能尽量到达各参考系，必然导致一组事务的共识效率难以提升。

事实上，狭义相对论并不完全适用于网络空间。在不考虑量子技术的情况下，虽然绝对实时和绝对客观的排序共识也是不可能实现的，绝对时间也被认为不存在，但是在地球尺度上、在人类对事务定序准确度和时效性的可接受范围内、在现有计算和网络技术能力下，构造出一种接近绝对时间的事务定序共识系统并非是不可能的。

在这种思路下，选主就不再必要，确定一种客观公正的定序依据就成为新路径。在一个事务的生命周期中，事务由某个主体发起，从某个节点进入共识网络，再传播到达其他节点，然后形成事务因果顺序的共识。其中，事务发起的时机由主体自主控制，事务进入共识网络的时序是客观形成的，而事务经过物理网络到达其他节点的时序则不再可控，并且，事务进入网络的时刻表示事务的发起主体与共识网络的交互关系建立，即事务所承载的意志已经传达到了共识网络。因此，将事务进入共识网络的时序作为事务因果顺序是最为客观公正的。

在确定了事务定序的客观依据后，还要避免节点对事务定序的主观操控。至少有四层思路避免节点操纵事务因果顺序。第一，将节点操纵事务因果顺序的机会窗口压缩到小于节点

操纵事务因果顺序所需的反应时间，也就是让节点不能操纵；第二，让节点操纵因果顺序所需的机会成本大于节点操纵事务因果顺序所能获得的预期收益，也就是让节点不敢操纵；第三，让节点不操纵事务因果顺序所能获得的收益大于操纵事务因果的预期收益，也就是让节点不想操纵；第四，不设主节点，让节点操纵的事务因果顺序只作为中间结果而无法侵入到最终结果，也就是让节点操纵无效，根本上杜绝操纵的负面影响。

事务定序共识的本质是消除事务因果的不确定性并形成对事务因果顺序的公共知识，也就是加工信息、输出知识，因而需要确定信息构成、信息来源、信息捕获方式、知识构成、信息向知识的转化方法。为提高知识输出的能力上限，就要提高信息输入能力，信息越充分，消除的不确定性就越多，就越能输出确定的知识。进一步的，同步、共识的目的就是要保持各节点有相同的信息和知识储备，这样各节点才能做出一致的行为。

在涉及选主的方案中，由于主节点的产生存在不确定性，并且主节点在输出知识时存在主观性，就这个整个网络来说，在知识输出环节引入了新的不确定性；主节点对于自己输出的知识有天然的认知优势，不仅引入了新的不公平，也可能引入新的不确定性；主节点输出的知识需要经过多个轮次才能“最终确认”，这个过程中存在较大被逆转的机会，也就长时间存在不确定性。这样就违背了通过定序共识来消除不确定性的初衷。

如果不设置主节点，就需要让每个节点都有足量的信息去生产出唯一相同的知识。通俗地讲，这个世界没有人能告诉你什么是绝对正确的，每个人都掌握着真相的碎片信息，每个人都可以共享碎片信息去帮助他人还原真相，每个人都会收集到足够的信息，并且按照相同的规则来处理信息，人们最终将得出相同的判断。

一组事务一般从多个节点进入共识网络，于是每个节点都无法直接测得所有事务进入共识网络的时序，但是每个节点都直接测得事务到达本节点的时序。这就需要找到一种方法，使各节点通过分析事务到达节点的时序信息判断出事务进入网络的时序信息，继而形成对事务进入网络的时序共识。第一，由于事务通过某一个共识节点进入网络，那么该节点就有能力判断该事务进入网络的时间必然晚于之前接收到的其他事务；第二，由于先进入网络的事务大概率先到达更多的共识节点，那么共识节点综合其他共识节点接收事务的顺序可以对比出哪个事务率先到达更多的节点，也就可以大概率正确判断事务进入网络的先后顺序；第三，节点间多轮次分享自己对事务进入网络的时序判断，并接受多数结果，最终可以收敛出对事务进入网络的时序的一致性判断，从而达成共识。

进一步地，已共识的事务序列可以作为时间参考系。序列中的事务按照客观规则进行编号，最新被共识的事务按照客观规则解释为当前时间，每新增一个共识事务视作一次时间推

进，由于每个节点都按照相同规则维护相同的已共识的事务序列，时间定义权、时间解释权、时间推进权就分散于每个节点，无法被少数节点主观操控。这样就形成了客观的时间机制。利用客观的时间机制可用于事务进入网络时的计时，从而又增加了一个信息维度。

技术思路

本文所述事务定序共识方法的基本思路为：第一，把整个网络作为一个参考系，将事务进入网络的时序作为确定事务因果顺序的依据，各共识节点致力于就事务进入网络的时序达成共识；第二，共识节点各自维护一个时间参考系，根据这个时间参考系记录下事务到达本节点的时序证据，并及时向网络同步所述证据；第三，每个节点获取到事务到达各共识节点的时序证据，根据预设规则判断事务进入网络的时序；第四，各共识节点相互共享对于事务进入网络的时序判断，接受多数结果而形成对于事务进入网络的时序共识；第五，各节点将已共识的事务序列作为另一个时间参考系，从而对事务增加了一个计时维度。

本文方案用于解决现有技术中因信息损失而造成的事务定序能力不足，因竞选主节点而造成不公平竞争和浪费，因主节点主观操纵而造成的事务定序失公平，以及由上述问题衍生的问题。为表述方便，事务进入共识网络的真实时序称作“本真序”，事务到达某节点的时序称作所述节点的“到达序”，节点对事务进入共识网络的时序的判断称作“还原序”，节点对事务进入网络的时序的共识称作“共识序”。

造成现有技术问题的原因是，第一，节点无法第一时间对事务计时举证而导致各个节点的排序能力不足，第二，选主环节的设计使时间推进权、时间解释权和事务定序权同时集中在一个节点。因此解决问题的基本思路是，设计一套机制，在事务进入网络的第一时间就能形成可举证的计时信息，并且将时间推进权、时间解释权和事务定序权进行分离和分散，不再产生主节点。要想设计这种机制就需要对分布式系统的基础理论作出突破。现有区块链局限于一定程度解决非竞争性事务的存证问题和双花问题，但竞争性事务对事务定序的客观公平性提出更高要求，以少数节点主观定序并强制共识为特征的现有技术不能满足要求，原因在于，第一，现有技术中节点缺乏可举证的计时手段导致在定序时缺乏信息支持，第二，借鉴狭义相对论的现有分布式定序基础理论不适用于绝对时间下的定序共识场景。

以时间和信息视角发明新技术是一种趋势。

从信息的视角，本公开内容的发明构思在于：由于事务进入网络的时序是客观形成的，通过一系列计算和通信手段，节点捕捉和收集事务与共识网络结合过程中产生的各碎片原始信息，结合已共识的推理方法，从所收集的信息中尽量复原出事务进入共识网络的时序信息

并达成共识，使事务能按照接近客观形成的时序被定序和执行，避免节点主观操纵事务定序，并且由于原始信息已经消除了大量不确定性，各节点更容易达成共识。

从时间的视角，本公开内容的发明构思在于：各节点自建一种高分辨率的计时手段，计时本节点到达序，通过共享和分析各到达序的相对时间信息，得出还原序，再就还原序达成共识序，并将共识序的增长作为一种推进共识时间的手段，从而在不依赖于网络外部时间源的前提下实现一种网络内生时间并提供时间服务。

所述发明构思更为具体而言，第一，为节点提供一种计时手段使节点能记录下本节点的到达序和事务来源信息，从而避免信息损失；第二，为节点提供一种方法，使节点能根据各节点到达序信息和事务来源信息推理出事务进入共识网络的时序的信息，从而各自给出还原序；第三，为节点提供一种方法，使节点就各还原序达成共识序；第四，由共识序定义一种新的计时手段来辅助定序共识——从而形成一种新的事务定序共识方法、系统和设备；进一步地，为了避免节点在提供信息时的不诚实行为，提供一系列优选方法、系统和设备；进一步地，为了提高共识效率、降低共识成本，提供一系列优选方法、系统和设备。

