2018-11-2

王浩然

复杂美（上海）

**目录**

雷电网络调研报告

[一、 引言 1](#_Toc528785913)

[二、雷电网络介绍 2](#_Toc528785914)

[2.1 基本概念 2](#_Toc528785915)

[2.2 基本术语 3](#_Toc528785916)

[2.3 基本原理 4](#_Toc528785917)

[2.3.1 状态通道概念 4](#_Toc528785918)

[2.3.2 状态通道及余额证明 5](#_Toc528785919)

[2.3.3 雷电网络的形成 7](#_Toc528785920)

[2.3.4 雷电通道智能合约 9](#_Toc528785921)

[2.3.5 通道生命周期 10](#_Toc528785922)

[2.3.6 雷电转账 10](#_Toc528785923)

[2.3.7 中介转账详解 12](#_Toc528785924)

[三、雷电客户端调研 13](#_Toc528785925)

[3.1 消息规范 14](#_Toc528785926)

[3.2 智能合约 19](#_Toc528785927)

[3.2.1 TokenNetworkRegistry合约 19](#_Toc528785928)

[3.2.2 TokenNetwork 合约 20](#_Toc528785929)

[3.2.3 SecretRegistry合约 29](#_Toc528785930)

[四、闪电网络总结 33](#_Toc528785931)

[4.1 雷电网络工作流程 33](#_Toc528785932)

[4.2 雷电网络的特点及优点 34](#_Toc528785933)

[4.3 雷电网络风险及缺点 36](#_Toc528785934)

# 引言

以太坊的拥堵问题由来已久，前段时间火爆的云养猫游戏让以太坊网络再度出现大规模的拥堵，受此影响，很多人不得不提高交易费来加速确认，而部分交易所甚至一度暂停了相关代币的提取。随着以太坊平台上 DApp 的增加，整个链上的交易量在不断激增，交易阻塞问题也愈发严重。为了解决这一刚需问题，众多扩展方案应运而生，包括**分片（Sharding）、侧链（Plasma）、雷电网络（Radien Network）**等等。这些方案都可以提升以太坊系统的整体交易能力，不过在大幅降低交易延迟与成本方面雷电网络更胜一筹，并且相较于还处在酝酿阶段的其它方案，Raiden 的完成度已经取得了实质性的进展。

雷电网络（Raiden Network）的开发人员已经在以太坊测试网络上发布了规模化解决方案的初期版本µRaiden（称为“微雷电”）。 µRaiden允许用户之间直接进行小额支付，而不是通过网络节点这种付款路径。此外，付款操作单方面有效, 尽管是简易版，但µRaiden能为发出申请的去中心化APP（DAPPs）提供支付渠道，比如用于新闻业和仓储业, 这些支付渠道具有erc-20标准的兼容性，这意味着最近在以太坊上发布的一系列代币均可以使用。

# 二、雷电网络介绍

## 2.1 基本概念

雷电网络（Radien Network）是基于以太坊智能合约的链下扩展方案，主要用来解决以太坊平台资产小额快速支付的问题，可支持所有符合以太坊ERC20标准代币的交易，具有即时到账，低转账费用，可大规模扩展，隐私保护的特点。雷电网络的节点在互相转账时，不需要通过以太坊主链的交易确认，仅需通过节点之间创建的支付通道便可快速完成链下交易。

类似于比特币的闪电网络（Lightning Network），雷电网络上的支付通道其实是一种基于以太坊系统的状态通道（State Channel），通过开启这种通道可以将部分链上小额交易分流到链下处理，当交易达到一定量或时间后再关闭通道并进行结算，最终交由链上进行确认。这样一来便可大幅减少链上的总交易量，达到降低交易费用和延迟的目的



雷电网络的设计目标：

* 即时支付：雷电网络的交易不需要区块确认，交易可即时在链下完成确认。
* 微支付低费用： 雷电网络使用Token交易，最小化交易结算（交易广播上链）的需要，可以以极低的费用实现交易。
* 大容量可扩展：雷电网络交易在链下确认，允许用户执行近乎无限的微支付交易，可以满足诸如物联网设备间的自动大量交易。

本次调研主要针对python开发的Raiden雷电网络测试版软件

## 2.2 基本术语

1.余额证明（balance proof）

余额证明是用于在区块链上证明从对方收到的最新转账金额和锁定金额的任意类型的消息。

2.通道容量（channel capacity）

通道的容量是其参与者的总存款总和。它也是通道参与者余额的总和。

3.直接转移（DirectTransfer）

直接转移是非锁定转移，这意味着转移不依赖于锁定来完成。一旦发送，他们应被视为已完成。

4.哈希锁（hashlock）

哈希锁是对某消息进行哈希加密的结果：sha3(secret)。

5.锁定时限（lock expiration）

锁定时限是可以交易结算时的最高区块高度

6.锁定余额（locked balance）

通道参与者的锁定余额是所有待定转移金额的总和

7.中介转账（MediatedTransfer）

中介转账是发起者与通过网络中的节点传播的目标之间带有哈希锁的转账。

8.锁根（locksroot）

merkle树的根，它保存通道中所有锁的哈希。

9.秘钥（secret、preimage）

32字节的随机加密安全数据，其keccak散列最终成为哈希锁。

10.退款转账（RefundTransfer）

退款转账是一种特殊类型的中介转账，当某节点不再转移转账并且需要完成路由回溯时使用。

11.揭露时限(reveal timeout)

代表通道中的区块高度，在该高度之前，节点可以获取secret并使用其获得转账

12.待揭露秘密(RevealSecret)

待揭露秘密被发送到已知有需求获取该秘密的节点

13.秘钥请求（SecretRequest）

请求获取秘密的消息由中介转账的目标发送到其发起者，以便请求秘密解锁转账。

14.结算窗口期（settlement window）

通道关闭后的区块数量。在该时间之前，交易对手能够调用updateNonClosingBalanceProof使用他们收到的最新余额证明。

## 2.3 基本原理

### 2.3.1 状态通道概念

对于状态通道我们首先要明白「状态」在此的含义，不妨把区块链想象成一本大的会计账本，上面记录了每一笔钱被分配到了哪里，比如张三给李四转了30个以太币，我们可以看到李四的账户余额多出了这笔钱，它从张三那边流转到李四这里，不再属于张三，账本上记得清清楚楚，此即为该笔加密资产的状态。

状态通道简单来讲就是从以太坊主网上抓取交易者的资产状态，然后"储存"在交易方之间开辟的双向通道上，双方以这个初始状态为基础进行通讯，通讯内容主要是经双方数字签名的交易信息，这些信息反映了双方资产的流动状态，从而达到降低交易确认门槛的目的，为离链交易提供了安全便捷的保障。

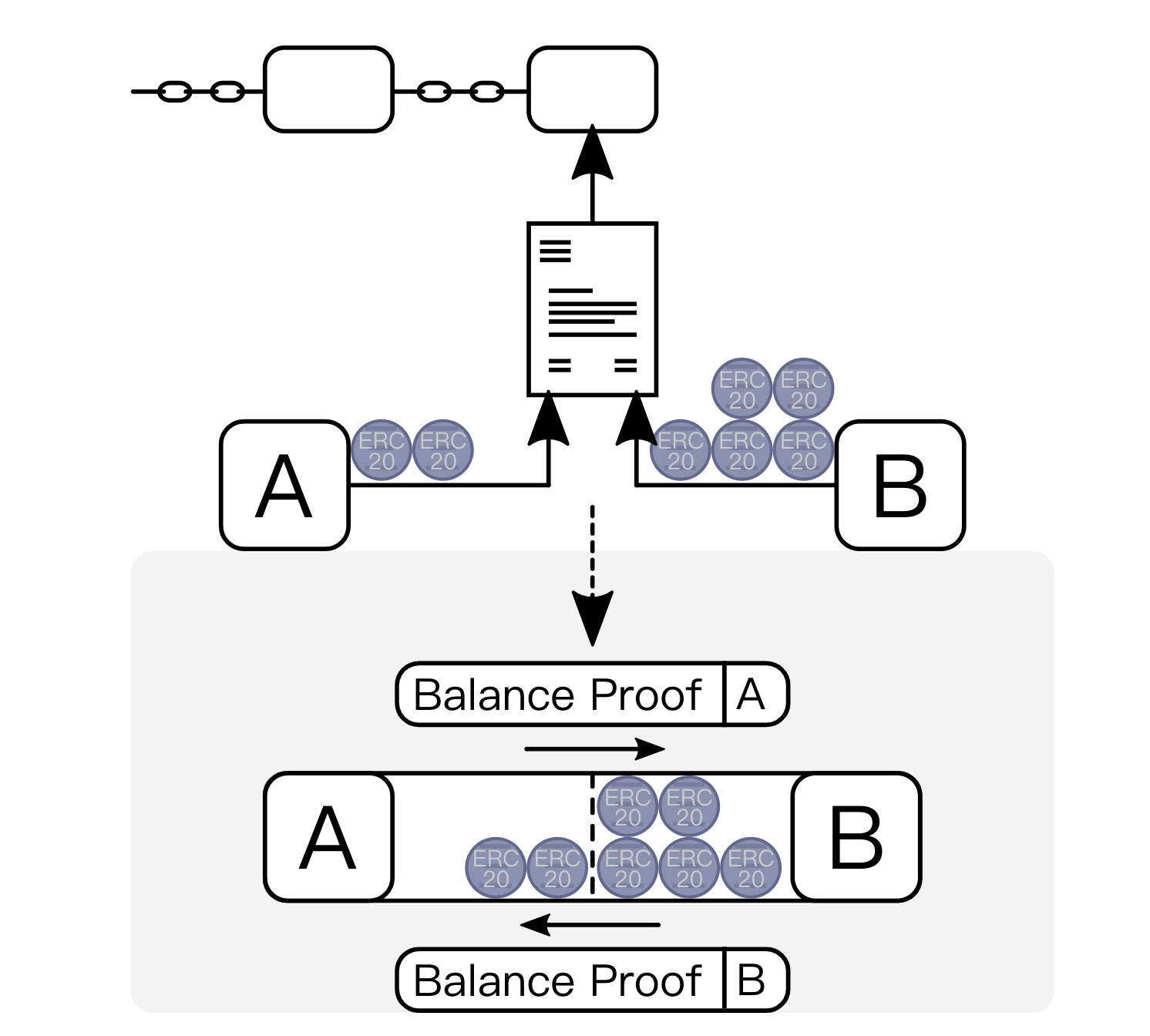


图1 雷电支付通道

在传统的经济活动中，当我们进行大额交易，比如买一栋房子，拍卖一幅名画，或签署一份重要合同时，通常需要某种类型的第三方担保或公证人，以声明资产转移的状态，确保交易的有效性。而当我们进行小额交易，比如吃一餐饭时的签单，或到学校的小卖部赊账时，我们并不需要一个公证人，否则代价实在太高，在这种情况下双方默认签个名即可。

