去信任的清算协议，以及在现代支付清算结算系统的应用 - 从闪电网络谈起

1 闪电网络的简介和现状

闪电网络(Lightning Network)最初是由 Joseph Poon 和 Thaddeus Dryja 在2015年的白皮书中提出的。这篇白皮书在比特币社区中产生了很大反响，在众多关于比特币的论文和白皮书中，被认为是第二重要的，其价值仅次于中本聪的创世论文。

由于闪电网络依赖于隔离验证，在2017年比特币隔离验证升级之前，一直停留在概念和内部开发阶段。2018年3月，Lightning Labs 开发并推出了第一个测试版，之后 ACINQ 和 Blockstream 两家公司也相继推出了不同的实现。从此以后闪电网络的发展就步入正轨。根据 统计网站 的数据，闪电网络目前有 6,761 个节点，30,622 个支付通道，支付通道总计有 729.89 BTC(约281万美金)。说明闪电网络在过去的一年中取得了显著增长。

闪电网络的愿景是解决比特币网络的扩容问题。众所周知，比特币的初衷是实现一个端到端的电子现金系统，为全世界提供一个去信任的、7x24小时服务的电子支付网络。但是比特币的性能却远远达不到要求。按照平均每个交易300字节计算，比特币的平均吞吐量是 5.6 TPS。然而 Visa 的峰值吞吐量可以达到 47000 TPS。如果对标这个吞吐量，比特币的区块大小要扩张到 8GB 左右，每年要新增 400 TB的区块数据。这显然是不现实的。

除了闪电网络，比特币社区同时也提出了众多的扩容解决方案，比如大区块、DPoS、DAG、分片等。这些方案试图修改比特币协议本身，例如调整配置参数、优化数据结构、修改共识算法、账本分区处理、优化网络资源管理等等。但是效果都不好，在付出了高昂的代价(增加存储量、增加网络流量、增加逻辑复杂度、弱化去中心化)之后，却只是获得了非常有限的性能提升，和 Visa 相比依然还有几个数量级的差距。

唯有闪电网络脑洞大开、另辟蹊径，使用了新的价值转移范式。在比特币的智能合约基础上，结合链下去信任的清算协议，构建了二层支付系统。彻底摆脱了“去中心化-安全-性能”的三元悖论约束，将系统的吞吐量上限提升到了几十万 TPS 级别，同时可以做到类似于支付宝、微信支付的实时支付体验。而且难能可贵的是，它对比特币网络本身几乎没有带来任何负面影响(隔离验证对于比特币的负面影响很小)。

闪电网络并没有使用类似于零知识证明那样的高难度技术，但是它的巧妙设计依然令白皮书晦涩难读。市面上也缺乏简单易读而且讲解透彻的科普文章，对于广大金融科技与区块链爱好者和投资者来讲，有很高的学习门槛。所以闪电网络技术的价值长期被误解、被低估。本文重新梳理了闪电网络的思想，用通俗易懂的文字，为大家介绍闪电网络的技术原理，总结技术优势和劣势，分析它的适用的场景，最终阐述它在现代电子支付系统中的潜在应用价值。希望能帮助广大读者更深入的认知闪电网络。

此文的章节结构如下：

- 第二节从金融的角度，介绍两种不同的价值转移范式，找出闪电网络的创新思路；
- 第三节详细介绍闪电网络的技术原理：如何使用链上智能合约、配合链下密码学协议实现去信任的债务清算；
- 第四节分析闪电网络技术的优点，解释为什么交易延时可以达到实时支付？为什么吞吐量的理论上限可以达到几十万TPS？应用闪电网络有哪些限定条件？
- 第五节介绍闪电网络还有哪些缺点？在哪些方面可以改进、可以拓展？
- 第六节结合现代电子支付、清算、结算系统，讨论闪电网络技术的应用前景；
- 第七节总结闪电网络技术对于区块链和金融科技的价值和意义。

2. 价值转移的范式：所有权交割与债权清算

闪电网络的实现包含智能合约与密码学协议，技术细节比较复杂。为了便于读者理解，我们先从货币银行学的角度重新审视比特币的转账机制是什么？扩容的难点在哪里？闪电网络的解决问题思路是什么？从宏观上理解闪电网络的原理，非常有助于理解微观的技术细节。

价值转移有两种不同的范式，一种是所有权的交割，另一种是债权的变更。二者既互相关联，又截然不同的转移方式。

2.1 所有权的转移

先谈一下所有权的交割。对资产宣示所有权意味着某一样资产是“我的”，而不是“你们的”。这是个体与集体之间的契约，而不是个体与个体之间的契约。集体中的任何一个人都有可能对所有权发起质疑或者挑战，只有当集体中的所有参与者都认可之后，所有权才能成立。

一对多是一种不对称关系，建立所有权的契约往往不是一件轻松的事。比如在动物世界里，一头狮子对它刚刚捕获的猎物宣示所有权，必须借助于尖牙和利爪，暴力驱赶所有其它的觊觎者。在人类社会也一样，银行间转移大量现金，必须要武装押运，借助于全副武装的军警“说服”其它人认可所有权的归属。一个不太令人愉快的事实就是，暴力是最基本的、最普遍的所有权共识机制。

除了暴力之外，一对多的通信的成本也是很重要的因素。所有权交割的过程中，交易信息必须要广播给其它所有相关者，并且得到集体的见证。很多读者都了解雅浦岛石头币的故事。这些很笨重的石头币就放在公共场合，或者在路边，或是一家房子面前，并不需要人守候。当交易石头币的时候，并不是把这些币搬运到买家的住所。雅浦岛上的居民之间有很好的信用关系，这套货币体系是通过信用建立起来的。但是每一次更新所有权的时候，必须要通知岛上的其它居民，石头币的新主人是谁。否则石头币可能会被二次卖给不知情的人。

这种支付体系的效率是很低的。好在雅浦岛上的居民人数不多，可以交换的商品也不多，交易频次也很低，一个居民一辈子也交易不了几次。石头币可以满足岛民的需求，但是它不适合在大范围内普及。

比特币协议本质上是一种资产所有权的转移系统。每一个节点代表集体中的一个参与者，共同维护一个资产所有权的账本。它的创造性和先进性在于：

第一，用密码学代替了武力保护，大大降低了所有权维护的成本。现代密码学充分利用了数学中的不对称性，防守者只需要付出极小的代价，而攻击者要消耗天文数字的算力才能攻破防守者的防御。

