查看我们实施的 Swap 协议 AirSwap！

交换：交易以太坊令牌的点对点协议

ENCNJPKRRUUA

Michael Oved， Don Mosites

2017 年 5 月 10 日发布

抽象

我们提出了一种点对点方法，用于在以太坊区块链上交易 ERC20 令牌。首先，我们概述了区块链订单簿的局限性，并在点对点令牌交易中提供了强有力的替代方案：脱链谈判和在线结算。然后，我们描述了一种协议，通过该协议，各方能够向他人发出交易令牌的意图。一旦连接，交易对手就可以自由地传达价格并传递订单。在此过程中，各方可以要求独立第三方 oracle 的价格来验证准确性。最后，我们提出以太坊智能合约来填补以太坊区块链的订单。

目录

介绍

随着越来越多的用例被实施为智能合约，过去 12 个月以来以太坊的数字资产数量激增。我们的论点是，这种趋势将持续到未来; 因此，当用户在用例之间移动或重新平衡其标记化的投资组合时，我们相信这种增长将增加交换资产的需求。基于区块链订单的交易并非没有固有的局限性，其中许多可以通过本文概述的设计决策来缓解。我们寻求通过指定一组解决资产流动性的协议来提供区块链订单的替代方案，并使以太坊生态系统在没有这些限制的情况下自由发展。

订购书籍

订单簿提供了一种高度自动化的方式来匹配给定可交易资产的供需。传统上，这些是集中的并且与订单执行相结合，这允许在中心来源处创建，执行和取消订单。本着分散化的精神，对区块链重新设计了订单簿。但是，在区块链上部署订单会产生一些限制。

区块链订单簿不能扩展。在区块链上执行代码会产生成本，因此自动化的订单 - 取消 - 订单周期很快变得昂贵，并且作为高性能，自动化匹配系统而破坏了订单簿的强度。实际上，如果该匹配算法在区块链上运行，则下订单的一方将产生执行成本，该执行成本随着订单的大小而显着增加。

区块链订单是公开的。由于在区块链上创建订单的交易由矿工处理，因此这些矿工在将订单发布到图书之前就知道该订单。这为正面运行创造了可能对原始订单产生重大影响的机会。此外，由于订单是公开发布的，订单价格对每个人来说都是相同的，从而消除了供应商调整流动性的能力。

区块链订单书是不公平的。物理分布式系统固有地遭受其节点之间的延迟。由于矿工在地理上分布，复杂的各方可能能够共同定位，检测订单，并超越区块链延迟，有效地在其他方面之前对订单信息采取行动。这种信息不对称可能会使不太复杂的政党根本无法参与生态系统。

点对点（P2P）

或者，点对点交易使各方能够直接相互交易。我们日常进行的大多数交易都是点对点的：在咖啡馆买咖啡，在 eBay 上卖鞋，或在亚马逊上买猫粮。因为这些是人或企业之间的私人交易，所以每一方都知道并最终选择与谁交易。

点对点交易规模。订单在各方之间传递，并且是“一个完成”而不是公共交易所的订单，而不保证完全填写。这使得订单上的取消定期发生，而点对点订单可能被填充，因为它们被提供给已经表达了兴趣的各方。此外，点对点供应和需求匹配可以通过轻量级对等点发现来解决，而不是昂贵的算法匹配 - 无论是在链上还是在链外。

点对点交易是私有的。一旦双方找到并选择相互交易，就不需要第三方进行谈判。在谈判期间，这些方之间的沟通仍然是私密的，从而使其他方无法对订单请求行为采取行动。只有在提交订单时，它才会成为公共知识。

点对点交易是公平的。由于订单是在双方之间直接创建和传输的，因此没有外部参与者可以获得优势。只要他们与多个独立的政党合作，参与者就可以获得与他们在交易所实现的价格相当或更好的价格。此外，这些定价订单可以积极地进行，而不必担心被自动化，低延迟交易策略利用。

区块链订单书所施加的可扩展性，隐私和公平性限制使得必须有另一种选择。今天的以太坊生态系统需要一个开放的点对点解决方案来进行资产交换。

介绍交换

交换是一种协议，用于促进真正的点对点生态系统，用于在以太坊区块链上交易令牌。以下是可扩展的规范，支持有效的对方发现和协商。这些协议旨在成为资产交易生态系统的基础，并加速以太坊生态系统的发展。通过发表本文并开始讨论，我们寻求生态系统利益相关方的意见，旨在生成高质量的协议，以支持各种各样的实际应用。