因此同理，在区块链网络中，链上的记账共识扮演了第三方公证的角色，这对于大额交易是必要的，对于大量的小额交易，则会占用过多资源，拉高整条链上的交易成本。而雷电网络为了降低这部分小额交易的信用成本，开辟了专门的状态通道，参与到通道中的节点，仅凭数字签名便可确认交易状态。可以说状态通道就相当于一套约定俗成的规矩，这套规矩省去了信用环节，大大提高了经济效率。

### 2.3.2 状态通道及余额证明

雷电的状态通道是有生命周期的，从开启到关闭为一个周期，为了确保通道中的双方最终成功结算，在创建状态通道时双方需要把一定量的代币锁定在通道所对应的智能合约中。在这个状态通道周期结束之前，智能合约确保了锁定在里面的代币只能在两个参与者之间流动，从而防止其中一方参与者将代币二次花费给其他人。

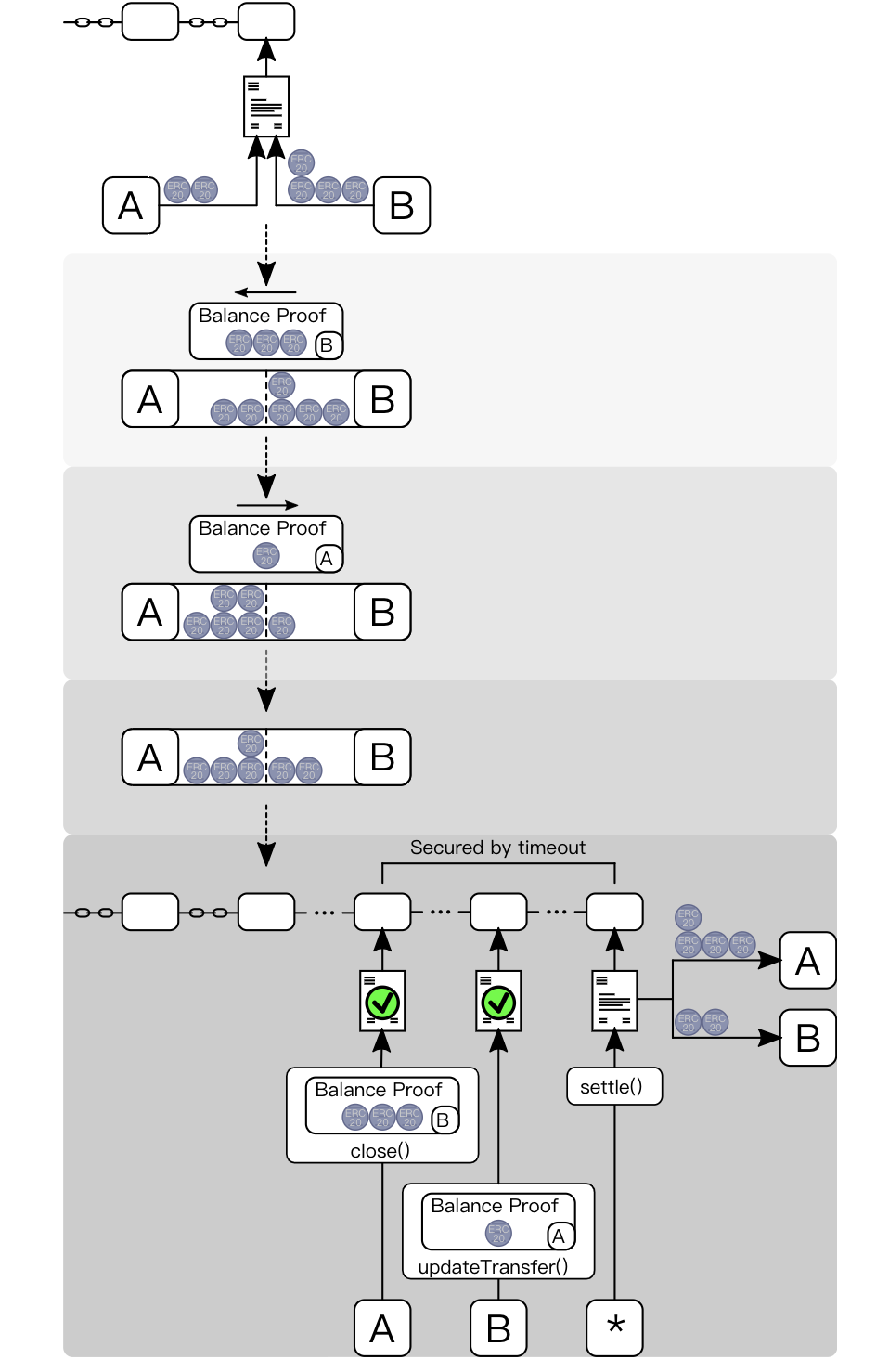


图2 雷电支付通道生命周期

在状态通道创建后，参与者可以自由地来回发送经过数字签名的交易信息，然后只保留最新的资产状态。这里引入了一个叫做余额证明（Balance of Proof）的机制，它包含了直至此刻所有发送给参与者的转账总合，即双方在通道中的余额状态，由发起方A进行数字签名后发给接收方B，以证明双方更新后的资产状态是有效的。由于每个通道有且仅有两个参与者，所以余额证明永远只有两个。

余额证明就像是每个参与者在此状态通道的酒吧消费额度卡一样（当你在酒吧喝酒的时候，可以将你的信用卡交给调酒师，他们会先将你的信用卡收起来，等你要离开时再结账，这样就不用每点一次酒就付一次钱）。多次信用交易来回进行，不断改变参与者的总余额。包含：

1. 一个随机数

2. 转账金额

3. pending锁定merkle树的根节点

4. 包含以上所有内容的签名

最后，当状态通道中一方决定在区块链上结算时，可以随时向智能合约提供其余额证明来拿回属于自己的那部分余额，同时关闭此通道。同理，另一方也需要向智能合约提供其余额证明。在双方都提交余额证明后，存款可以被提取出来。

如果另一方参与者没有及时提交余额证明，智能合约则会根据首先提出结算一方的余额证明来进行验证，只要它的解锁签名合法，智能合约就可以进行结算。通过这种方式，雷电网络确保了每个状态通道的参与者能及时提取出他们的资产。

### 2.3.3 雷电网络的形成

作为雷电网络的基础，状态通道为交易双方的互通有无提供了便捷渠道，但如果每个节点都需要跟网络中的所有节点开辟通道的话也会使成本变得很高。因为每条通道的生命周期都需要经过两次链上确认，而且由于雷电网络有余额证明，在整个状态通道生命周期间需要把用户的代币锁定在智能合约上。用户不会愿意把自己的较大金额在状态通道冻结，所以说全部这样做肯定是不合理的，不仅费事还会锁定过多的代币。

为了进一步提高效率降低成本，雷电采取了一种拓展通道的解决方案，即每一个存在多条状态通道的节点都可以成为其交易节点的中介，为这之中没有开启过状态通道的两个节点牵线搭桥，使所有状态通道都可以像水管一样联通起来，进而形成网络，让其中的小额代币自由流动起来

