Celer Network

构建匹配互联网规模的区块链应用入口平台

中文白皮书（社区翻译版）

摘要

正如九十年代的拨号上网无法支持 4K 视频直播，扩展性不足正严重制约着当今区块链技术的应用落地。现有区块链效率低下是因为每个计算操作都需要经过绝大多数节点的重复处理从而达到链上共识，而这恰恰违背了设计高性能可扩展分布式系统的最基本原理。除此之外，链上共识也缺乏对隐私的保护，因为所有节点的全部历史交易记录都对外可见。虽然共识算法自身也在不断地改进发展，但其无法摆脱链上共识根本上的局限性。

链下扩容技术允许相互不信任的主体之间在链下而非链上进行智能合约交互。参与方共同维护并各自复制一个多签名无法篡改的链下状态机，仅在绝对有必要（如多方无法达成一致）时才会诉诸于链上共识。链下扩容是在保障区块链无需信任与去中心特性的同时实现能够横向扩展且保护隐私的分布式应用（dApp）的唯一方式。它是区块链技术大规模普及落地的转折点，并将成为所有可扩展 dApp 的引擎与基石。

Celer Network 是一个可达互联网规模、无需信任且保护隐私的区块链平台，能够让所有人在该平台上快捷地开发、运行与使用可高度扩展的分布式应用。它并不是一个独立的区块链，而是一个可以广泛运行在现有和未来区块链之上的网络系统。Celer 以其在链下扩容技术和加密经济学上的创新为区块链平台提供了前所未有的高性能和灵活性。

Celer Network 采用了具有清晰抽象的分层架构，使该平台每个单独部分都能够快速迭代，包括：

- 能够支持任意链下状态快速转换的广义状态通道与侧链组件。
- 具有最优证明可提供数十倍于当今最先进链下支付方案之速度的价值传输路由算法。
- 功能强大且易用的链下应用开发框架与运行环境。
- 与能够为链下生态提供网络效应、稳定的流动资金和高度安全可用性的新型加密经济模型。

目录

1.介绍

现代的很多经济活动本质上是信息和价值的流动和交换。在过去的两百年里，信息的传递已经从需要通过信鸽式传输的离散事件转变成光速般持续性传输的的互联网，然而，价值的传输方式却明显落后：离光速般的传输速度相差很远，并且依旧是由独立的经济储备所控制。这种不匹配导致了在经济演变过程中的一个毁灭性瓶颈：不管信息流动多快，昂贵且缓慢的价值交易都会限制信息和价值的有效交换。

随着区块链技术和激励平衡的分布式共识的到来，对互不信任方建立起了革命性的信任基础，打破了独立的金融孤岛，大幅度扩大了全球价值流动的范围和自由性。然而，尽管潜力巨大，但与传统价值传输工具相比，由于其处理能力较低，区块链实际上远达不到“光速般”的需求，所以提高可扩展性是区块链技术被大规模采用所必经历的一个挑战。

我们设想的未来的分布式生态系统是：人类、计算机、移动终端和物联网设备之间可以大规模的进行安全、私密的且去信任化的信息和价值交换。为了实现这一点，区块链的扩展性应该和互联网的规模相匹配，支持每秒数亿或数十亿的交易。但是鉴于现有区块链的处理速度（每秒几个或几十个交易），是否真的有可能像互联网这样？答案是肯定的，但只可能是在链下扩展的情况下实现。

虽然链上共识是区块链的基础，但它的局限性也很明显。从某种意义上来说，共识和可扩展是对立的。对于任何分布式系统，如果所有节点都需要在每个事务中达成一致，那么它的性能不应该比通过单个节点处理每个事务的中心化系统更好（事实上，在通信上的开销会使这种情况更糟糕），这也意味着系统最慢的节点最终会成为整个共识系统的瓶颈所在。链上共识也存在隐私上的隐患，因为所有的交易都是永久公开的。已经提出的一些链上共识的改进方案，包括分片和各种变种的 POX（Proof-of-X）证明机制，为了使区块链更快，对性能、分布式、安全性和不可篡改进行了不同程度的折衷处理，但无法从根本上改变链上共识的局限性。

为了使区块链系统拥有互联网一样的高扩展性，具有更好的隐私性，并且无需在可信任和分布式之间进行妥协，我们必须对链上共识方案进行更有远见的改进。设计高扩展的分布式系统的核心在于使不同节点的操作达到最大的独立。这种看似简单的见解意味着，完全扩展分布式应用程序的唯一方法就是将大多数的

事务处理引入到链下，尽可能地避免在链上达成一致，并且只有在最坏的情况下才会在链上使用它相关的技术包括状态通道（cChannel），侧链（side-chain）和链下计算。尽管潜力巨大，但链下扩展技术仍处于起步阶段，遇到的一些技术和经济方面的挑战仍未解决。

为了使链下扩展技术达到最好的使用效果，我们提出了 Celer Network，一种紧密结合的将互联网的扩展性移植到到现有的和未来区块链世界的架构。Celer Network 由精心设计的链下技术栈组成，该技术栈通过强大的安全性和隐私性来保证实现高度的可扩展性和灵活性，并通过博弈的加密经济模型来平衡各方面的权益。

1.1 Celer 技术栈

作为一个可以建立在现有或未来区块链上的全面综合的平台，Celer Network 包含一个清晰的分层架构，将复杂的链下平台分离为各个层级模块。这种架构大大降低了系统设计、开发和维护的复杂性，以便每个组件都可以轻松迭代并适应变化。

一个设计良好的分层架构应该具有开放的接口，只要它们支持相同的跨层接口，就鼓励在每个层上启用不同的实现。每一层只需要专注于实现自己的功能。受互联网成功的分层设计的启发，Celer Network 采用了一种链下技术堆栈，可以在不同的区块链上构建，命名为 cStack，它由以下几层组成，按自底向上的顺序排列：

- cChannel：广义状态通道和侧链套件
- cRoute：最佳价值传输路由
- cOS：链下应用程序的开发框架和运行环境

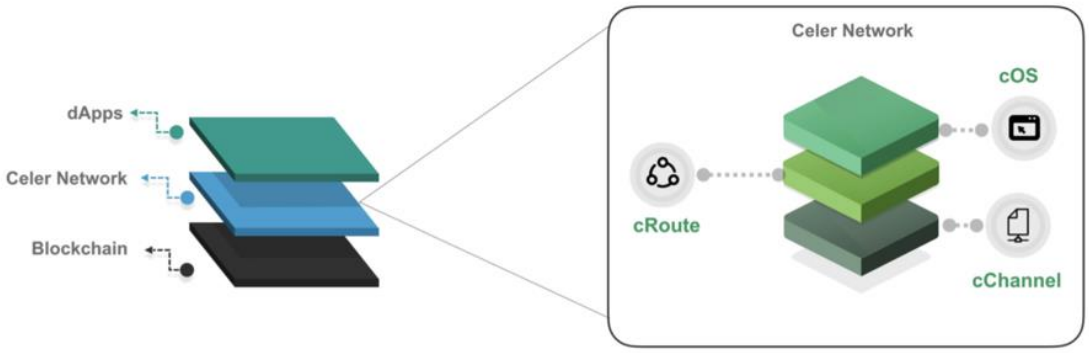


图 1.Celer Network 分层架构

Celer 架构为所有层提供创新解决方案。下面我们详细介绍 cChannel，cRoute 和 cOS 的技术挑战和关键特征。

cChannel.

这一层是离底层区块链最近的一层，直接与基础公链产生交互，并且在有限的时间内，为有共同抽象结构的上层提供实时状态更新。cChannel 使用状态通道和侧链技术，这些都是链下扩展平台的基石。

状态通道允许互不信任的各方在链下程序上迅速与最新约定的状态达成一致，并通过链上的债权合约保证其不可篡改的安全性。这一理念最初是由闪电网络引入，以支持高吞吐量的链下比特币的小额交易。自闪电网络理念提出以来，已经有一些研究工作在支付通道网络的背景下解决了不同的问题，例如路由算法，时间锁定优化。然而，链下网络仍旧处于早期发展阶段，在其模块化、灵活性和成本效益方面面临一些重大的挑战。cChannel 通过创建一系列新的功能来应对当前的挑战。

• 广义链下状态传输。链下传输可以是依赖于 DAG 的任意状态传输。这使得 Celer Network 能够支持复杂的高性能离线 dApp，例如游戏，在线拍卖，保险，市场预测和去中心化交易。

• 灵活和高效的价值传输。提供了多种状态通道和侧链结构，多样的效率和不可篡改性的权衡策略，以支持具有通用条件的快速价值传输，最小化的链上交互以及最小化资金锁定。

• 纯链下合约。任何与链上存款不直接相关的合约都不需要任何链上的操作或初始化，除非会由此引发争议。每个纯粹的链下合约或对象都有一个唯一可识别的链下地址，只有在需要时才在区块链上进行部署，并且由内置的链下地址转换器分配链上地址。

cRoute.

Celer Network 是一个高度可扩展的 dApps 平台，在平台上提供支持高吞吐量的价值传输是其最重要的承诺之一。链下价值传输是许多链下应用的基本要求。虽然 Celer Network 有着比支付解决方案更大的愿景，但它还是对链下支付路由进行突破性的改进，因为它直接决定了生态系统内可以传输多少价值以及传输多快。

所有现有的链下支付路由，都可归结为传统的“最短路径路由”算法，由于链路模型的根本差异，这可能会让链下支付网络具备较差的性能。计算机网络的链路容量是稳定且无状态的（不受过去传输的影响）。然而链下支付网络的链路容量是有状态的（即由链上存款和过去支付确定），这导致在高度动态的网络中拓扑结构和链路状态会不断变化，这使传统的最短路径算法很难收敛，从而产生低吞吐量、高延迟甚至中断的结果。

Celer Network 的支付路由模块意识到了这一根本挑战，cRoute 介绍了使用分布式拥塞梯度的的分布式路由平衡算法（DBR）。我们在 3.2.2 章节详细介绍了 DBR 一些特性。

- **可证明的最佳吞吐量** 我们证明了对于任何一笔支付交易请求速率，如果存在可支持该速率的路由算法，则 DBR 就一定能算出这个算法。根据我们的评估显示，与最先进的解决方案相比，DBR 的吞吐量提高了 15 倍，信道利用率提高了 20 倍。
- **透明通道平衡** 自“闪电网络”以来，“保持通道平衡”一直是一种直觉要去做的事。然而，现有的尝试都是探索式的将大量的链上或链下进行协调来达到低保证的平衡。DBR 将信道平衡过程与路由结合起来，并保持网络的平衡，而不需要任何额外的协调。
- **完全分散** DBR 算法是一种完全分散的算法，其中每个节点只需要在状态通道网络拓扑中与其临近的节点进行通信。DBR 在协议中的消息传递成本也很低。
- **故障弹性** DBR 算法对故障具有很高的弹性：它可以对无响应的节点进行快速检测并适应，支持剩余可用节点上的最大可能吞吐量。
- **隐私保护** 由于其多路径性质，DBR 算法自然而然地保留了有关传送值的隐私，而不需使用任何额外的隐私保护技术（例如 ZKSNARK）。更重要的是，DBR 算法可以与洋葱路由，无缝集成以保护源和目的地的匿名性。

cOS.

链上的 dApp 只是一个简单的连接区块链的前端，链下 dApp 虽然具有高扩展性的巨大潜力，但要将它构建在传统的公链上并不容易。Celer Network 引入了 cOS，这是一个能让每个人都可以轻松开发、操作和与可扩展的链下 dApp 进行交互的开发框架，从而减少了由于链下扩展带来的额外的复杂性。Celer Network 让开发人员更专注于应用程序逻辑的开发，并创建最佳的用户体验，而 cOS 则处理繁重的工作，包括以下任务：

- 找出任意链下状态和链上状态的依赖关系。
- 处理链下状态的跟踪，存储和争议。
- 中间节点故障容错和透明。
- 支持多个并发的链下 dApp。
- 统一实施到不同的链上和链外模块。

1.2 Celer 加密经济

Celer Network 的加密经济机制——cEconomy，是基于一个基本原则设计的：一个好的加密经济模型(通证模型)应该提供额外的价值并且可以引入新的动态博弈，否则就不那么好，或者没有用。然而任何解决可扩展问题的链下解决方案，都是在做折衷，如果这种方案没有可以使新动态来平衡这些折衷的加密经济的话，它也不会被采用。

新的折衷方案

一切都是有代价的。通过做出以下折衷，链下平台可获得可扩展性。

- **链下可扩展 vs 流动性**

链下操作网络首先通过牺牲链上网络的流动性来获得可扩展，这对潜在的链下服务提供商来说尤其具有挑战性，因为需要大量的流动性来提供高效的状态通道服务，然而，持有加密资产的大户，可能没有商业兴趣或者没有技术能力去构建一个状态通道服务的基础设施。有能力构建可靠的可扩展的状态通道服务能力的人，又通常没有如此大量的资金，存放在通道或欺诈证明债券合约中。这种不匹配给大量链下操作网络运用的技术发展带来了巨大障碍。

- **链下可扩展 vs 可用性**

尽管链下扩展不会对区块链的无信任属性做出任何妥协，但它确实牺牲了可用性。每个状态通道或链下合约都会关联一个时限，并且当一方长时间保持离线且超过一定时限，相关方将处于危险中，或者丢失当前状态。

因此，我们需要一个激励机制来提供足够的流动性，帮助那些有能力运行可靠和可扩展的离线服务基础设施的实体确保离线状态总是可用于可能的链上争端。

新的加密经济

为完成链下扩展解决方案,我们引入了一套名为cEconomy的加密经济机制，通过Celer Network的协议token（CELR）和三个紧密耦合的组件来提供网络效应、稳定的流动性和高可用性，以及不可或缺的价值。

- **流动性承诺证明 （PoLC）**

PoLC是一个虚拟的“挖矿”过程,为链下生态系统获得丰富而稳定的流动性。任何人要参与进来，只需将他闲置的流动性锁定一段时间即可获得奖励。

- **流动性支持拍卖 （LiBA）**

LiBA链下状态通道服务提供商能够通过获取一定谈判利率的“众包贷款”方式获得流动性。贷方根据称为“happiness scores”的积分进行优先级排列，这些积分由期望利率、提供的流动性数量和CELR代币的数量决定。拥有更多CELR代币（作为他们过去对生态系统贡献的指标）的贷方有更高的优先权。

- **状态守护网络 （SGN）**

SGN是一种特殊而严谨的侧链，在用户离线时守护用户的状态，以便用户的状态始终可用于处理纠纷案件。状态守护者需要将他们的CELR锁入SGN，以获取守护者资格并赚取服务费用。