由于本公开内容从信息和时间的视角定义了新的时间技术和事务定序共识技术，发明构思已不同于区块链技术，结合本公开内容中的技术特征，可以称本技术为“时间结绳”。当时间结绳技术与具体应用场景结合时，可以产生区块链技术所不具备的解决能力，例如，不依赖少数节点主观定序的去中心化证券交易所、不依赖人类时间的去中心化拍卖平台。

有益效果

时间结绳技术方案能够达到以下有益效果：

1.本文所述共识节点构造本地到达绳记录事务到达本共识节点的时序的方法，使得共识网络保留了事务到达共识节点的时序证据信息，避免了信息损失，从而使得共识节点可以根据事务的到达序各自还原出事务进入网络的时序，共识节点的事务定序能力增强。

2.本文所述共识方法中不再设有主节点也不再需要竞选主节点，事务定序由共识节点共同完成，从而避免了因竞选主节点而造成的非理性竞争和资源浪费。

3.本文所述共识方法中，共识节点根据事务到达各节点的时序的证据，通过预设的客观分析方法完成事务定序，保证了定序过程的客观性，事务定序不再由某个节点主观操控，事务定序的依据不再是和事务时序无关的因素（如交易手续费高低），定序的结果也基本接近事务真实发生的顺序。同时，事务发起者获得了公平的记账待遇，不需要为了争取被记账而提高手续费，并且不再担心所发起的事务先进入网络反而长期不被记账或者后记账，从而丧

失事务的时效性和先发优势的情况，于是增强了对共识网络的使用信心。

4.本文所述共识方法中，对于单股结绳来说，本地到达绳的一次打结时间约等于共识节点的一次哈希计算的时间，由于本地到达绳每次打结都可以看做时间推进，本地到达绳可以作为事务到达的计时器，这种设计就为事务到达计时提供了很高的时间分辨率；同时，共识绳也能为事务提供全排序。相比于比特币 10 分钟出一个块，而块中事务都视作同时发生，本文不仅能实现事务的全排序，还能精确分析事务到达共识节点的频率变化，进而可以根据事务到达时序和共识时序进行大数据分析和优化处理。

5.本文所述共识方法中，事务被共识等同于事务被确认，事务共识不需要像比特币方案那样以出块间隔为周期批量完成，实现了共识的平滑推进，也不需要像比特币方案那样等待 6 个区块的时间才能确认一笔交易，极大缩短了事务的确认延迟，也就避免了因为提前人为确认一笔交易而被双花攻击的安全隐患。

6.本文所述共识方法中，共识节点必须不间断地构造到达绳，这种设计下，如果共识节点希望逆转记录两个事务的到达时序就必须花费相同的时间重新构造所涉及的绳段，并且同时要处理在重新构造绳段的过程中到达的事务，因此极大增加了共识节点恶意篡改事务时序的难度。尤其是采用多股结绳和变股结绳的时候更加难以伪造事务到达时序；恶意篡改共识绳时间和篡改事务到达时序的行为会在共识节点的到达绳处理阶段就被识别，从而无法影响到后续共识绳的生成，保证了共识系统的容错性。

7.本文所述共识方法中，共识节点通过客观分析事务到达共识节点的时序，还原出事务进入网络的时序，并对事务进入网络的时序达成共识，按照共识的时序执行事务，这种设计使得事务的执行时序逼近于事务进入共识网络的真实时序，在资源竞争性的场景中（如股票交易、能源交易、碳排放权交易、订票、知识产权抢注）实现了“先到先得”的交易公平性。

8.本文建立了一种去中心化的网络内生时间机制，其中时间推进不再由主节点完成，事务定序不再需要频繁变换参考系，避免了节点为垄断时间推进权而产生的中心化趋势。单一节点的到达序并非最终的共识序，从而避免了单一节点的到达序直接侵入共识序。

理论框架

本节希望从时间、能量、认知（而不是网络、数据、加密）的层面建立时间结绳技术的基础理论。本文所述的“事务定序共识”是指共识网络中各节点不信任外部时间，而仅通过网络内部的计算和通信手段形成不被少数操控的时间机制，对进入网络的一组事务的因果顺序达成不可逆的一致性共识，从而使各数据副本状态有一致的变化轨迹。

大一统模型

各种区块链系统是事务定序共识系统在不同时间分辨率情况下的特例。

所有的事务定序共识系统都在维护一种表达时间戳的有向无环依赖关系和事务与时间戳绑定关系的数据结构，并使数据实现一定程度上不可逆且不可篡改的增量存储。

一个时间戳至少要前向依赖一个时间戳，但可以绑定或不绑定事务。通过时间戳的有向无环依赖关系可以排列出时间戳的次序，再根据时间戳与事务的绑定关系可以排列出事务的次序，最终完成事务的定序——这就是事务定序机制。事务定序共识系统本质在维护一个时钟。每个时间戳的产生都意味着时间推进，时间戳的产生机制决定了共识机制。

时间模型

状态：在一个有记忆的系统中，一个或一组主体设定若干个参考项来描述一个事物，当这些参数项的值一定时，则称这个事物“处于一个状态”，当这些参考项中任意一个值发生了变化，或者参考项的个数发生了变化时，则称这个事物发生了“状态变化”。其中，

采用C表示可选的参考项的键名的集合，s 表示一个状态，由一组参考项的键值对表达，右上角（n）为状态编号。

$$C = \{c_0, c_1, \cdots, c_k, \cdots\}$$
$$s^{(n)} = \left(x_0^{(n)}, x_1^{(n)}, \cdots, x_k^{(n)}\right), \quad \forall x_i^{(n)} \in s^{(n)} \Rightarrow x_i^{(n)} = (c_j, v^{(n)}), c_j \in C$$

一个事物的状态从这个事物上一个状态变化而来，也就是每次状态变化都是从一个“始态”变为一个“终态”。采用S表示这种状态变化关系，而所述状态变化关系也是一种事物，因而S也可以表示所述状态变化关系作的状态，右上角的（n）为状态的编号。

$$S^{(n+1)} = \left(S^{(n)}, s^{(n+1)}\right)$$

状态变化轨迹：当一个事物的状态不断变化，这个事物也随之产生新的状态变化关系，历次产生的状态变化关系按照产生的顺序形成一个序列，称为这个事物的“状态变化轨迹”，记作Ψ。一个事物的状态变化轨迹必然是单向延长的。

$$\Psi = S^{(0)}, S^{(1)}, \cdots, S^{(n)}, \cdots$$

通过一组连续的自然数与状态变化轨迹中的元素建立一一映射的关系，自然数的大小顺序与状态变化轨迹中的元素产生的顺序一一对应。

进一步地，约定，状态展开集合是状态中所有参考项的集合，记作，

$$\widetilde{S}^{(n)}=\left\{x_0^{(n)},x_1^{(n)},\cdots,x_k^{(n)},\cdots\right\}$$

状态轨迹的展开集合是轨迹中所有元素的展开集合的元素的集合，记作，

$$\widetilde{\Psi}=\left\{x_0^{(0)},x_1^{(0)},\cdots,x_k^{(0)},\cdots x_0^{(n)},x_1^{(n)},\cdots,x_k^{(n)},\cdots\right\}$$

事件：将一个事物的一次状态变化称为一个事件，也把一组事件称为一个事件。

时间：时间是一个或一组主体对所在系统的状态变化的感受。当任意一个事物的状态发生变化，或者一个主体认为一个事物的状态发生了变化时，则被主体视为时间变化。

连续时间：因为现实中人类无法穷尽认识所有事物的所有变化，但是根据经验，事物一定在发生变化，所以人类认为时间是连续变化的，也就认为时间是有长度的。

时间流逝：由于一个事物的状态变化轨迹必然是单向延长的，由所述状态变化轨迹所定义的时间也必然是单向变化的，而当时间向前推进时，将一个事件标记为当前时间的机会窗口也不断关闭消失，因此用“流逝”来形容时间的变化。

时间参考系：由于现实中任何一个主体都无法穷尽认识所有事物的所有状态变化，为了简化了主体感知时间变化和描述时间感受的难度，一个或一组主体协定将且仅将一个或多个事物的状态变化轨迹作为时间参考系。时间参考系记作：

$$\Theta=\left\{\Psi_i\middle|i\in N,\Psi_i=S_i^{(0)},S_i^{(1)},\cdots,S_i^{(n)},\cdots\right\}$$