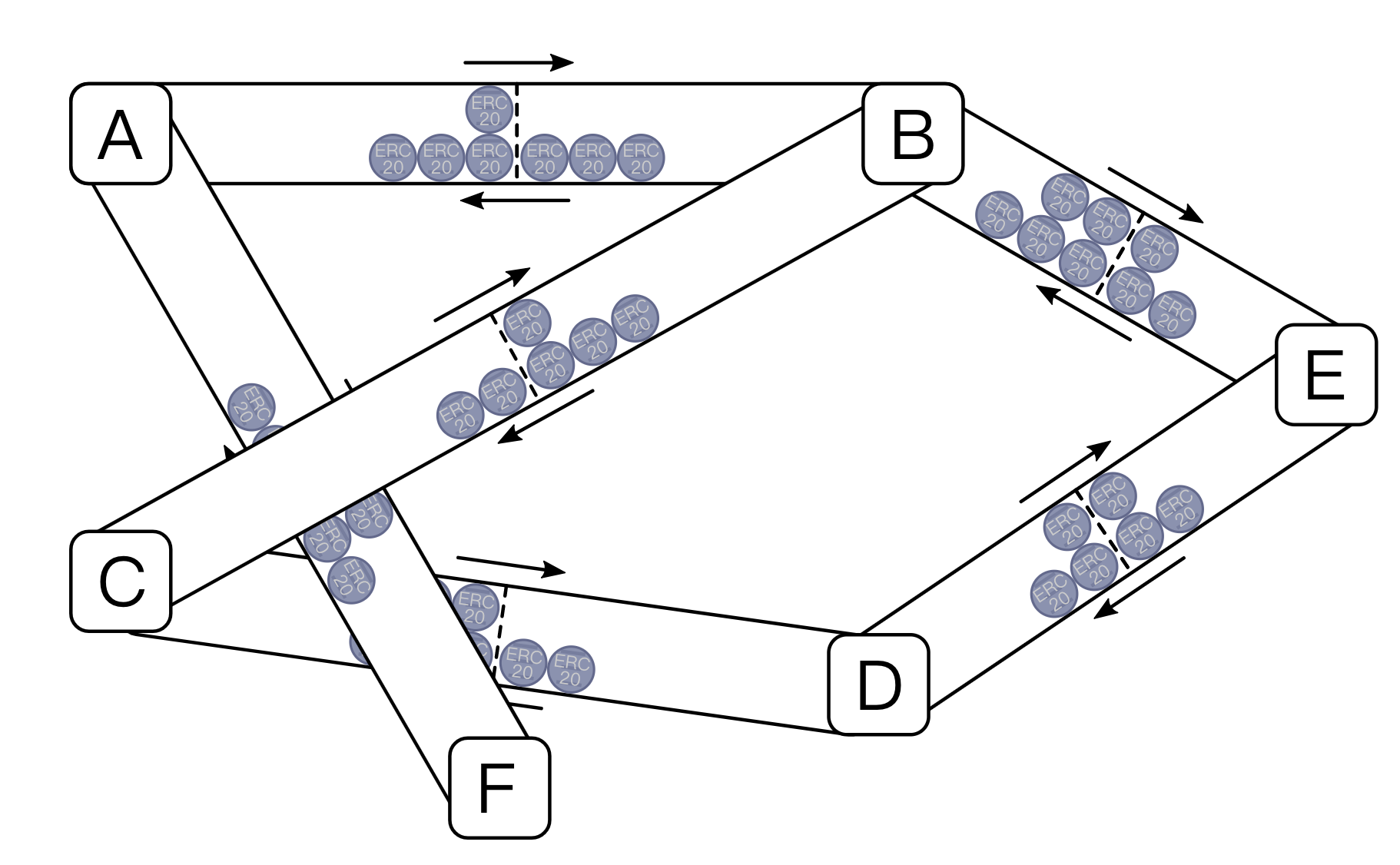
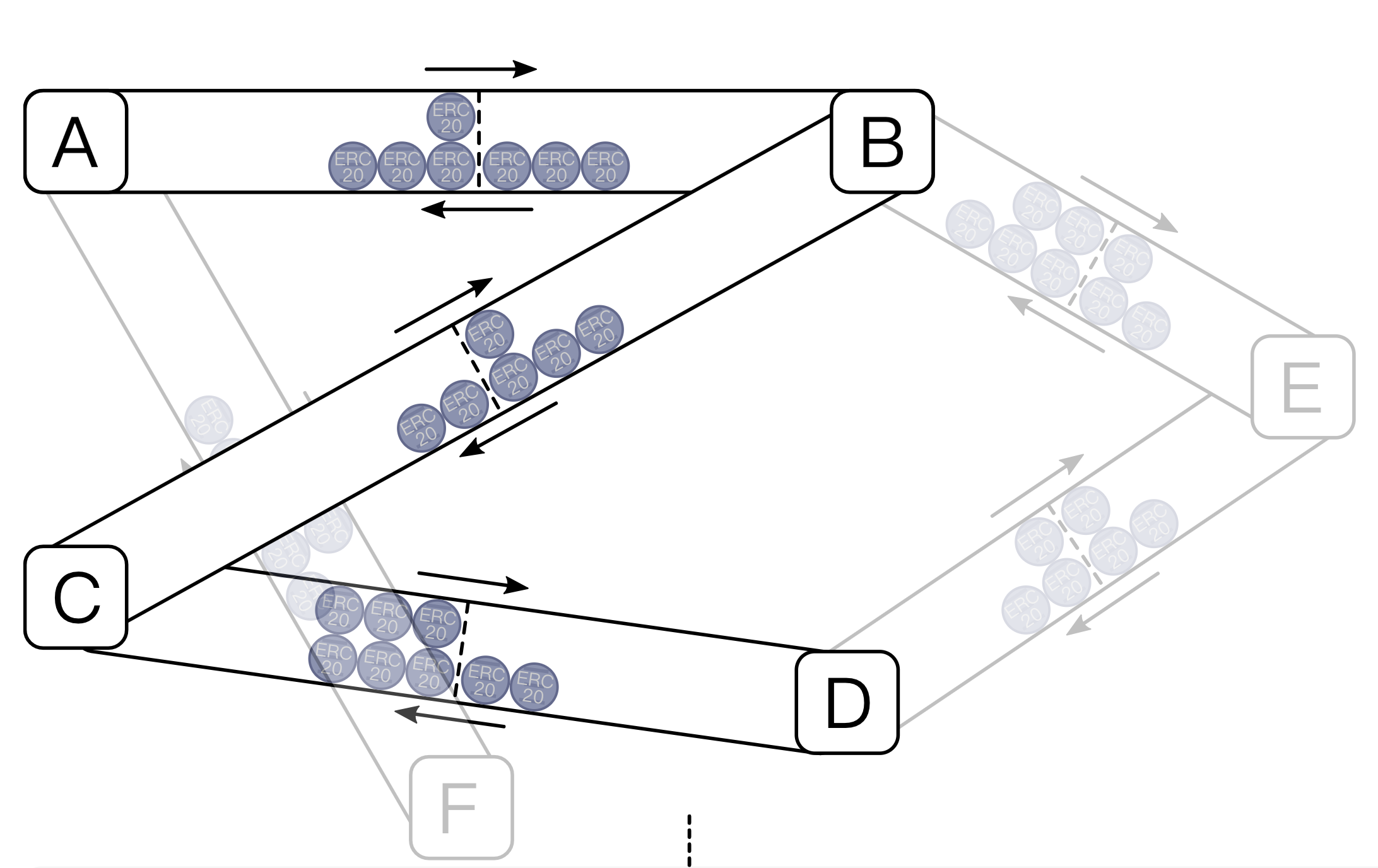
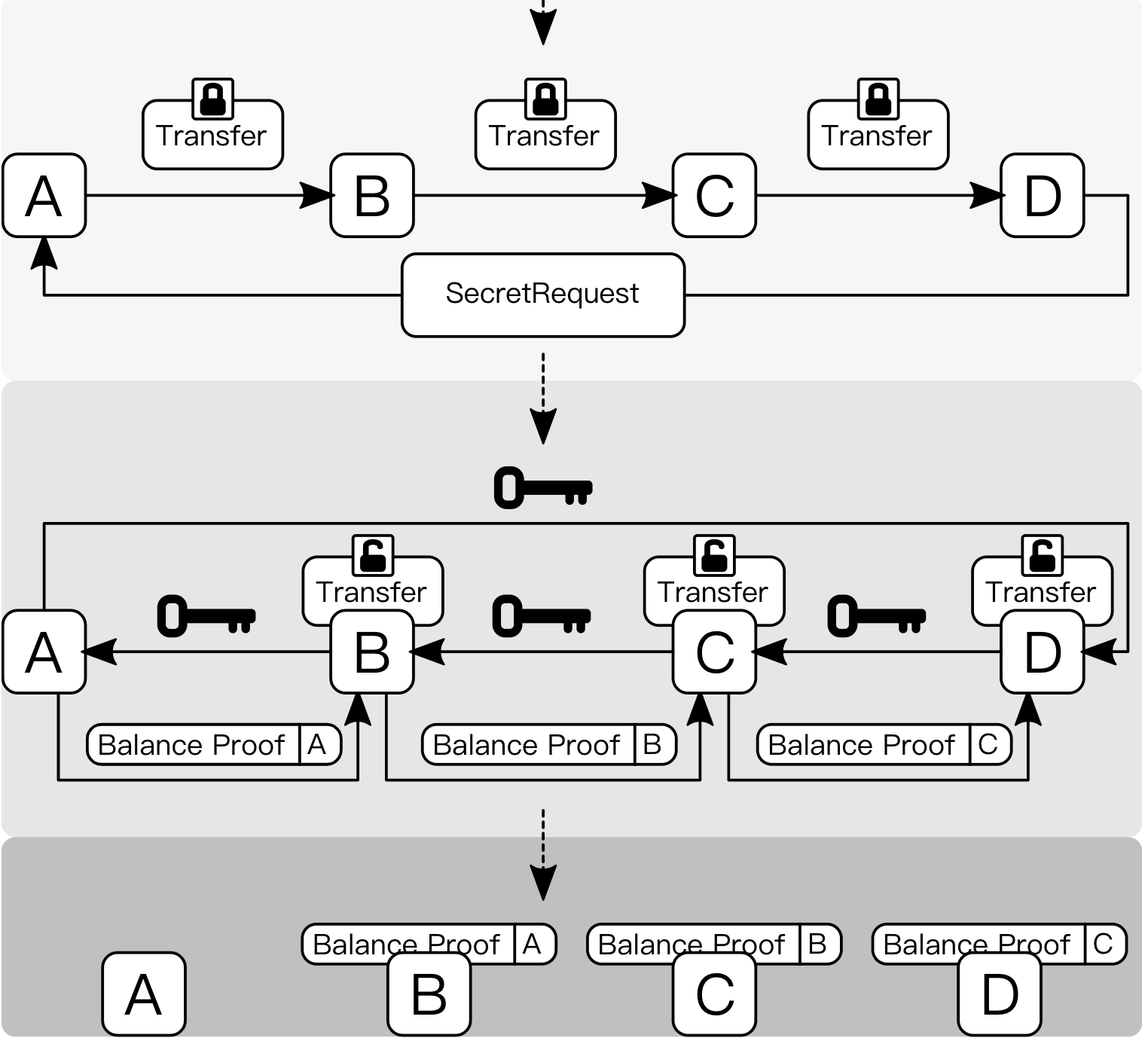


图3 雷电支付通道网络

这样一来，参与的中介节点越多，网络效应就越明显，交易容量也就越大。一旦状态网络形成规模后，任意节点间的转账会变得异常高效，交易成本和抵押代币也会大大降低，雷电网络的强大之处正在于此。不过也有人担心这种中介节点会产生额外的问题，比如私吞经手的代币，或者少数节点过于强大导致的中心化问题，事实上这种担心是多余的：

打个比方，A想把代币发给D，如下图所示。她首先必须在状态通道网络中找到一条通往D的路线。然后，该线路上的每个中介节点通过把支付渠道租借给A，来一起帮A把代币转发给D。每个中间转账都被套上了一把锁（加密哈希锁），这个锁可以防止所有这些中间转账在D确认他收到了A的付款之前被使用。一旦A决定解锁付款，她会把能够解锁的钥匙（密钥）给D。如果D现在想拿到付款，他则必须将该钥匙传递给路线上的上一个中间商C，同理C又需要把钥匙传递给其上一个中间商B来拿到属于自己的付款。

由于该路线上的每个参与者都会想立即解锁哈希锁来拿到属于他们自己的收款，所以这个密钥自然地通过状态通道路线向后传播回A。所有锁定的转账都可以使用A提供的密钥在链上进行兑换。一般来说，在状态通道未关闭前，参与者会将所有锁定的转账合并到一个余额证明中。因此，在接收到密钥之后，每个中介节点将用密钥解锁其收到的加锁转账，并且签署一个新的余额证明来把这些加锁转账合并进去，使转账间接完成。

当然，雷电网络中的中介节点很可能不会免费地提供状态通道给其他人使用。毕竟，转账将导致额外的网络流量和付款通道的资金不平衡。这些提供状态通道的参与者可以给他们的状态通道设置服务费。这些费用可以用来覆盖网络流量费用和激励中介节点重新平衡那些不平衡的状态通道，从而确保状态通道能长期被使用。好在，这是一个自由竞争的市场，由于每个参与者提供状态通道的成本很低，通过激烈的竞争，预计收费将比线上交易的收费低好几个数量级。

另外，第三方中心化确实可能存在，因为更大的节点将比小节点发送更多的转账交易，并且拥有更多的支付通道与代币存款。然而，中介节点无论多大都不能进行欺诈行为，大的节点也不能阻止较小的节点加入到网络中。一个节点停止接收和转发转账交易的时刻，就是它与网络中其他部分节点断开链接的时刻。大机构可能建立大型的转账中心来赚取转账费，但这只会有助于维护网络流动性与竞争性交易费，而不会威胁到去中心化。