第5节详细介绍了cEconomy机制，并分析了CELR价值和模型的可兼容性激励。

2. cChannel：链下扩容的基础

Celer Network 中 cChannel 的作用是提供一个框架，以实现高灵活性与高效的状态通道和侧链网络。本节从广义通道的构造开始，并概述了支持链上可验证状态之间任意条件依赖的关键元素。然后我们将视野扩展到传统的状态通道之外，研究如何将侧链封装到暴露于上层的统一接口中。

2.1 广义状态通道

2.1.1 核心思想和一个简单例子

现有支付网络解决方案的一个主要局限在于缺乏对广义状态转换的支持。随着智能合约平台（如以太坊）的兴起，对广义状态转换的需求也随之而来。智能合约能支持基于任意合约逻辑的异步价值传输。可以利用链下状态通道的概念来提高区块链的可扩展性，将链上状态转换放到链下状态通道中，并且相应的价值传输应该知道这种状态转换。

我们用一个简单的有条件支付的例子来说明如何将链上状态转换为链下状态转换的核心思想。假设 Alice 和 Carl 想要下一盘棋，同时以一种无需信任的方式打赌游戏结果：如果 Carl 赢了，Alice 会给 Carl 1 美元，反之亦然。

简单的链上实现逻辑是这样的：我们可以创建一个智能合约，在游戏开始前 Alice 和 Carl 各自存一笔钱到合约里。Alice 和 Carl 通过调用链上合约的函数来下棋。当其中一人输掉、投降或超时时，获胜的一方将赢得对方的赌注金。赌注金可以被看作是条件触发时（即对方赢的条件）的奖励。但是，链上合约调用又慢又贵，因为每笔交易都涉及链上交易。

链下状态通道可以在保持相同语义的同时显著的提高可扩展性。假设 Alice 和 Carl 之间有一个支付通道。为了实现上述语义，我们需要扩展通道的状态证明的功能,包括一个取决于游戏赢家状态的条件锁。然后，Alice 可以向 Carl 发送一个有效地链下条件支付，并说：“如果合约判定 Carl 赢得游戏，我将支付 Carl 1 美元”。游戏状态转换也可以移到链下。最直接的方法是仍然有一个管理游戏规则的链上合约，并且该合约的地址在有条件的支付中被引用。所有的状态转换都发生在链下相互签名的游戏状态中，这些状态可以在必要时写到链上合约中。

但事实上，由于对程序状态没有任何形式的价值保证要求，所以整个游戏合约和相关状态可以始终保存在链下，只要双方是相互合作的。唯一的要求是相关游戏状态在需要时可在链上验证。链上可验证的状态意味着其他合约或对象可以明确的引用它。为了实现这一点,我们需要一个引用转换合约,将链下引用(如合约代码中的哈希、构造函数参数和随机数)映射到链上引用(合约地址)。有了这些构造,Alice 和 Carl 之间的游戏只涉及一个长期的链上合约,它不是特定于游戏逻辑,也没有链上操作或进行游戏的初始化。

上面的例子专门反映了一个简单链下实例的设计模式，它可以更复杂。有条件支付比简单的布尔条件更复杂,可以设计为基于任意合约逻辑重新分配锁定的流动性。事实上，条件支付只是一个更广义的条件状态转换的特例。要实现多跳状态中继的通用模式，通道依赖性会比一对一依赖性更复杂。我们将在以下章节详细介绍技术规范。

2.1.2 设计目标

我们的首要目标是实现快速，灵活和无需信任的链下交互。我们希望大多数情况下，链下的状态转换将一直保持直到最终解决。因此，我们的目标是优化常用的链下模式，使其支持与内置的链上组件之间进行简洁的交互。

我们的第二个目标是设计适用于不同区块链的数据结构和对象交互逻辑。Celer Network 旨在构建一个底层区块链之上的平台，并支持智能合约在不同的区块链上运行。因此，需要一个通用的数据结构模式和中间层。

除了这两个突出的目标之外,我们还计划使用通道状态机的形式规约,并验证安全属性以及更改这些状态的通信协议。我们还应尽可能提供有效的链上解决机制。

2.1.3 一般技术要求

在本节中，我们采用自顶向下的方式为 cChannel 广义状态通道的核心组件提供规范说明，并描述了**通用状态通道接口**，该接口适用于任何具有价值传输和任意合约逻辑的状态通道。对于不同的具体用例，可能会进行广泛的专业化和优化，但原则保持不变。

在详细说明广义状态通道之前，我们首先介绍将在本节中使用到的几个重要的符号和术语。

• **（状态）**。 通过 s 表示通道的状态。对于一个双向支付通道，s 代表双方的可用余额;对于棋盘游戏，s 代表棋盘状态。

• **（状态证明）**。 状态证明作为链上合约和链下通信协议之间的桥接数据结构。 状态证明 sp 包含以下字段

$$sp = \{\Delta s,\ seq, merkle\_root,\ sigs\}, \tag{1}$$

其中 Δs 表示到目前为止累积的状态更新。请注意，当给定一个基本状态 S0 和状态更新Δs，我们能生成一个具有唯一性的新通道状态 S。比如，在一个双向支付通道中，S0 表示双方的存款，状态更新Δs 表示从一个参与者到另一个参与者 token 数量转移的映射。seq 是状态证明的序列号。具有较高序列号的状态证明将禁用较低序列号的状态证明。 merkle_root 是所有未决条件组的的默克尔树的根，且对于在 cChannel 中创建状态之间的条件依赖是至关重要的。sigs 代表各方对于此状态证明的签名，状态证明仅当各方签名都存在时才有效。

• **（条件）**。 条件 cond 是条件依赖基本单位的数据结构，这也是有条件依赖的 DAG 被构造的地方。一个条件可以如下表示.

$$cond = \{timeout,\ *IsFinalized(args),\ *QueryResult(args)\} \tag{2}$$

这里，timeout 是指条件到期后的超时，比如说，对于依赖于棋局结果的条件，超时可以对应于棋局的最大持续时间（例如，数十分钟）。布尔函数指针 IsFinalized(args)被用于在条件超时之前检查条件是否已解决，函数的调用参数是特定于应用程序的。例如，在下棋时，参数可能像 args = [blocknumber]一样简单，用于查询是否在 blocknumber 之前已确认获胜者。另外，QueryResult（args）是一个结果查询函数指针，返回任意字节作为条件的解析结果。比如，在下棋时，参数可以是 args = [player1]用于查询 player1 是否是获胜者；在第二价格拍卖中，参数可能为 args = [participant1, participant2, ..., participantN]查询谁是赢家及各参与方应支付的金额（一般情况）。条件的解析过程首先是执行 IsFinalized(args)然后通过执行 QueryResult（args）查询结果。

• **（条件组）** 条件组 cond_group 是对于表示广义状态依赖的一组条件的高级抽象，一个条件组可以如下表示.

$$Cond\_group = \{\wedge, ResolveGroup(cond\_results)\}, \tag{3}$$

其中，∧表示在该条件组中包含的一组条件，每一个条件 cond ∈ ∧解析为一个任意大小的字节数组（即，cond.QueryResult（args）的输出），这些字节数组被一组解析函数 ResolveGroup(cond_results)进行处理，它们将所有的条件解析结果作为输入返回状态更新Δs。对于一个支付通道，每一个条件组相当于一个条件支付。比如条件是“如果 B 赢了五子棋游戏，A 付给 B 1 美元”的条件支付，对应于包含两个条件的条件组：哈希时间锁条件（用于多跳中继）和五子棋游戏条件（“B 赢得游戏”）。在两个条件都为真的情况下 ResolveGroup 函数返回由 A 转给 B 的 1 美元的结果。

现在我们可以为状态通道指定接口，状态通道 C 可以被指定为以下元组：

$$C = \{p, s0, sp, s, F, T\}, \quad (4)$$

p = {p1, p2, ...,pn}是指通道中参与者的集合，s0 是该通道在链上的原始状态（例如，支付通道中每个参与者的初始存款）。sp 表示通道中最新的已知状态证明。s 是指状态证明 sp 完全确定后的已更新的通道状态。 T 是将在稍后指定的状态证明的结算超时增量。F 包含了一组应该被每个状态通道实现的标准函数：

• ResolveStateProof(sp, cond_groups).该函数通过解析传进来的条件组去更新当前状态证明；

• GetUpdatedState(sp, s0).该方法通过链下的状态证明 sp 和链上初始状态 s0 来获取最新的状态；

• UpdateState(s).该方法允许将状态通道的当前解析状态 s 更新到链上；

• IntendSettle(new sp).该方法在结算超时之前开启一个新的挑战期，在这个挑战期内，方法会接收一个状态证明作为输入，如果输入的状态证明较新则更新当前的状态证明；

• ConfirmSettle(sp).在当前时间超过结算超时时间的情况下，该函数会验证并确认当前状态是否已结算完成；

• IsFinalized(args) 和 QueryResult(args)是解析条件依赖的入口，它接受带有必要参数的外部查询，来查询合约的相关解析。事实上，有些模式被使用的频率很高，在 cChannel 的实现中，我们将它们分离成预定义的函数接口；

• CloseStateChannel(s).该函数用于结束状态通道的生命周期，并根据最后确定的状态 s 分配必要的状态；

结算超时取决于上次调用 ResolveStateProof 或 SettleStateProof 的时间，结算超时的增量为 τ 。

依赖约束.

当我们在不同的状态通道之间创建依赖关系时，需要强制执行一些约束，以保证依赖性 DAG 的正确解析。假设状态通道 C1 依赖于状态通道 C2。那么要求 C1 的参与者应该是 C2 的参与者的子集，使得 C1 的参与者具有解析其依赖于 C2 的必要信息。

2.1.4 基础功能

上述抽象定义了广义状态通道的通用构造模式。在不同的区块链中，实际的实现可能会有所不同。例如，在以太坊，合约间的调用会有返回值，但在 Dfinity 中，合约间的调用仅触发已注册的回调函数。回顾状态转换虚拟机在多个区块链上的实现，我们确定了在实践中运行广义状态通道所必需的两个实用的通用程序，如下所示：

- **链下地址转换器(OAT)**。在上面的抽象中，条件和条件组与不同的函数相关联。这些功能应该是链上合约功能的引用，但由于程序（智能合约）状态并不一定与区块链的约束绑定在一起，所以不应该有链上存在的基本要求。将它们完全脱链的唯一障碍是对诸如 IsFinalized 和 QueryResult 等函数的引用可能存在歧义。为了解决这种歧义，我们可以在链上定义一系列规则来将链下引用和链上引用关联起来。链下地址转换器是为此而构建的。对于一个没有任何价值的合约，可以通过它的合约代码、初始状态和特定的 nonce 生成的唯一标识符来引用它。我们称这种独特的标识符为链下地址。在解析链上的条件时，需要部署所引用的合约，相关的功能（例如，IsFinalized 和 QueryResult）应该能够从链下地址转换到链上地址。为了实现这样的功能，OAT 需要能够部署合约代码和初始链上状态来获得链上的合约，并建立起链下地址到链上地址的映射。
- **散列时间锁定注册表(HTLR)**。 HTLR 通常用于涉及多个状态通道的情况下保证事务的原子性。例如，多跳中继支付（无条件或有条件）、不同 token 之间的原子性交换、多链之间的交叉链桥等等。HTL 可以完全在链下实现，但正如 Sprite 指出的那样，这是一种过度优化，实际上限制了链下的可扩展性。因此，Sprite 提出了一个中心注册表，所有的锁都可以引用。我们扩展并修改了 Sprite 的方法以适应 cChannel 的通用模型。实际上，HTLR 为充当锁的条件提供依赖端点（IsFinalized，QueryResult）。IsFinalized 接受散列值和区块编号，如果在区块编号之前注册了相应的 pre-image，则返回 true。

QueryResult 接受一个散列值，如果 pre-image 的散列值被注册则返回 true。这两个功能可以进一步简化为一个功能，但为了通用性，我们可以将它们作为两个简单地独立功能。请注意，HTLR 以及相关的 IsFinalized 和 QueryResult 始终处于链上。

2.1.5 即用特性

此外，我们需要研究通用模式，并利用即用特性增强某些链上组件，以简化相应的链下交互。**广义支付通道（GPC）**就是一个很好的例子。广义支付通道是符合广义状态通道规范的支付通道，因此可以进一步支持各种基于链上或者链下的条件支付。

我们首先在 GPC 的背景下确定更具体的抽象模型。s0 代表 p 中每个参与方的静态存款映射。s 代表每一方最后各自赚到的钱。SubmitStateProof 是在 SettleStateProof 被调用并确认状态证明之前提交状态证明并触发一个超时挑战期的函数。 IsFinalized 和 QueryResult 被用于检查此支付通道的状态是否已完成并查询当前余额。大家可能会疑惑为什么支付通道会有一个用于外部查询的接口。这是因为某些付款或状态可能依赖于锁定在 sp 中的某个特定条件支付或 s。ResolveStateProof 是最有趣的部分，因为这里面做了很多专门的优化，并且大大减少了链下交互的复杂性。我们稍后会对此进行讨论。GetUpdatedState 是一个用于根据初始存款和最终解析的 sp 计算各参与方净支出的函数。 CloseStateChannel 仅仅只是关闭通道并为每一方分配最终余额。通过这个基本模型，我们将进一步讨论如何优化 GPC 结构，以支持开箱即用的特性。

- **合作解决**

在大多数情况下，状态通道应用中双方是竞争的。因此，这给挑战期和最后结算增加了复杂性和费用，所以，cChannel 可以实现合作安排，其中交易对手不仅要签署最新的状态证明，还要对结果签名以表明此状态证明中描述的状态更新确实是最终状态。由此，解决状态证明的交易数量可以从 2 减少到 1.

•单交易通道开放.

cChannel 带来的另一个优化是，将在链上打开一个通道的操作次数由 3 减少到 1。这是通过使用一个依赖的合约来保存交易对手的存款来实现的，N-1 个

交易方只需要签署能在链下取款的授权，另一个交易方提交授权到链上即可完成通道的开启。

•直接最终状态索赔

通常使用条件状态依赖来构建广义状态通道程序。在最终确定 GPC 时，一方可能希望避免遍历所有的条件依赖关系，这是为了限制对手的作恶行为，对方可能会下线拒绝合作并将一些条件组转化为无条件状态更新。为了减少争议方所需的工作，我们引入了直接最终状态索赔的方法。它允许在线方直接声明最终状态，而无需实际执行任何的额外依赖关系遍历，也不需要对方签名，为了防止滥用，索赔者也要绑定部分防欺诈金额。当一个挑战期过后，这个状态会成为最终的状态而不需要执行其它的额外的操作。

• 动态存取款.