时刻：当时间参考系被当作一个整体时，时间参考系也有状态变化轨迹，则时间参考系的状态变化轨迹中的每一项是一个时刻。

时延：两个时刻定义的时间区间的时间长度。

时序：利用所述时间参考系的状态变化轨迹来标记事件发生时间或预期发生时间的顺序。已知事件 0 在时刻 i 和时刻 m 之间发生，且事件 1 在时刻 m 和时刻 n 之间发生，且已知时刻 j 早于时刻 m，则可以确定事件 0 早于事件 1 发生。

$$S^{(i)}\prec e_0\prec S^{(j)}\wedge S^{(m)}\prec e_1\prec S^{(n)}\wedge S^{(j)}\prec S^{(m)}\Rightarrow e_0\prec e_1$$

时间戳：为了书写交流方便，采用符号与时间参考系的时刻一一对应，所述符号为时间戳。由于所述时间参考系状态变化轨迹不断延长，所述时间戳也需要有相应生成方式。根据时间参考系状态变化轨迹的时刻对应生成时间戳的方法，称为“历法”。

$$t_n\leftrightarrow n\leftrightarrow S^{(n)},\quad n\in N$$

$$t_n=(n,\varphi(S^{(n)})),\quad s.t.\forall i\neq j\Leftrightarrow t_i\neq t_j$$

$$t_n\prec t_{n+1}\Leftrightarrow S^{(n)}\prec S^{(n+1)}$$

$$\Gamma=t_0,t_1,\cdots,t_n,\cdots$$

事件计时：将事件与一个时间戳建立关联的行为，则称为事件计时。

同时：如果一组事件与相同的时间戳建立关联，则认为这组事件同时发生。

时钟：一套模拟时间参考系状态变化且能直接或间接输出当前时间戳的设备。主体对于时间参考系的认识，也是一种对时间参考系的模拟，也可以视作一个时钟。

时间分辨率：一个时钟所能表达的最小时延，称为所述时钟的时间分辨率。时间分辨率体现了时钟帮助主体分辨时间变化和事件时序的能力。

时间定义：当时间参考系和历法确定以后，主体对于时间的理解就确定了，因此把确定时间参考系和历法的行为称为“时间定义”。

时间解释：当时间被定义之后，把确定当前时间的行为称为“时间解释”。

时间推进：在确定当前时间后，把产生当前时间戳的行为，称作“时间推进”。

时间发布：也称“授时”。

时间接受：也称“受时”。

人类时间：人类定义的时间。例如，人类将地球公转、月球公转、地球自转的状态变化轨迹集合作为时间参考系，对应记作 X 年、阴历 X 月、X 日。（为了使年月日的周期关系稳定，引入了闰年闰月机制）。再利用机械、晶振模拟时间参考系的变化来制作时钟。

网络时间：将网络内部的事物状态变化轨迹作为时间参考系所定义的时间。进一步地，网络内生时间，指的是，相对于人类依靠天文观测而建立的历法时间（例如根据地球公转、月球公转、地球自转来建立的年、月、日历法）来说，网络内生时间是依赖且仅依赖于在网络内部产生的计算和通信事件而标记和推进的时间。采用网络内生时间可以摆脱对网络外部时间源的依赖。而去中心化的网络内生时间指的是时间不由某个中心控制和授时，而是由网络内的节点通过某种分散的机制共同推进的时间。

本地时间：由本地到达绳的状态变化轨迹作为时间参考系定义的时间。

共识时间：由共识绳的状态变化轨迹作为时间参考系定义的时间。在共识网络中，已共识事务的数量是单向递增的，且每共识一个事务都意味着一段时间的流逝，则可以在共识系统中设置一种以已共识事务数量为基础的时间，称为共识时间，一种简单的共识时间戳表示方法是直接用一个整数表示，所述整数为当前已共识事务的数量。

认知模型

事务：主体所产生的变更某个目标事物状态的意志的载体。主体通过发起事务来表达自己变更某个事务状态的意志，当事务被接受且满足执行条件后就会被执行，事务的执行会改

变目标事物的状态。

$$T: s^{(n)} \to s^{(n+1)}$$
$$T: S^{(n)} \to S^{(n+1)}$$

强/弱事务：一个事务能否被执行存在不确定性，主体在发起事务时往往并不能确定事务执行所对应的始态和终态。把明确表达了始态和终态的事务称为强事务，否则为弱事务。

事务输入模块：向系统输入事务的模块。在具体表现形式上，可以是手机客户端、网页、设备板卡、服务器应用、无人机、智能汽车等任何具备向共识网络输入事务的功能的整体或部分，可以从设备角度理解，也可以从程序角度理解，具体形态和位置不做限定。本文中的“事务输入模块”可能指一个模块，可能指一组模块，可能指所有的事务输入模块，具体的含义会配合语境确定。

待共识事务：事务编号、事务输入模块标识、入口节点标识、事务输入模块对系统指定状态数据的变更请求的描述。所述事务输入模块可能是用户客户端，也可能是其他设备，这里不做具体限定。系统通过密码学手段实现对事务内容是否被篡改的可验证性。其中，事务待执行内容中一般包括：用户 ID、指定的事务执行器 ID、待执行状态更新描述指令。

状态数据：状态数据是事务执行的结果，状态数据可供状态数据获取模块获取，在本文中，状态可以指状态数据集合，可以指状态数据集合的子集，可以指状态数据集合中的特定元素。例如，在“为用户 A 初始化账户 1”事务执行后，在状态数据集合中会新增一个编号为 1 且余额为 0 的账户实体数据，并新增一个表示“用户 A 拥有账户 1”的关系数据。之后，在“用户 B 向账户 1 转入 1 元钱”事务执行后，账户 1 的余额会从 0 增加到 1 元。进一步地，共识应用和共识节点的程序代码也是一种状态数据，程序代码的内容、版本、开发者、所有者、执行者信息也可以在状态数据集合中存储，也可以通过事务执行来更新状态。

状态获取模块：从系统获取状态的模块。可以是手机客户端、网页、设备板卡、服务器应用、无人机、智能汽车等任何具备从共识网络获取状态的功能的整体或部分，可以从程序角度理解，也可以从设备角度理解，可以和事务输入模块组成一个程序或一个设备，也可以与事务输入模块分布在不同的程序或设备，具体形态和位置不做限定。

主体：拥有状态获取能力和事务输出能力的个体或一组个体的整体。一个主体一般包括若干个事务输入模块和若干个状态获取模块，并有一个数据库存储已获取的系统数据，一个缓冲区用于存放对系统数据的处理结果，并有一组决策模块用于分析理解系统状态并决定发起事务。M 表示主体，I 表示事务输入模块集合，F 表示状态获取模块集合，Σ 表示以获取的系统状态的集合，B 表示缓冲区，D 表示决策模块集合。

$$
\begin{aligned}
M &= (I, F, \Sigma, B, D), \\
I &= \{i_0, i_1, \cdots, i_m, \cdots\}, \\
F &= \{f_0, f_1, \cdots, f_m, \cdots\}, \\
\Sigma &= \{x_0, x_1, \cdots, x_m, \cdots\} \subseteq \widetilde{\Psi}, \\
D &= \{d_0, d_1, \cdots, d_m, \cdots\}, \\
B &= \{d(\Sigma' \cup B') | d \in D, \Sigma' \subseteq \Sigma, b \subseteq B'\}
\end{aligned}
$$

一个主体已获取的系统数据是系统状态变化轨迹的展开集合的子集，未必包含系统的当前状态，也不一定完整。也就是主体对系统的认识往往是片面、滞后、残缺的。

事务因果顺序：当事务 1 基于的始态是事务 0 直接或间接造成的终态，则两个事务构成因果关系。一组事务按照因果关系排列成事务因果顺序。

$$O_{ce} = T_0 \prec T_1 \prec T_2 \prec T_3 \prec \cdots$$

事务执行顺序：事务被实际执行或计划执行的顺序，一般由事务因果顺序决定。

$$O_{ex} = T_0 \prec T_1 \prec T_2 \prec T_3 \prec \cdots$$

事务定序：一种流程，输入一组事务并输出所述事务之间的因果顺序。

共识节点：简称节点，一种特殊的主体，相互直接或间接连接成一个网络，对网络内一组事务进行定序，并就这组事务的因果顺序和执行顺序达成共识。共识节点可以主动发起事务，也可以接受其他主体的委托向网络内输入事务，也可以同步其他节点发来的事务。