### 2.3.4 雷电通道智能合约

雷电通道智能合约是为了用户操作链下支付通道包含通道规则的可执行代码。当支付通道被使用的时候，相关规则被默认生效。该通道允许：

1. 通道参与者之间的大量双向代币转移。

2. 具有到期条件和预定义规则的条代币转移。

3. 确定转移顺序的规则。

每个雷电通道都支持双向离线支付渠道。它们处理预定的代币，每个代币都有自己的结算期配置。两个参与者中的任何一个可以存入任意次数，任何数量的指定代币。

转移可以有条件地完成（条件支付），这意味着在任何给定的时间点可能有多个条件转移等待完成。这些传输由包含代币数量，到期时间和散列锁的锁结构表示。所有待处理传输的集合在merkle树中编码，并在每次传输中由其根表示。

通道容量等于由两个参与者的总存款。容量既是转移可以具有的最大值，也是待处理转移中的代币总量。容量按可用和锁定的余额划分给每个参与者。可用余额在渠道的生命周期内有所不同，具体取决于完成的转移的方向和价值。可以通过参与者的存款或对方的付款来增加。锁定的余额取决于锁转移的方向和值。它随着每次锁传输而增加，并且在转移完成时成功或以其他方式减少。

### 2.3.5 通道生命周期

1. 部署智能合约

2. 注资（使用）

3. 关闭

4. 结算

在部署之后，该频道可以从任一参与者接收多个存款。一旦交易对手承认，存款人可以使用可用余额进行转账。

一旦任何一方想要撤回其代币或出现争议，该渠道必须关闭。调用close函数后，将打开结算窗口。在结算窗口内，参与者必须更新对方状态并撤回解锁锁。一方不能进行部分撤回。

调用updateTransfer（）函数接收一个已签名的余额证明，其中包含一个通道特定数据的envelope。 这些是merkletree根，转移量和nonce（随机数）。由于节点只能提供来自交易对象的签名消息，因此用户数据未被篡改且有效。为了阻止节点提供较旧的消息，撤销余额从转移的金额中扣除，这是一个单调递增的值。 因此，没有负值转移，如果参与者提供较旧的信息，则违法者的净余额最终会清零。

### 2.3.6 雷电转账

* 直接转账

DirectTransfer不依赖于锁来完成。 一旦网络数据包被发送，它将自动完成。 由于Raiden运行在无法保证传递的异步网络之上，因此传输无法自动完成。关于直接转移的主要考虑因素如下：

消息未被锁定，这意味着transfer\_amount递增，并且消息可用于撤销令牌。 这意味着无论是否获得服务，付款人都无条件地转移令牌。 在付款人/收款人中假设信任以完成交易。

一旦将消息发送到网络，发送方必须假定传输已完成，没有解决方法。一旦收到消息，付款人将只知道转移。

例如：

* A 需要向B转账 n token
* **A用以下信息创建一个转账：**
  + transferred\_amount = current\_value + n
  + [locksroot](https://raiden-network.readthedocs.io/en/stable/glossary.html#term-locksroot) = current\_locksroot\_value
  + nonce = current\_value + 1
* A对这笔转账进行签名并发送给B，这时候应该认为转账已经完成
* 中介转账

中介转账是一种hashlocked转账。 目前raiden只支持一种锁。 锁具有正在传输的数量，用于验证解锁密钥的密钥锁以及用于确定其有效性的锁期限。

中介传输具有发起者和目标以及其间的跳数。跳数也可以为零，因为这些传输也可以发送给直接合作伙伴。假设N个跳数，中介传输将需要6N + 8个消息才能完成。这些是：

* N + 1 mediated or refund messages
* 1 secret request
* N + 1 secret reveal
* N + 1 secret
* 3N + 4 PROCESSED

例如：

* A 需要向B转账 n token
* **A用以下信息创建一个转账：**
  + - transferred\_amount = current\_value
    - lock = Lock(n, hash(secret), expiration)
    - locksroot = updated value containing the lock
    - nonce = current\_value + 1

A对转账进行签名，发送给B；B发送SecretRequest请求获取用来取回转账资金的Secret；A把秘密发送给B，此时她必须假设转账完成了; B接受了这个秘密，此时有效地保证了N令牌的转移; B向A发送一条秘密消息，通知她这个秘密是已知的，并且作为离链同步的请求。最后，A给B发了一个秘密。这还充当同步消息，通知B锁将从merkle树中移除，并更新transferred\_amount和locksroot值。

* 退款转账

RefundTransfer是在节点无法向前传递信息的特殊情况下使用的中介传输，必须进行路由回溯。

### 2.3.7 中介转账详解

Raiden不能依赖直接通道进行大部分操作，特别是如果大多数操作只针对只接收一次传输的目标节点。中介转移是一种价值转移形式，允许Raiden节点之间的无信任合作，以促进价值流动。

中介转账依靠锁来确保安全。锁只能通过了解其secret才能解锁。此信息用于确定传输是否已完成，并在中介链中的所有节点之间共享。锁定操作允许每个参与者安全地完成他们的传输而不需要信任。

假设Alice < - > Bob < - > Charlie的路径。 Alice与Charlie没有直接通道，因此Alice可以打开新频道或通过其他节点中介转账。Bob是一个hop，Alice拥有一个开放的频道并被认为适合路由。

Bob的角色是调解发起人Alice和目标Charlie之间转账的中介人。给定路径中的节点数可能会有所不同，但角色和工作方式相同。

Bob将首先从Alice收到中介转账t1。这种转账有条件地锁定了Alice生成的secret，所以由Alice决定是否完成转账。 Alice发送的转账是有效的余额证明，可以由Bob在链上随时用于声明当前收到的金额，如果知道了解锁秘密，也可以取回待处理的转账t1。因为Bob知道他可以声明转移价值，并且Bob已经正确地完成了所有验证检查以保证余额证明包含正确的转移金额，merkle root有效地表示所有待处理的转账，并且Alice确实具有可用的余额以供使用通过给定的转移，Bob可以安全地转发转移。

然后Bob将在频道Bob < - > Charlie上创建一个新的传输t2。 t2通过转移t1备份其值，因为Bob不是付款人。在给定的示例中，Alice是Bob的付款人，Charlie是Bob的付款人。在跳数增加的情况下，将有更多的付款人/收款人对，每个对有一个中介。传输t2将是另一个有条件锁定的转账，并且中介人负责为此转账使用相同的锁定量和hashlock。

一旦Charlie已经接收到中介转移，它将从Alice请求解锁转账的secret。此时，Alice知道某个节点将支付Charlie。Charlie通知Alice有关收到的锁的数量，令牌和hashlock，因此可以确定它是正确的传输。现在，发起人Alice通过向Charlie透露secret来完成转账。

一旦Charlie知道secret，支付就会从支付链的后面流到前面。这意味着他们从Charlie开始，他将向Bob获取资金，告知Bob有关已知秘密，允许Bob请求从Alice获取资金。

2.3.8 哈希锁

锁具有两个部分，一个用于确定锁定token数量的amount，以及用于定义如何解锁token的规则。锁本身独立于与其关联的通道或令token。将哈希锁绑定到特定通道的是余额证明的merkle树。

Raiden目前严重依赖哈希时间锁。它们是安全无信任中介转移的基本要素。这种锁有两个额外的数据属性，一个哈希原像和一个到期时间。如果在其到期之前显示散列的原像，则锁被解锁。在raiden中，preimage被称为secret，它的hash被称为hashlock。secret只是32字节的加密安全随机数据。散列锁可以是任何加密安全散列函数的结果，但Raiden目前依赖于keccak散列函数。

使用此锁定构造，可以：

通过依赖相同的hashlock但不同的到期时间来调解token传输。

执行令牌交换。使用相同的散列锁进行针对不同token的两次中介传输

# 三、雷电客户端调研

雷电目前包含三个独立的项目：μRaiden（微雷电）、雷电网络（Raiden）和Raidos（雷电 2.0）。其中μRaiden在2017年11月30日，正式在以太坊主链上线，μRaiden（Micro Raiden）是一个支付渠道框架，用于双方之间频繁，快速和免费的基于ERC20令牌的微支付，不支持多跳传输费用，因此只允许将Token单向发送到预定的接收器。雷电网络正在开发过程中。开发者预览版已经发布，可以让Dapp开发人员对雷电网络API和系统属性有一个初步印象，构建与雷电基于Ropsten测试网络交互的原型。在当前状态下，这项技术目前并不适用于产业应用。仍需研发一系列有关核心协议的主要开发工具，甚至是对核心协议本身进行改良。Raidos（雷电 2.0）目前只是在计划阶段，开发还没有正式启动。

## 3.1 消息规范

1. 余额证明（Balance Proof）

用于智能合约来更新支付通道的信息，定义如下：

ecdsa\_recoverable(privkey, keccak256(balance\_hash || nonce || additional\_hash || channel\_identifier || token\_network\_address || chain\_id)

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| nonce | uint256 | 从1开始单调增，用于确定转账顺序 |
| transferred\_amount | uint256 | 通道历史中的总转移量（单调增值） |
| locked\_amount | uint256 | 当前锁定金额 |
| locksroot | bytes32 | 锁定哈希的merkle树的根 |
| token\_network\_identifier | address | TokenNetwork合同的地址 |
| channel\_identifier | uint256 | TokenNetwork合同中的通道标识符 |
| message\_hash | bytes32 | 消息的哈希值 |
| signature | bytes | 以上数据的椭圆曲线256k1签名 |
| sender | address | 消息的发送人 |
| chain\_id | uint256 | EIP155中定义的链标识符 |

2. 哈希锁（HashTimeLock）

到期时间必须大于当前的区块高度且小于通道的结算期。定义如下：

keccak256(expiration || amount || secrethash)

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| expiration | uint256 | 可以转账结算时的区块高度 |
| locked\_amount | uint256 | 锁定的token量 |
| secrethash | bytes32 | keccak256 hash of the secret |

3. 签名消息（SignedMessage）

SignedMessage是一种基础数据结构，几乎包含在由通道参与者发送或接收的每条消息中

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| Cmdstruct | uint256 | 本地储存结构 |
| sender | uint256 | 用户 |
| signatrue | []byte | 用户签名 |

4. ACK

用于确认已收到某些消息的数据结构。它只是回显了收到的消息的哈希值。

| **Field Name** | **Field Type** | **Description** |
| --- | --- | --- |
| sender | uint256 | 收到信息的用户地址 |
| ECHO | Hash | 消息的哈希值 |