GPC 的一个常见需求是，当交易对手不在线时，可以实现无缝的链上交易。对于取款，为了满足上述需求，我们将介绍 IntendWithdraw 和 ConfirmWithdraw 这两个函数，IntendWithdraw 会在一个挑战期内改变初始状态 s0，对方可以提交反对意见 sp 去阻止改变。如果在挑战期 τ 之内没有争议，ConfirmToWithdraw 会被调用去确认分配提款。这两个函数的作用与 IntendSettle、ConfirmSettle 很像。存款很简单，因为它只改变基本状态 s0。

• 布尔条件组.

我们推测 GPC 的最常见用例是基于布尔值的条件支付。比如，“如果函数 X 或者函数 Y 返回 true，A 将付给 B 钱”，为了优化这种情况，我们调整了条件组和条件的接口。特别地，我们可以专门优化 ResolveGroup 函数来发布一个预定义的条件支付，如果任何条件解析结果（或任何条件结果的布尔值）都为 true。这样，我们省去了为 ResolveGroup 创建额外对象的麻烦和相应的多方通信开销。我们还将条件专化为布尔条件，以便我们要求依赖对象应该具有”isSatisfied”效果的接口，该接口基于查询的状态返回 true 或 false。

•存款分配条件组.

GPC 的另一个使用更广泛的场景是广义状态分配。我们通过引入另一种不同类型的条件组来实现，它只有一个条件。QueryResult 将直接返回一个Δs 更新的状态分配映射。这为 GPC 提供了一个更通用的插件。人们可以插入一些初始化时锁定部分资产的链下合约，这个合约不仅可以检查谁赢了这个游戏（布尔值），还可以检查获胜者用了多少步赢得游戏，然后通过执行某种计算来分配

资产。 有关各方可以生成一个条件组，该组引用他们各自同意的链下合约地址的检查函数。

可以为不同的模式定义更多的通用模式，但是上面的例子说明了这种优化的设计原则。

2.2 带侧链的可选通道模型

除了上面提到的广义状态通道模型，cChannel 也引用了一个由侧链促进的替代状态通道模型[1],例如，考虑多个用户需要互相支付的场景。用户可以将他们的存款集中到一个中央合约里，该中央合约就像一个侧链合约，与链下服务提供者一起扮演区块提议者的角色（与链下的中心服务提供者开成一个多方的中心），因此可以在一个中心内实现一对多的支付关系，通过参与者可以接受的一定级别的防欺保证金来确保链下服务提供者的诚实性。

具体来说，在 CelerNetwork 中，每个链下服务提供者都可以运行一个侧链辅助的状态通道

$$C = \{s,\ p^*,\ b,\ \tau\}, \tag{5}$$

其中 s 表示侧链状态，p*是单个区块提议者（链下服务提供者），b 是欺诈保证金，τ 代表终止超时。每个节点 i 都可以发送侧链交易到通道中的其它节点以更新最新状态。就像任何侧链交易一样，节点 i 不仅会签署这个交易，而且还会签署另一个交易，以证明它已经看到了这个交易包含在由 p*创建的区块中，签署的第二笔交易可以看做是节点 i 的证明。只要参与者拥有完整可用的区块数据，则侧链交易的最终状态就可以很快被确认。

与前面提到的通道模型相比，这个侧链辅助通道模型可能会带来以下好处：

•接收方不需要链上交易也不用在线。这继承了侧链的天然优势，原因是，接收者可以在侧链辅助的通道上兑换他们的资金，而不用执行任何的侧链存款操作。

•参与方的资金不必锁定。这在支付通道的场景下会非常有优势，当以侧链为基础的通道用于多方支付时，各方不需要在对方付款之前事前将其存款锁定（除了那些需要存放防欺诈保证金的区块提议者）。

但是，系统应该清楚的意识到该通道模型的以下缺点：

•仍然需要防欺诈保证金。在基于侧链的通道中，无论是直接的区块提案人还是其它提供审计和保险服务的参与者都需要防欺诈保证金。我们应该清楚的认识

到在最坏的情况下对区块提案者（即链下服务提供者）的流动性要求实际上是非常大的，因为，如果有足够多的人勾结，恶意的一方可以创造出无限的重复支出。

•数据可用性问题可能使最终结果更复杂。即使没有恶意的节点，侧链模型固有的最终性延迟仍然困扰着这个通道模型，特别是相关方不总是提供区块数据，数据可用性成为一个问题的时候。侧链将不可避免地重新组织，因此最终的结果将被延迟，而整个侧链在最坏的情况下可能被抛弃。

这些基于侧链的通道可以经由通用状态通道进一步相互连接。

3. cRoute：可证明效率最优的价值传输路由

3.1 状态通道网络路由的挑战

创建状态路由（或支付通道网络中的“支付路由”）是必须的：因为直接在每对节点之间建立状态通道的办法，成本太高并且需要锁定流动性，没有太多实用价值。因此，有必要建立一个由状态通道组成的网络，其中状态传输的应该以无需信任的方式进行。如何设计状态路由是最为关键的，它决定了状态通道网络的可扩展性，即，有多少交易、以多快速度在当前网络中传输。然而，现有的方案都满足不了海量交易和高速传输的需求。

现有的方案中，一些去中心化的支付通道网络采用了地标路由方案。例如，闪电网络采用地标路由方案，称作闪光。分布式借据信用网络 SilentWhispers 也采用了类似的算法。地标路由的关键思想是，通过中间节点确定从发送方到接收方的最短路径。中间节点，也称作地标节点，是一类具有高链接能力的重要节点。

雷电网络(一种支付通道网络)提出过几种路由实现方案，比如 A* 树搜索，它是最短路径路由方案的在分布式网络中的实现。另外，由于路由非常关键，但发现路由路径却较困难，所以雷电网络中的节点可以通过提供路径发现的服务，来收取一定的费用。

最新提出的 SpeedyMurmurs ，把支付通道中的可用余额也考虑了进来，使得最短路径路由算法（用于闪电网络和雷电网络）效率有所提高。具体地，SpeedyMurmurs 使用了 P2P 网络中常用的嵌入式路由算法，该算法首先构造一个前缀树，然后为每个节点分配一个坐标。每次转发支付都需要事先知道当前节点和终点节点的坐标。如果需要移除（当链接的资金不平衡时）或者增加（当被移除的链接收到新的资金时）节点间的链接，则需要调整网络的前缀树和所有节点的坐标。

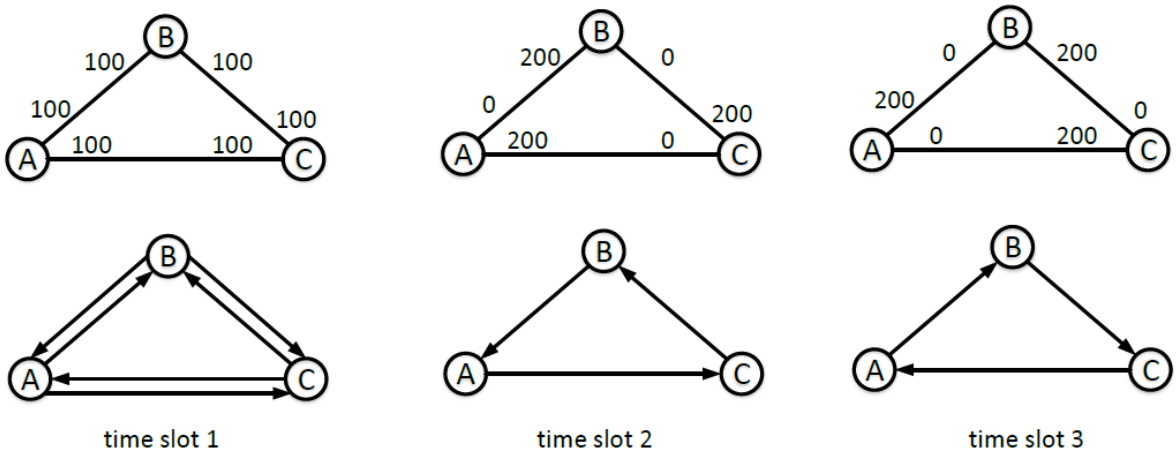


图 2. 最短路径算法导致网络拓扑频繁发生变动，导致通道不平衡。

如上所述，现有的路由机制都是“关注可用余额的最短路径算法”。在传统的信息网络中，最短路径算法之所以有高吞吐量和低延迟，是基于以下假设：网络拓扑保持稳定且链接是无状态的（不会因为以前传输过信息，现有传输信息的能力就发送改变）。但是，上述假设不再适合状态通道模型，因为状态通道模型是有状态链接模型，它传输的是价值而不是信息，每个链接的传输能力会因为传输过价值而发生改变。请注意，最短路径没有考虑通道的平衡性，这导致了每个链接会过快的耗尽其传输价值的能力，进一步的会导致网络拓扑频繁发生变化。如图二所示，它指出了最短路径算法导致网络拓扑上发生变动的一种情况，并展示了不同时刻上网络拓扑。假设，在时刻 1，节点 A、B、C 可以各自向其他两个节点支付 100 个币（如，A 有 200 个币，其中 100 个币用于 B 的支付，另外 100 个币用于 C 的支付）。此时，通道各端的价值是平衡的，每一对节点通过双向链路进行连接。使用最短路径路由算法，每个节点之间都可以通过直接路由进行支付。在时刻 2，节点 A、B、C 已经分别向节点 B、C、A 支付过 100 个币。此时，通道各端的价值分布变得不平衡，每一对节点的现在只能通过单向链路进行连接，其底层拓扑是逆时针方向的。在当前的拓扑下，使用最短路径路由算法，只能继续进行单向链路的支付。如果 A 持续不断地向 B 进行支付，并通过 C 进行路由 （A → C → B），这时状态通道的平衡性走向了另一个极端，底层拓扑变成了顺时针方向了（时刻 3）。相反，如果 C 向 A 只进行一次支付（100 个币），并通过 B 进行路由（C → B → A），那么每条通道就会重新得到平衡，网络拓扑也恢复最初的状态。对于任何去中心化的最短路径路由算法，如此频繁的拓扑变化将会导致性能变得很差劲，因为它的拓扑恢复需要时间（比如，SpeedyMurmurs 重

建前缀树）。因此最短路径路由算法，不是最优的算法。更糟糕的情况下，网络拓扑不停地在改变，不能恢复到最初的状态，最终就会让整个网络处在一个跪地不前的状态。

最近，Revive 项目提出了一种新的通道重新平衡的方案。但是，Revive 并没考虑到状态路由，这意味着通道是如何重新平衡，底层路由并不知道，并且该方案需要额外的协调。此外，Revive 的方案，只能用于循环的网络拓扑结构，不能保证其在通用的拓扑中可行。而我们提出了一种路由算法，其路由状态对通道是可见的，且其通道平衡能力是最好的。

3.2 分布式平衡路由(DBR)

我们提出一种，在链下状态通道网络上进行的，高性能价值传输路由协议：分布式平衡路由（Distributed Balanced Routing ,DBR）。DBR 算法源于无线网络中使用的背压路由算法。它的设计理念和传统的最短路径路由算法完全不一样。DBR 不会确切地计算，出发节点到目标节点的路径。取而代之的是，其路由策略基于当前网络的拥堵梯度（congestion gradients）。高山流水，奔向大海。水流的方向是从高往低，但是水自己不需要提前知道它到大海的路径，只需要跟着重力往下流就行了。

DBR 算法的设计理念也是类似的，它不需要考虑状态通道网络中链接的状态。需要特别指出的是，DBR 算法拥有持续平衡状态通道的能力，这使得价值传输能够高效的进行。与现有的路由算法相比，DBR 算法具有以下优势：

- **可证明效率最优的吞吐效率。** 如果有一个全知全能的“上帝”能成功让这个网络达到一个高效的吞吐效率，那么 DBR 也能做到这一点。也就是说，可以用数学证明其吞吐效率是最优的。
- **完全透明的通道平衡机制。** DBR 算法可以做到，在路由的过程中，不需要额外的协调方法，可以自然而然维护通道的平衡性，以保证价值转移能够长期平稳的进行。
- **完全的去中心化。** DBR 算法是一种完全去中心化的算法，在状态通道网络拓扑中，每个节点只需要与它的邻居进行通讯。同时， DBR 可以保证通信成本也非常低。
- **失败容忍性。** DBR 算法有非常高的鲁棒性来容忍失败。它可以快速检测到无响应的节点，并作出相应的调整。通过剩余的可用节点，达到最优的吞吐量。

- **隐私保护。** 因为 DBR 算法特性，从发出节点到终点节点可能有非常多的路径可以选择，所以自然而然的就保护了价值传输过程中的隐私，而不需要任何额外的保护隐私的技术（比如，ZKSNARK）。更重要的是，DBR 算法可以与洋葱路由无缝集成，来保护出发节点和终点节点的匿名性。

接下来，我们首先会介绍状态通道网络模型，然后介绍 DBR 算法，最后证明 DBR 算法效率最优。为了便于说明，我们只讨论了只有两个节点的双方支付通道，但是同样的想法适用于任何用于价值传输的状态通道网络。

3.2.1 系统模型

在我们的模型中，时间被分割成一个个固定长度的时刻。每个时刻的长度就路由一跳传输信息所消耗的时间。假设网络中有 N 个节点。对于每一对节点 i 和 j，如果存在一个双向支付通道，那么存在一对双向连接：i → j 和 j → i。记时刻 t 链接 i → j 的能力为： c(i,j,t)，其意义是在时刻 t 开始时，支付通道中节点 i 可用于向节点 j 支付的余额。每个节点 i 和节点 j 组成的双向通道中存款总和，记作：

$$c_{ij}(t) + c_{ji}(t) = B_{i \leftrightarrow j}(t),$$

B(i,j,t)的意义是在时刻 t 开始时，双向支付通道 i↔j 中存款的总和。注意，由于不断地有资金加入/退出链上资金池，B(i,j,t) 存款总和可能会随着时间而改变。

任意时刻 t，每个节点 i 都会收到该网络外的一个支付转移请求，这时所有需要传递给节点 k 的币的总量记作 a(i,k,t)。通常也用 μ(i,j,k,t) 来表示，意义是在时刻 t ，通过链接 i ↔ j 需要传递到节点 k 的币的总量。