共识网络：简称网络。由一组数量可变的共识节点组网而成，共识节点可以按照预设的规则加入或者退出共识网络。所述事务输入模块向共识网络输入待共识事务，具体来说就是事务输入模块向共识网络中的共识节点输入待共识事务。组成一个共识网络的共识节点之间不一定两两直接相连，但是每两个共识节点之间都有至少一条通路。

用 G 表示共识网络拓扑，V 表示共识网络的节点集合，E 表示节点连接关系集合。例如，一个拥有 4 个共识节点的共识网络表示为，

$$
\begin{aligned}
G &= (V, E) \\
V &= \{n_0, n_1, n_2, n_3\} \\
E &= \{\langle n_0, n_1\rangle, \langle n_0, n_2\rangle, \langle n_1, n_2\rangle, \langle n_1, n_3\rangle, \langle n_2, n_3\rangle\}
\end{aligned}
$$

入口节点：主动发起事务或接受其他主体的委托向网络内输入事务的共识节点，称为所述事务的入口节点。一个事务输入模块向共识网络的某个入口节点输入事务，被表达为，

$$i : T^{(n)} \mapsto n, i \in I \in M, n \in V$$

事务进入网络：一个事务被入口节点完成计时并签名之后从入口节点向其他节点发出，视作该事务进入了共识网络。本文提供的实施例假设事务被入口节点盖时间戳后立即签名并

立即发出，于是用事务的共识时间戳表示事务进入网络的时间。

事务同步：一个过程，即事务在节点之间转发，以尽量使得网络内所有节点都获取到所述事务。事务在节点之间的转发关系表达为，

$$n_j : T^{(n_i)} \mapsto n_k,\ n_i, n_j, n_k \in V, j \neq k, i \neq k$$

事务到达节点：一个事务以本节点为入口节点或者被其他共识节点转发向本节点，被节点获取，视作事务到达了本节点。本文提供的实施例假设事务被节点获取后被立即处理并填写本地时间戳和共识时间戳并立即引入本地到达绳，于是用事务的本地时间戳和共识时间戳表示事务到达本节点的时间。

事务共识延迟：一个事务从进入网络的时刻到被写入共识绳的时刻所经历的时延。

事务执行器：用于执行事务中待执行内容的装置、程序、设备。

事务待执行队列：一个队列，队列元素为待执行的事务。

状态合成：将从一个或多个共识节点获取的状态数据做合成处理，形成节点对于状态数据的自主认知。由于本文所述共识系统中，各共识节点的共识进度会有细微差别，状态数据获取模块从单一共识节点所获取的状态数据可能不是最新数据、可能由于节点故障无法获取数据，为保证获取到最新数据以及保证获取能力，状态数据获取模块可能同时从多个共识节点中获取状态数据，再由状态合成模块从获取的多个状态数据中选择可接受的状态数据，发送给状态数据呈现模块进行处理。

顺序模型

本真序：事务输入模块各自向共识网络输入事务，客观上产生了事务进入共识网络的时序，称为“本真序”。共识网络是分布式的，事务输入模块输入的事务会以共识网络中不同的共识节点为入口进入共识网络，在现有技术条件下不存在某个时钟能立即记录下所有事务进入网络的时序，而共识节点也不会以任何外部的时间戳作为事务排序的依据。因此，所述的本真序不会在事务进入网络后立即被共识网络的所有共识节点获取。记作，

$$\mathrm{O}_T = T_0 \prec T_1 \prec T_2 \prec T_3 \prec \cdots$$

到达序：共识网络中的各共识节点获取事务并向其他共识节点同步事务，客观上产生了事务到达各共识节点的时序，称为“到达序”。各共识节点通过各自构建的时钟记录下到达序。所述事务“到达”一个共识节点的方式包括：事务以所述共识节点为入口共识节点从而到达了所述共识节点，或者事务进入共识网络后经过同步机制被传播到了所述共识节点。显然，各共识节点记录的到达序可能不同，到达序也不等同于本真序。打比方说，池塘上的浮

标有两种方式知道一个雨滴落在了池塘上，一种是雨滴刚好落在了浮标上，另一种是雨滴落在水面上后散出来的水波推进到了浮标。所述到达一般指的是首次到达。记作，

$$\mathrm{O}_A^{n_0} = T_0^{(n_0)} \prec T_1^{(n_1)} \prec T_2^{(n_4)} \prec T_3^{(n_0)} \prec \cdots, \quad n_i \in V$$

还原序：共识节点之间同步各自的到达序记录，通过预设规则分析出每个到达序记录中蕴含的部分本真序信息，并将各部分本真序信息整合，从而各自还原出对本真序的判断。所述共识节点还原出的对本真序的判断，称为“还原序”。记作，

$$\mathrm{O}_R^{(n_0)} = T_0^{(n_0)} \prec T_1^{(n_1)} \prec T_2^{(n_4)} \prec T_3^{(n_0)} \prec \cdots, \quad n_i \in V$$

共识序。共识网络对一组事务的还原序的一致认识。共识节点之间共享各自的还原序，通过预设的共识处理方法形成对本真序的一致性判断。一般的，共识序一旦形成后不允许逆转，只能在末端加入新的共识事务而单向延长。于是，共识节点可以将共识序的一次单向延长作为共识网络的一次时间推进，即把共识序作为共识网络公用的时钟，共识节点可以利用共识序记录时间。共识序记作，

$$\mathrm{O}_C = T_0 \prec T_1 \prec T_2 \prec T_3 \prec \cdots$$

时序还原度：一种度量指标，即通过事务定序共识流程所输出的共识序与所述事务的本真序之间的相似度，反映的是事务定序共识系统多大程度上还原了本真序。

结绳模型

结绳：一种数据结构，包括：一组连续生成的数据区块，其中，每个数据区块在上一个区块生成后立即开始创建，且依赖所述上一个区块的数据（一般是将所述上一个数据区块的哈希值写入本区块），且每个数据区块有连续的编号。记作：Φ

绳结：结绳中的每个数据区块称作一个绳结。所述绳结的结构包括：

绳结编号，一般为连续的自然数，且在同一个到达绳中后产生的绳结编号要比之前生成的绳结的编号大。例如：第一个绳结的编号为 0，第二个为 1，第 N 个为 N-1。

前序结引用，一般为同一个本地到达绳中紧邻本绳结的上一个绳结的数字摘要值。例如，当前的绳结编号为 9，则前序结为 8 号绳结，前序结引用为 8 号绳结的哈希值；

事务引用，一般为一个待共识且已验证的事务的数字摘要值，表示引入了所述事务，或者为空值或特定值，表示本绳结没有引入事务；

校验项，可选的，为了增加到达绳的重建难度并易于检查被篡改的绳结，可以设置零个或多个校验项，所述校验项可以是当前绳结所见证的一些特征状态值，例如：

(1) 到当前绳结为止所在的本地到达绳中引入的事务的总数；

(2) 当前绳结的上一个引入了事务的绳结的编号；

(3) 当前绳结向前一定数量的绳结中引入的事务的总数；

(4) 当前绳结引入的事务的入口共识节点的 ID；

(5) 当前绳结之前第 N 个满足某特征的绳结的数字摘要值；

(6) 其他预设的特征值。

校验规则编号，可选的，为了灵活地在不同绳结中增减或变更校验项，可以设置校验规则的编号，指定每个校验项按照哪个校验规则来校验。从程序的角度来看，所述校验规则为一段可以验证所述校验码真伪的程序，在共识节点中预设了一组校验规则，给每个规则设置一个编号，那么通过绳结中的校验规则编号即可知道执行那段校验规则。

在本文中，一个绳结可以用一个结构体表示，存储在内存中，也可以用 JSON、XML 等形式存储在文件、数据库、磁盘中，只要有预设格式保存上述编号、前序结引用、事务引用、校验项等内容即可，对绳结的格式和存储位置不做限定。

绳结的符号记作：K

打结延长：一个流程，为一个结绳新创建一个绳结。

引入绳结：将某个内容 X 作为打结的输入称作“将 X 引入结中”。

绳段：一个结绳中连续的一组绳结组成的集合称作一个绳段。例如，在一个本地到达绳中，存在 0-100 号绳结，其中，0-10 号绳结是一个绳段，89-93 号绳结也是一个绳段。