同伴协议

交易对手之间只传递了一些消息，交易可以快速，公平和私下进行协商。出于本文档的目的，Maker 是提供订单的一方，而 Taker 是填写订单的一方。由于每一方都是同行，任何一方都可以随时担任 Maker 或 Taker 的角色。以下规范中的标记符合 ERC20 标准，任何实现该标准的标记都可以使用此协议进行交易。

核心协议在下图中排序。Maker 和 Taker 执行脱链贸易谈判。以下合同是以太坊智能合约，Taker 在准备填写区块链订单时调用。

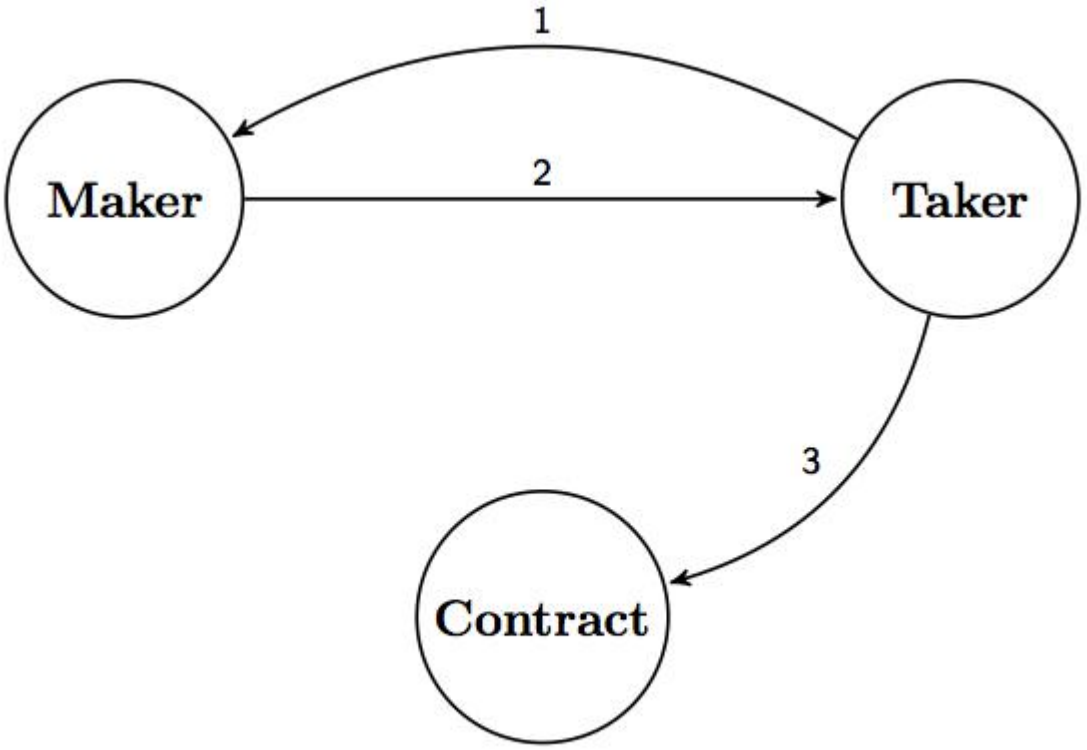

Figure 1: Request, provide, and fill an Order

1. Taker 在 Maker 上调用 getOrder。
2. 制造商回复订单。
3. Taker 在合同上调用 fill（[order]）。

订单 API

以下 API 是用于在对等方和服务之间进行通信的与传输无关的远程过程调用（RPC）。示例使用令牌代码而不是地址，但实际调用需要 ERC20 兼容令牌的地址。以下呼叫签名仅供讨论之用，因为进一步的技术细节将在另一份文件中公布。

Order API 是脱链的，并指定在交易协商期间在交易对手之间进行的异步调用。实现者可以选择将请求提供周期作为同步请求 - 响应来提供。由于订单是由制造商签署的，因此 Taker 可以稍后将其提交给要填写的智能合约。

getOrder

getOrder（makerAmount，makerToken，takerToken，takerAddress）

由制造商的 Taker 调用，请求订单交易代币。

```
Example: “I want to buy 10 GNO using BAT.”
```

```
getOrder(10, ‘GNO’, ‘BAT’, <takerAddress>)
```

provideOrder

provideOrder（makerAddress，makerAmount，makerToken，takerAddress，takerAmount，takerToken，expiration，nonce，signature）