第二，利用互联网的通信和存储能力，极大的扩展了地域限制。世界上的任何一个人，只要能链接互联网，就能使用比特币的支付网络。

第三，利用POW共识算法模拟了参与者对交易的见证过程，令交易过程完全自动化，不需要人工操作。

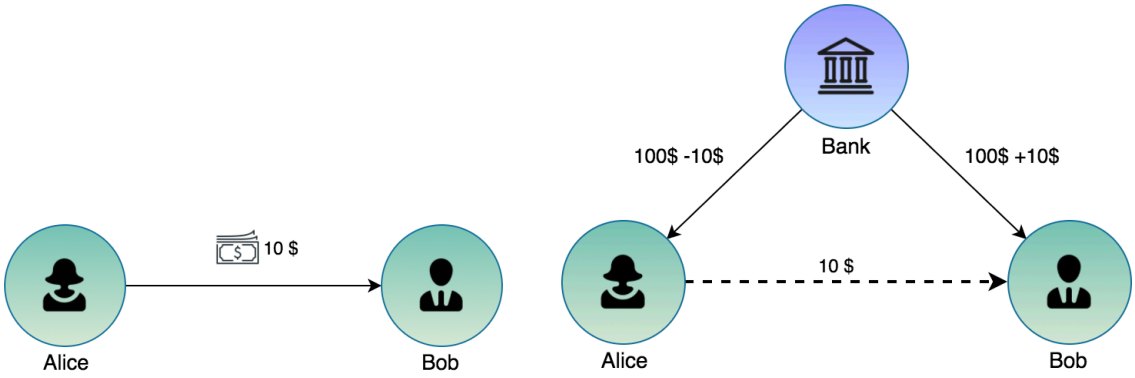
相对于雅浦岛的石头币，比特币技术大大优化了支付体系的安全性、稳定性、便利性和使用范围。但是并没有改变一对多的所有权关系，而且由于扩大了使用范围，反而大大增加了所有权关系的规模。每一个交易都要广播给更多的参与者，达成共识的的成本增加了。比特币性能瓶颈的本质就在于此。

2.2 债权的清算

从人类历史的早期，一直到近代货币银行体系建立之前，是以实物货币的所有权交割作为主要的支付手段。曾经使用过的实物货币包括黄金、白银、玉器、纸钞等。近代出现了银行、钱庄这样的组织，他们作为金融中介，可以通过债务清算的方式为用户提供支付服务。其工作原理非常简单，我们用下面两幅图对比两种支付方式的差异。

我们假设 Alice 要向 Bob 支付 10 美元，左图表示现金的支付方式，Alice 通过把 10 美元的钞票转交给 Bob 完成支付。

右图表示债务清算的方式，增加了第三方银行作为公共债务人，Alice 和 Bob 分别在银行里存入 100 美元。或者说银行分别欠 Alice 和 Bob 100 美元。当 Alice 需要向 Bob 支付 10 美元的时候，银行对 Alice 的债务余额减10美元，对 Bob 的债务余额加10美元。支付过程中 不需要 Alice 和 Bob 之间交割任何实物货币，只需要银行居间调整债务余额就可以了。在金融领域里，债务余额的调整的过程属于清算，债务终结的过程属于结算。这种价值转移方式称之为：银行通过清算间接帮助 Alice 和 Bob 完成了结算。

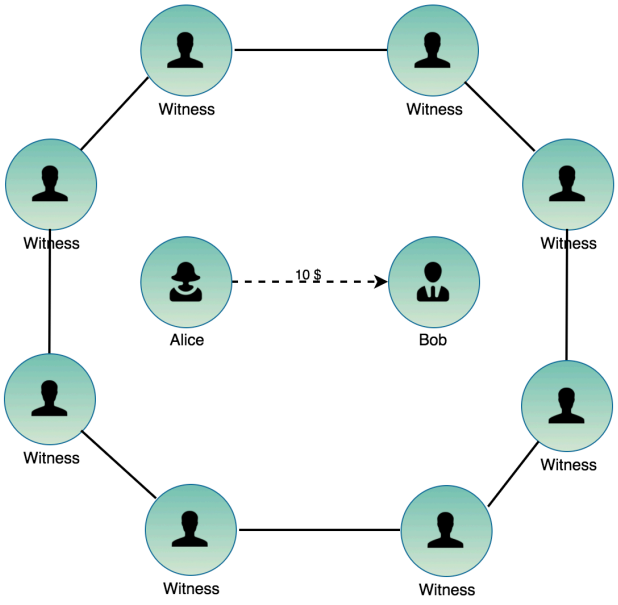


相对于事物交割的直接结算，债务清算的间接结算的优势非常明显。

首先，促进货币去实物化。虽然去实物化的过程从2千年前就开始了。货币从粮食、牲畜，发展到贵金属，再到纸钞，货币越来越轻便，内在价值越来越小。但是依然无法彻底脱离实物载体。到了债务货币阶段才彻底的变成数字。节省了保存、转移等不必要的成本。

其次，数字化的债务货币非常适合计算机处理和网络传输。随着IT技术的发展，债务清算可以全自动化处理，并且可以支持随时随地的远程支付。

最后，也是最有启发意义的是，债务关系是1-1的关系，债务余额的更新只需要银行一方执行，三方达成一致。相对于比特币协议，不需要 Gossip 通信让众多与交易无关的见证者参与共识。这一方面节约了通信和存储成本，保护了交易的隐私性，另一方面多笔互不相关的支付可以的并行执行，突破了吞吐量的瓶颈。

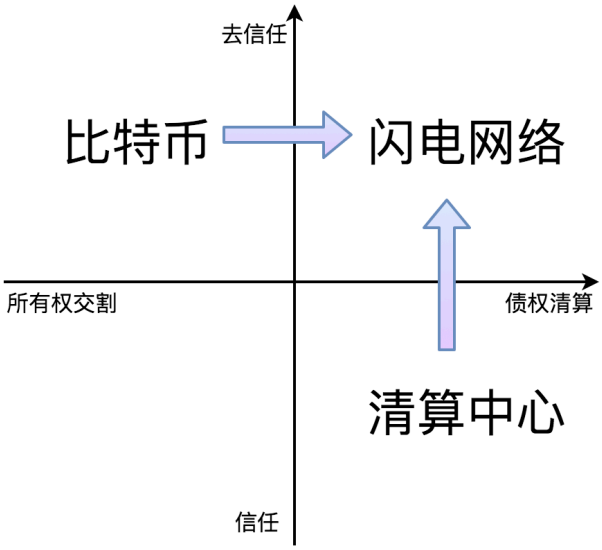


通过可信的清算中心大大提高支付的性能，这种支付方式的创新几十年前就已经开始应用，并且获得了很大

的成功。它帮助 Visa 每天能够处理几十亿笔交易。但是这种创新也要付出代价。银行和客户之间的债务关系必须长期续存，意味着客户的资产长期由银行托管。银行必须有强大的信用背书和风控制度，防范道德风险、管理风险、信用风险，保证银行有充足的偿付能力，以免银行被挤兑。为此现代金融系统建立了一整套法律和监管体系，防范这些风险的累积和爆发。然而随着银行体系越来越庞大、越来越复杂，金融监管的成本也越来越高，这些成本最终转化交易的摩擦有消费者买单。