初始结：一个时间结绳初始化生成的第一个绳结（一般编号记作 0）。一个本地到达绳初始化时创建的位于最开始的绳结，就是所述本地到达绳的初始绳结，初始绳结在所述本地到达绳中没有前序结。

当前结：在一个持续延长的结绳中，当前正在创建的绳结，当前结的编号已经生成。或者是当前正在考察的绳结。

开头结：一个绳段中最早创建完毕的绳结。

末尾结：一个绳段中最晚创建完毕的绳结。例如，在 99-106 号绳结组成的绳段中，第 106 号绳结为最后一个生成的绳结，也就是末尾绳结。

事务结：引入了事务且已创建完毕的绳结。例如，事务结的“事务引用”属性值为被引入事务的数字摘要。例如，编号为 i，引入事务 T 的事务结记作：

$K_{(T)}^{(i)}$

空结：没有引入事务且已创建完毕的绳结。例如，空结的“事务引用”属性值为空或者

表示未引入事务的特定值。例如，编号为 i 的空结记作：

$$K_{(\varnothing)}^{(i)}$$

前序结：如果绳结 0 的哈希值被引入绳结 1，则绳结 0 为绳结 1 的前序结。

后继结：如果绳结 0 的哈希值被引入绳结 1，则绳结 1 为绳结 0 的后继结。

时间结绳：一种用于定义时间和事件计时的结绳，所述结绳的状态变化轨迹作为时间参考系。在本文实施例中存在两种时间结绳，到达绳和共识绳。时间结绳蕴含的信息本质是时间戳集合、事务集合、时间戳的顺序关系集合、事务与时间戳的绑定关系集合组成的四元组，其中，一个时间戳可以绑定 0 个或 1 个或多个事务。区块链是时间结绳的一种特例，其中，一个区块实际上就是一个时间戳和绑定这个时间戳的事务的集合。

$$\Phi \to \left(\Lambda, T, \left\{\left\langle t_i, T_j \right\rangle \middle| t_i \in \Lambda, T_j \in T\right\}, \left\{t_i \prec t_j \middle| t_i \in \Lambda, t_j \in \Lambda\right\}\right)$$

$$\Lambda = \left\{t_0, \cdots, t_n\right\}$$

$$T = \left\{T_0, \cdots, T_m\right\}$$

当时间结绳中的一个时间戳至多绑定 1 个事务，并且任意两个时间戳之间的时序都是确定的，并且不存在多个时间戳的时序形成环状结构，那么这个时间结绳就支持事务的全排序，也就是可以确定任意两个事务之间的时序关系。

到达绳：是用于承载到达信息的数据及数据结构的一种可能的实现方式，一种由共识节点构造，定义本地时间，并记录事务到达序的时间结绳，是一种共识节点以结绳方式记录的事务到达时序数据。由于所述结绳记录了事务到达共识节点的时序证据，故称作所述结绳为“到达绳”。其中，一般的，一个共识节点只构造一条到达绳。例如，一条由节点 n0 构造的本地到达绳可以被记作：

$$\Phi_A^{(n_0)} = K_{(T_0)}^{(0)} \prec K_{(T_1)}^{(12)} \prec K_{(T_2)}^{(60)} \prec K_{(T_3)}^{(130)} \prec K_{(T_4)}^{(400)} \prec K_{(T_5)}^{(425)}$$

其中，空结被省略不计，只列出事务结。

所述到达绳相比于比特币方案的交易缓冲区，至少增加了事务到达共识节点、事务到达共识节点的时间、事务输入模块三个信息维度，可以极大地扩展了信息表达能力。

到达绳不仅能反映事务到达共识节点先后时序，还能一定程度上反映事务到达共识节点的频率变化，进一步还能反映当前网络拥堵情况，还能近似反映不同共识节点的相对打结频率，还能反映出本地事务晚于此前到达的外地事务进入网络。

从安全性角度讲，由于结绳的每个绳结都是前向依赖的，如果想逆转两个事务的顺序就必须重新花费时间构造所涉及的绳段，而在构造的过程中又需要把不断新接收的事务引入结

绳，否则会积压多个待引入的事务而不得不连续地引入事务，这种伪造行为会在结绳上留下痕迹并且很容易被其他节点发现。然而，单一节点对事务到达时序的伪造难以影响到整个网络最终对事务进入网络时序的共识。可见，节点伪造到达绳是徒劳的。

本地到达绳：由本节点构造的到达绳，记录了事务到达本共识节点的时序数据。

到达绳同步：一种流程，使到达绳数据尽可能到达所有的节点。

外地到达绳：由其他节点构造的且被同步到本节点的到达绳。外地到达绳是本共识节点获取的由其他共识节点发来的事务到达所述发出到达绳数据的共识节点的时序数据。

本地到达绳未同步绳段：在一个本地到达绳中未被写入同步消息进行同步的绳段。

打结器：用于创建绳结的装置、程序、设备。

到达绳的并行度。

多股到达绳：一种在构造过程中并行度固定不变且并行度大于 1 的到达绳。

变股到达绳：一种在构造过程中并行度可以自适应调整的到达绳。

还原绳：一种由共识节点构造，记录事务还原序的数据结构。例如，由节点 n0 构造的地到达绳可以被记作：

$$\Phi_R^{(n_0)}$$

本地还原绳：由本节点构造的还原绳。

外地还原绳：由其他节点构造的还原绳。

本地还原绳的稳定绳段：在本地还原绳中，一组被判断连续的事务构成的序列，且经判断不可能有额外的事务被判断加入到所述序列的非末尾位置，则视作所述序列已稳定，用以表达所述序列的还原绳被称为本地还原绳的稳定绳段。

还原绳中的冗余时序数据。

共识绳：一种由共识节点构造，定义共识时间，记录事务共识序的时间结绳。

本地共识绳：由本节点构造的共识绳。

外地共识绳：由其他节点构造的共识绳。

共识绳同步：一种流程，使本地共识绳数据更新到最新状态。

本地共识绳未执行绳段：在本地共识绳中，从最早写入所述本地共识绳的已共识但未被解析执行的绳结到所述本地共识绳的末尾绳结所组成的绳段。

应用模型

共识应用：与共识网络和用户进行交互，满足特定应用场景需求的主体。例如：在一个

共识系统中包括了共识网络和若干共识应用，其中共识应用包括：电影订票应用、股票交易应用、物品拍卖应用、检测试验应用、无人机群应用，在每个应用中都包括事务输入模块、状态获取模块。例如：电影订票应用会输入“订票”“退票”“取票”等事务，也会获取“影院”“场次”“座位”等状态数据。从设备角度而言，每个应用都可以是整体或部分的形式安装在一个或多个设备上。例如：用户 A、B、C、D 的手机上都安装有电影订票应用 APP，也可以通过网页打开应用，在电影院大厅里安装有电影订票取票触控屏。

去中心化交易所：一种程序、设备或系统，搭建运行在一种去中心化网络上，其中，交易时序不被网络节点主观操控，已形成的交易记录不可被篡改内容和逆转时序。由于交易时序严重影响交易结果，任何一笔交易都不希望自己客观产生的时序被任何人操纵。通过设置更高的交易手续费来获取优先交易权也会被认为是不公平行为，不仅会侵占交易者的利益，也会产生竞相提高交易手续费的“剧场效应”。因而，现有的选举主节点进行主观定序的区块链方案不能满足对交易时序公平性有严苛要求的去中心化交易的场景。

文明模型

人类文明活动的本质就是把大自然能量转化为人类智慧，继而用人类智慧改变大自然能量的循环过程。其中又划分为多个步骤：能量有序地转移转化就形成了信号，信号有序地解析记录就形成了数据，数据有序地分析理解为信息，信息有序地总结提炼就形成了知识，知识有序地组织运用就形成了智慧，如果继续下去，开动智慧会形成决策，执行决策会形成动作，施加动作会改变能量，如此形成迭代闭环。