由 Taker 上的 Maker 调用，提供签名的执行订单。

```
Example: “I’ll sell you 10 GNO for 5 BAT.”
provideOrder(<makerAddress>, 10, ‘GNO’, <takerAddress>, 5, ‘BAT’, <expiration>, <nonce>, <signature>)
```

索引器协议

Indexer 是一种脱链服务，根据交易意图汇总和匹配同行：未来的 Makers 和 Takers 是否希望买入或卖出代币。索引器是脱链服务，汇总了这种“交易意图”，并根据购买或出售特定代币的意图帮助匹配同行。许多潜在的制造商可以发出交易意图的信号，当 Taker 要求 Indexer 找到合适的交易对手时，可能会有多个结果。一旦 Taker 找到了他们想要交易的制造商，他们就会继续使用上面的对等协议进行谈判。一旦 Maker 和 Taker 达成协议，该订单将在智能合约上填写。

Maker，Taker 和 Indexer 之间的交互如下图所示。Maker，Taker 和 Indexer 都可以远离区块链进行操作，并通过任何首选的消息传递介质进行通信。

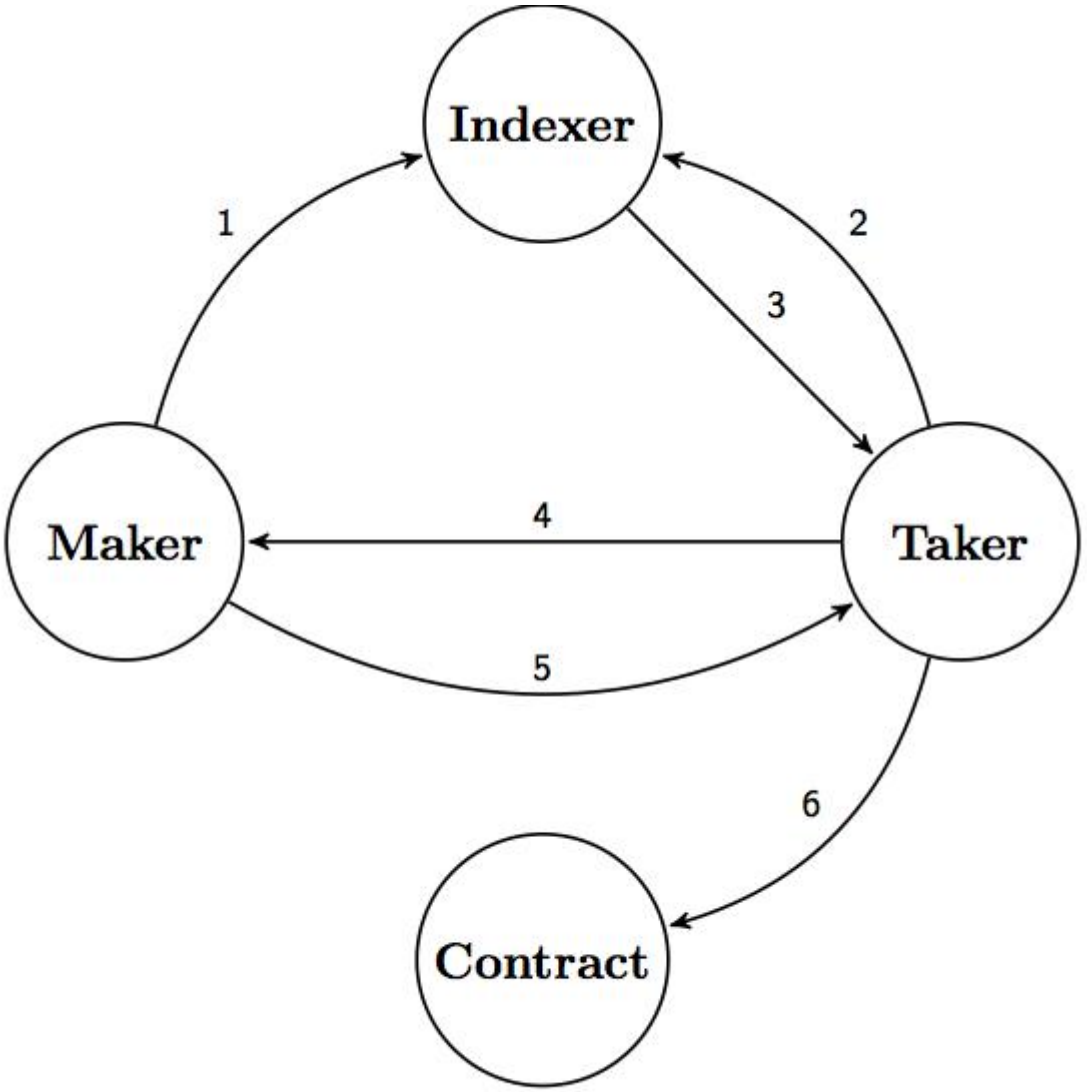