3 去信任的清算协议

如果从支付方式和去信任这两个维度对比清算中心和比特币协议，我们看到二者优缺点恰好是互补的。比特币协议是去信任的，不需要考虑金融中介的信用风险问题，但是它的性能很难扩展；反之清算中心可以支持高并发、大吞吐量的性能，但是依赖于金融中介的信任，要承担监管和合规的成本，以及市场垄断对创新的伤害。



能否存在一种解决方案综合清算中心和比特币协议的二者优点呢？既能支持海量的小额支付场景，同时又不依赖于中介的信用？闪电网络给我们提供了可行的去信任清算解决方案。

本节先阐述去信任清算的大致思路，然后在下一节再介绍技术实现的细节。首先介绍一些基本概念：

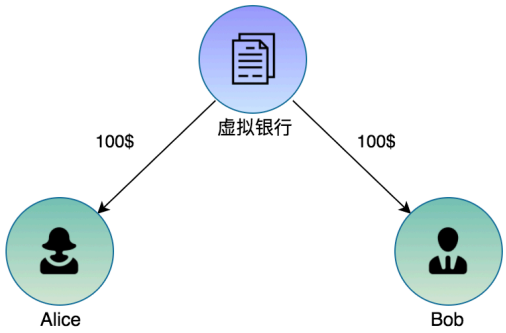
3.1 基本概念

1. **虚拟银行 Virtual Bank**

   虚拟银行是运行于区块链上的一个智能合约，按照支付双方共同协商的方式创建，它模拟一个银行机构作为支付双方的公共债务人。虚拟银行部署之后，按照预先协商的额度，双方向虚拟银行的智能合约注入资金，完成虚拟银行的筹建。将来如果虚拟银行被清盘结算，资金都返还给了支付双方，那么虚拟银行的服务自行终结。

   和传统的银行相比，有三个特点：
   微型： 一个虚拟银行只有两个账户，只管理特定交易双方的资产。所以资产负债表也只有2条数据。

无需信任：虚拟银行是通过智能合约实现的，继承了智能合约的公开、透明、不可篡改、不可伪造、不可撤销等特点。所以作为公共债务人，虚拟银行没有传统金融机构的风险：比如对手风险、道德风险、流动性风险等。虚拟银行筹建完成之后，它的债务偿付能力永远是100%，自然也不没有金融监管的成本。提供了无需信任的资产托管服务。
双重签名：和一般银行的运营模式不同，虚拟银行不接受客户单方面取款请求。银行只有两个账户，而且总资产不变。一方的资产增加，意味着另一方的资产减少，这是一个零和博弈。为了防止单方面作弊行为，侵害对方的权益，双方需要共同协商，对每一次新的资产分配方案达成一致性的意见，并且都要对分配方案签名。虚拟银行智能合约在处理资产调整请求的时候，要验证双方的签名，保证每一次资产调整都是双方共同的真实意愿。



- **共同承诺 (Mutual Commitment)**

每一次微支付，虚拟银行中的资产负债表要做一次调整。双方在链下对债务调整方案达成一致，形成向虚拟银行智能合约的请求消息，并且双方签名。此消息并不立刻广播到链上，而是由双方存储在本地，称之为**共同承诺**。共同承诺是双方真实意愿的表达，是彼此之间对资产分配方案的承诺。共同承诺一旦达成，在任何时候、任何一方都可以将承诺方案广播到链上，虚拟银行都会兑现承诺，按照预定方案结算银行的资产，分配到双方的账户中。共同承诺的作用类似于银行支票，虽然没有兑现，但是持有共同承诺就可以无风险的、随时的从虚拟银行中兑现响应的资产。

共同承诺的实现必须满足以下几个要求：
不可伪造：共同承诺表达虚拟银行双方当事人的真实意愿，任何第三方无法伪造当事人的身份，生成虚假共同承诺。
不可篡改：对于已经达成的承诺，其中的所有条款无法篡改。虚拟银行智能合约会检查双方的签名，确保承诺的完整性。
可以撤销：共同承诺只要没有提交到虚拟银行兑现，双方可以达成新的承诺，并且撤销旧的承诺。对于交易双方来讲，只有最后的一份共同承诺是有效的，之前达成的历史承诺都应该撤销。可撤销性是通过撤销锁机制实现的。

闪电网络协议中有两种承诺方案：RSMC 承诺与 HTLC 承诺。他们的区别我们后面会讲，但是都满足上述3个条件。

- **承诺编号 Sequence**

在共同承诺兑现之前，双方可以达成多次共同承诺，撤销旧的承诺，建立新的承诺。这些承诺按照时间顺序编号，以 Sequence 表示。

- **进攻方(Attacker) & 防御方(Defender)**
  如果一方将共同承诺广播到链上，主动向虚拟银行发起申请, 重新结算资产，此方称之为承诺方案的进攻方。被动的接受他人资产分配方案的一方，称之为防御方。

  虚拟银行的资产清算，相当于瓜分银行的总资产，是一种零和博弈。假设双方都是理性的决策者，任何一方都不会做出于对方有利，于己方不利的决策。双方需要一种公平的机制管理共同承诺，规避对手风险，防止对方作弊。

  在闪电网络的协议里，一个新的承诺方案先由防御方初审。如果防御方接受此承诺方案，就对此签名，然后发送给进攻方进行二审。进攻方持有多个防御方已初审的承诺方案，有权放弃对己方不利的方案。同时有权选择广播共同承诺的时间，当他觉得合适的时候，再加上自己的签名，广播到链上，向虚拟银行请求结算资产。虚拟银行智能合约检验双方的签名，根据共同承诺的条款，公开、透明的结算双方的资产。

- **对偶承诺方案 Dual Commitment**

  对于同一个承诺编号，共同承诺都是一式两份，双方各持有一份。两份承诺内容一致，但是攻守位置互换。比如说 Alice 持有的那一份，Alice 是进攻方，Bob 是防御方(Bob已经初审签名)；反之，Bob 持有的那一份中，Bob 是进攻方，Alice 是防御方(Alice 已经初审签名)。这两份承诺方案是一对，互为对偶承诺方案。

  这样设计的好处有两个：一是保持共同承诺的活性，避免单点故障造成的死锁。因为防御方只能被动等待进攻方向虚拟银行发起请求。假如进攻方故障，不能行使进攻方的职责，防御方可以翻转角色，使用对偶承诺方案，以进攻方的身份完成资产的结算。二是保持灵活性和对称性，任何一方都可以随时主动兑现共同承诺，降低对手风险。