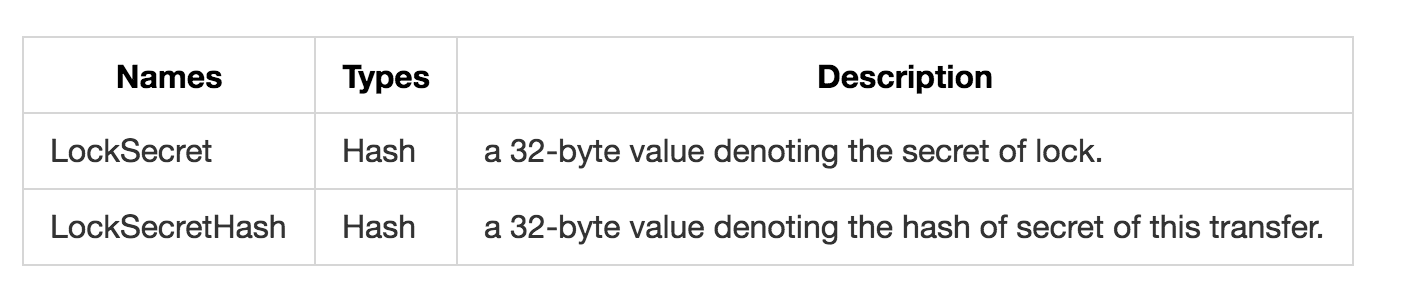
5. 密钥请求（SecretRequest）

SecretRequest是主要在支付渠道中的转账接收者想要获得转账秘密时使用的消息

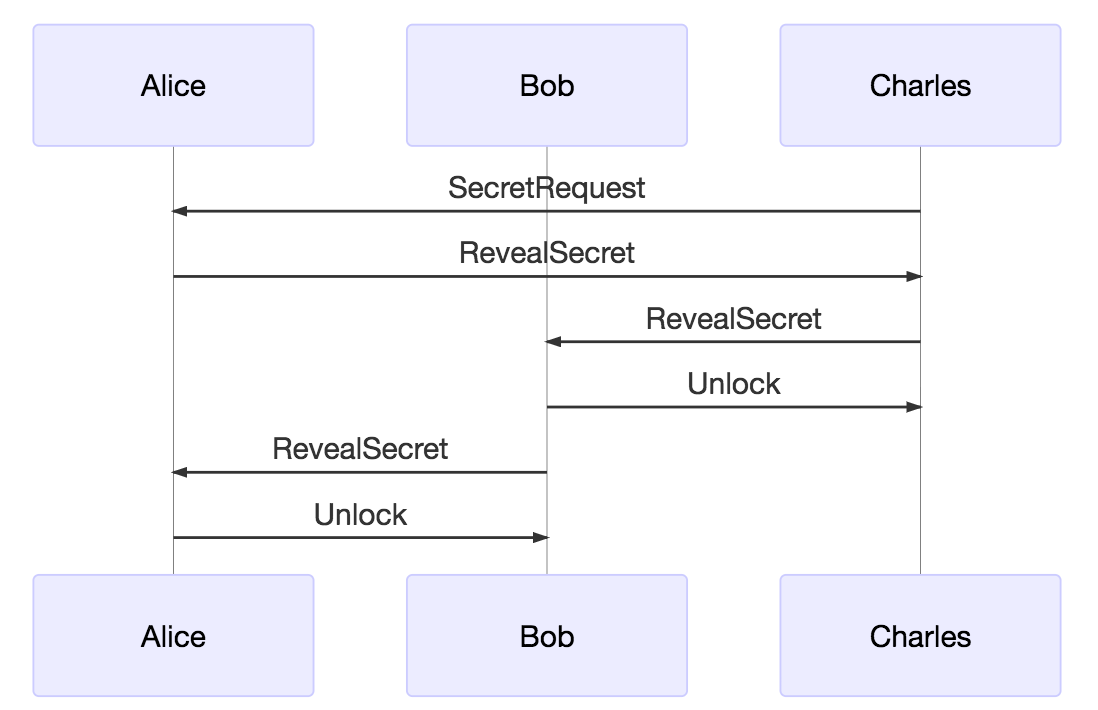


6. 密钥揭露（RevealSecret）

RevealSecret是在支付渠道中转账发起人的情况下使用的消息，一旦他从需要该秘密的参与者那里收到SecretRequest，就会揭示该参与者的秘密。



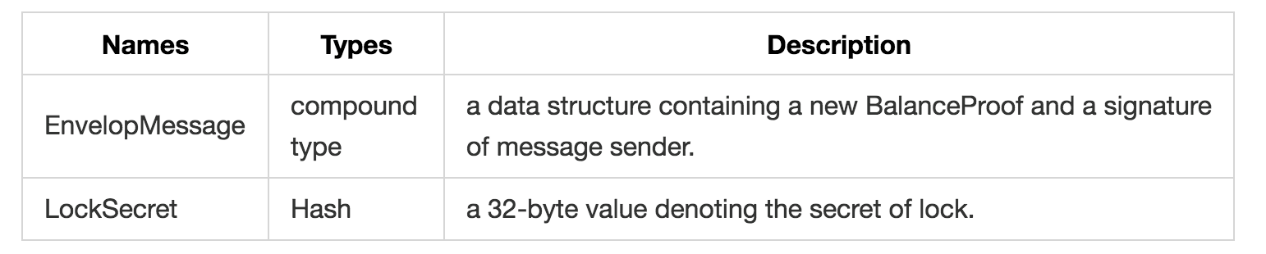
7.解锁（unlock）



以中介转账为例：

某频道有三个参与者，A，B和Cs。整个工作流程是

* 在C收到A的转账后，他希望通过SecretRequest向A获取秘密。
* 一旦A收到这个SecretRequest，如果没有问题，那么A将把这个秘密提供给C，通过RevealSecret。
* 至于C，当他收到A的秘密，以获得他应得的钱时，C准备透露他前任跳跃节点B的秘密，通过RevealSecret。
* 当B收到这个秘密时，在他完成验证后，他有两个选择：
  + 首先，如果B可以确保A是一个诚实的人，并将解锁锁定在她的BalanceProof中的钱，B立即发送Unlock给C以解锁转账。但这不是我们的情况。
  + 实际上，B不能依赖A的诚信，所以他应该确定他的钱不会被欺诈行为者偷走。任何知道这个秘密的人都可以在链上注册这个秘密，以获得他们应得的资金。所以B不担心他的token，他可以直接解锁他的转账给C。



8. 直接转账（DirectTransfer）

DirectChannel是主要用于链接两个频道参与者的直接路径的情况下的消息。两者都只是将他们的转移通过发送到他们的频道对应DirectTransfer。DirectTransfer只有在此直接渠道仍然开放的情况下，任何时候都可以通过此直接付款渠道发送任何人。

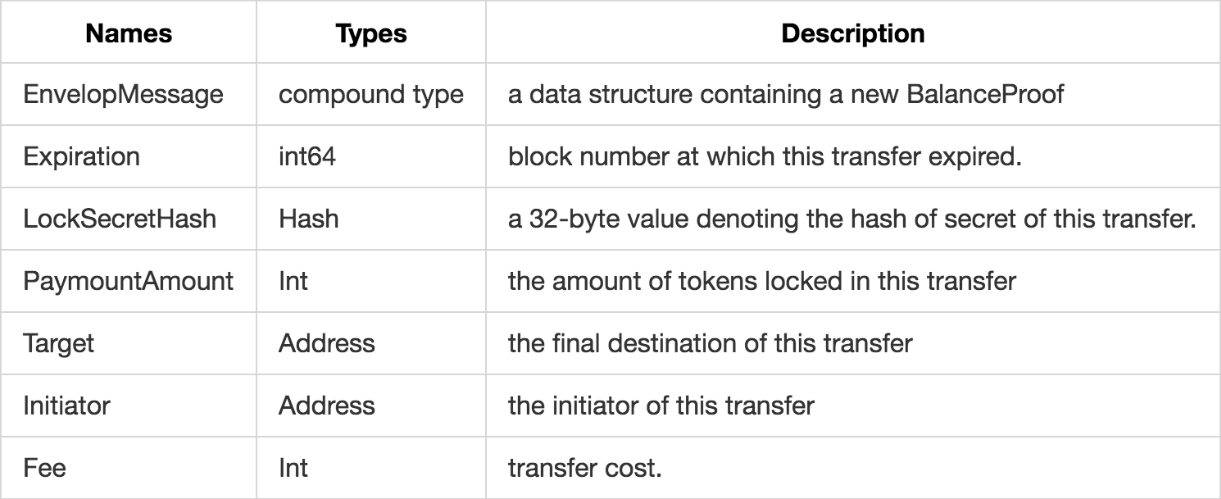
1. 存在与给定的chain ID，token网络地址和通道标识符匹配的通道。
2. 相应的通道处于打开状态。
3. 转移总量是比以前的值大，并将其增加的量小于或等于所述参与者的当前余额。
4. 相对于以前的余额证明，nouce随机数增加1。
5. 该locksroot没有改变
6. 该锁定量没有变化



9. 中介转账（MediatedTransfer）

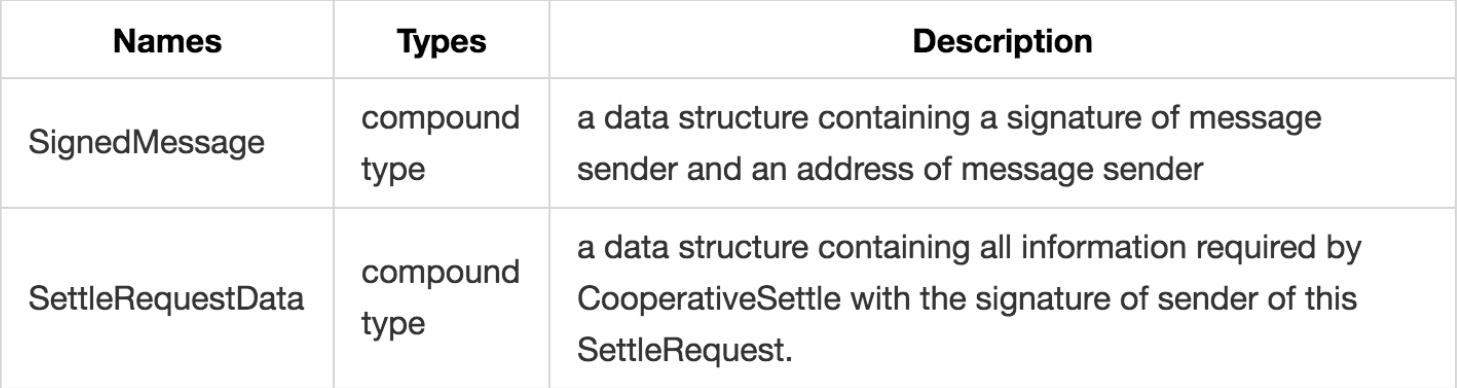
MediatedTransfer是主要在参与者没有直接路由链接到其转账接收者的情况下采用的消息。