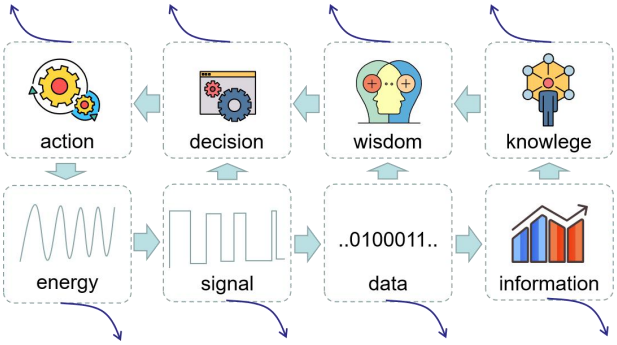

图 2 有序态迭代模型

解决方案

详细技术方案，见《时间结绳技术白皮书》。

原型演示

理论模拟

模拟一个事务定序共识系统，包括一个共识网络 G，若干个事务输入模块 I 和若干个状态获取模块 F。其中，

事务输入模块有 3 个，记作：

$$I = \{i_0, i_1, i_2\}$$

状态获取模块有 3 个，记作：

$$F = \{f_0, f_1, f_2\}$$

节点 0、1、2、3 组成一个共识网络，记作：

$$G = (V, E)$$
$$V = \{n_0, n_1, n_2, n_3\}$$
$$E = \{\langle n_0, n_1\rangle, \langle n_0, n_2\rangle, \langle n_1, n_2\rangle, \langle n_1, n_3\rangle, \langle n_2, n_3\rangle\}$$

为了便于计算和说明，假设各个节点之间的延迟 D 分别为一个固定值，具体为：

$$D(n_0, n_1) = 3, D(n_0, n_2) = 2, D(n_1, n_2) = 2, D(n_1, n_3) = 4, D(n_2, n_3) = 2$$

为了模拟一组事务在共识网络中被共识的过程，假设客观存在一个本真序，

$$\mathrm{O}_T = T_0 \prec T_1 \prec T_2 \prec T_3 \prec T_4 \prec T_5$$

并且，假设存在一种绝对时间，按照绝对时间和本真序设计一个事务进入网络的计划表。其中，本真序中的事务由事务输入模块在指定时刻输入给指定节点。例如，在时刻 10，事务输入模块 1 以节点 0 为入口节点输入事务 T0，以此类推如下：

$$t(i_1 : T_0 \to n_0) = 10.0$$
$$t(i_0 : T_1 \to n_1) = 12.8$$
$$t(i_1 : T_2 \to n_2) = 14.6$$
$$t(i_0 : T_3 \to n_3) = 16.4$$
$$t(i_1 : T_4 \to n_0) = 18.2$$
$$t(i_0 : T_5 \to n_1) = 20.0$$

开始计时后，按照计划表由指定事务输入模块将指定事务输入共识网络的指定入口节点。假设 t=0 时，各节点的本地事务到达序开始打结，分别如下：

$$\Phi_A^{(n_0)} = K_{a(n_0)}^{(0)}$$
$$\Phi_A^{(n_1)} = K_{a(n_1)}^{(0)}$$
$$\Phi_A^{(n_2)} = K_{a(n_2)}^{(0)}$$
$$\Phi_A^{(n_3)} = K_{a(n_3)}^{(0)}$$

由于各节点打结频率不同，当 t=10 时，各节点的本地事务到达序如下，

$$\Phi_A^{(n_0)} = K_{a(n_0)}^{(0)} \prec K_{\varnothing}^{(1)} \prec \cdots \prec K_{\varnothing}^{(1000)}$$
$$\Phi_A^{(n_1)} = K_{a(n_1)}^{(0)} \prec K_{\varnothing}^{(1)} \prec \cdots \prec K_{\varnothing}^{(2000)}$$
$$\Phi_A^{(n_2)} = K_{a(n_2)}^{(0)} \prec K_{\varnothing}^{(1)} \prec \cdots \prec K_{\varnothing}^{(3000)}$$
$$\Phi_A^{(n_3)} = K_{a(n_3)}^{(0)} \prec K_{\varnothing}^{(1)} \prec \cdots \prec K_{\varnothing}^{(4000)}$$

此时，T0 被入口节点 0 获取，节点 0 判断 T0 为本地事务，验证、解析、存储和同步本地事务 T0。节点 0 将事务 T0 打结到本地事务到达序的 1001 号 Knoten，记作：

$$\Phi_A^{(n_0)} = K_{a(n_0)}^{(0)} \prec K_{\varnothing}^{(1)} \prec \cdots \prec K_{\varnothing}^{(1000)} \prec K_{(T_0)}^{(1001)}$$

此时，节点 0 将本地事务到达序的未同步 segment 进行同步。

当 t=11 时，由节点 0 同步的 T0 到达了节点 1，节点 1 判断 T0 为外地事务，验证、解析、存储和同步外地事务 T0。此时，节点 1 的本地事务到达序末尾编号为 2200，于是节点 1 将事务 T0 打结到本地事务到达序的 2201 号 Knoten，记作：

$$\Phi_A^{(n_1)} = K_{a(n_1)}^{(0)} \prec K_{\varnothing}^{(1)} \prec \cdots \prec K_{\varnothing}^{(2200)} \prec K_{(T_0)}^{(2201)}$$

此时，节点 1 将本地事务到达序的未同步 segment 进行同步。

经过一段时间后，各事务都已经输入到了共识网络中且同步到各节点，且各节点的本地事务到达序也都同步到了共识网络的其他节点。

为简化计算，将节点为了同步一个事务而在本地产生的处理时间忽略不计，则根据节点之间的延迟设定可以知道事务到达各个节点的“绝对时间”为：

$$t(T_0 \to n_0) = 10,\quad t(T_0 \to n_1) = 13,\quad t(T_0 \to n_2) = 12,\quad t(T_0 \to n_3) = 14$$
$$t(T_1 \to n_0) = 15.8,\quad t(T_1 \to n_1) = 12.8,\quad t(T_1 \to n_2) = 14.8,\quad t(T_1 \to n_3) = 16.8$$
$$t(T_2 \to n_0) = 16.6,\quad t(T_2 \to n_1) = 16.6,\quad t(T_2 \to n_2) = 14.6,\quad t(T_2 \to n_3) = 16.6$$
$$t(T_3 \to n_0) = 20.4,\quad t(T_3 \to n_1) = 20.4,\quad t(T_3 \to n_2) = 18.4,\quad t(T_3 \to n_3) = 16.4$$
$$t(T_4 \to n_0) = 18.2,\quad t(T_4 \to n_1) = 21.2,\quad t(T_4 \to n_2) = 20.2,\quad t(T_4 \to n_3) = 22.2$$
$$t(T_5 \to n_0) = 23,\quad t(T_5 \to n_1) = 20,\quad t(T_5 \to n_2) = 22,\quad t(T_5 \to n_3) = 24$$

各节点的事务到达序数据如下：

$$\Phi_A^{(n_0)} = K_{a(n_0)}^{(0)} \prec K_{(T_0)}^{(1001)} \prec K_{(T_1)}^{(1581)} \prec K_{(T_2)}^{(1661)} \prec K_{(T_4)}^{(1821)} \prec K_{(T_3)}^{(2041)} \prec K_{(T_5)}^{(2301)}$$
$$\Phi_A^{(n_1)} = K_{a(n_1)}^{(0)} \prec K_{(T_1)}^{(2561)} \prec K_{(T_0)}^{(2601)} \prec K_{(T_2)}^{(3321)} \prec K_{(T_5)}^{(4001)} \prec K_{(T_3)}^{(4081)} \prec K_{(T_4)}^{(4241)}$$
$$\Phi_A^{(n_2)} = K_{a(n_2)}^{(0)} \prec K_{(T_0)}^{(3601)} \prec K_{(T_2)}^{(4381)} \prec K_{(T_1)}^{(4441)} \prec K_{(T_3)}^{(5521)} \prec K_{(T_4)}^{(6061)} \prec K_{(T_5)}^{(6601)}$$
$$\Phi_A^{(n_3)} = K_{a(n_3)}^{(0)} \prec K_{(T_0)}^{(5601)} \prec K_{(T_3)}^{(6561)} \prec K_{(T_2)}^{(6641)} \prec K_{(T_1)}^{(6721)} \prec K_{(T_4)}^{(8881)} \prec K_{(T_5)}^{(9601)}$$