- **撤销锁 Revocation Lock**

  为了标识一个共同承诺方案已经被撤销，闪电网络协议设计了撤销锁机制。在每一份承诺方案中，进攻方必须要放置一个撤销锁。不同的承诺编号、不同的承诺方案镜像有不同的撤销锁。如果一共有 N 对承诺方案，那么需要有 2N 个不同的撤销锁。

  撤销锁由承诺方案的进攻方管理，它实际是一个随机账户地址，对应的私钥由进攻方保管。如果进攻方要撤销某一个承诺方案，他必须公开对应的撤销锁私钥。反过来说，如果防御方从进攻方拿到了一个撤销锁的私钥，那么他可以相信进攻方确实放弃了对应的承诺方案。

  一般来讲，从虚拟银行开始，一共有 N 对共同承诺方案，前面的 (N - 1) 对承诺方案已经被撤销，他们的撤销锁私钥是公开的，每一方都会保留一份对方的**撤销锁私钥列表**。只有最后一对承诺方案是有效的，其撤销私钥还没有公开。

  撤销锁的安全机制：当一个承诺方案提交给虚拟银行的时候，防御方可以查看此撤销锁的编号(Sequence)。如果防御方发现此承诺方案是已经被撤销的，那么从**撤销锁私钥列表**中，找到对应的私钥，并且生成签名作为凭证，向虚拟银行证明对应的承诺方案是已经被撤销的。虚拟银行将判定进攻方违约，对进攻方处以罚金。这种情况称之为“**破解撤销锁**”。所以如果进攻方是理性的，他就不会冒着撤销锁被破解的风险提交已经撤销的承诺方案。反之，提交未撤销的承诺方案是安全的，因为防御方不知

道撤销锁私钥，也就无法破解撤销锁。

当双方创建一对新的承诺方案的时候，需要交换旧承诺方案的失效私钥，表示双方都只承认新的方案，放弃旧的方案。在承诺方案兑现之前，双方都要妥善保存对方的**撤销锁私钥列表**，防止对方使用对己方不利的承诺提案结算虚拟银行中的资产。

- **诚信保证金 Fidelity Bond**

  为了保证承诺方案的公平性，承诺方案会设立**诚信保证金**条款，和撤销锁机制配合使用。当虚拟银行兑现一份承诺方案的时候，防御方被动接受进攻方的资产分配方案，所以防御方的资产份额可以优先结算。而进攻方的所有资产作为**诚信保证金**冻结一段时间。目的是防止进攻方提交一份已经被撤销的承诺方案，侵犯防御方的利益。

  在**诚信保证金**的冻结期间内，虚拟银行会等待防御方破解该方案的撤销锁。如果破解成功，防御方可以取走所有的诚信保证金作为惩罚。进攻方就会损失所有资产。反之，冻结期满之后，进攻方可以取走诚信保证金。

- **三权分立**

  在承诺方案的决策过程中，首先是防御方对承诺方案签名，拥有**初审权**，可以拒绝对防御方不利的条款；然后进攻方对承诺方案签名，拥有**复审权**，可以放弃对于进攻方不利的方案。二者的权利是对等的。

  在承诺方案的执行过程中，进攻方拥有**提交权**，有权选择什么时候提交、提交哪一个承诺方案；防御方拥有**监督权**，有权检查承诺方案的有效性，挑战**撤销锁**，惩罚进攻方的违约行为；虚拟银行智能合约拥有**执行权**，公开、公平的按照承诺方案的条款处置双方的资产。三者的权利是不同的，既互相独立、又互相制约。

  无论是决策阶段、还是执行阶段，基于虚拟银行智能合约的共同承诺协议，构建了一种均衡的二元博弈体系。此博弈的纳什均衡点，要求双方必须诚实的遵守共同承诺。对于对手风险的防范机制是内生的，可靠的，不需要外部的第三方监管与信用保障。

3.2 RSMC承诺方案与支付通道

支付双方以虚拟银行托管双方的资产，通过共同承诺重新清算双方的存款余额，已达到价值转移的效果，这种支付工具称之为支付通道。虚拟银行筹建完成标志着支付通道的开启；虚拟银行根据任何一方提交的承诺方案结算双方的资产，就标志着支付通道关闭。

在闪电网络中定义了两种承诺方案。第一种称为 RSMC(Recoverable Sequence Maturity Contract)承诺方案，定义了最基本的承诺方案，其中包括承诺编号、撤销锁、诚信保证金、对偶承诺方案等。RSMC 承诺方案是无时间期限的：只要没有被撤销，承诺永久有效。我们在下一章再详细介绍 RSMC 承诺方案的技术实现。

3.3 HTLC承诺方案与支付通道网络

- **HTLC 协议**

  RSMC 协议的局限性在于虚拟银行自有连个账户，只能服务于两个人之间的往来支付。支付双方必须建立直连的支付通道，如果在N个人之间建立支付通道，那么每个人需要管理(N-1)个支付通道，总计一共有(N - 1)*N/2 个支付通道。闪电网络进一步提出了 HTLC (Hashed Timelock Contract)协议，可以将支付通道的负责度从$O(N^{2})$ 降低到 O(N)。

  和 RSMC 一样，HTLC 中的承诺方案同样也具有不可伪造性、不可篡改性、可撤销性。这两种承诺方案的差异在于：HTLC 承诺方案除了撤销锁之外，还额外添加了时间锁和 Hash 锁条款。所以HTLC 的承诺方案是带有限制性条件的临时承诺方案。具体的来说：

  - **时间锁**：承诺只在规定时间内有效。
  - **Hash锁**：支付的发送方向接收方出示一个 Hash 值 H，接收方必须公开对应的暗语R，使得 Hash(R) = H。

  接收方当且仅当满足这两个条件，就能获得约定的款项。否则将回退到上一个承诺方案的状态。

  举个例子，假设根据当前的共同承诺，Alice 和 Bob 当前的余额是 <60美元, 40美元>，Alice 向 Bob 支付10美元。他们约定使用 HTLC 承诺方案，Bob 先生成一个随机数 R，计算其Hash值：H = hash®，然后将 H 分享给 Alice，而且约定时间锁 T。那么HTLC 承诺方案的条款应该是这样的：