3.2.2 协议描述

在描述协议之前，我们先介绍几个重要的概念：债务队列，通道失衡，拥堵失衡（congestion-plus-imbalance, CPI）度。

债务队列

在 DBR 算法操作时，每个节点 i 维护着一个债务队列，记录着需要转发给节点 k 的支付。队列长度 Q(i,k,t) 代表着节点 i 需要转发给下一跳的币的总量（需要转给节点 k，但是在时刻 t 开始，还没有来得及转）。简单地说，债务队列的长度就是每个链接拥堵的指标。该队列引入时间变量后的公式如下：

$$Q_i^{(k)}(t+1) = \left[Q_i^{(k)}(t) + a_i^{(k)}(t) + \sum_{j\in\mathcal{N}_i} \mu_{ji}^{(k)}(t) - \sum_{j\in\mathcal{N}_i} \mu_{ij}^{(k)}(t)\right]^+, \tag{6}$$

[x]+ = max{0, x} (队列的长度不可能为负数)，并且 Ni 表示节点 i 的所有邻居。上面公式意思是，在时刻 t，队列长度的变化由三个因素组成：(1) 从外界网络中转入的新的支付的所涉及币(比如，a(i,k,t))，(2) 邻居转给节点 i 的币，(3) 节点 i 转给邻居的币。此外，终点节点上的债务队列长度永远为 0. Q(i,i,t) = 0 这意味着，在 DBR 算法中，每个包最终会被传递到终点节点上。

通道失衡

对于每对链接 i → j，我们定义通道失衡为：

$$\Delta_{ij}(t) = \sum_{\tau<t}\sum_{k}\left(\mu_{ji}^{(k)}(\tau) - \mu_{ij}^{(k)}(\tau)\right). \tag{7}$$

Δ(i,j,t) 就是，在时刻 t 开始时，支付通道中节点 j 支付给节点 i 的币的总量，减去节点 i 支付给节点 j 的币的总量。如果 Δ(i,j,t) < 0 这意味着节点 i 支付给节点 j 的币，多于它从 j 那查收到的。简单的讲 ，Δ(i,j,t) 就是用于衡量通道上节点 i 的失衡指标。我们的 DBR 算法会尝试平衡每个 i ↔ j 支付通道，也就是随着时间的推进，Δ(i,j,t) 最终会等于 0，也就是将时间维度拉长来看，i 支付给 j 的速率等于 j 支付给 i 的速率。

拥堵失衡度

定义链接 i ↔ j 和终点 k 的拥堵失衡（CPI）度为

$$W_{ij}^{(k)}(t) = Q_i^{(k)}(t) - Q_j^{(k)}(t) + \beta\Delta_{ij}(t), \tag{8}$$

上述参数 ß > 0 ，是通道失衡度重要的影响因子。简单说，拥堵失衡（CPI）度，等于节点 i 和节点 j 需要支付给节点 k 的差异积压（二者债务队列长度的差值），加上二者之间的通道失衡值 Δ(i,j,t)。前者用于减少网络拥堵并提高网络吞吐量，后者用于平衡通道的余额。

分布式平衡路由(DBR)

每个节点 i 都会在本地执行下列协议：
在每个时刻 t，节点 i 首先和邻居交换队列长度信息，并计算拥堵失衡度 CPI。然后每条链接 i → j，节点 i 计算出经过这条链接的最佳的支付流：

$$k^* = \arg\max_k \quad W_{ij}^{(k)}(t). \tag{9}$$

If $W_{ij}^{(k^*)}(t) > 0$, then $\mu_{ij}^{(k^*)}(t) = c_{ij}(t)$ otherwise $\mu_{ij}^{(k^*)}(t) = 0$. For any $k \neq k^*$, set $\mu_{ij}^{(k)}(t) = 0$.

备注：在每个时间点 t，DBR 最终会尝试解决整体拥堵失衡（CPI）度最优的问题。

$$\begin{aligned} \max \quad & \sum_{ij}\sum_{k} \mu_{ij}^{(k)}(t) W_{ij}^{(k)}(t) \\ \text{s.t.} \quad & \sum_{k} \mu_{ij}^{(k)}(t) + \sum_{k} \mu_{ji}^{(k)}(t) \le B_{i\leftrightarrow j}(t), \ \forall i,j. \end{aligned} \tag{10}$$

上述最优问题也叫作 MaxWeight，上述算法描述给出了 MaxWeight 的近似解。在我们对 DBD 的理论分析部分（参见下一章节）会有更详细的解释。

3.2.3 DBR 的吞吐量性能

为了分析 DBR 的吞吐量性能，我们首先介绍一些定义。

- 一个状态通道网络被认为是**稳定**的，假如满足以下条件：

$$\lim_{t\to\infty}\frac{Q_i^{(k)}(t)}{t}=0,\ \forall i,k,$$

这蕴含着每一个欠债队列的远期抵达率等于那个队列的远期离开率。

- 一个状态通道网络被认为是**平衡**的，假如满足以下条件：

$$\lim_{t\to\infty}\frac{\Delta_{ij}(t)}{t}=0,\ \forall \text{ channel } i\leftrightarrow j.$$

换言之，对于每一个支付通道 i <-> j，从节点 i 到节点 j 的远期发送率等于从节点 j 到节点 i 的远期发送率。

- 定义

$$\lambda_i^{(k)}\triangleq\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}a_i^{(k)}(\tau)$$

为从节点 i 到目的地 k 的支付能力的长期平均抵达率。一个抵达率向量 lambda = (lambda_iˆ(k))_(i, k)被称作是可支撑的如果存在一个路由算法使得在这个抵达率向量以下网络依旧能保持稳定和平衡。

- 一个状态通道网络**的吞吐量范围**是可支持的达率向量的集合。
- 一个路由算法是否是吞吐量最优的评判标准取决于，在吞吐量范围内，它是否可以支持任意的支付到达率　能够保持任何支付抵达率向量始终在吞吐量区域当中。

为了方便阐述，我们假设外部支付抵达过程{a_iˆ{k}(t)}_t>=0 是平稳的而且具有稳定状态分布，而且对于每一个支付通道整体存款保持恒定，也就是对于任何 t>=0, B_i<->j(t)= B_i<->j。当抵达过程不平稳而且通道存款是随着时间变化的时候（例如动态链上存款/取款），我们的分析依旧能够成立，只是符号使用会不方便。下一个章节展示了 DBR 的吞吐性能。

定理 3.1 DBR 算法是吞吐量最优的。

换句话说，只要存在一个路由算法使得支付网络能够保持稳定和平衡，那么 DBR 算法也能实现它。3.2.3 剩下的部分就是对定理 3.1 进行证明。我们首先来引入一个论点，这个论点阐述了一个状态通道网络的吞吐能力。

论点 3.2 一个抵达率向量是可支撑的当且仅当存在流量变量使得下述条件得到满足

$$\lambda_i^{(k)} + \sum_{j \in \mathcal{N}_i} f_{ji}^{(k)} - \sum_{j \in \mathcal{N}_i} f_{ij}^{(k)} \leq 0, \ \forall k, i \neq k \tag{11}$$

$$\sum_k f_{ij}^{(k)} = \sum_k f_{ji}^{(k)}, \ \forall i, j \tag{12}$$

$$\sum_k f_{ij}^{(k)} + \sum_k f_{ji}^{(k)} \leq B_{i \leftrightarrow j}, \ \forall i, j. \tag{13}$$

证明：上述条件的必要性是微不足道的。不等式（11）对应着流量守恒要求。如果它被违反，在节点 i 的抵达率会大于离开率，状态通道网络将会呈现不稳定状态。等式（12）对应着通道平衡要求。如果它被违反，那么通道 i <-> j 就会变得不平衡。不等式（13）对应着通道容量限制要求，因为在每一个通道 i <-> j 当中传递的令牌总和不能超过通道存款总和 B i <-> j。

为了证明上述条件的充分性，我们建立了一个当抵达率向量 lambda 满足（11）-（13）时能够使得状态通道网络稳定且平衡的算法。这个算法直截了当：在每一个空隙 t，对于任意的 i, j, k 置路由变量 miu_ijˆk = f_ijˆk。显然，在这个路由算法下，因为 sigma = = =sigma，每一个通道 i <-> j 在每一个空隙 t 处都保持平衡。更进一步讲，这个网络之所以在这个算法下保持稳定是因为在每一个节点处流量守恒条件都得到了满足。注意到每一个连接 i <-> j 有可能因为没有足够的余额（也就是 c_ij < sigma）而导致路由决定不可行。在这种

情况下，为了使得余额在状态通道的两端保持平衡，我们可一个让节点 j 在开始时刻转移一部分令牌到节点 i。这样一个调整过程会导致对于每一个通道 i <-> j 有至多 B_i <-> j 个子优化转账，而且长期而言不会影响到网络稳定性和通道余额。

因此，方程（11）-（13）是抵达率向量 lambda 成为可支撑的充分必要条件。

应该指出，因为我们不能提前知道外部抵达率向量，在引理 3.2 的证明过程中提到过的路由算法不能在现实中实现出来。接下来，我们证明在提前不知道支付流量统计的条件下，DBR 能够达到相同的吞吐量性能。

根据论点 3.2，如果一个抵达率向量 lambda 属于吞吐量区域，它一定满足（11）-（13）而且可以被由在引理 3.2 证明过程中提到过的算法（称之为优化 Oracle 算法）来支撑。接下来，miu_ij^k(t)表示在空隙 t 处由优化 Oracle 算法做出的路由决定。根绝优化 Oracle 算法的本质，我们有对于任意的 t，miu_ij^k(t)=f_ij^k（如果忽略初始资金调整过程）。

我们定义李雅普诺夫函数如下：

$$\Phi(t) = \sum_{i,k} \left(Q_i^{(k)}(t)\right)^2 + \frac{\beta}{2} \sum_{i,j} \Delta_{ij}^2(t). \tag{14}$$

我们也定义条件李雅普诺夫漂移为，这里的期望是对抵达的随机性而言。为了方便分析，我们假设对于每一个空隙，抵达网络的新支付的数量被常数控制。根据方程（6），我们有

$$\begin{aligned}\left(Q_i^{(k)}(t+1)\right)^2 &= \left[Q_i^{(k)}(t)+a_i^{(k)}(t)+\sum_j \mu_{ji}^{(k)}(t)-\sum_j \mu_{ij}^{(k)}(t)\right]^2 \\ &= \left(Q_i^{(k)}(t)+\sum_j \mu_{ji}^{(k)}(t)-\sum_j \mu_{ij}^{(k)}(t)\right)^2+\left(a_i^{(k)}(t)\right)^2 \\ &\quad +2a_i^{(k)}(t)\left(Q_i^{(k)}(t)+\sum_j \mu_{ji}^{(k)}(t)-\sum_j \mu_{ij}^{(k)}(t)\right) \\ &\le \left(Q_i^{(k)}(t)\right)^2-2Q_i^{(k)}(t)\left(\sum_j \mu_{ij}^{(k)}(t)-\sum_j \mu_{ji}^{(k)}(t)\right) \\ &\quad +2a_i^{(k)}(t)Q_i^{(k)}(t)+const,\end{aligned} \tag{15}$$

这里不等式成立是因为我们有假设在每一个空隙 t 处，抵达 a_iˆk(t)是被常数控制以及事实在每一个空隙处被转移令牌的数量也是有界的（因为 miu_ijˆk(t)<=B_i<->j）。现在我们有

$$\begin{aligned}&\sum_{i,k}\left(Q_i^{(k)}(t+1)\right)^2-\sum_{i,k}\left(Q_i^{(k)}(t)\right)^2 \\ &\le const-2\sum_{i,k}Q_i^{(k)}(t)\left(\sum_j \mu_{ij}^{(k)}(t)-\sum_j \mu_{ji}^{(k)}(t)\right)+2\sum_{i,k}a_i^{(k)}(t)Q_i^{(k)}(t) \\ &= const-2\sum_{i,j}\sum_k \mu_{ij}^{(k)}(t)\left(Q_i^{(k)}(t)-Q_j^{(k)}(t)\right)+2\sum_{i,k}a_i^{(k)}(t)Q_i^{(k)}(t),\end{aligned}$$

这里我们重新排列了上述等式的求和。类似地，注意到

$$\Delta_{ij}(t+1)=\Delta_{ij}(t)+\sum_k \mu_{ji}^{(k)}(t)-\sum_k \mu_{ij}^{(k)}(t),$$

我们可以证明

$$\begin{aligned}\frac{\beta}{2}\sum_{i,j}\Delta_{ij}^2(t+1)-\frac{\beta}{2}\sum_{i,j}\Delta_{ij}^2(t) &\le \beta\cdot const-\beta\sum_{i,j}\sum_{k}\Delta_{ij}(t)\Big(\mu_{ij}^{(k)}(t)-\mu_{ji}^{(k)}(t)\Big)\\ &=\beta\cdot const-\beta\sum_{i,j}\sum_{k}\mu_{ij}^{(k)}(t)\Big(\Delta_{ij}(t)-\Delta_{ji}(t)\Big),\\ &=\beta\cdot const-2\sum_{i,j}\sum_{k}\mu_{ij}^{(k)}(t)\beta\Delta_{ij}(t),\end{aligned} \tag{16}$$

这里我们用到事实 Delta_ij(t)=-Delta_ij(t)。结合（15）和（16）我们得到条件李雅普诺夫漂移能够被如下所控制

$$\begin{aligned}D(t) &\le \beta c_1+c_2-2\sum_{i,j}\sum_{k}\mu_{ij}^{(k)}(t)\Big(Q_i^{(k)}(t)-Q_j^{(k)}(t)+\beta\Delta_{ij}(t)\Big)+2\sum_{i,k}\lambda_i^{(k)}Q_i^{(k)}(t)\\ &\le \beta c_1+c_2-2\sum_{i,j}\sum_{k}\widetilde{\mu}_{ij}^{(k)}(t)\Big(Q_i^{(k)}(t)-Q_j^{(k)}(t)+\beta\Delta_{ij}(t)\Big)+2\sum_{i,k}\lambda_i^{(k)}Q_i^{(k)}(t)\\ &=\beta c_1+c_2-2\sum_{i,j}\sum_{k}f_{ij}^{(k)}\Big(Q_i^{(k)}(t)-Q_j^{(k)}(t)+\beta\Delta_{ij}(t)\Big)+2\sum_{i,k}\lambda_i^{(k)}Q_i^{(k)}(t)\\ &=\beta c_1+c_2+2\sum_{i,k}Q_i^{(k)}(t)\Big(\lambda_i^{(k)}+\sum_{j}f_{ji}^{(k)}-\sum_{j}f_{ij}^{(k)}\Big)-\beta\sum_{ij}\sum_{k}\Delta_{ij}(t)\Big(f_{ij}^{(k)}-f_{ji}^{(k)}\Big)\\ &\le \beta c_1+c_2,\end{aligned}$$