Figure 2: Find a counterparty and make a trade

1. Maker 在 Indexer 上调用 addIntent。
2. Taker 在 Indexer 上调用 findIntent。
3. Indexer 在 Taker 上调用 foundIntent（[Maker]）。
4. Taker 在 Maker 上调用 getOrder。
5. 制造商回复订单。
6. Taker 在合同上调用 fill（[order]）。

下图说明了几个 Makers，Taker 和 Indexer 之间的交互。每个制造商都独立宣布他们的意图。Taker 要求查找具有特定意图的 Makers，并且 Indexer 返回以太坊地址和详细信息的列表。

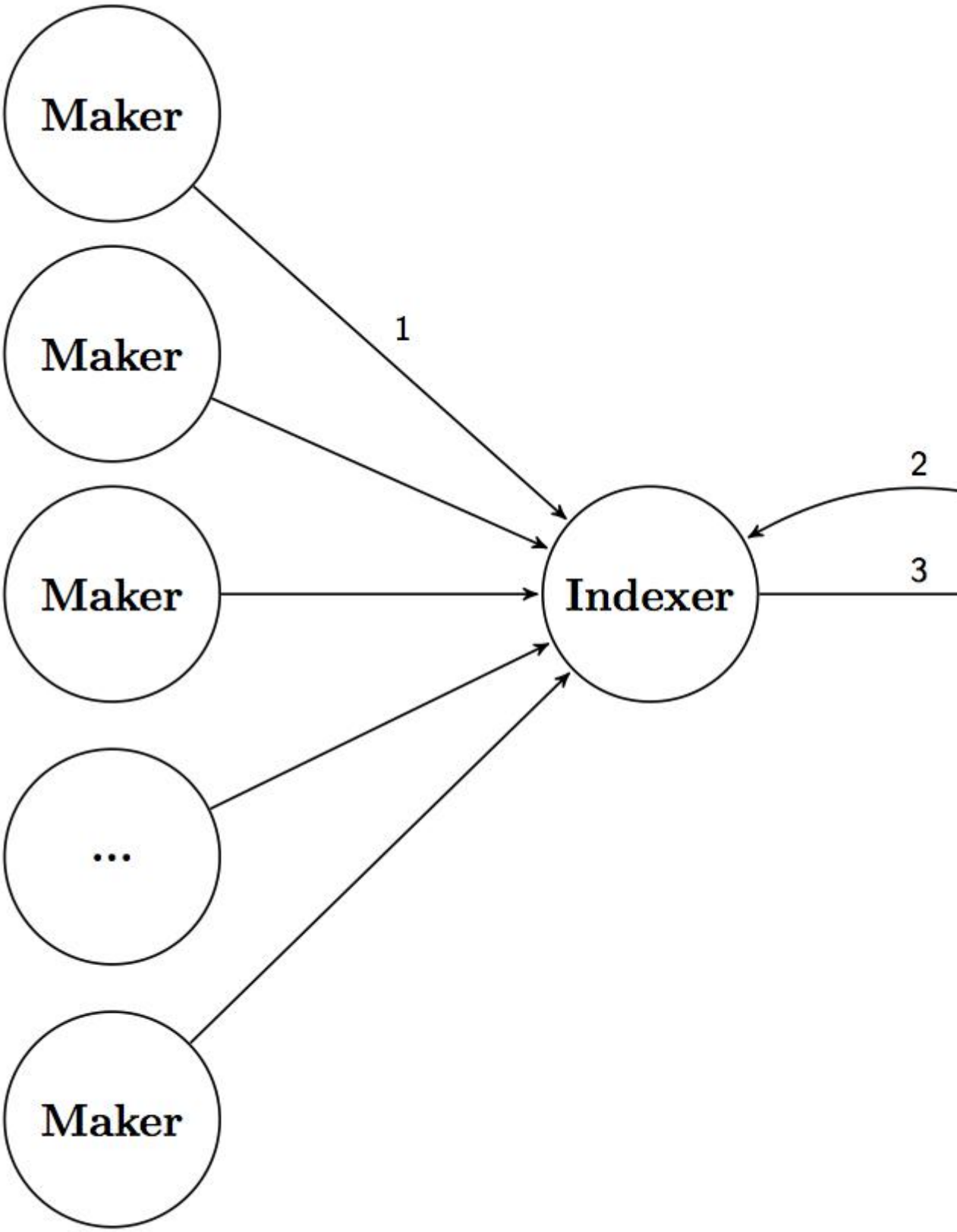

Figure 3: Makers call addIntent, a Taker calls findIntent

1. 几个 Makers 在 Indexer 上调用 addIntent。
2. Taker 在 Indexer 上调用 findIntent。
3. Indexer 在 Taker 上调用 foundIntent（[Maker]）。