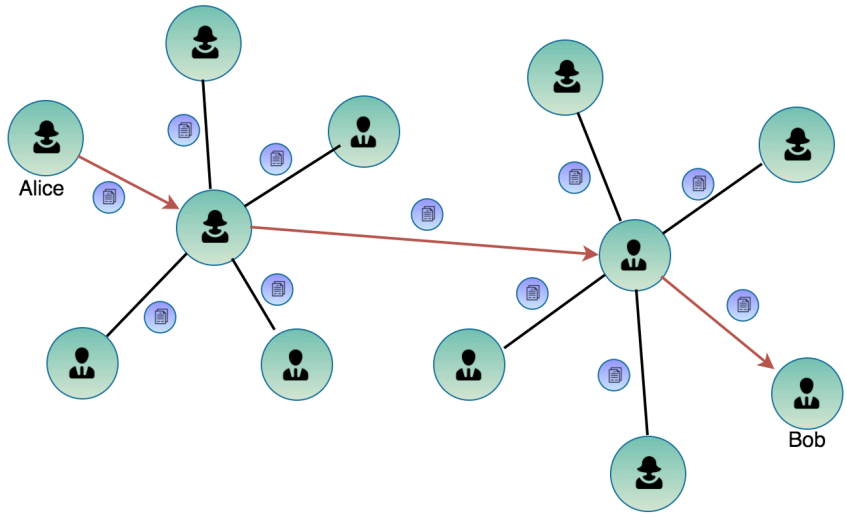
  1. 如果在时间 T 之前，任何一方提交承诺方案，检查暗语 R 是否符合 Hash 锁

     1.1. 如果符合，那么按照<50美元，50美元>的方案结算 Alice 和 Bob 的资产
     1.2. 如果不符合，那么此次承诺方案提交无效。

  2. 如果在时间 T 之后，任何一方提交承诺方案，那么按照<60美元, 40美元>的方案结算 Alice 和 Bob 的资产。

- **支付路径**

  HTLC 承诺方案的价值在于可以将首尾相连的多个支付通道串联起来，建立支付路径。如图所示，Alice 和 Carol 没有直接关联的支付通道，但是他们分别与 Bob 建立了支付通道。那么Alice - Bob - Carol 形成了一条支付路径，他们可以通过支付路径也可以完成支付。假设Alice 向 Carol 支付10美元，基本过程如下：



1. Carol 随机产生暗语 R，计算Hash 值： H = hash®，并将 H 发送给 Alice。
2. Alice 和 Bob 达成 HTLC 承诺：如果Bob能在时间 **2T** 内出示暗语 R，使得 hash® = H, 那么Alice 向 Bob 支付 10 美元。
3. Bob 再和 Carol 达成 HTLC 承诺：如果 Carol 能在更短的时间 **T** 内出示暗语 R，使得 hash® = H, 那么Bob 向 Carol 支付 10 美元。
4. 由于 Carol 知道暗语 R 值，在规定的期限T内，可以把R出示给 Bob，获得 Bob 的10美元。之后 Bob 和 Carol 可以再签署一份 RSMC 承诺方案，代替临时的 HTLC 承诺方案。
5. Bob 从 Carol 那里拿到暗语 R 的时候，和 Alice 之间的 HTLC 承诺还没有过期，向 Alice 出示 R 之后，可以获得 10 美元。然后也可以再重新签署一份无期限的 RSMC 承诺，替换临时的 HTLC 承诺方案。

最终的结果就是：Alice 通过 Bob 向 Carol 支付了 10美元，Bob 作为中间人资产并没有变化。HTLC 承诺方案可以在两个联通的支付通道上传递交易，整个过程是可信的。我们从支付通道内和支付通道间两个方面考察对手风险问题。

- 支付通道内的对手风险

  首先，支付的接收方无法伪造暗语 R 而欺诈对方，在规定的时间内破解 Hash 锁是不可能的。

  其次，双方在达成 HTLC 共同承诺之后，可以安全的撤销原来的承诺方案。对于接收方来讲，及时公开暗语 R 值，就可以获得 10 美元，所以放弃原承诺方案没有风险。对于发送方来讲，如果没有及时公开暗语 R，那么就回退到原来的状态；如果及时公开暗语 R，他就可以从前面的一个支付通道获得 10 美元，总之也可以无风险的放弃原有的承诺方案。

  再次，双方在达成 HTLC 共同承诺之后，发送方不需要担心接收方在获得 10 美元的情况下，不及时把暗语 R 发送给自己。否则接收方必须在规定时间内向虚拟银行提交HTLC 承诺方案，其中包含暗语 R。从智能合约的公开交易数据中，发送方依然可以获得暗语 R。

  最后，公开暗语 R 之后，双方可以在时间锁 T 到期之前，按照新的资产负债表建立无时间期限的 RSMC 承诺方案，撤销临时的 HTLC 承诺方案。接收方无疑不会反对替换新承诺方案；对于发送方来讲，拒绝替换方案也没有意义，因为如果发送方想拖延时间，等时间锁过期之后，HTLC 承诺

回退，那么接受方为了保护自己的利益，可以及时提交 HTLC 承诺方案，兑现属于自己的10美元。发送方不但无法取回10美元，而且需要重新建立支付通道。

- 支付通道间的对手风险

  注意到 HTLC 承诺方案是沿着支付路径，从发送方向接收方建立；然后暗语是反方向，从接收方向发送方传递。对于任何一个中间节点，必须向接收端一侧支付 10 美元才能知道暗语 R，否则无法从发送端一侧获得 10 美元的补偿。另一方面，由于发送端的时间锁比接受端的长，当他拿到暗语 R 之后，也有充足的时间，可以无风险的获得 10 美元。

对于更长的支付路径，HTLC 承诺方案依然有效。需要注意的是，资金从接收方向发送方传播，每经过一段支付通道，对应的时间锁要增加一个 T，保证资金的发送方有足够的时间向接收到下一个发送方的资金。进一步推广，所有的支付通道链接在一起构成了**支付通道网络**，其中某一些特定的节点作为中心枢纽，链接其它普通节点。一个节点要向另一个节点支付，只要找到一条联通二者的路径，而且路径上的每一段都都有充足的额度，就可以通过这条路径完成价值转移到过程。

4. 技术详解：去信任的清算协议

本节具体阐述闪电网络清算协议的技术细节。对于技术细节不感兴趣的读者可以略过此部分，只需阅读 2.3 节。

闪电网络的技术细节晦涩难懂，而且工程实现的复杂度也比较大。部分原因是由于闪电网络的清算协议基于比特币协议，其智能合约是通过堆栈式指令编写，类似于汇编语言的风格。而以太坊的智能合约编程语言 Solidity 的语法接近于 JavaScript，是一种高级编程语言的风格，相对来讲有更好的可读性。所以本文基于 Solidty 重新表达闪电网络协议。在保持技术原理一致性的前提下，尽量提高可读性，帮助读者降低学习的门槛。

根据 2.3 节的讨论，清算协议的技术实现分为三部分: 虚拟银行智能合约，RSMC 可撤销清算协议，HTLC 支付通道串联协议。

4.1 虚拟银行智能合约

不失一般性，假设 Alice 和 Bob 两个用户在某一段时间内需要频繁的往来支付。于是双方协商建立共同的虚拟银行智能合约。这个合约模拟了一个微型银行，只有 Alice 和 Bob 两个账户。双方约定分别在虚拟银行中存入 100 美元，用 <100, 100> 表示 Alice 和 Bob 在资产负债表的初始余额。