这里 c_1 和 c_2 是某个常数，第二个不等式是由于 DBR 的操作（参看 10），最后一个不等式是由于（11）和（12）。利用迭代期望法则，我们可以得到

$$\mathbb{E}[\Phi(\tau+1)]-\mathbb{E}[\Phi(\tau)]\le\beta c_1+c_2.$$

加和指标 tao=0,…t-1,我们有

$$\mathbb{E}[\Phi(t)] - \mathbb{E}[\Phi(0)] \leq (\beta c_1 + c_2) \cdot t.$$

然后我们有

$$\sum_{i,k} \mathbb{E}\left[\left(Q_i^{(k)}(t)\right)^2\right] + \frac{\beta}{2} \sum_{i,j} \mathbb{E}\left[\Delta_{ij}^2(t)\right] \leq (\beta c_1 + c_2)t + \mathbb{E}[\Phi(0)]. \tag{17}$$

为了证明 DBR 可以实现通道平衡，我们从（17）注意到

$$\frac{\beta}{2}\left(\mathbb{E}[\sum_{i,j} |\Delta_{ij}(t)|]\right)^2 \leq \frac{\beta}{2} \sum_{i,j} \mathbb{E}\left[\Delta_{ij}^2(t)\right] \leq (\beta c_1 + c_2)t + \mathbb{E}[\Phi(0)],$$

这里第一个不等式成立是因为|Delta(t)|的变化量不能是负值

$$\mathbf{Var}(|\boldsymbol{\Delta}(t)|) = \mathbb{E}\left[\sum_{i,j} \Delta_{ij}^2(t)\right] - \left(\mathbb{E}[\sum_{i,j} |\Delta_{ij}(t)|]\right)^2 \geq 0.$$

因此我们得出

$$\mathbb{E}[\sum_{i,j} |\Delta_{ij}(t)|] \leq \sqrt{2c_1 t + \frac{2c_2 t}{\beta} + \frac{2\mathbb{E}[\Phi(0)]}{\beta}}.$$

因为 $\mathbb{E}[\Phi(0)] < \infty$ ，对于任何的支付通道 i <->j，我们有

$$\lim_{t\to\infty} \mathbb{E}\left[\frac{|\Delta_{ij}(t)|}{t}\right] = 0,$$

也就是在 DBR 算法下网络保证了通道平衡。

类似地，我们可以证明

$$\sum_{i,k} \mathbb{E}[Q_i^{(k)}(t)] \leq \sqrt{(\beta c_1 + c_2)t + \mathbb{E}[\Phi(0)]},$$

这蕴含了

$$\lim_{t\to\infty} \frac{\mathbb{E}[Q_i^{(k)}(t)]}{t} = 0, \ \forall i, k,$$

也就是说在 DBR 算法下，网络是稳定的。

3.3 DBR 的讨论

3.3.1 失败容忍性

由于 DBR 的适应性和多路径的特性，它有非常高的鲁棒性来容忍网络失败。例如，当一个节点不响应时，DBR 可以快速的适应，并通过其他的节点继续保持高吞吐量。

3.3.2 隐私

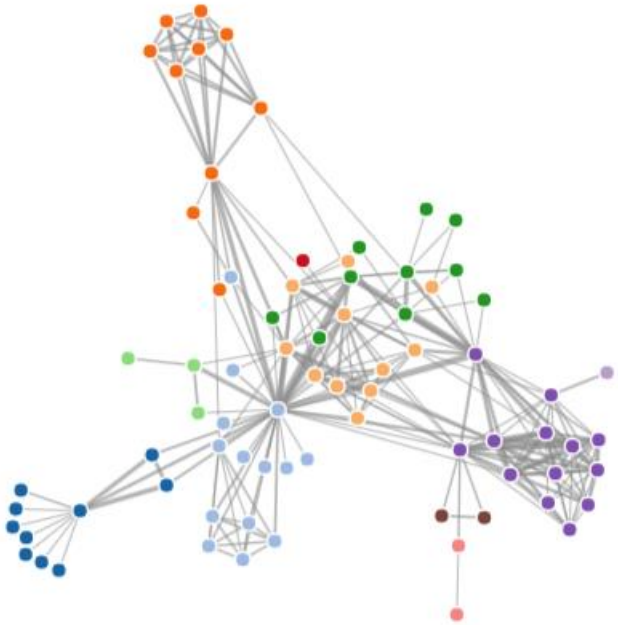
由于 DBR 多路径的特性，任何中间节点只能访问每个价值传输请求的一小部分信息。因此，在价值传输方面，DBR 算法本身天然就提供了很好的隐私保护。然而，在 DBR 中，为了将一笔价值传输请求放入合适的债务队列中，每一个节点都应该知道这笔请求的目的地址，如果我们还需要隐藏付款的最终地址，洋葱路由（TOR）可以与 DBR 一起配合使用。在洋葱路由中，信息会被封装在加密层中。加密信息会通过一系列网络节点，也叫洋葱节点，每一个节点需要“剥”(解密)一层“洋葱皮”，才知道下个节点是谁。当最后“一层皮被剥开”时，信息也就到达终点。我们可以在包含洋葱节点的混合网络中直接进行支付行为，，通过 DBR 算法去优化洋葱节点的支付路由。

3.4 模拟结果

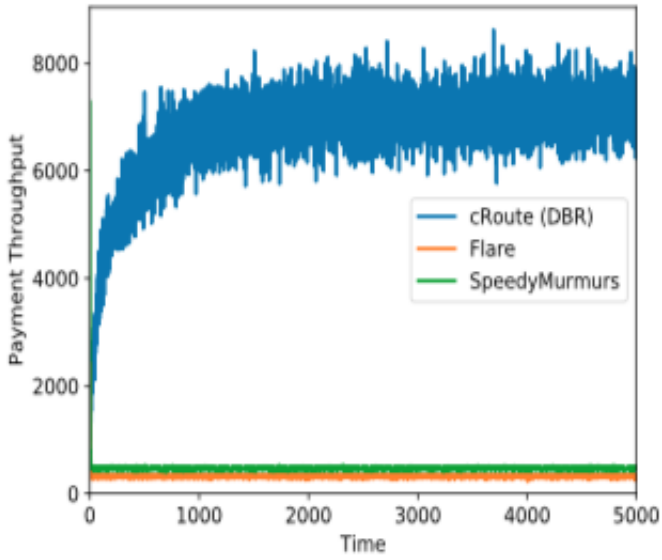
我们引入了现有的两种路由算法，通过模拟跑分来进行横向对比，包括：SpeedyMurmurs 和 Flare（闪电网络）。图三，给出了三者的模拟拓扑。

图四，给出了吞吐性能的对比。可以看出，与现有的路由算法相比，DBR 平均支付吞吐量提高了 15 倍。这是因为，DBR 算法拥有通道平衡和拥堵感知的特性。图五，给出了三者的通道利用率，DBR 有 90% 左右的通道利用率。与此同时，由于缺乏通道平衡的能力其他路由算法的通道利用率连 5% 都不到

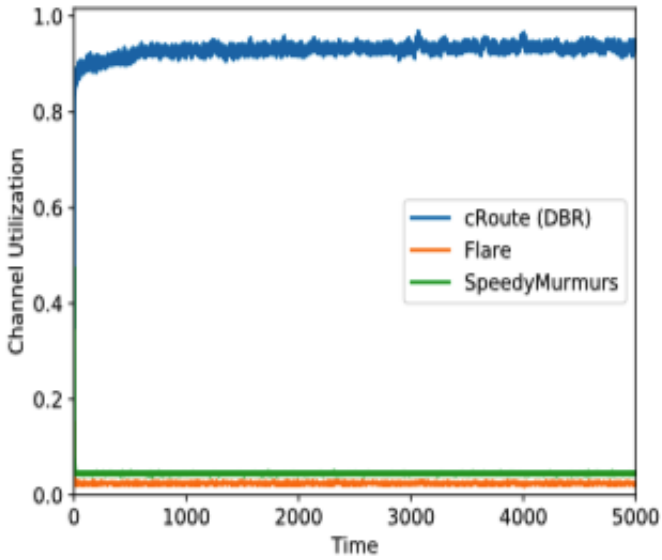
(通道利用率：在同一时刻，用于支付转移的币的总量，比上通道的币的存款总量。例如，通道中存了 100 个币，有 50 个币用于支付转移，那么通道利用率就是 50%。)



图三. 用于模拟的支付通道网络拓扑（77 个节点，254 个双向付费通道）。支付通道网络运行 40 个支付流，随机选择一对出发—终点节点。每个通道分配的初始存款在 100 到 200 个 token 之间。付款到达符合泊松过程，每笔支付的均值为 3 个币，其遵循几何分布。



图四. 三者支付吞吐量对比。DBR（平均值：6748 笔/时刻），SpeedyMurmurs（平均值：467 笔/时刻），Flare（平均值：316 笔/时刻）.



图五. DBR,SpeedyMurmurs 及 Flare 的通道利用率对比。更高水准的通道平衡性，造就了更高的通道利用率。

4. cOS：链下去中心应用操作系统

为了让每个人在不被由链下扩容而引入的额外复杂性所搅扰的同时都能够快速搭建，运营，使用可扩容链下去中心化应用，Celer Network 革命性地引入了更高一层的抽象： cOS，一个应用开发框架与运行时环境的结合体。这部分提纲挈领地阐释了 cOS 的愿景与设计目标。

4.1 条件依赖状态的有向无环图

在这一部分，我们不仅会提供一个关于链下应用搭建的抽象模型的看法，而且会阐述这个模型是如何与状态通道网络整合的。为了支持除 P2P 简单支付以外的更多使用场景，我们使用条件依赖状态的有向无环图作为链下应用系统的模型，这里每条边代表了它们之间的依赖关系。

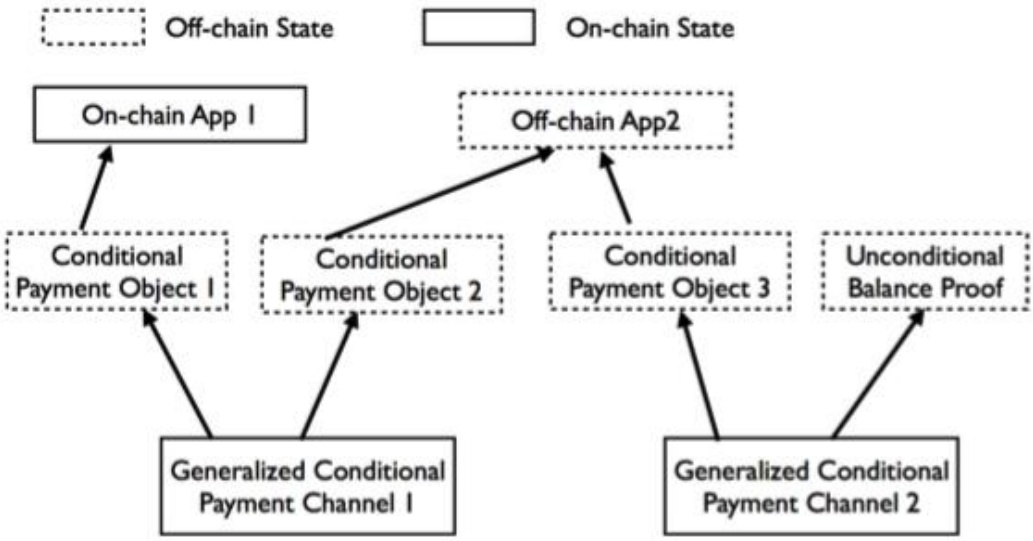


图6. 有条件依赖状态的DAG

图 6 诠释了这个系统模型，这里广义条件支付通道是和链上状态在支付网络里唯一建立起来的合约。这些链上状态的确定依赖于一个或者多个条件支付对象（例如条件支付对象 3），而这些对象是纯粹链下但却可以强制转化为链上的。我们想特意强调这些条件支付对象不仅是经过简单时间哈希锁死过的交

易，而且可以被设置为依赖链下应用合约状态，例如在图 6 中的“链下应用 X”。

这些条件支付对象能够通过多次类如简单无条件支付对象的连接来被相继传递。例如，支付通道 1 可以是连接 Alice 和 Bob 的通道，同时支付通道 2 可以是连接 Bob 和 Carl 的通道。假设“链下应用 2”是 Alice 和 Carl 玩得一场链下象棋游戏，并且假设 Alice 想表达“如果 Carl 赢得这场游戏，Alice 将会支付 Carl 10 ETH”。尽管没有一条直接连接 Alice 和 Carl 的通道，但是 Alice 可以用两层条件锁通过 Bob 向 Carl 发送一个条件支付。第一层经过简单时间哈希过的死锁是用来确保 Bob 在一个合理的时间内传递并且处理了这笔支付。第二层会根据象棋游戏的结果来锁定这笔支付。通过这个两次相继传递，尽管 Bob 没有参与到象棋游戏，但 Alice 和 Carl 之间的条件支付也能够通过 Bob 来完成结算。这个简化的例子阐释了由广义状态通道形成的依赖图，条件支付对象和链下应用是如何支持任意复杂的多方交互的。

这里需要注意的是，链下状态并不一定只能依赖于别的链下状态。举个例子来说，Alice 可以对 Carl 进行一个“条件支付”，这个”条件支付“依赖于 Carl 将一个 ENS 域名转移给了 Alice 这件事情在链上发生了。换句话说，在这里，一个链下状态（条件支付）依赖于一个链上状态（谁是这个 ENS 域名的所有者）。

而且，链下支付对象不必总是有条件的：一个条件支付对象能够在一个应用运行当中“退化”成一个无条件余额检验。更一般地说，条件依赖在本质上说是短暂的：一个应用状态能够通过一对在底层状态图上的拓扑遍历而完成更新。第一个是前向遍历，第二个是反方向遍历。向前遍历从链上状态通道合约开始，创建额外短暂条件条件依赖边的同时，修改已有的边。向后遍历有可能移除已有的短暂条件依赖边，因为后向遍历的过程中一些条件的值总是布尔值为真。

4.2 链下应用开发框架

正如现代高级语言和操作系统抽象掉关于底层硬件的细节，和条件状态依赖图交互的复杂性使得一个专用开发框架成为一个必要。谨守便于使用的原则，Celer Network 现在展示这套 cOS SDK，一个对于创建，跟踪和解决链下应用状态的完整工具链解决方案。我们希望这套 SDK 可以加快链下扩容解决方