一旦 Taker 找到合适的制造商，他们就可以使用订单 API 来请求每个制造商的订单相互权衡。如果 Taker 决定填写给定订单，他们将在智能合约上进行填写。

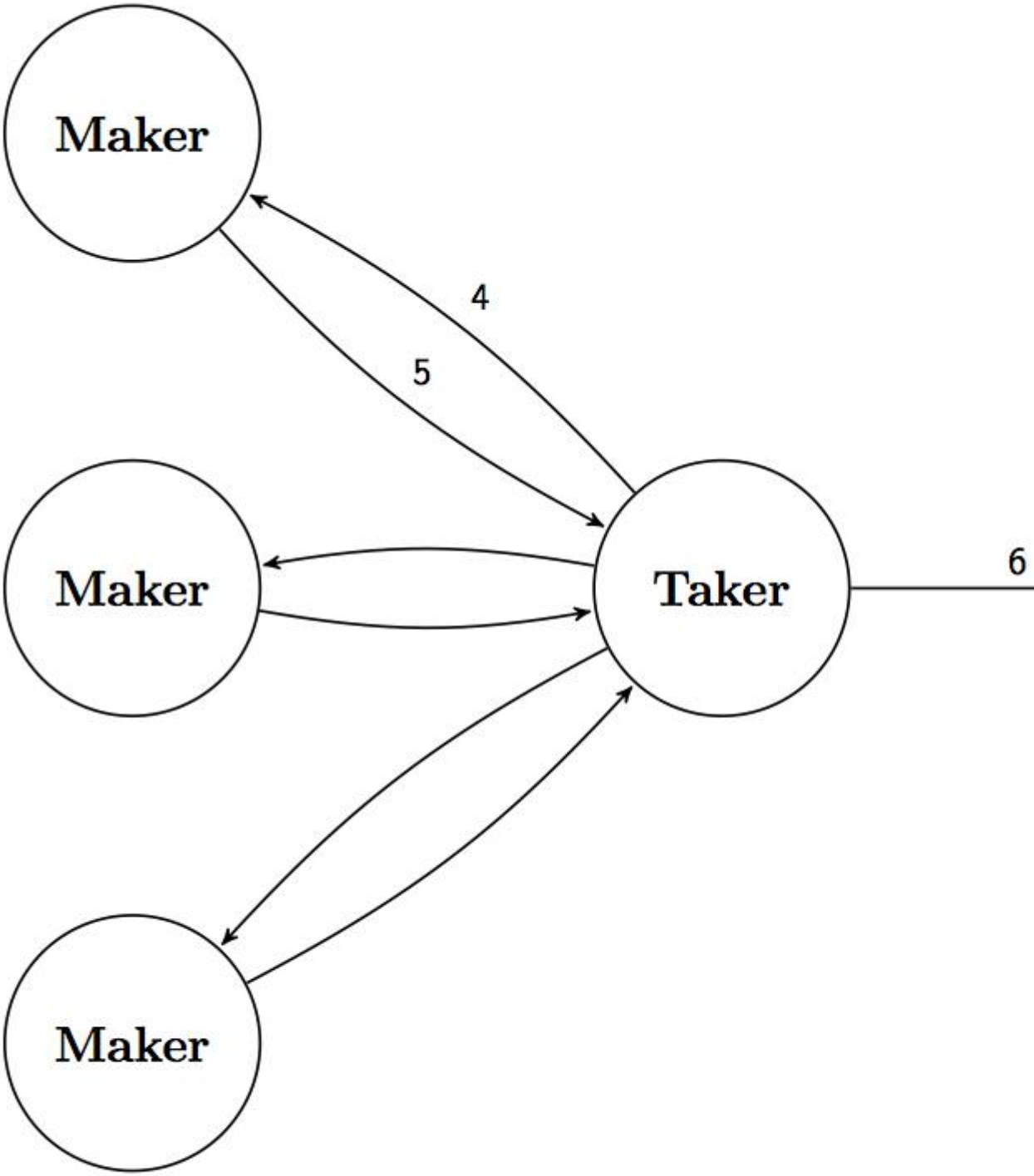

Figure 4: Taker calls getOrder on Makers, Taker calls fill

1. Taker 在几个 Makers 上调用 getOrder。

2. 制造商回复订单。
3. Taker 选择订单并在合同上调用 fill（[order]）。

Indexer API

Indexer API 管理交易意图，在同行之间发出信号。以下调用在对等方和索引器之间进行。

addIntent

addIntent（makerToken，takerTokens）

添加购买或出售一定数量令牌的意图。

```
Example: “I want to trade GNO for BAT.”
addIntent(‘GNO’, [‘BAT’])
```

removeIntent

removeIntent（makerToken，takerTokens）

删除交易令牌的意图。

```
Example: “I am no longer interested in trading GNO for BAT.”
removeIntent(‘GNO’, [‘BAT’])
```

getIntent

getIntent（makerAddress）

列出与地址关联的活动意图。

```
Example: “List the tokens that [makerAddress] wants to trade.”
getIntent(<makerAddress>)
```

findIntent

findIntent（makerToken，takerToken）

找一个愿意交易特定代币的人。

```
Example: “Find someone trading GNO for BAT.”
findIntent(‘GNO’, ‘BAT’)
```

foundIntent

foundIntent（makerAddress，intentList）

Indexer 找到了有意交易的人。

```
Example: “Found someone selling 10 GNO for BAT.”
foundIntent(<makerAddress>, [{ makerAmount: 10, makerToken: 'GNO', takerTokens: ['BAT'] })
```

Oracle 协议

Oracle 是一种脱链服务，为 Makers 和 Takers 提供定价信息。在将订单交付给 Taker 之前对订单进行定价时，制造商可能会向 Oracle 询问其认为合理的价格建议。同样，收到订单后，Taker 可能会要求 Oracle 检查订单上的价格以验证其是否公平。Oracle 提供此定价信息，以帮助 Maker 和 Taker 做出更有根据的定价决策，并顺利完成贸易谈判。