1. 存在与给定的chain ID，token网络地址和通道标识符匹配的通道。
2. 相应的通道处于打开状态。
3. 相对于以前的余额证明，nouce随机数增加1
4. 所述locksroot必须改变，则新值必须等于一个新的树的根，它具有所有以前的锁加上消息中提供的锁。
5. 该锁定量必须增加，新值等于旧值加上本次锁定的数量。
6. 本次锁定的数量必须小于参与者的余额。
7. 锁定到期时间必须大于当前区块高度。
8. 转移总量不得改变。



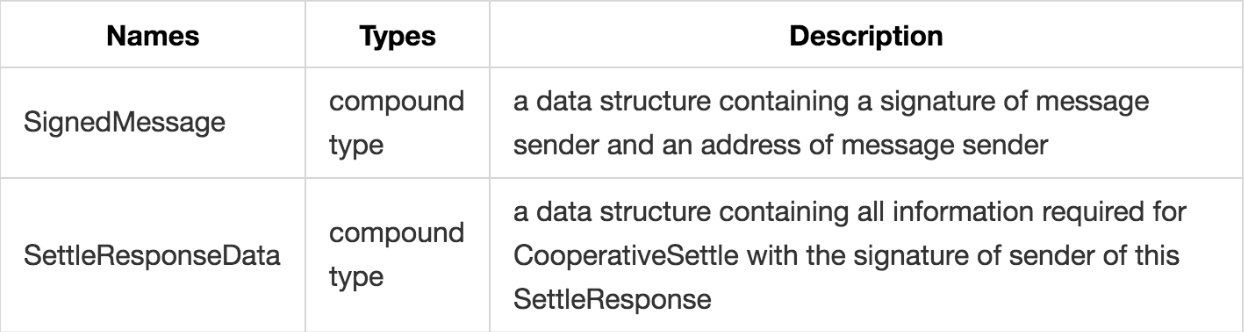
10. 结算请求（SettleRequest）

SettleRequest是渠道参与者在需要CooperativeSettle他们之间的支付渠道时所采用的消息



11.结算回应（SettleResponse）

SettleResponse是渠道参与者在需要确认意图并同意时所采用的信息



## 3.2 智能合约

### 3.2.1 TokenNetworkRegistry合约

首先是需要向雷电网络注册Token, 调用如下函数，输入已在以太坊上部署的Token地址，以此部署新的TokenNetwork合约

function createERC20TokenNetwork(address token\_address) public

会触发solidity事件

event TokenNetworkCreated(address token\_address, address token\_network\_address)

token\_address为已在以太坊上部署的Token地址

token\_network\_address为新部署的TokenNetwork合约的地址

假设用户拥有一定数量的ERC20或ERC223 token，但该token尚未在raiden上注册。该用户必须先将其token注册到TokenNetwork合约中，然后才能在链下支付中使用其token。Raiden为每种token提供单独的TokenNetwork合约。一旦注册了特定类型的token，该token的用户就能够连接到该token网络并调用该TokenNetwork的相关功能

### 3.2.2 TokenNetwork 合约

TokenNetwork主要是一个提供与支付渠道交互的接口的合约。付款渠道只能从本合约中的token\_address转移代币。TokenNetwork集成了支付渠道在其生命周期中与之交互的大多数功能，包括ChannelOpen，Deposit / Withdraw，ChannelClose，ChannelSettle等。除此之外，TokenNetwork合约定义了支付通道的全局变量，以便在调用某些功能时使用。这些变量包括

* Token public token : TokenNetwork中使用的代币实例
* SecretRegistry public secret\_registry : SecretRegistry的合约实例，用于在链上提交secret

uint64 constant public punish\_block\_number = 5 : 提交欺诈证明的期限，如果欺诈节点提交updatetransfer或在unlock结算时，诚实节点有时间提交证据以保证其资金

* uint256 public chain\_id : 在BalanceProof中使用防止双花
* channel\_identifier: 32位bytes，代表频道参与者及TokenNetwork地址的组合哈希值。

通道信息获取函数：

假设用户希望通过aiden进行离线支付，以A和B为例。A计划将30个TOKEN转移给鲍勃。为了进行安全转移，A必须使用getChannelIdentifier知道她之前没有与Bob建立支付渠道。如果没有，A需要知道通道状态getChannelInfo。如果A能够使用该频道，那么A使用getChannelParticipantInfo检查B以及A存储在本地中的数据一致性。

1. 标识符

function getChannelIdentifier(address participant, address partner)

view

public

returns (uint256 channel\_identifier)

2. 通道状态

function getChannelInfo(

uint256 channel\_identifier,

address participant1,

address participant2

)

view

external

returns (uint256 settle\_block\_number, ChannelState state)

state比较重要 返回结果有：NonExistent - 0, Opened - 1, Closed - 2, Settled - 3, Removed - 4.

3. 通道参与者信息

function getChannelParticipantInfo(

uint256 channel\_identifier, （通道标识符，合约进行分配）

address participant,（通道发起者一方）

address partner（通道发起者另一方）

)

view

external

returns (

uint256 deposit,（通道资金，Settled，Removed状态必须为0）

uint256 withdrawn\_amount,（撤回资金）

bool is\_the\_closer,（是participant关闭通道则为TRUE）

bytes32 balance\_hash,

uint256 nonce,

bytes32 locksroot,

uint256 locked\_amount

)

**通道开启函数：**

假设用户有意使用raiden从链中付款。我们以A和B为例。A计划向B发送30个Token。如果这是A和B第一次使用raiden进行离线支付，并且没有通道直接连接它们。因此，作为转账发起者，A必须调用openChannel合同，并将她的地址和B的地址作为函数参数传递。一旦它无错误地返回，就会创建一个连接A和B的直接通道。那时并没有Token存放在这个频道中，A和B可以在创建频道后进行存款。

function openChannel(address participant1, address participant2, uint256 settle\_timeout) public returns (uint256 channel\_identifier)

触发event：

event ChannelOpened(

uint256 indexed channel\_identifier,

address indexed participant1,

address indexed participant2,

uint256 settle\_timeout （雷电网络通道维持时长，用区块数表示）

);

**注资函数：**

function setTotalDeposit(

uint256 channel\_identifier,

address participant,

uint256 total\_deposit,

address partner

)

public

触发event:

event ChannelNewDeposit(

uint256 indexed channel\_identifier,

address indexed participant,

uint256 total\_deposit

);

无论对方是否在线，都可以向通道注资

**撤资函数：**

允许频道参与者从频道撤回Token而不关闭它。可以被任何人调用。每个已签名的撤销消息只能被调用一次。

假设A计划向B转让30个的Token。一旦此传输成功完成，A的存款金额为70，Bob的金额为30。此时通道中没有任何Token被锁定。如果现在其中一个想要撤回部分存放的代币，该怎么做？假设A希望从该频道撤回20个Token，她可以直接调用Withdraw，因为当前没有Token被锁定。通过协商方式，A与B就她目前的余额（70个代币）和提取金额（20个代币）达成协议。Withdraw完成后，A和B的频道余额变为A的50个令牌和B的30个令牌。然后转移可以继续。

function Withdraw(

uint256 channel\_identifier,

address participant,

uint256 total\_withdraw,

bytes participant\_signature,

bytes partner\_signature

)

external

触发event:

event ChannelWithdraw(

uint256 indexed channel\_identifier,

address indexed participant,

uint256 total\_withdraw

);

需要通道参与者双方签名才能实现撤资

**通道关闭函数：**

如果其中一个通道参与者不想继续使用此通道，那么他可以要求关闭此通道并更新其参与者的BalanceProof。这时会触发挑战期，在挑战期超时之前，此通道无法结算。

例如在A和B完成了大量转账之后，A希望停止转账并计划关闭该通道。因此，Alice决定调用closeChannel关闭此通道。为了避免丢失Token，A必须将最新的BalanceProof从B更新到主链以保留记录。

function closeChannel(

uint256 channel\_identifier,

address partner,

bytes32 balance\_hash,（余额数据哈希，keccak256(transferred\_amount, locked\_amount, locksroot)）

uint256 nonce,

bytes32 additional\_hash,

bytes signature

)

public

触发event：

event ChannelClosed(uint256 indexed channel\_identifier, address indexed closing\_participant);

无论partner是否在线，participant都可以关闭通道

**更新余额证明函数：**

例如Bob收到Alice计划关闭它们之间的支付渠道的消息，为了避免丢失令牌，Bob还必须提交Alice发送的最新BalanceProof。然后Bob可以调用updateBalanceProof将该证明更新到主链并保存记录。

function updateNonClosingBalanceProof(

uint256 channel\_identifier,

address closing\_participant,

address non\_closing\_participant,

bytes32 balance\_hash,

uint256 nonce,

bytes32 additional\_hash,

bytes closing\_signature,

bytes non\_closing\_signature

)

external

触发事件

event NonClosingBalanceProofUpdated(

uint256 indexed channel\_identifier,

address indexed closing\_participant,

uint256 nonce

);

**解锁函数：**

通过提供待处理转账数据的整个merkle树来解锁所有待处理的转账，merkle树用来计算merkle根，必须和最新的余额证明中的locksroot一样。

例如在Alice与Bob进行转账的过程中，可能有部分转移尚未完成。假设Alice和Bob的通道中的token数量分别为50和20。在Alice的部分，有10个token被锁定，Alice打算发送给Bob。在Alice关闭支付渠道后，如果此锁定转移的secret已在链上公开，则Bob可以调用unlock以获取这10个令牌。如果Bob是潜在的欺诈者，并且他计划通过之前的BalanceProof解锁此转移，那么上层可以验证Bob的欺诈意图并采取相应的行动。

function unlock(

uint256 channel\_identifier,

address participant,

address partner,

bytes merkle\_tree\_leaves

)

public

event ChannelUnlocked(

uint256 indexed channel\_identifier,

address indexed participant,

address indexed partner,

bytes32 locksroot,

uint256 unlocked\_amount,

uint256 returned\_tokens

);