案以及由 Celer Network 支持的支付网络的接纳程度，进而繁荣整个生态系统。

一般来讲，我们将去中心化应用归为两类：简单的按使用付费的应用以及更复杂的多方应用。按使用付费的应用有如 Orchid 协议，这个协议使得用户从真实世界的实体中不断接受微服务（也就是数据相继传递），并且从支付网络当中不断获取支付。因为不需要条件依赖于其他链下状态，因此一个在路由层上的精简的传输层足够处理这中情形，而二者都会由 Celer Network 提供。

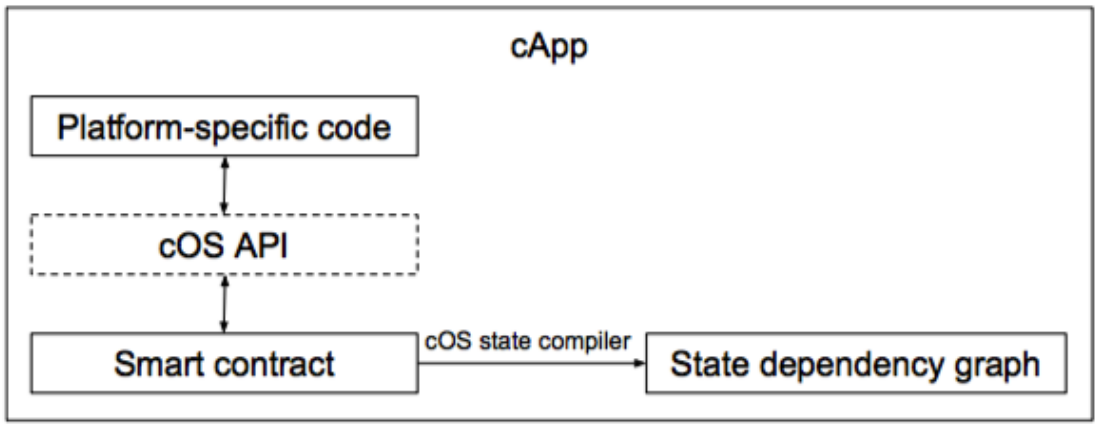


图7. Celer网络上的分布式应用程序的结构（cApp）

在由图 7 所阐释的多方应用的一般结构中，条件状态依赖图的思想才真正大放异彩。为了表达条件依赖，这套 SDK 为开发人员定义了一系列设计模式以及一个常规框架。我们计划通过现代软件开发技术例如元编程，注解处理和依赖注入来延伸现有的智能合约语言，从而使得依赖信息能够被鲜明的写出来而不至于突兀。接下来编译器处理应用代码，剥离已申明链下对象并且产生条件依赖图。编译器会检测出无效或者不能实现的依赖信息并且产生人类可以阅读的报错信息来辅助开发人员调试。为了帮助开发人员进一步推理依赖关系，这套 SDK 能够将图序列化成普遍格式例如 Graphviz，利用它那些图可以很容易被可视化进而展现出来。

这套 SDK 也能提供一个代码生成器来生成一系列用于和智能合约互动的“桥接方法”，而且在编译时这些智能合约的代码已经可以使用。这个代码生成器解析应用二进制接口（ABI），这个接口指名了在一个智能合约当中可被调用函数的签名，而且生成器在特定平台语言例如 Java 当中生成相应的桥接方法。这个方法主要的优点就是类型安全：胶水方法在智能合约当中可靠地复制

了函数的方法签名，而且在分派给将要执行的 cOS 运行环境的方法之前提供了一个静态且稳健的编译环境检查。

4.3 链下应用运行环境

cOS 运行时环境是 cAPPs 和 Celer Network 的传输层的接口。就网络通信和局部链下状态管理而言，它支撑着 cAPPs。图 8 阐释了它的总体框架。

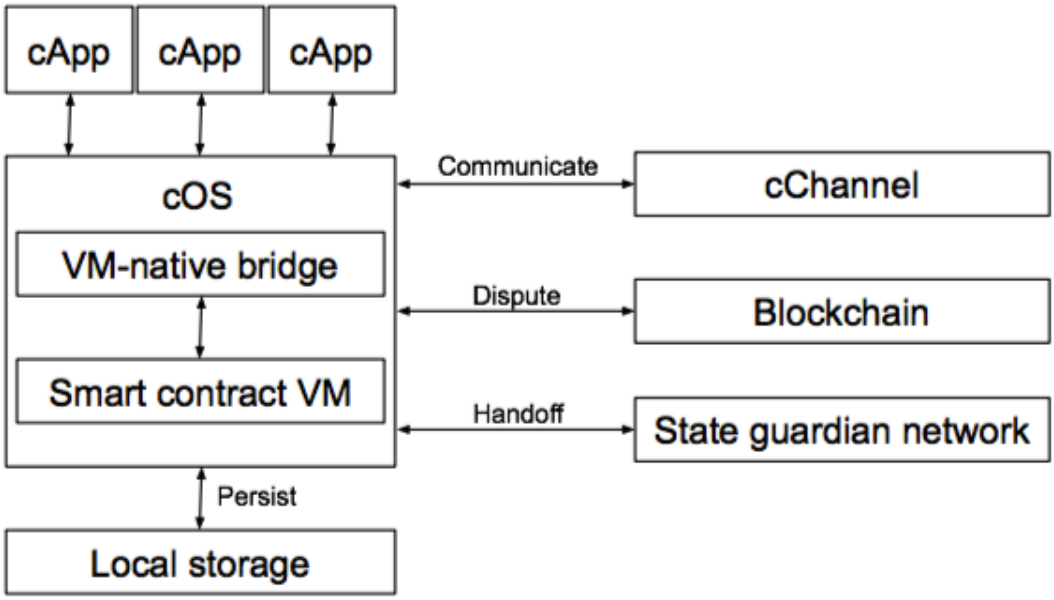


图8. cOS 运行体系结构

在网络方面，运行时环境在一个 cApp 的生命周期当中处理多方通信。为了支持如游戏这种需要确保多方计算安全的使用案例，提供了一组原语。在对方失败的情况下，无论失败停止或者拜占庭，运行环境会把争端相继传递到链上状态。在客户端掉线的情形下，运行环境通过卸载到状态保护网络来处理可用性。当客户端重新上线，运行环境会从状态保护网络同步局部状态。

对于局部线下状态管理，由 cOS SDK 合成的条件状态图被绑定到了 cApp 里，进而被传送至运行环境等待链下执行。在 Celer Network 客户端中，运行环境充当着本地创建，更新，保存和监控链下状态的基础设施。它既跟踪运行于它之上的应用的内部逻辑，也运行如 4.1 节所述状态更新的 DAG 遍历。同时，它还很优雅地处理着支付可靠性问题，例如路由支付中的能力不足。

在 cOS 内核当中，它绑定了一个能够运行智能合约的原生虚拟机。我们的核心理念是“一次编写，到处运行”，将 cOS 部署到桌面、网页、移动终端和

物联网等设备上。换句话说，开发者只需要编写一次业务逻辑，就可以在每个环境中运行完全相同的链上智能合约代码，而不必针对不同平台写多分代码。通过采用这一原则，我们致力于消除重复代码，确保跨平台的高度一致性。

一些依赖特定平台的 cApp 逻辑，比如用户界面（UI），可以用最适合每种平台的语言构建（例如 Android 的 Kotlin 和 iOS 的 Swift）。UI 代码可以自由地使用平台特定的工具和库，使得 cApp 的界面和用户体验与各自平台的设计指南相符。

cOS 运行时环境，提供了 VM-Native 桥，实现了不同平台的应用代码和底层业务逻辑的交互。例如，一个 iOS 上运行的国际象棋游戏的 cApp，其中用 Swift 编写的用户界面，用 Solidity 编写的业务逻辑。当 UI 层需要查询 cOS VM 中象棋游戏状态时，需要通过 Solidity-Swift 桥来进行查询。因为编译时可以知道调用合约的代码，所以 cOS SDK 可以生成一个桥接方法，名字叫 chess.getBoardState，由它进行桥接在 VM 上进行查询。 只要有可能，我们就会利用语言的外部函数接口（例如：JNI）来减少智能合约和本地代码之间来回调用的开销。开发人员也可以使用相同的调试和分析工具，在链下开发环境中，开发链上智能合约。

为了真正复制在链下环境中发生的链上状态变化，VM 需要用链上的相同方式来处理字节码，有一些问题需要注意。第一， VM 需要在本地而不是区块链上更新和存储状态。为了 VM 和 cOS 其他部分的能够直接地进行相互操作，我们实现了一组 API，将不同平台的数据存储在 VM 层下面。第二，用户可能会突然掉线，本地的 VM 可能会在任何时候停摆，比如软件 BUG、硬件失效、或者没电了。为了避免本地状态丢失，我们需要实现一套具有鲁棒性的日志系统，检查点和通讯协议。第三，运行在本地的逻辑，是没有必要收取 Gas 费用的。

运行在手机端和物联网设备上的 VM 必须体积小、效率高，这些设备处理性能、内存、电池都十分有限。我们目前在 cOS 中嵌入的是一个轻量级的以太坊虚拟机 EVM。后续会考虑使用更通用的、支持多种智能合约语言、多种公链的字节码技术(比如，WebAssembly)。

最终版本的 cOS VM，会应用到前沿的 VM 技术，比如提前编译(AOT)和即时编译(JIT)，使得链下智能合约的执行效率接近原生代码。大多数的以太坊虚拟机 EVM 的做法，是解释(interpreting)执行智能合约字节码，我们是将字节码编译成更接近原生代码的底层表现形式。如果某个合约的代码在编译时可用(比如合约已经部署到了链上)，我们提前执行编译，并将二进制文件与应用程序的

其余部分进行静态链接。对于在运行时动态加载的合约，我们针对经常调用的函数（即“热门”代码）对它们进行分析并执行即时编译。我们相信这两种技术的结合，将在性能和能耗之间取得很好的平衡，这对于移动端和物联网设备至关重要。

5．cEconomy：链下加密经济学算法设计

Celer 网络（CELR）的原生数字加密代币是 Celer 网络生态系统的主要组件，并且只应用于 Celer 网络内部。CELR 是一款不可退款的功能性程序代币，将被用作 Celer Network 生态系统中的平台货币。CELR 不以任何方式代表基金会，其附属机构或是任何其他公司，企业或企业的任何股权，参与权，所有权或利益，也不许诺代币持有人任何费用，收入，利润或投资回报，且无意愿成为新加坡或任何相关司法权区的证券。CELR 只能在 Celer 网络上使用，CELR 的所有权除使用权之外，不具有任何明示或默示的权利，CELR 本身，是作为激活 Celer 网络的使用并与之交互的手段而存在。在咨询了广泛的法律顾问，并持续分析虚拟货币的发展和法律结构之后，基金会将对出售 CELR 采取谨慎的态度。因此，在 CELR 的销售方面，基金会可能会不断调整销售策略，以做到尽可能的避免相关的法律风险。基金会同时正在积极的与 Tzedek Law LLC 展开合作，该公司是新加坡的精英律师事务所，且在区块链领域享有盛誉。

特别需要指出的是，您了解并接受 CELR：

(a) 不可退还，不能兑换成现金（或是等值的任何其他虚拟货币）或基金会以及任何分支机构的任何付款义务;
(b) 并不代表或授予代币持有人任何有关基金会（或其任何附属公司）或其收入，资产的任何形式的权利，包括但不限于任何未来收入，股份，所有权或股权，股权或担保，任何投票，分发，赎回，清算，专有（包括所有形式的知识产权）或其他财务，法律上的同等权利，知识产权以及与 Celer Network，基金会，分销商和/或其服务提供商有关的任何形式的参与权;
(c) 并非旨在代表货币（包括电子货币），证券，商品，债券，债务工具或任何其他种类的金融工具或投资;
(d) 不是对基金会或其任何分支机构的贷款，也不代表基金会或其任何分支机构欠下的债务，没有获利的期望;
(e) 不向代币持有者提供基金会或其任何附属机构的任何所有权或其他权益。

在下文中，我们介绍了 Celer Network 的加密经济机制，cEconomy，其设计基于一个原则，即一个好的加密经济模型（代币模型）应该提供额外的价值，并引入新的动态博弈理论，否则是不可能实现的。在下文中，我们首先阐述链下生态系统的基本权衡（或者说折衷）（第 5.1 节），然后展示 cEconomy 如何在带来价值的同时，去动态的“平衡”这些折衷(5.2 节)。

5.1 链下生态系统中的折衷

任何链下解决方案，在获得可扩展性的同时，也做出了相应的折衷。在下文中，我们描述了在链下生态系统中的两个基本折衷：可扩展性 - 流动性折衷和可扩展性 - 可用性折衷。

5.1.1 链下可扩展性 vs 流动性

链下平台通过牺牲网络流动性而获得可扩展性。例如，在双向支付状态通道中，双方可以高速安全地互相支付，而不会触及底层区块链，因为它们一开始就已将代币存入链上债券合约。从而锁定了流动性。这种流动性锁定对最终用户来说没问题，因为用户只需要将自己的流动性存入开放的通道，并享受因此而带来的 dApp 的可扩展性，然而，对于那些希望成为链下服务提供商（OSP）的人来说则是一个重大挑战。以状态通道为例，OSP 需要把资产存入每个通道，并考虑其未来牵扯到的可能支付数量。这些抵押资产很容易聚集到一个天文数量。尽管 Celer Networks 的侧链通道可以显著降低流动性的需求水平，但每个区块提交者仍然需要抵押一定比例的防欺诈证明债券合约，作为“押金”，以用来价值转移。

总而言之，需要大量的流动性来为全球区块链用户提供高效的链下服务。然而，拥有加密资产的大户（“鲸鱼”）可能没有商业兴趣或技术能力来运营一个链下服务基础设施。而有技术能力运营可靠和可扩展的链下服务的人，通常又没有足够的资金用于通道存款的抵押或是防欺诈债券合约。这样的不匹配会给链下平台的大规模使用和技术演化带来巨大障碍。如果不能得到缓解，最终将只有富人可以充当链下服务商。这种成为 OSP 的高资本壁垒将导致中心化的网络，从而破坏了区块链去中心化愿景的大前提。从更加现实的角度来看，审查制度，差劲的服务质量和隐私破坏，也会像今天的中心化服务一样伤害到用户。