每个节点在接收到外地事务到达序后，就开始进行事务还原序处理。以节点 0 为例，根据本地和外地事务到达序可以获知事务到达各节点的时序，如下：

$$\mathrm{O}_A^{(n_0)} = T_0 \prec T_1 \prec T_2 \prec T_4 \prec T_3 \prec T_5$$
$$\mathrm{O}_A^{(n_1)} = T_1 \prec T_0 \prec T_2 \prec T_5 \prec T_3 \prec T_4$$
$$\mathrm{O}_A^{(n_2)} = T_0 \prec T_2 \prec T_1 \prec T_3 \prec T_4 \prec T_5$$
$$\mathrm{O}_A^{(n_3)} = T_0 \prec T_3 \prec T_2 \prec T_1 \prec T_4 \prec T_5$$

根据还原规则 1，目前尚未有已共识事务，无法利用所述规则 1。

根据还原规则 2，已知，节点 0 的本地事务有 T0 和 T4，节点 1 的本地事务有 T1 和 T5，节点 2 的本地事务有 T2，节点 3 的本地事务有 T3，则，

$$\left(T_0^{(n_0)} \prec T_4^{(n_0)}\right) \in \mathrm{O}_A^{(n_0)} \Rightarrow \left(T_0 \prec T_4\right) \in \mathrm{O}_R$$

$$\left(T_1^{(n_1)} \prec T_5^{(n_1)}\right) \in \mathrm{O}_A^{(n_1)} \Rightarrow \left(T_1 \prec T_5\right) \in \mathrm{O}_R$$

根据还原规则 3，

$$\left(T_1^{(n_1)} \prec T_4^{(n_0)}\right) \in \mathrm{O}_A^{(n_0)} \wedge \left(T_1^{(n_1)} \prec T_4^{(n_0)}\right) \in \mathrm{O}_A^{(n_1)} \Rightarrow \left(T_1 \prec T_4\right) \in \mathrm{O}_R$$

$$\left(T_2^{(n_2)} \prec T_4^{(n_0)}\right) \in \mathrm{O}_A^{(n_0)} \wedge \left(T_2^{(n_2)} \prec T_4^{(n_0)}\right) \in \mathrm{O}_A^{(n_2)} \Rightarrow \left(T_2 \prec T_4\right) \in \mathrm{O}_R$$

$$\left(T_0^{(n_0)} \prec T_5^{(n_1)}\right) \in \mathrm{O}_A^{(n_1)} \wedge \left(T_0^{(n_0)} \prec T_5^{(n_1)}\right) \in \mathrm{O}_A^{(n_0)} \Rightarrow \left(T_0 \prec T_5\right) \in \mathrm{O}_R$$

$$\left(T_2^{(n_2)} \prec T_5^{(n_1)}\right) \in \mathrm{O}_A^{(n_1)} \wedge \left(T_2^{(n_2)} \prec T_5^{(n_1)}\right) \in \mathrm{O}_A^{(n_2)} \Rightarrow \left(T_2 \prec T_5\right) \in \mathrm{O}_R$$

$$\left(T_0^{(n_0)} \prec T_2^{(n_2)}\right) \in \mathrm{O}_A^{(n_2)} \wedge \left(T_0^{(n_0)} \prec T_2^{(n_2)}\right) \in \mathrm{O}_A^{(n_0)} \Rightarrow \left(T_0 \prec T_2\right) \in \mathrm{O}_R$$

$$\left(T_0^{(n_0)} \prec T_3^{(n_3)}\right) \in \mathrm{O}_A^{(n_3)} \wedge \left(T_0^{(n_0)} \prec T_3^{(n_3)}\right) \in \mathrm{O}_A^{(n_3)} \Rightarrow \left(T_0 \prec T_3\right) \in \mathrm{O}_R$$

根据还原规则 5，结合规则 2 和 3 的结果，

$$\left(T_0 \prec T_2\right) \in \mathrm{O}_R \wedge \left(T_2 \prec T_5\right) \in \mathrm{O}_R \Rightarrow \left(T_0 \prec T_2 \prec T_5\right) \in \mathrm{O}_R$$

$$\left(T_0 \prec T_2\right) \in \mathrm{O}_R \wedge \left(T_2 \prec T_4\right) \in \mathrm{O}_R \Rightarrow \left(T_0 \prec T_2 \prec T_4\right) \in \mathrm{O}_R$$

目前还不确定 T0 与 T1，T1 与 T2，T1 与 T3，T2 与 T3，T2 与 T4，T3 与 T4，T4 与 T5 的还原序关系。

根据还原规则 4：对于 T0 与 T1 来说，在节点 0、2、3 中均有 T0 早于 T1 到达，只有节点 1 是 T1 早于 T0，则判断 T0 早于 T1 进入网络，同理可以判断出 T1 早于 T3，T2 早于 T3，T2 早于 T4，T3 早于 T4，T4 早于 T5，此时再根据规则 5，有，

$$\left(T_0 \prec T_1 \prec T_3 \prec T_4 \prec T_5\right) \in \mathrm{O}_R$$

$$\left(T_0 \prec T_2 \prec T_3 \prec T_4 \prec T_5\right) \in \mathrm{O}_R$$

但是，T1 和 T2 各有两个节点的到达序早于对方，目前还不确定 T1 与 T2 的还原序关系。

根据还原规则 6，获取 T0 和 T1 在本地和外地事务到达序的 Knoten 编号并计算。

$$\Phi_A^{(n_0)} : \frac{\left\| K_{(T_1)} - K_{(T_2)} \right\|}{\left\| K_{(T_0)} - K_{(T_5)} \right\|} = \frac{-80}{1300}, \quad \Phi_A^{(n_1)} : \frac{\left\| K_{(T_1)} - K_{(T_2)} \right\|}{\left\| K_{(T_1)} - K_{(T_4)} \right\|} = \frac{-760}{1680},$$

$$\Phi_A^{(n_2)} : \frac{\left\| K_{(T_1)} - K_{(T_2)} \right\|}{\left\| K_{(T_0)} - K_{(T_5)} \right\|} = \frac{60}{3000}, \quad \Phi_A^{(n_3)} : \frac{\left\| K_{(T_1)} - K_{(T_2)} \right\|}{\left\| K_{(T_0)} - K_{(T_5)} \right\|} = \frac{80}{4000},$$

$$\frac{-80}{1300} + \frac{-760}{1680} + \frac{60}{3000} + \frac{80}{4000} \approx -4.526 < 0 \Rightarrow (T_1 \prec T_2) \in \mathrm{O}_R$$

至此，节点 0 可以得出还原序，

$$\left(T_0 \prec T_1 \prec T_2 \prec T_3 \prec T_4 \prec T_5\right) \in \mathrm{O}_R$$

同理，节点 1、2、3 根据相同的方法可以得出还原序，

$$\left(T_0 \prec T_1 \prec T_2 \prec T_3 \prec T_4 \prec T_5\right) \in \mathrm{O}_R$$

节点 0 将本地还原序发送给其他节点，其他节点回复本节点的本地还原序，节点 0 对比所收到的外地还原序，发现完全一致，则认定还原序达成共识序。

节点 0 将共识序按顺序写入共识 Knoten，则得到共识 Knoten 为，

$$\Phi_C^{(n_0)} = K_{init}^{(0)} \prec K_{(T_0)}^{(1)} \prec K_{(T_1)}^{(2)} \prec K_{(T_2)}^{(3)} \prec K_{(T_3)}^{(4)} \prec K_{(T_4)}^{(5)} \prec K_{(T_5)}^{(6)}$$

节点 0 按共识序解析共识 Knoten 中的事务，写入事务执行队列，并逐一调取事务对应的事务执行器执行事务执行队列中的事务，完成对状态数据的变更。其他节点也是做相同的操作。例如，所述事务的待执行内容分别为：