**惩罚函数：**

主要是惩罚欺诈性解锁行为，通道结算完成之间，会留有一定的惩罚申请时间。如果任何参与者解锁已经被解锁的转账以获得额外的好处，那么他们必须受到惩罚。

在解决窗口之后，Alice调用punishObsoleteUnlock函数来检查Bob是否解锁了废弃锁。假设在通道关闭之前Alice的通道状态是50令牌，而Bob的通道状态是20令牌。Alice从存储信息中检索有关Bob的废弃锁的信息，并将其与链上的解锁结果进行比较。如果Bob已经解锁了丢弃的锁，则根据惩罚机制，Alice获取所有Bob的令牌，即Alice 70 token，Bob 0令牌。

function punishObsoleteUnlock（

address beneficiary，

address cheater，

bytes32 lockhash，

bytes32 additional\_hash，

bytes cheater\_signature

）

public

触发事件：

event ChannelPunished（

bytes32 indexed channel\_identifier，

address beneficiary

）;

**通道结算函数：**

单边结算通道会将token返回给频道参与者。任何人都可以调用该功能，只能调用一次

如果Alice在惩罚窗口期间没有找到Bob解锁被放弃的锁，那么Alice可以在结算窗口期和惩罚窗口期之后调用settleChannel函数。根据两个参与者提交的参数，计算需要转移的令牌数量并发送到相应的地址。如Alice 40令牌，Bob 30令牌。结算完成后，该频道将被关闭。

function settleChannel(

uint256 channel\_identifier,

address participant1,

uint256 participant1\_transferred\_amount,

uint256 participant1\_locked\_amount,

bytes32 participant1\_locksroot,

address participant2,

uint256 participant2\_transferred\_amount,

uint256 participant2\_locked\_amount,

bytes32 participant2\_locksroot

)

public

event ChannelSettled(

uint256 indexed channel\_identifier,

uint256 participant1\_amount,

uint256 participant2\_amount

);

**合作通道结算函数：**

允许参与者合作并提供他们的余额证明和签名。这会立即关闭通道，而不会触发一个挑战期。任何人都可以调用该功能，只能调用一次。合作结算时通道不能包括锁。

function cooperativeSettle(

uint256 channel\_identifier,

address participant1\_address,

uint256 participant1\_balance,

address participant2\_address,

uint256 participant2\_balance,

bytes participant1\_signature,

bytes participant2\_signature

)

Public

event ChannelCooperativeSettled（

bytes32 indexed channel\_identifier,

uint256 participant1\_amount,

uint256 participant2\_amount

）

### 3.2.3 SecretRegistry合约

该合约储存了中介转移中secret揭露时的区块高度，充当安全措施，以允许第三方在某些节点可能脱机时解锁锁定的token

function registerSecret（bytes32 secret）public returns（bool）

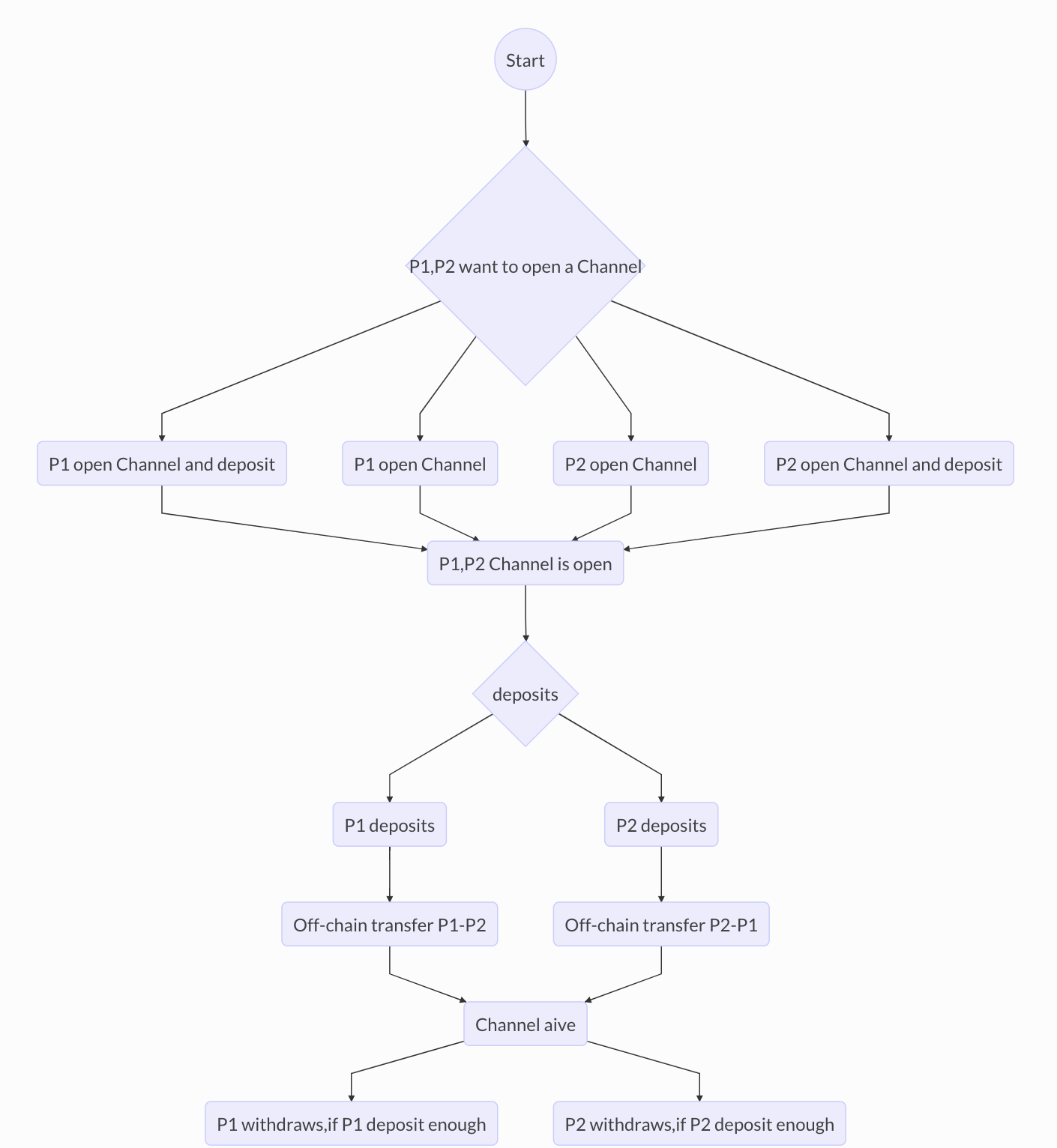
event SecretRevealed（bytes32 indexed secrethash，bytes32 secret）;

function getSecretRevealBlockHeight（bytes32 secrethash）public view returns（uint256）

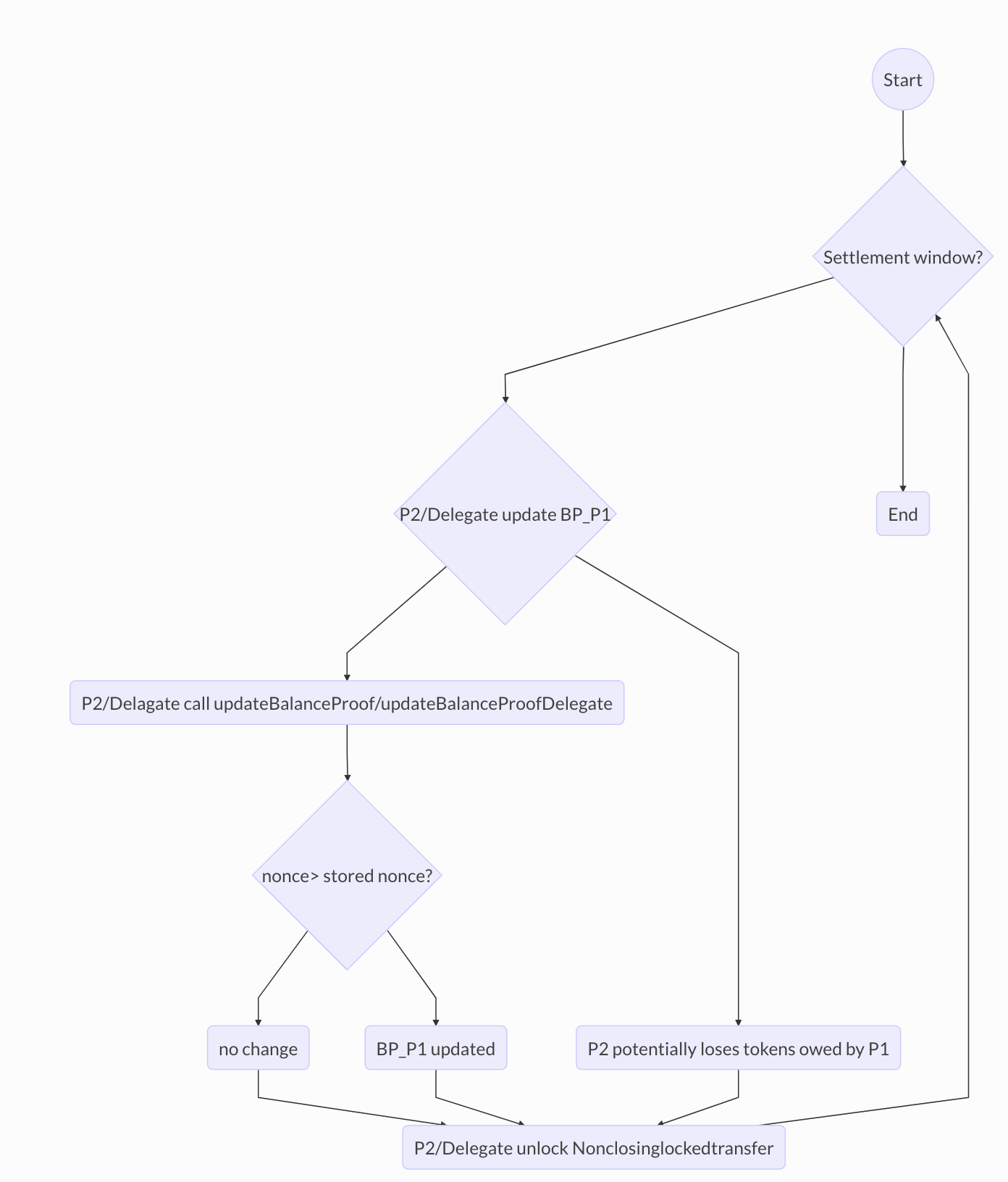
在Alice和Bob的转移过程中，可能有一些交易没有完成。假设Alice的频道状态是50（10）令牌，Bob的频道状态是20令牌，括号中的10个Token为Alice转账给Bob的，并且被锁定。如果Bob得到一个秘密并要求Alice在秘密到期之前回答解锁消息，并且没有得到Alice的回复，Bob可以选择在链上注册该秘密（call registerSecret函数）并获取注册时的块数（调用getSecretRevealBlockHeight函数）。