虚拟银行智能合约的源码位于：[GitHub: Solidity Bidirection Payment Channel](#)。

4.1.1 合约数据结构

首先介绍虚拟银行的数据结构，一共分为三部分(如下图所示):

1. 资产负债表 Client[] _clients。

由于虚拟银行只有两个账户，Alice 和 Bob。所以此数组的长度永远是 2。每一项保存的用户的地址、应存入的余额，以及是否已经足额存款。

2. 虚拟银行状态 State _state。

   虚拟银行一共有4个状态。

   1. 智能合约初始化完成之后进入 Funding 状态，Alice 和 Bob 根据约定的额度，往虚拟银行里转账。
   2. Alice 和 Bob 都足额注入100美金之后，虚拟银行开始运营，进入 Running 状态。Alice 和 Bob 在链下可以进行微支付交易。
   3. 在 Running 状态下，Alice 和 Bob 都可以提交资产清盘请求。虚拟银行检验清盘请求之后，将清盘发起人的资产冻结，对方的资产立刻返还。然后进入 Liquidating 状态。
   4. 在 Liquidating 状态下，清盘发起人的请求被对方审核。根据审核结果，冻结资产最终都被转给 Alice 或者 Bob。虚拟银行的所有资产被清偿，进入 Closed 状态。

3. 资产清盘信息 Liquidation _liquidation。

   此数据结果包含：清盘的时间，清盘发起人的索引、其资产的锁定期限，豁免地址，以及资产负债表的序号。

```
contract VirtualBank {
    struct Client {
        address addr;
        uint balance; // 余额
        bool deposited; // 是否足额存款
    }
    struct Liquidation {
        int32 lockPeriod;   // 锁定期限
        int32 liquidateTime;  // 清盘时间
        address waiverAddr;  // 豁免地址
        int   master;        // 清盘发起人索引
        int   nounce;
    }
    enum State { Funding, Running, Liquidating, Closed }

    // 用户列表
    Client[] _clients;

    // 状态: init, running, redemption, closed
    State _state;
```

4.1.2 合约代码

下面根据虚拟银行的状态演化的顺序，逐一介绍虚拟银行智能合约的代码。

构造函数

创建虚拟银行之前，Alice 和 Bob 需要提前协商相关的配置参数，这些参数包括：

1. Alice 和 Bob 的个人账户地址: address[] addrs。
2. Alice 和 Bob 欲存入的账户余额。
3. 冻结资产的锁定期限，根据具体情况，可以是几分钟到几天不等。

构造函数检查输入参数的合法性，初始化内部数据结构之后，进入 Funding 状态。

```
    // 构造函数，初始化参数Client[],  state
    constructor(address[] addrs, int[] balances, int32 lockPeriod){
        require(addrs.length == 2);
        require(addrs[0] != address(0) && addrs[1] != address(0));
        require(balances.length == 2);
        require(balances[0] > 0 && balances[1] > 0);
        require(lock > 0);

        _clients = new Client[2];
        _clients[0].addr = addrs[0];
        _clients[0].balance = balances[0];
        _clients[0].deposited = false;

        _clients[1].addr = addrs[1];
        _clients[1].balance = balances[1];
        _clients[1].deposited = false;

        _liquidation = Liquidation(lock, 0, address(0), 0, 0);
        _state = State.Funding;
    }
```

资金存款

虚拟银行构建完成之后，Alice 和 Bob 调用 deposit() 函数向银行转入预订额度的资金。双方的资金都转入之后，虚拟银行进入 Running 状态。双向支付通道搭建完成，Alice 和 Bob 现在可以使用 RSMC 协议进行多笔微支付。在下一节具体阐述 RSMC 协议的机制。

```
    function deposit() payable {
        require(state == State.Funding);

        if(msg.sender == _clients[0].addr
         && msg.value == _clients[0].balance
         && !_clients[0].deposited) {
            _clients[0].deposited = true;

        } else if (msg.sender == _clients[1].addr
                 && msg.value == _clients[1].balance
                 && !_clients[1].deposited) {
            _clients[1].deposited = true;

        } else {
            throw;
        }

        if (_clients[0].deposited && _clients[1].deposited) {
            state = running;
        }
    }
```

清盘

Alice 和 Bob 任何一方都可以作为发起人关闭支付通道，调用 liquidate() 函数向虚拟银行合约发起清盘请求。假设 Alice 发起请求，清盘需要递交以下参数：

1. nounce: 微支付的序号
2. balances：最终的资产负债表
3. peerSignature：对方，也就是 Bob 对于清盘请求的签名。证明 Bob 认可新的资产负债表。
4. waiverAddr：锁定资产豁免地址。

liquidate() 首先检查所有输入参数是否有效。然后根据**先发起、后结算**的原则，先返还 Bob 的资产，把 Alice 的资产暂时冻结，从当前开始计算冻结时间，虚拟银行进入 Liquidating 状态。

设立资产冻结的目的是防止清算发起人的作弊行为。虚拟银行的资产清盘涉及到双方的资产结算，相当于瓜分银行的总资产，这是一种零和博弈。假设 Alice 和 Bob 都是理性的决策者，无论谁作为清盘发起人分割资产，都不会做出于对方有利，于己方不利的决策。所以在清盘的过程中，冻结发起人的资产作为一种保障机制，被动清盘的一方的资产可以立刻结算。

```
    function liquidate(int nounce, int32[] balances, address waiverAddr, bytes peerSi
        require(state == State.Running)
        require(balances.length == 2);
        require(waiverAddr != address(0));

        // identify master liquidator
        int master, peer;
        if (msg.sender == clients[0].addr) {
            master = 0;
            peer = 1;
        } else (msg.sender == clients[1].addr) {
            master = 1;
            peer = 0;
        } else {
            throw;
        }

        // check peer's signature
        bytes32 hash = keccak256(abi.encodePacked(address(this), nounce, balances[0],
        require(checkSignature(hash, peerSignature, _clients[peer].addr));
```

资产解冻

在 Alice 资产的冻结期之内，Bob 有充足的时间审核 Alice 提交的清盘申请是否符合最后的资产负债表。正常情况下，Alice 是诚实的，按照双方最终达成的协议分割资产，Bob 不会对清盘的结果有异议。在冻结期满之后，Alice 可以调用 withdrawByMaster() 函数赎回被冻结的资产。

```
    function withdrawByMaster() {
        require(state == State.Liquidating);
        int master = _liquidation.master;
        require(msg.sender == _clients[master].addr);
        require(NOW >= _liquidation.liquidateTime + _liquidation.lockPeriod);

        // update state;
        state = State.Closed;

        // send fund to master
        int value = _clients[master].balance;
        msg.sender.send(value);
    }
```

但是如果 Alice 不是诚实的，资产清算的方式不是双方最终达成的结果，而是之前某一次支付后的结果。在冻结期间内，Bob 可以调用 withdrawByPeer() 抢先取出被冻结的资产。