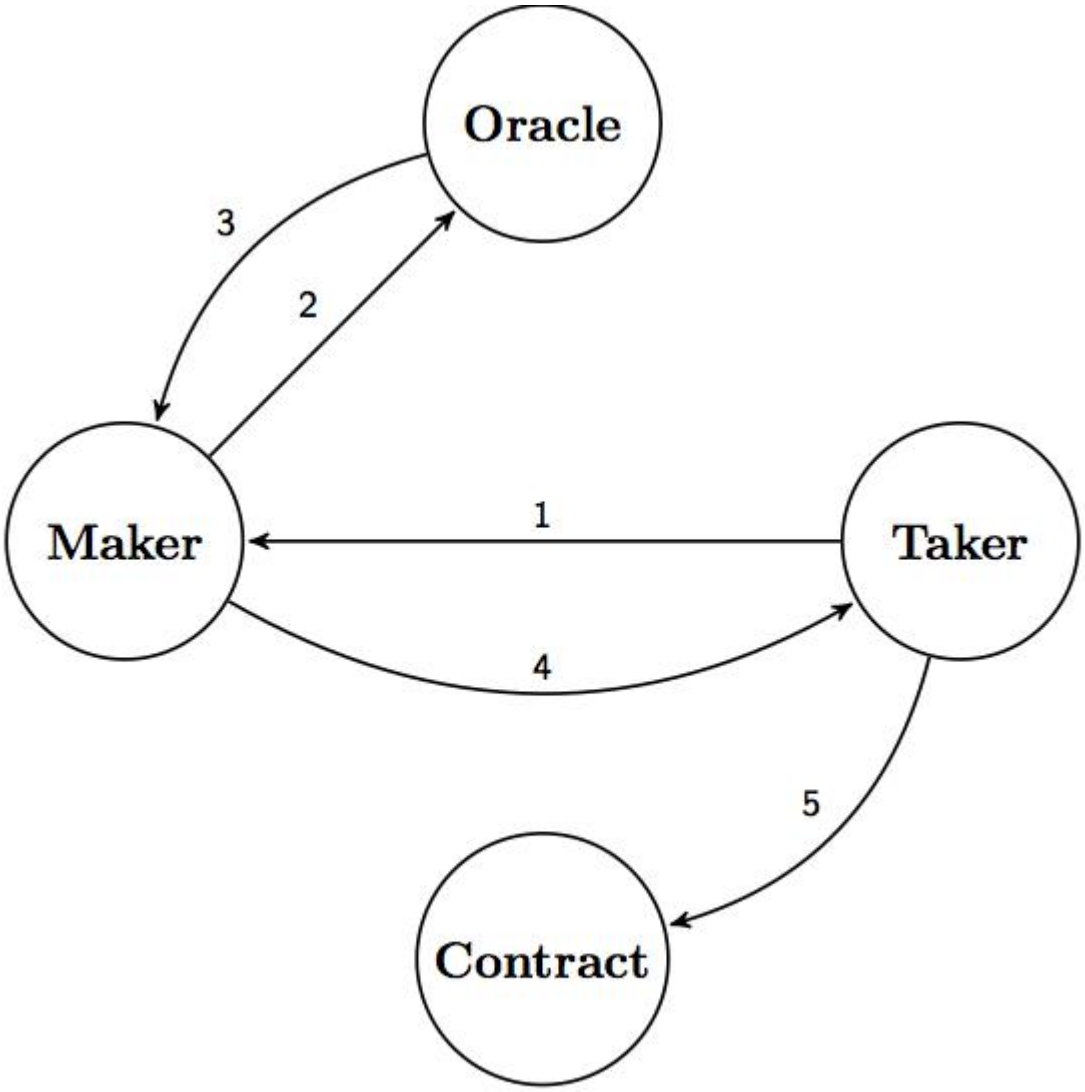

Figure 5: Maker querying Oracle before providing order

1. Taker 在 Maker 上调用 getOrder。

2. Maker 在 Oracle 上调用 getPrice。
3. Oracle 向制造商返回价格。
4. 在分析价格信息后，Maker 回复订单。
5. Taker 在合同上调用 fill（[order]）。

当 Taker 收到订单时，Taker 和 Oracle 之间发生了非常类似的交互。

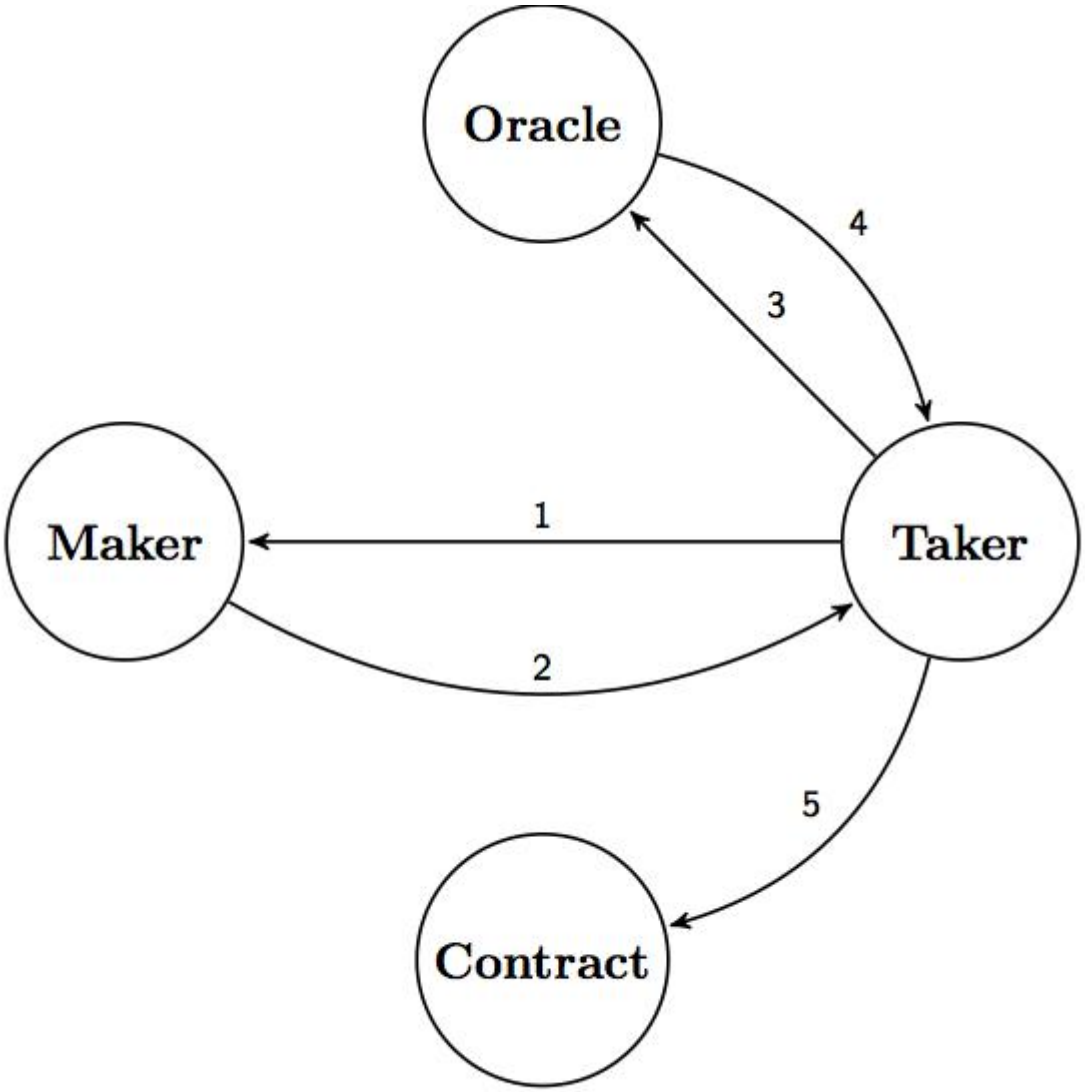

Figure 6: Taker querying Oracle before filling order

1. Taker 在 Maker 上调用 getOrder。
2. 制造商回复订单。
3. Taker 在 Oracle 上调用 getPrice。
4. Oracle 向 Taker 返回价格。
5. 在分析价格信息后，Taker 在合同上调用 fill（[order]）。

Oracle API

Makers 和 Takers 使用 Oracle API 来确定订单价格。价格是建议，不可执行。

用 getPrice

getPrice（makerToken，takerToken）

由 Oracle 上的 Taker 或 Maker 调用以获得价格。

```
Example: “What is the current price of GNO for BAT?”