**附图：**

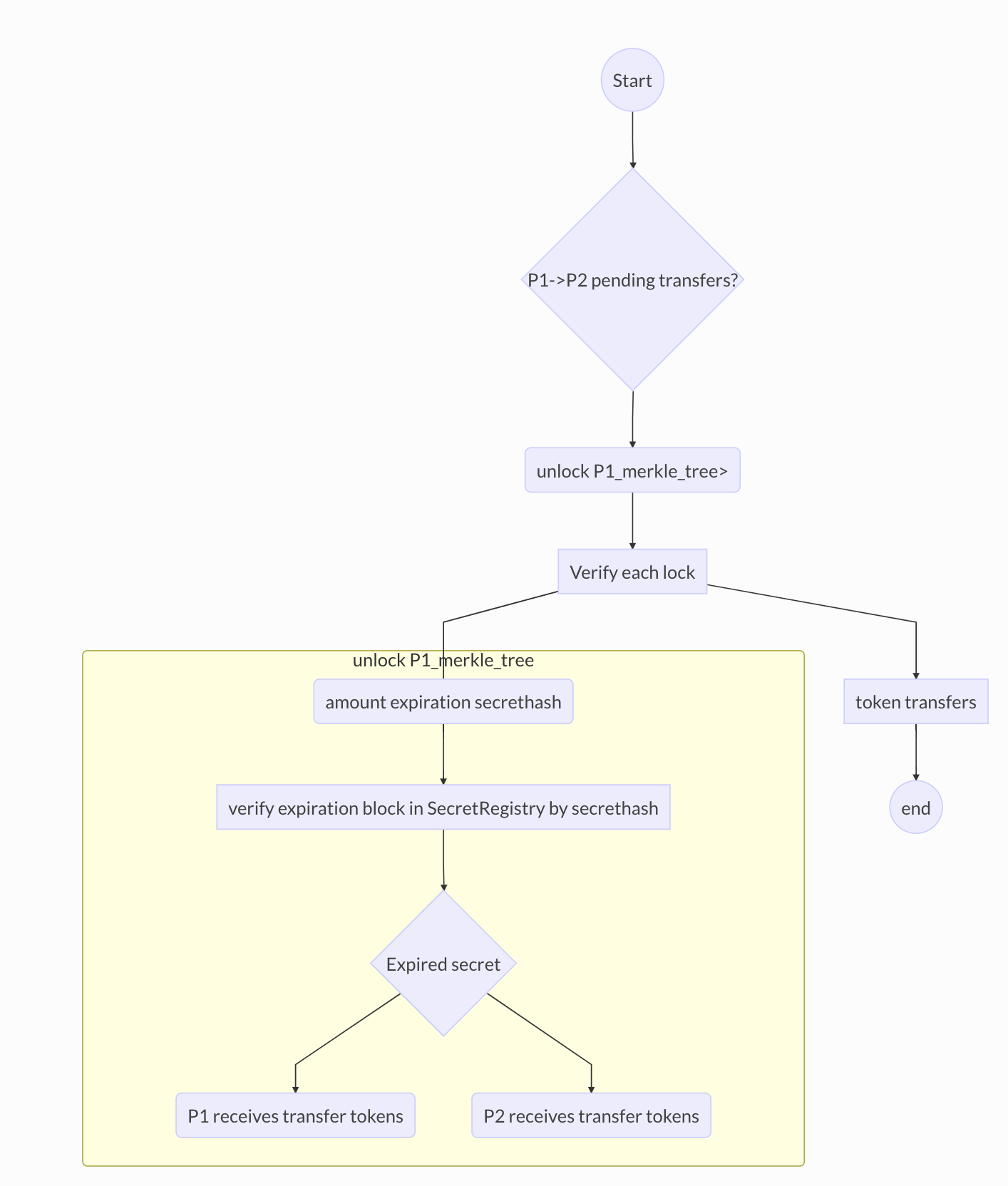
1.通道生命周期



2. 通道结算



3.交易解锁



# 四、雷电网络总结

## 4.1 雷电网络工作流程

以太坊是一个图灵完备的智能合约平台，很多复杂的工作以太坊在底层就已完成了，所以雷电网络的实现其实比闪电网络更简单。雷电网络不像闪电网络，必须依赖多重签名地址予以实现，而是直接靠以太坊的智能合约来实现。

第一步：在以太坊上布署一个智能合约，理论上来说，既可以每建一个通道，就布署一个合约，也可以为一种代币布署一个合约。前者实现比较简单，但耗费gas太多，后者实现起来复杂一点，但便宜一些。

第二步：通道双方将一定量的代币发送到合约地址进行锁定，构建资金池，在闪电网络那，由多重签名地址代持代币，而在雷电网络，则由合约地址代持。同时，双方会向合约发送一条双方签名的报文，这条报文的作用是向全网公示，通道双方各自锁定了多少代币。到此时为止，通道便建立起来了。

第三步：通道建立后，双方所有的交易都可以在链下进行。比如A和B建立了雷电通道，双方最初各自锁定了10个以太币，现在A要转1个以太币给B，那么双方一起签名一条新的信息：A减去1个以太币，B增加1个以太币（闪电通道的是记录余额，雷电通道的是记录净增减，比如A再发1个以太币给B，闪电通道会记录A还有8个，B有12个，但雷电通道会记录A减少2个，B增加2个）。这条信息不需要发到区块链上，只需A和B保留就行。

第四步：A和B之间在链下的每一次双方签名的转账信息，都有一个序列号，比如第一次是1，第二次是2，如果要结束通道（假如A发起 ），A可以将最新的，也就是序列号最大的那个信息发送到智能合约，同时提供一个锁定时间。如果在锁定时间到期前，B提供了一个更新的信息，那说明A作弊（比如，A在倒数第二条信息时收到了B的1个以太币，在最后一条信息发给B两个以太币，但A结束通道时，只提交倒数第二条信息），合约会将锁定的币全部给B，用以惩罚A，如果到期时对方没有异议，合约根据最后这条信息的净增减额计算双方的最终余额并发还给他们。

## 雷电网络的特点及优点

1. 智能合约：

闪电网络是通过2-2的多重签名地址建立channel，雷电的支付通道通过智能合约，此处智能合约的作用是： 1）为通道建立指定参与方可共识的共享规则； 2）以托管形式持有代币，以支持链下支付； 3）使用防止一方滥用规则的仲裁协议；

智能合约也负责在通道关闭时，根据链下双方签名的余额证明，进行链上的余额结算。智能合约对比闪电网络中的多重签名脚本实现更简单，条件盘点也更灵活。

1. 智能转账（smart transfers）

HTLC是闪电网络实现支付路径的关键，雷电网络利用了以太坊支持智能合约的特点，发明了更为通用的智能条件（smart condition），实现智能转账。

智能条件可接受任何格式的报文为参数，执行后对通道上的余额进行调整。而HTLC是smart condition中可实现的条件之一。

智能转账能够根据链上智能合约可读取的条件进行结算，提供比HTLC更为丰富的功能，如支持预测市场、期货等条件，实现更为丰富的应用。

1. 组合锁

闪电网络中，HTLC的解锁取决于到期时间和收款人能否出示符合哈希值的secret两个条件。

HTLC的secret一般由收款人设置，将secret的哈希值提供给付款人。但是这样做存在一定问题，即转账中，如果Alice临时改变路径，不想通过C转账。或者C离线，无法转账，Alice需要更改路径。更改路径后，在上次的HTLC没有到期之前，Bob和C有合谋的可能，在Bob从新路径拿到代币后，Bob向C透露secret，使Alice在旧路径被锁定的代币损失。

而雷电网络开发者考虑为锁设置由三个锁组成的组合锁，解决此问题。包括：

（1）重试哈希锁（retry hashlock）。重试哈希锁的secret由付款人提供，可能会被付款人重新生成。重新生成一般发生付款人想改变路径的情况。

（2）收据哈希锁（receipt hashlock）。收据哈希锁的secret由收款人提供。

（3）时间锁（time lock），用于由付款人控制的锁到期时间。  
在到期前要解锁这笔转账，就需要重试哈希锁和收据哈希锁的两个secret，可称之为secretR和secretE。

改造后，以Alice通过C中介转账给Bob为例，新的转账方式是：

（1）收款人Bob发送收据哈希锁给Alice，自己保留收据哈希锁的secretE。

（2）付款人Alice使用收据哈希锁和自己的重试哈希锁构造给C的转账，Alice自己保留secretR。

（3）C按照同样的锁向Bob构建转账。

（4）Alice确认以上转账构造完毕后，向Bob提供secretR。

（5）Bob拥有secretR和secretE，向C出示，解锁转账，获得代币。C也获知了两个secret，向Alice出示，解锁转账，获得代币。

这样做的好处是，只要Alice不提供secretR，Bob不能在转账中途向转账中介C提供secret，使Alice遭受资金损失。

1. 其他雷电网络在设计细节方面与闪电网络也有一定不同。

其中，用来更新通道余额分配的报文，增加了序号字段和等待期字段用来识别作废的报文。如果A向链上提交更新余额的报文后存在等待期，在等待期中，如果有序号更新的报文提交，A将受到惩罚。惩罚一般是罚没A在通道中锁定的代币。

另外，在余额分配中，申明新余额分配的方式是出示余额分配的净增减，而不是重新申明余额，和闪电网络有形式上的差别。由于这个缘故，雷电网络的通道称之为“The Netting Channel（净通道）”。

优点如下：

 1.**可扩展性**: 它与参与者的数量呈线性关系（每秒可能有1,000,000+次转帐)。

 2.**快速**： 转账在一秒钟的时间内被确认和确定。

 3.**匿名性**：单次转帐不会显示在全局分布式帐本中。

 4.**互操作性**： 以太坊上任何实现了标准代币的接口的代币都可以互相交易。

 5.**低额交易费**： 交易费比区块链上的交易费低7个数量级。

 6.**小额支付**： 低交易费允许有效的转移小额资金。

## 4.3 雷电网络风险及缺点

雷电网络目前的局限性主要如下：

 收款时要求必须在线：如上所述，收款人在收款之前需要签名回收交易，以便付款人知道他们可以在发生恶意通道关闭或拒绝签名的情况下回收资金。因此，要收款就需要一个热钱包，这意味着如果发生安全事故，私钥可能会暴露。

 对于大额付款并不理想：尽管可能存在通过各种支付渠道的路线，但同行多个钱包中的资金可能不足以转移大额资金

 集中化：雷电网络可能会鼓励支付中心的集中化（类似于矿工集中），这也是为什么有人说雷电网络实际上会出现类似银行的超级节点的原因。