抢先赎回需要使用豁免私钥签名，此豁免私钥是由 Alice 创建的，不同的支付对应不同的豁免私钥。在每一次支付的时候，Alice 要向 Bob 发送上一次支付对应的豁免私钥。Bob 获得此私钥之后，可以相信 Alice 已经放弃了上次支付对应的资产分配方式。具体的操作过程请参考 RSMC 协议。

```
    function withdrawByPeer(int nounce, bytes waiverSignature) {
        require(state == State.Liquidating);
        int peer = 1 - _liquidation.master;
        require(msg.sender == _clients[peer].addr);

        // check signature of redepmtionPubKey
        bytes32 hash = keccak256(abi.encodePacked(address(this), nounce));
        require(checkSignature(hash, waiverSignature, _liquidation.waiverAddr));

        // update state;
        state = State.Closed;

        // send fund to peer
        int value = _clients[master].balance;
        _clients[peer].addr.send(value);
    }
```

无论是 Alice，还是 Bob 取出冻结的资产，虚拟银行都进入关闭状态，对应的支付通道随之也关闭。

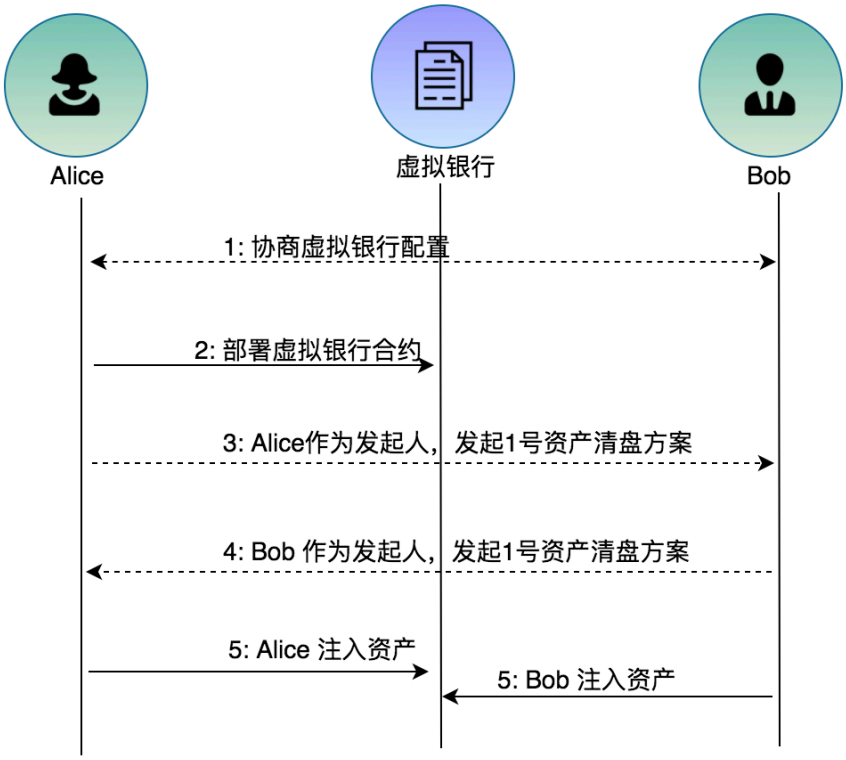
4.2 RSMC 可撤销清算协议

交易双方把资产存于虚拟银行智能合约，建立了支付通道，可以在链下共同协商资产的清算方案，通过资产清算的方式完成支付的结算过程。RSMC 协议规范了链下资产清算的过程。每一次支付会达成一次资产清算方案，经过一段时间的积累，Alice 和 Bob 会有多份资产清算方案，但是只有最后的方案才是有效的结果。假设双方都是理性的，会尽力保存对于己方有利的方案。必须要设计一个机制，让双方彼此证明已经放弃旧的清算方案，只保留最后的版本。这是 RSMC 协议的核心思想。

下面按照支付通道的建立，使用，关闭三个环节介绍 RSMC 协议的具体过程。

建立支付通道

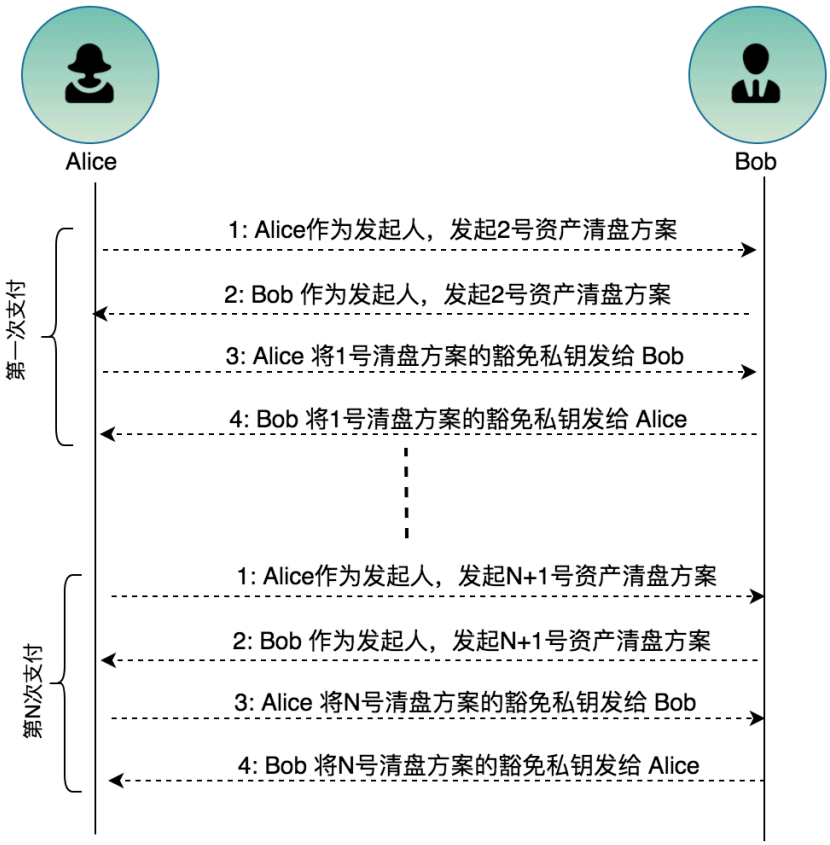
建立支付通道的流程包括：Alice 和 Bob 建立虚拟银行智能合约，约定初始的资产清盘协议，并且按照预定的额度向虚拟银行注资。交互流程如下面的序列图所示：