getPrice(‘GNO’, ‘BAT’)
```

providePrice

提供价格（makerToken，takerToken，价格）

由 Oracle 在 Maker 或 Taker 上调用以给出价格。

```
Example: “The current price of GNO for BAT is 0.5.”
providePrice(‘GNO’, ‘BAT’, 0.5)
```

智能合约

用于填写或取消订单的以太坊智能合约。

填

fill（makerAddress，makerAmount，makerToken，takerAddress，takerAmount，takerToken，expiration，nonce，v，r，s）

由 Taker 调用的令牌的原子交换。合同确保邮件发件人与收件人匹配，并确保过期时指示的时间尚未过去。要填写订单，同行必须已经在指定的代币上调用批准，以允许合约至少撤回指定的金额。对于令牌转移，合同会在相应的令牌上调用 transferFrom。成功完成此功能后，Filled 事件将广播到区块链。参数 `v`，`r` 并 `s` 构成制造商签名。

```
Example: “I want to fill this order of 5 GNO for 10 BAT.”
fill(<makerAddress>, 5, 'GNO', <takerAddress>, 10, 'BAT', <expiration>, <nonce>, <v>, <r>, <s>)
```

取消

取消（makerAddress，makerAmount，makerToken，takerAddress，takerAmount，takerToken，expiration，nonce，v，r，s）

取消已经传达给 Taker 但尚未填写的订单。由订单的制造商调用。将订单标记为已经填写合同，因此后续填写订单的尝试将失败。成功完成此功能后，将取消的事件广播到区块链。

```
Example: “I want to cancel this order of 5 GNO for 10 BAT.”
cancel(<makerAddress>, 5, 'GNO', <takerAddress>, 10, 'BAT', <expiration>, <nonce>, <v>, <r>, <s>)
```

以太订单

智能合约支持为令牌交易以太（ETH）。如果订单包含 null takerToken 地址（0x0），智能合约将检查通过函数调用发送的 ether 的值，并将其代表 Taker 传送给 Maker。

概要

Swap 协议满足了以太坊网络上分散资产交换的不断增长的需求。基于区块链的订单虽然新颖，当然也符合我们生态系统的精神，但我们认为最终使它们难以与当前可用的集中式解决方案竞争。交换提供了一种分散且不受这些限制影响的方法。

通过实施该协议，参与者可以以可扩展，私密和公平的方式获得流动性，而不会牺牲对优惠价格的访问权限。协议和 API 是可扩展的，我们鼓励社区与我们一起构建应用程序。我们欢迎反馈，并期待与您一起推动以太坊社区的发展。

如有问题，意见或反馈，请通过 team@swap.tech 与我们联系。