5.1.2 链下可扩展性 vs 可用性

虽然链下平台通过将应用程序的状态带到链下来提高可扩展性，但它会给用户带来了不切实际的“始终保持在线”的责任，因为链下状态必须始终可用，才能解决链上争端。例如，在双向支付状态通道中，如果一方掉线，对方也许或是被黑客攻击，或是主动恶意行事，然后提交一个对自己更有利的旧的状态。数据可用性问题在侧链通道中更为关键，在侧链通道中，需要在一方参与者离线时单独监视和验证另一方的区块提交；这是一个安全问题，应该仔细审查。在物联网那种设备不可能一直保持在线的机器对机器通信方案中，这一问题则更为关键。因此，设计合适的机制以保证链下平台中的数据可用性至关重要。要解决这一难点，需要对整个链下生态系统和进行系统型思考，而现有的解决方案都还无法做到提供分散，高效，简便，灵活和安全这些重要属性，我们将会在接下来的部分做进一步探讨。

5.2 经济设计

为了平衡上述折衷，我们提出了一套名为 cEconomy 的加密经济机制，其中包括三个紧密相关的组件：流动性承诺证明（PoLC）挖矿，流动性支持拍卖（LiBA）和状态卫士网络（SGN）。图 9 显示了这三个组件之间的关系。

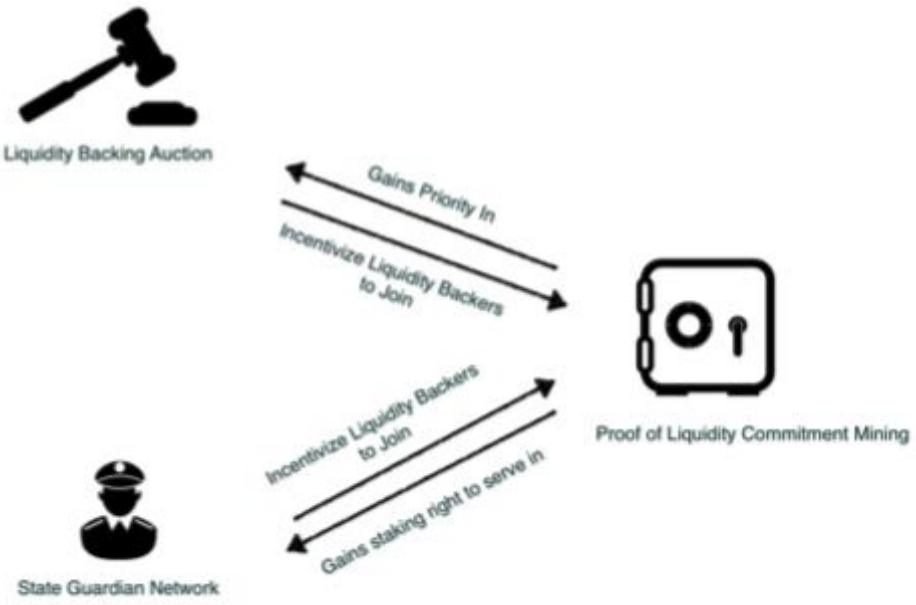


图9. cEconomy组件之间的关系

在讨论这些组件的细节之前，我们首先介绍将在本节中使用的几个术语。具体而言，我们 cEconomy 系统中的用户可以扮演三种角色：链下服务提供商（OSP），最终用户（EU），网络流动性支持者（NLB）和状态守护者（SG）。链下服务提供商（OSP）是具备运行高度冗余，可扩展和安全链下基础设施的技术能力的实体。最终用户（EU）可以访问由 OSP 提供的链下服务（例如支付和接收加密货币）。他们既可以是普通消费者，也可以是物联网设备，VPN 供应商，直播视频流媒体供应商和 CDN 供应商，机器对机器（M2M）系统中的交易双方，甚至是一个链下/链上的智能合约。网络流动性支持者（NLB）是通过锁定其在系统中的流动性以支持链下基础设施运营的实体。状态守护者则通过状态守护网络来为用户提供分散，安全，灵活和高效的守卫服务。

5.2.1 流动性承诺（ PoLC ）证明挖矿

我们的第一个目标是通过降低流动性阻碍来平衡可扩展性与流动性的折衷，让技术上有能力的各方成为链下服务提供商，从而为优质可靠的链下服务创建一个高效和富有竞争的市场。这个想法的主旨是让服务提供商在需要时能够获取大量的流动资金。实现这一构想的第一部分，是提供一个能够缓解短期流动性供给波动的，充裕且稳定的流动性池。为此，我们设计了流动性承诺证明（PoLC）虚拟挖矿流程。

从一个高层次上来看，PoLC 的挖矿过程，是通过 CELR 代币的奖励，来激励网络流动性支持者（NLB）进行 Celer Network 内流动性的长期锁定，从而建立稳定和充裕的流动性池。

更具体地说，挖矿过程就是让 NLB 在一段时间内将其闲置的流动性（为简单起见，以 ETH 为例）提交（或者说锁定）到叫做 “担保承诺合约”（CCC）的“木偶盒子”里。在这个锁定的时间段内，他们的资产只能用于流动性支持，不能挪作他用。直观来说，随着更多的价值和更长的时限锁定到 CCC，也会得到更多的 CELR 作为奖励。正式来说 - PoLC 挖掘过程定义如下。

定义 5.1（PoLC 算力）。如果 NLB i 锁定了区块链（比如 ETH）中 T i 时长，S i 数量的本币，那么其 PoLC 算力 M i 就被计算为

$$M_i = S_i \cdot T_i. \tag{18}$$

定义 5.2（PoLC 奖励机制）。在有限的时间内，Celer Network 计划给那些把 CCC 锁定用来支持系统的 NLB 相比预挖多 20%的 CELR 作为激励，激励跟每个 NLB 的 PoLC 算力成正比。如果 R i 表示 i 的激励，那么它就有：

$$R_i = \frac{R \times M_i}{\sum_{j=1}^{N} M_j}, \tag{19}$$

其中 R 是当前区块的总奖励。

要注意的是，锁定 CCC 的流动性不会带来任何内在的交易风险，因为它只是单纯展示了对与 Celer Network 的流动性承诺。另外还要注意一点，系统不允许提前解锁 CCC，因为人们可能会试着创建一个“虚假结算”，使其看起来就像是因为假 OSP 发起的“黑客入侵”，从而使 CCC 再次流动。为了防止这种欺骗行为，在 CCC 解锁之前，新开采的 CELR 不可用于提现和使用。任何提前解锁都会导致已经挖出来的 CCC 被没收并重新分配给其他矿工。PoLC 流动性共同标准的构建也是一个重要问题。在平台刚启用时，我们将使用目标区块链的本币，未来将会通过外部的价格预测机，来使用更多的加密资产。

通过这些机制，PoLC 挖矿过程确保了 PoLC 算力 随着 CELR 的系统和效用的增长而增长，形成一个正向循环。

关于这一点，有人可能会问，为什么 CELR 有可以作为激励作用的价值？我们会在接下来描述流动性支持拍卖和状态守护网络的章节中给予解释。

5.2.2 流动性支持拍卖（ LiBA ）

解决流动性难题的第二部分是通过流动性支持拍卖（LiBA），使得链下服务提供商可以访问全球流动性池。LiBA 使的链下服务提供商能够通过“民间借贷”的方式来争取流动资金。从本质上来书说，一家链下服务提供商在 Celer Network 上启动一个 LiBA，在特定时间内“借到”一定数量的流动性。有兴趣的流动资金支持者可以提交一个投标书，里面包含利率询价，流动性数量和她愿意在该段时间内押注的 CELR 数量。流动性数额可以通过 CCC 提交。也就是说，CCC 具有充当流动性支持资产的功能。借入的流动资金将用作防欺诈债券或将来的通道抵押资产

LiBA 是一个通常意义上的多属性维克里-克拉克-格罗夫斯机制（VCG 拍卖，第二价格密封拍卖）。要启动拍卖流程，OSP 需要通过 Celer Network 的中央 LiBA 登记处创建标准 LiBA 合约，并提供有关请求流动性总额（q），请求期限（d）和可以接受的最高利率（rmax）等信息。关注登记处的 NLB 将会注意到这个新的 LiBA 合约，并可以开投标。Celer Network 要求竞价时，所有加密资产都必须包含在 CCC 里。请注意，这里的 CCC 可以单纯充当支持资产，不用锁定，但同时也相应的失去了 PoLC 的挖矿功能。CCC 充当加密资产的容器，并为各种加密资产提供统一的可验证价值。除此之外，使用 CCC 可以让 NLB 参与 LiBA 变得容易，无需在每次出价时将加密资产转来转去，既简化了支持流程，又提高了安全性。NLB i 以数组 b i =（r i ，t i ，c i ）的形式提交出价，其中 r i 是利率，t i 是愿意在合约期间锁定的 CELR 总额，c i 是与这次出价的 CCC 中包含的总货币价值。提交投标后，相应的 CCC 将被暂时冻结。密封投标后，LiBA 合约会使用反向第二价格拍卖，通过以下三个步骤确定中标者

- **得分规则**。 投标记作 bi = (ri,ti,ci) ，投标组合记作 ß={B1,B2,B3}，记 fi = ti/ci。得分 si 计算公式如下：

  公式(20)

  其中 fmax = max{f1,f2,...,fn} ， rmax = max{r1,r2,...,rn}。w1 和 w2 是两个部分的权重因子。我们首先考虑更高利率，再考虑更多 CELR。
- **确定中标者**。 LiBA 合约按照投标得分，对其进行降序排序。记 ß*={B1*,B2*,B3*} 为排序后的投标组合，其中 s(b1*) ≥s(b2*)≥s(b3*)。中标者是 ß* 的前 K 名，其中 Σ(K)t≥q 且 Σ(K-1)<q 。
- **以第二价格质押/消费 CELR**。 当中标者确定后，他们的 CCC 将被锁定在 LiBA 合约中，锁定期限为 d (招标的要求时限)，然后他们会收到相应的利息，该利息是 OSP 在招标时预支的。然而，并不要锁定或消费，投标时标注的全部的 CELR。每个中标者真正支付的 CELR 会被重新调整，使得中标者的得分和排名第一的非中标者的得分匹配。在头 5 年，新的代币将通过 PoLC 挖矿生成，而 LiBA 只需要质押代币。当 PoLC 挖掘结束后，LiBA 将开始消费令牌，消耗的令牌将作为持续的 PoLC 挖掘奖励注入系统。需要强调的是，以第二价格质押/消费 CELR 的机制，参与者会倾向于

提交与商品(在这里是，提供流动性的机会成本)的真实估值相匹配的投标。

举例：假设一个 OSP 以如下参数（600ETH，30 天，1%）发起一个 LiBA 而且对于这个 LiBA 有三个潜在的竞拍者（我们假设是 A，B，C）。这三个竞拍者的竞拍分别是 bA = (1%, 800 CELR, 400 ETH); bB = (0.5%, 800 CELR, 200 ETH); bC = (1%, 100 CELR, 400 ETH). 根据得分规则，我们有 sB > sA > sC。因为 A 和 B 可以执行整个请求，他们被选为赢家。同时注意到尽管 A 和 C 有相同的利率（1%）而且提供相同的流动性（400ETH），竞拍者 A 被选为赢者而竞拍者 C 输掉竞拍。这是因为他们承诺的 CELR 代币，作为一种他们对平台贡献的标志，是显著不同的。最后，根据第二得分赌注原则，A 和 B 在 30 天内锁住（或者消费）了他们的 CELR 代币来匹配 C 的分数。

在拍卖过程结束之后，发起流动性请求的 OSP 通过往 LiBA 契约中存款来支付赢得流动性支持者利息。一旦受到利息支付，LiBA 契约支付利息给相应的流动性支持者，同时 1：1 发放和流动性请求数量匹配的被支持的 cETH（用 ETH 作为例子）。尽管 cETH 从本质上讲是 IOU，但是它对用户而言没有任何风险，因为这些 IOU 是 100%被在 LiBA 契约中网络流动性支撑者们所保证。

在正常的情况中，当 OSP 把所有 cETH 代币发送回来时，LiBA 契约会在超时前被确定下来。基本上，在超时之前，通过分别从上游通道提款，OSP 会用真实的 ETH 来把所有支付过的 cETH 转换成 EU。

在 OSP 有可能被攻击的情形，Celer 网络可以改变它的信任模型。没有任何的协议层的开销情况下最简单的信任模型是声誉驱动的，这里 NLB 在没有任何失误历史的情形下选择一个声誉良好的 OSP。在这个简单的模型中，NLB 被暴露在了丢失钱财和资产的风险中，因为如果 OSP 犯错误，他们的 CCC 就是 EU 的保险。然而，在这个简单的信任模型中，是否真的可能操作一个可依赖和有信誉的 OSP，这件事仍具有争议性。所有的支持都丢失是几乎不可能的。在 LiBA 周围可以添加更多的安全功能来缓解潜在的风险。例如，新发行的 CETH 仅仅被允许存入一个状态通道契约的白名单；cETH 仅仅被允许使用在渐渐增长且有上限的花销速度。一个 OSP 依旧可以有很多种方式来维护一个安全的基础设施，例如分区多节点部署，网络基础设施的安全接入规则的正常验证，以及更多。

另外，我们开启了增强安全模型，这里一个随机选取的 NLB 的法定人数会需要多人签署一个 OSP 操作（例如支付）。这些 NLB 只允许一个向外转账等价于他们看到一个等额的向内转账。这些 NLB 也被 OSP 的收入支付所缠绕。如果

OSP 最终失败地重新支付，那么 NLB 有第一优先权去从其他通道中收回收入资金给 OSP。然而，我们也意识到这个操作模型会最终损失掉网络中一部分效率。

已经说了这么多，我们相信这个模型中最终的余额会被市场需求所定义。我们开放信任模型给市场为了让它能够有机增长。我们相信那些无信任模型会在网络开始的早期比较受人喜爱，然后它将会变得更加的基于信任。

出于 LiBA 的信任模型，我们想特别强调 LiBA 过程保证了**终端用户不会承担任何风险**，因为所需的流动性是 100%的被 LiBA 合约所保护的。在 Celer 网络的系统中，我们努力确认善良的终端用户不需要担心他们收到资金的安全性而且 LiBA 会实现那些。POLC 和 LiBA 一起激励一个丰富的流动池，降低了成为链下服务提供者的门槛和集中化的风险，加速了网络采纳。

5.2.3 状态守护网络

另一种 CELR 代币的用法是在新奇的保险模型和简单互动下提供链下数据可用性,使在 5.1.2 章里提过的可扩展性和可实践性的两者取舍达到平衡。

简单从表面上来看，可用性的问题是比较容易解决的。其中一个可能的答案可能是：让我们在未来建立一些监控服务，然后当人们不在线的时候，他们便需要给这些监控服务付费。这在表面上来看，感觉上像是一个合理的答案，但是让我们把这列思想的火车再稍微往前开一点，我们马上就会看见一个断裂的轨道一般的漏洞存在着。