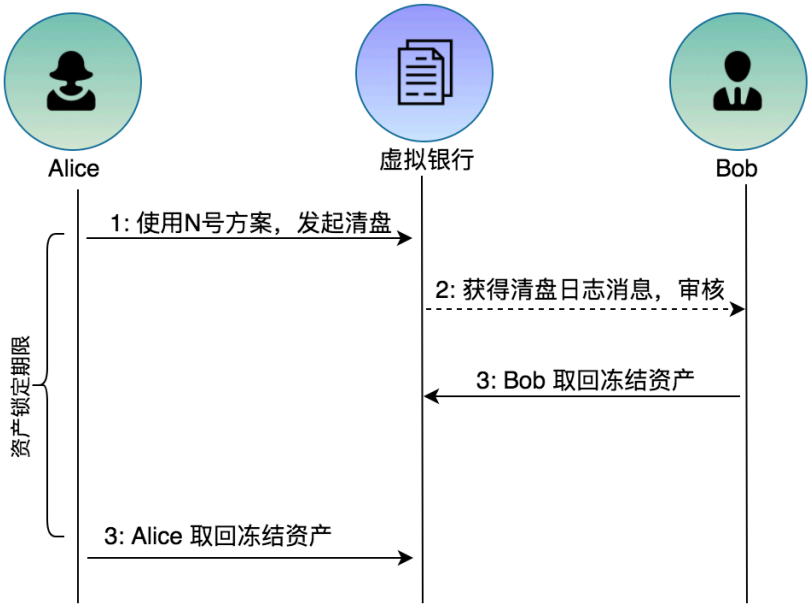
1. 协商虚拟银行配置：在建立虚拟银行智能合约之前，Alice 和 Bob 互相交换资产地址，协商共同出资的额度、和冻结资产的锁定期。假设双方分别出资 100 美元，锁定期为1小时。
2. 部署虚拟银行：Alice 或者 Bob 按照协商好的配置参数，部署虚拟银行智能合约。
3. Alice 作为清算发起人，生成一对公私钥 KeyWaiverA1，计算对应的 WaiveraddrA1。Alice 将一份清算方案发给 Bob，清算方案包括：序号、资产负债表、豁免地址。Bob 确认清算方案之后，给这份方案签名并且返回给 Alice。
4. 类似的，Bob 也作为清算发起人，生成一对公私钥 KeyWaiverB1，计算对应的 WaiveraddrB1。Bob 将一份清算方案发给 Alice，清算方案包括：序号、资产负债表、豁免地址。Alice 确认清算方案之后，给这份方案签名并且返回给 Bob。至此双方互相交换了清盘方案。
5. Alice 和 Bob 分别向虚拟银行转入 100 美元。

至此虚拟银行的状态为 Running，Alice 和 Bob 之间建立了双向支付通道。

支付



关闭支付通道



4.3 HTLC 支付通道串联协议

5. 技术优势和劣势分析

5.1 场景限制

- 即时支付，买卖双方在链下达成了支付协议，银货对付(Delivery Versus Payment)，买方需要在交付当时支付有关款项，或者卖方在收款后立刻交付产品或者服务。
- 双方大量小额往来账户，支付通道的利用率就高。

6. 技术的进一步拓展

多方清算所合约

Uniform Resource Locator 地址，路由优化

半同态Hash - 闭环攻击

广义支付通道

- 论文：cChannel Generalized State Channel Specification
- GitHub 源码: general-state-channels
- 媒体报道 counterfactual: generalized-state-channels-on-ethereum

Sprites and State Channels: Payment Networks that Go Faster than Lightning

1. 降低资产抵押
2. 支付通道的重用，可以部分存款和取款

7. 现代支付清算结算系统与应用

8. 总结

在比特币的基础上，闪电网络提出了以债务清算的方式实现价值转移的新方式。一方面有，协议定义了一种新的价值转移方式，通过智能合约管理博弈论的设计，支付双方权利制衡，形成的纳什均衡使得双方必须保持诚信，以完全去信任的方式实现了支付通道网络。

当参与者把他们所有的钱都放在他们的渠道上时，它们是不会冒险进行欺诈的，而中间商在完成交易的时候也无法窃取用户哪怕是一丁点的比特币。

去信任的智能合约代替了传统的银行机构，负责托管用户的资产；基于密码学的 RSMC 和 HTLC 协议，代替了中心化的清算所，负责清算虚拟银行中的资产负债表。这三种技术整合在一起，构建了去信任的、高效的清算系统。

- 底层逻辑基础层面

- 产品功能
- 宏观金融层面

比特币通过所有权交割的模式实现电子支付，转账过程中必须要多方共识，性能很难达到实际的需求。闪电网络在比特币的基础上提出了去信任的清算协议，使用债务清算的方式完成支付的结算，一次支付只需要少数参与者之间达成共识，不但大幅提升了吞吐量，而且

```
在1930年出版的上下两卷的《货币论》第一卷第一章开篇第一句，
凯恩斯就说：“记账货币是表示债务、物价与一般购买力的货币。这是货币理论中的原始概念。”

事实上货币的本质是一种债，在每张美元的钞票的正面上都明确标识出来：
“This note is legal tender for all debts,  public and private.”
在每一张英镑的钞票的正面，也明确标识着英国女王的承诺：
“I promise to pay the bearer on demand the sum of XX Pounds.”

从大范围的人类社会的货币制度史来看货币，就会发现人类社会经济运行中的一些深层次的东西。
把货币看成一种债，一种可转让的信用，一种支付承诺，用现代制度经济学的话来说是一种“债务支付契约
就会发现许多之前看不清楚的人类社会经济运行的基本法则。
```

债务货币、实物货币

1. 现代社会由庞大、复杂的协作关系网络组成，其中一些关键节点和周边其它节点形成了稳定的信用关系。同时法律制度健全也促使信用关系的建立和稳固。所以现代社会为债务货币提供了充足的信用基础条件。
2. 实物货币制度需要高昂的成本、不能支撑大规模的支付。非现金支付已经成为必然的趋势。现代债务货币

区块链目前还只是少数创业者和极客玩家的游戏，未来逐步普及到大众的日常经济活动中。

可以理解，成熟的同业竞争对手对银行的竞争威胁越来越少，而 新进者造成的威胁却越来越多，新进者几乎全部来自热门的互联网公司。

我们将见证，在未来几年的创新、动荡和改革中，市场和消费者肯定受益颇丰。

从某种意义上说，比特币通过计算机程序还原了最原始的价值转移方式，而闪电网络又把我们拉回到了现代。

Reference

- 闪电网络白皮书 Joseph Poon,Thaddeus Dryja, The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments
- 闪电统计
- 闪电网络工作原理， 中文翻译

- BitMEX：从中期来看，闪电网络不会带来什么大变革
- 简述闪电网络历史：一个与比特币同样伟大想法的历程
- 闪电网络如何工作 英文原文
- 社群对闪电网络的看法
- The growth of the Lightning Network
- 简述闪电网络历史：一个与比特币同样伟大想法的历程