T0:要求对变量 A 加 1；

T1:要求对变量 A 加 2；

T2:要求对变量 B 从 2 加 3；

T3:要求对变量 A 减 3；

T4:要求对变量 A 减 2；

T5:要求对变量 B 从 2 加 1；

存在约束条件：A 和 B 不能小于 0；

在所述事务输入网络之前，节点的状态数据包括：

A：值为 1；

B：值为 2。

当共识序开始执行后：

执行 T0，A 的值变为 1+1=2；

执行 T1，A 的值变为 2+2=4；

执行 T2，B 的值为 2，从 2 加 3 变为 5；

执行 T3，A 的值变为 4-3=1；

执行 T4，A 的值 1-2<0，不符合所述约束条件，T4 舍弃；

执行 T5，B 的值为 5，无法从 2 加 1，T5 舍弃；

同时，状态获取模块从任意节点中获取状态数据。例如：

状态获取模块 0 向节点 0 请求 A 的值，A 向节点 0 返回此时 A 的值；

状态获取模块 1 已向节点 2 订阅了 B 的值，每当 B 的值更新时，节点 2 就向所述状态获取模块 1 推送 B 的值；

状态获取模块 2 同时从节点 0、1、2、3 请求 A 的值；

状态获取模块 3 同时从节点 1、2 订阅 A 的值；

实验模拟

为了更好地获取数据，本实验对事务的数据结构进行了简化，每个事务省略去了具体事务信息，用编号代替。通过 100 笔事务，4 个全节点来模拟整个原型系统。

原型适用 Go 语言开发，在 Centos+Docker 环境中运行，实验输入数据如图所示。由客户端（事务输入模块）按顺序向网络发送事务，每个事务随机选择入口节点。

```
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:134    ----- Analyze Sorted Knot List -----
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[6]Payload="0"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[8]Payload="1"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[10]Payload="2"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    e11dfe7f9c2f4[11]Payload="3"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    6d8aced649a52[11]Payload="4"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    e11dfe7f9c2f4[15]Payload="5"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    e11dfe7f9c2f4[18]Payload="6"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    6d8aced649a52[17]Payload="7"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    f8df87497b145[22]Payload="8"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[23]Payload="9"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[26]Payload="10"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[28]Payload="11"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    6d8aced649a52[25]Payload="12"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    f8df87497b145[31]Payload="13"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    f8df87497b145[33]Payload="14"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    6d8aced649a52[32]Payload="15"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    f8df87497b145[37]Payload="16"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    f8df87497b145[39]Payload="17"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    e11dfe7f9c2f4[38]Payload="18"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    f8df87497b145[42]Payload="19"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    e11dfe7f9c2f4[44]Payload="20"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    e11dfe7f9c2f4[47]Payload="21"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    e11dfe7f9c2f4[49]Payload="22"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    6d8aced649a52[48]Payload="23"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[52]Payload="24"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[55]Payload="25"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    6d8aced649a52[53]Payload="26"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    6d8aced649a52[56]Payload="27"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    e11dfe7f9c2f4[60]Payload="28"
2020-11-15T00:57:58.833+0800    INFO    calc/analyze_knot.go:136    b72b0f66b294d[61]Payload="29"
```

图 3 实验原型输入数据

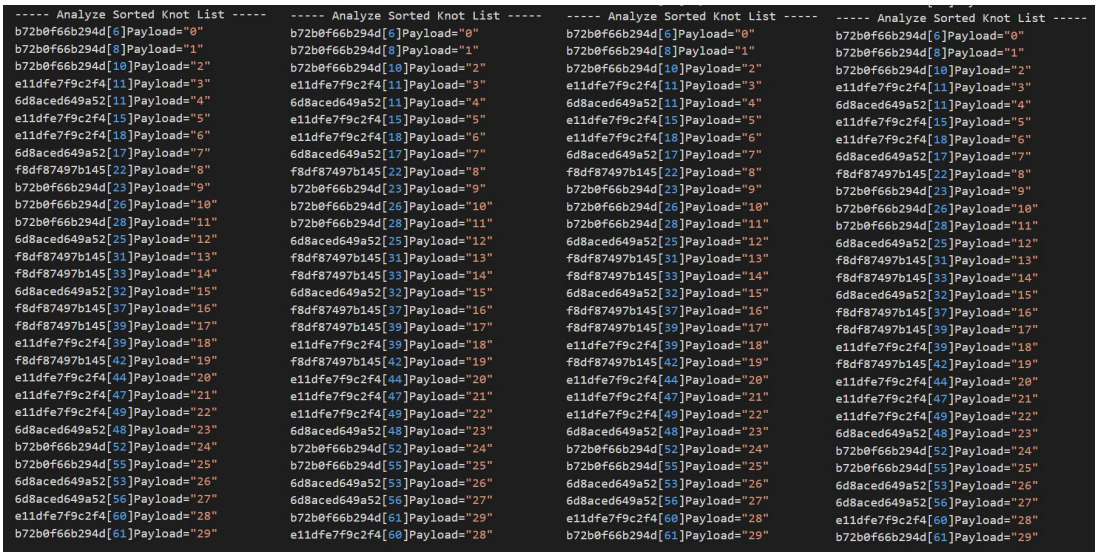

图 4 实验原型输出数据

实验结果证明，本原型事务共识顺序与输入顺序基本一致，事务时序得到有效还原。

经济模型

我们认为，一个事务被打结到时间结绳上，本身就是一件很有意义的事。

时间结绳平台本身不发行同质化的数字货币，是真正的“无币”项目。但是共识绳可以向应用层无限供应 NFT，应用层可以自行决定是否发行应用内数字货币。所有基于时间结绳技术发行数字货币的项目需自行解决合规性问题，且自愿使用时间结绳技术。时间结绳平台与应用解耦，只对交易的数字摘要进行定序，对交易内容不知情，予以免责声明。

我们已建立了时间结绳社区，接受时间结绳技术支持者的捐赠（优先接受数字人民币），捐赠记录会被打结到时间结绳上生成 NFT，并将捐赠内容向全网公开。

时间结绳社区欢迎志愿者加入，积极为公社发展做出贡献，贡献记录会打结到时间结绳上生成 NFT，并按照志愿者意愿决定是否公开。

时间结绳社区编制并实时发布时间结绳生态指数。合规上线时间结绳生态股指期货。优先采用数字人民币及其合规稳定币计价。

时间结绳社区不设立实体办事处，声明注册地为中国，遵守中华人民共和国法律，接受中国政府监管，积极从事公益慈善事业。

路线图

2019 年 10 月，在 NASAC2019 区块链论坛与专家讨论“网络内生时间”话题；

2019 年 12 月，“一种网络内生时间协议”立项，开展预研；

2020 年 7 月，在陕西省国资委党校干部培训班上作为讲师介绍项目立意；

2020 年 8 月，项目更名为“时间结绳”，并申请“时间结绳”商标；

2020 年 10 月，项目获得 200 万元人民币种子轮投资；

2020 年 10 月，“时间结绳”原型代码开始开发；

2020 年 11 月 15 日，“时间结绳”原型跑出第一组测试数据，符合原理预期；

2021 年 4 月 7 日，“时间结绳”商标 9 类和 42 类授权；

2021 年 4 月 16 日，正式向中国专利局提交《一种事务定序方法和系统》发明专利申请书，全文共 97 页，7.1 万字，40 张附图；

2021 年 4 月 17 日，在中关村区块链产业联盟社群发布《时间结绳技术白皮书》。

2021 年 4 月 26 日，在 github 上开通 JieSheng 开源项目。

2021 年 5 月 18 日，正式发布时间结绳项目白皮书，开启白皮书滚动更新发布。

2021 年 5 月 18 日，宣布成立“时间结绳社区”，暂由西安宽勤标准化技术服务事务所有限公司（Wide Service Standard Lab）代管，接受全球捐赠，开启应用生态建设。

2021 年 6 月，筹备在中国大陆境外设立“时间结绳社区基金会”。

2021 年 6 月，发布首个基于时间结绳的 NFT 应用项目白皮书。

2021 年 6 月 30 日，计划初步完成时间结绳内部测试网络，陆续发布更多测试数据。

2021 年 8 月 31 日，计划开放时间结绳测试网络和测试应用。

2022 年 4 月 16 日，正式上线时间结绳主网。

结语

当人类涌入硅基文明之后，稀缺性开始凸显，事务因果不可忽略，公平竞序就成了必然要求。发端于狭义相对论的传统分布式共识理论和区块链技术无法满足这一要求。为此，本文提出了时间结绳理论、技术、系统、项目，未来将不断更新。

参考文献

[1] Lamport, Leslie. Time, Clocks, and the Ordering of Events in a Distributed System[J]. Communications of the Acm, 1978, 21:558-565.