让我们从这个问题着手:这些监控服务是不是建立在信任之上的呢？如果答案是对的话，那么这就又制造了另一个中心化的的令人窒息的问题。单点失败且是不安全的。恶意的对手完全可以轻松的贿赂这些提供服务的公司使其攻击伤害善良的客户端使用者们。

那我们可不可以建造一个完全信任的监控服务呢？举个例子，我们可以惩罚那些提供监控服务的人如果他们不能保护使用者的状态。然而，当我们趋向于这种想法的时候，我们马上会看见一些，“这个方法是不可取的”之类的警告。我们该让那些监控服务的提供商付出多少以作惩罚呢？忽略那些额外因素不计，对监控服务提供商所有叠加在一起的惩罚应该和那些下线后的人群所可能遭受的最大损失相等。

这样有效的加倍了一个链下网络对流动性的需求因为任何时候当一个人下线时，除在频道中已存在的被锁住的流动性或者是在侧链中防诈骗的纽带，随着惩罚存款的增加，监控服务的提供商同时也需要锁住相同数量的流动性。

更糟糕的是，监控服务的提供商需要为不同的监控任务保持不同的资产。当所参与的状态变得复杂起来，当所存在的资产级别变得多样化起来，事情很容易就变得复杂而艰难。有的时候，在对于广义状态通道给定所有的复杂状态依赖性时，连一个简单的从状态变为底层价值的转换都没有。

即使有了足够的流动性，保险模型在这也还是很死板的：简单来说就是你会一次性拿回 x%一旦监控服务提供商没有保护好你的状态。如果你选择一个较大的 x 值，由于额外的流动性锁定，费用将会变得很昂贵。但是如果你选择一个较小的 x 值，又会变得很不安全。

在这些不利条件之上，因为市场信息现在依然是低效率的被隔离着，对于这些状态监控服务该如何定价还是很不明朗的。这样的的低效率以及每个独立团队在异质财产上的自我捆绑，只会造成更长远的链上和链下对监控服务互动的复杂性，以及摧毁任何链下平台的的使用性。更多的问题依然存在着，但是光上述的已经够糟糕了。

为了解决这些问题，我们提出状态守护网络（SGN）。当用户下线后，SGN 是一个特殊紧凑的来保护链下状态的侧链。CELR 代币持有者可以将 CELR 资助成 SGN，然后变成状态守护者。当一个用户下线前，她可以通过支付一些费用将她的状态提交给 SGN，让保护人替她在一定时间内保护她的状态。然后，根据状态散列和“责任分数”随机选择一些守护者来负责这个状态。选择守护者的具体规则如下：

- **状态守护请求**

  一个状态守护请求是一个向量 $\eta_i = (s_i, \ell_i, d_i)$ ，s_i 代表该被守护的状态，l_i 是付给守护者的服务费额度，然后 d_i 是这个状态被守护的长度。

- **责任分数**

  状态守护请求的责任分数是这样计算的：

$$\gamma_i = \frac{\ell_i}{d_i}.$$

一个用户的责任分数实际上就是这个用户对于 SGN 所产生的收入流动。

- **守护者筹码数量**

给出一组出色的状态监管，要求 $\mathcal{R}=\{\eta_1,\cdots,\eta_m\}$，CELR 在每一个请求抵押上的数量 $\eta_i \in \mathcal{R}$ 是

$$n_i = \frac{\gamma_i}{\sum_{j=1}^{m}\gamma_j} K,$$

K 是守护者放入 SGN 的 CELR 筹码值的总量。换句话说，可靠的 CELR 利益值数量是和需求的责任值和所有出色的状态责任分数的比例成正比的。

- **守护筹码的分配**

给一个状态守护请求，让 hi 成为相对应状态 si 的哈希值（hash value）。每一个 CELR 筹码 k 都和一个 ID pk（也是一个哈希值）联系在一起。让 $\delta(g_1,g_2)$ 成为两个哈希值 g1 和 g2 之间的距离。然后 CELR 筹码被按照他们对 h_i 哈希值距离的顺序排列。假设 $\delta(p_1,h_i)\le\delta(p_2,h_i)\le\cdots\le\delta(p_K,h_i)$ (联系被随机打破)。第一个有最小距离的 CELR 筹码 n_i 被选出，然后相对应的筹码持有者将会变成对这个请求的筹码守护者。

- **状态守护服务费的分布**。

对于每一个状态守护请求，附带的服务费是根据如下规则分配给状态守护者的。对于每一个状态守护者 j,令 z_j 是他/她所下注的 CELR 的被选为这个状态守护请求的代币的数量。那么守护者 j 从状态守护请求中获取的服务费是

$$q_j = \frac{z_j \times \ell_i}{n_i}.$$

注意到每一个筹码 CELR 有相同的概率被选为状态守护请求。作为结果，从状态守护者的角度来看，她往 SGN 中放越多的 CELR 代币筹码，她的筹码会更有可能被选中，因此赚取更多的服务费。这凸显 CELR 作为 SGN 的会员有显著的价值。

- **安全和抗密谋**

每个守护者根据结算超时分配一个争议时段。如果守护者未能及时对其进行争议，则后续守护者可以报告并获得失败的守护者的 CELR 股份。因此，只要至少有一个选定的守护者没有被破坏并完成工作，最终用户的状态就总是安全的且可用于争议。

SGN 机制还带来以下附加价值：

- **它不需要守护人进行大量流动性锁定**

守护者只会放置他们的 CELR 令牌，无论底层价值/代币的类型/数量如何，都可以用来保护任意状态。

- **它为任意状态监控提供统一的接口**

无论该状态是否与 ETH，任何 ERC20 令牌或复杂的状态相关，用户只需附加一笔费用并将其发送给 SGN。SGN 并不关心潜在的状态和相关的价值，只是将 CELR 的数量与支付的费用成比例地分配给该状态。

- **它实现了简单的交互**

Celer Network 的用户无需联系所属守护者，他们只需要向此侧链提交状态。

- **最重要的是，它使一个全新而灵活的状态能够保护经济活力**

SGN 为用户提供了一种“在 X 时间内收回我的钱”的新机制，并为该流动的保险模型提供了一种有效的定价机制，而不是强制使用刚性和不透明的“获得 X%回”模型。如果所有涉及的守护者都没有对用户提出异议，她将从这些守护者处获得 CELR 股份作为补偿。在稳定状态下，放在 SGN 中的 CELR 令牌表示进入流（例如，获得 x Dai /秒）。忽略状态监控和其他摩擦的成本，当用户将状态提交给 SGN 时，她可以通过选择每秒支付的费用（即责任分数）明确地选择 CELR 为其状态“覆盖”多少。

5.2.4 小结

系统考虑的话，cEconomy 涵盖了一个链下平台的完整生命周期。LiBA 和 PoLC 挖矿用一种低障碍的方式把仲裁交易带到链下。SGN 在任何需要的时候确保能够把最新的状态状态带回链上。这样的话，我们相信 cEconomy 是一个完整的链下加密经济平台，这个平台能够带来全新的价值以及前所未有的活力。

6. 结论

Celer Network 是一个统一的技术和经济架构，同时也是一个运用链下扩展技术将互联网规模扩展到现有和未来的区块链。它可以横向扩展，且去信任化、分布式、具有隐私保护。它包含了一个分层架构，每层都有重大的技术创新。另外，Celer Network 还提倡一个有原则性的链下加密经济设计，以此来平衡其扩展性。Celer Network 的使命是充分释放区块链的能量，并且在如何构建和使用分布式应用程序上进行创新。

7. 团队介绍

1. 董沫

董沫博士毕业于上海交通大学，并于 2017 年在 UIUC 计算机系获得博士学位。圈内人称老董，是区块链的早期布道者，开发者和投资人。他的公众号“老董区块链干货铺”(现 Celer Network 技术社区)用通俗易懂的方法传播区块链的艰深技术知识。从 2017 年开始他坚持教授全栈区块链智能合约开发课程，已为社区培养了 300 名合格的区块链开发者。他的研究和开发经验主要在网络和分布式系统协议设计，网络协议形式验证以及博弈论。董博士领导开发了基于 non-regret learning 的网路传输协议，可以提升互联网复杂网络环境下的数据传输速度 10 至 100 倍。其工作已被一些最大的互联网内容提供商和服务商所应用，也获得了第二代互联网创新应用奖，发表在了国际顶级网络与系统会议上。董博士曾是由 NEA、Menlo Venture 联合投资的网络形式验证与安全公司 Veriflow 的创始成员、产品经理、早期战略客户经理和工程团队负责人。他开发的分布式系统和网络协议形式验证软件已被部署在世界五十强企业当中。

2. 刘俊达

刘俊达博士毕业于清华大学并于 2011 年在 UC Berkeley 计算机系获得博士学位师从 SDN 创始人、美国工程院院士 Scott Shenker 教授。刘博士首先提出使用 DAG 来进行路由，并实现比业界领先 1000 倍的纳秒级网络修复能力。他之后加入 Google 并把先驱研究应用到 Google 全球的技术基础架构上，以技术负责人的身份开发出新一代数据中心动态拓扑结构，支持百万节点互联和超过 1000Terabits/s 的带宽。2014 年刘博士成为 Google 创新移动运营商 Project Fi 的创始成员，并在两年内实现超 1 亿美元的营收。他同时是安卓系统运营商服务的负责人，超过十五亿台手机运行他设计和实现的基于 UICC 数字签名的系统。刘博士拥有 6 项美国技术专利并在顶级会议上发表多篇论文。他于清华大学获得本科和硕士学位。

3. 李小舟

李小舟本科毕业于清华大学并在普林斯顿大学获得计算机博士学位。他在分布式系统、网络、数据存储等领域有多年深入研究，在 SOSP、NSDI、FAST、SIGMOD、EuroSys、CoNEXT、ToN 等顶级会议与期刊发表多篇论文，并获得 NSDI’18 最佳论文奖。他设计开发的算法已成为 Google TensorFlow 与 Intel DPDK 等多

个工业界系统的核心组件。李小舟博士曾在 Barefoot Networks(一家由 Google、高盛、阿里巴巴、腾讯等著名机构联合投资的网络创业公司)工作两年，期间主导开发了数个开创性的智能高速网络与系统解决方案，负责部分关键客户对接，并申请了六项美国专利。

4. 梁清凯

梁清凯本科毕业于上海交通大学并在麻省理工学院(MIT)获得博士学位，主要从事分布式系统和网络安全方面的研究。他提出了首个在恶意网络环境中可证明最优的高性能路由算法，并已经成功应用在 Raytheon BBN Technologies 和贝尔实验室。他在计算机网络系统和安全领域发表了数十篇一作论文，并且获得了 IEEE MASCOTS 2017 的最佳论文奖提 名，IEEE INFOCOM 2016 和 2018 的最佳报告奖。

5. 周末

周末(Michael)毕业于麻省理工学院(MIT)，获得计算机硕士学位。他的研究方向专注于并行数据流架构编程模型。Michael 在谷歌工作时研发了一系列 Java 和 JavaScript 的编译器，静态分析器，虚拟机与开发框架，支持网页端和移动端应用为数十亿用户提供服务。

6. 王子轩

王子轩于南京大学获得硕士学位，拥有 6 年安卓应用的资深研发经验。他曾是南京银行和江苏银行手机 APP 的核心开发和移动端技术负责人，后任职于中国前五的电商独角兽企业蘑菇街，在移动端架构组主要负责安卓端相关优化和安卓崩溃保障体系的创建与运行。他大幅改善并保持日活 200 万以上的蘑菇街 APP 的日平均崩溃率在万分之四左右，因此申请并获得了两个安卓崩溃保障体系的相关专利。他具备极其丰富的手机端开发、安全防护和优化经验，同时也是早期区块链技术探索者和布道师，对区块链共识技术、加密算法和智能合约都有较深程度的研究。

7. 张岩

张岩于巴黎高等电力学院 Supélec 获得工学硕士及法国工程师学位，该校在能源和信息领域排名位于法国第一。他本科毕业于北京邮电大学，通信工程专业。逾十年来，他在欧洲和美国多个著名创业公司和跨国企业担任主力开发和技术总监，专注于移动应用的开发和移动技术的创新。他从零开发并上线过数十个移动

应用及智能硬件产品，其中一些应用和产品在欧美家喻户晓。

8. 戴运嘉

戴运嘉毕业于哥伦比亚大学，于 2014 年加入谷歌展示广告团队，负责开发和演进广告基础设施平台。平台广泛服务于 AdSense 及多个 DoubleClick 核心产品。他曾担任技术带头人负责展示广告伺服-报告接口及基础设施，用于处理百万级 QPS (每秒请求数)。

9. 王鹏颖

王鹏颖(Windy)毕业于波士顿大学，获得国际关系和国际管理双学位。作为麻省理工学院中国创新与创业论坛(MIT-CHIEF)的副主席，Windy 已带领约 50 个美国优秀初创企业回中国参加路演，并与高新科技园区进行交流。她在 MassChallenge 加速器和中美波士顿创新中心(CUBIC)的战略规划和市场营销经历激发了她对初创公司的热情和加深了对市场的理解。

10. 李思蓉

李思蓉于罗切斯特大学获得商业分析硕士学位。她在市场调研、广告行业和快消行业拥有丰富的工作经验，并致力于开发新颖的市场战略以及制定数据驱动的业务决策。她曾在世界 500 强公司 TJX 协调策划数千名员工的组织转型。于世界领先的营销公司奥美，她为关键客户设计的营销策略巨幅提升了销售额与产品知名度。作为一位经验丰富的项目经理，她曾带领一个团队利用大数据和机器学习技术帮助客户公司快速提高知名度及销售额。

11. Alex Wu

Alex 于2016年毕业于纽约视觉艺术学院，获得社会创新硕士。他曾于康乃尔大学协助医疗相关的产品开发，毕业后在 SAP 担任用户体验设计师，参与项目包括开发企业内部工具、人资、与企业智能语音平台等，并在多家新创公司担任设计顾问。Alex Wu 在 2011 年加入 TEDxTaipei，协助创办人许毓仁(现任台湾立委、台湾区块链议会联盟、台湾区块链自治联盟发起人)建立设计与制作团队，成为当时新兴的新媒体之一。

8. 致谢

本文由 Celer 技术社区成员译自官方英文白皮书：https://www.celer.network/doc/CelerNetwork-Whitepaper.pdf

译者：蒋宏伟、王昕屹、高冰、陈威、梁超、黄鑫实、陈鹏、路朋飞。
注：以上排名不分先后