Novaswap 核心

摘要

这篇技术白皮书解释了Novaswap核心合约背后的设计决策。内容覆盖了合约的新特性，包括任意ERC20代币的交易对，强化的价格预言机制（price oracle），允许交易者先接收资产并使用，在未来可以开启的协议手续费（protocol fee）。为了减少受攻击面，重新设计了合约的架构。本白皮书描述了Novaswap核心合约的运行机制，包括储存流动性提供者资金的交易对合约，和用于实例化交易对合约的工厂合约。

1. 介绍

Novaswap是一个GeneChain区块链上的智能合约系统，基于常数乘积公式实现了自动流动性协议。每个Novaswap交易对在资金池里储存两种资产，并位这两种资产提供流动性，维持这两种资产的乘积不减少。交易者支付30bp(基点)的交易手续费给流动性提供者。合约是不能升级的。

Novaswap支持ERC20代币/ERC20代币交易对。除此以外，它也强化了价格预言，在每个区块开始处累积两个资产的相对价格。这允许其他GeneChain合约计算两个资产在任意时间区间上的TWAP(time-weighted average price时间加权后的价格)。

虽然合约通常来说不可升级，但是有一个私钥能够更新一个变量，打开链上5bp的手续费，这个变量一开始是关闭的，可以在未来打开，打开之后流动性提供者在每笔交易上赚取25bp，而不是30bp。

这篇论文描述了核心合约（core contract）的运行机制，以及用来实例化核心合约的工厂合约（factory contract）。实际上使用Novaswap需要通过一个“路由”合约（router contract），计算交易/存款数额并转移资金到交易对合约（pair contract）。

合约代码：https://github.com/novaswap-dev/Novaswap

2 新特性

2.1 ERC20交易对

Novaswap允许流动性提供商为任意两个ERC20创建对合约。

任意ERC20之间的交易对数量的激增可能会使寻找交易特定货币对的最佳路径变得更加困难，但是路由可以在更高的层面上处理（在链下处理或通过链上路由器或聚合器）。

2.2 价格预言(Price oracle)

Novaswap提供的t时刻的边际价格（marginal price，不包括手续费）可以用资产a的储备(reserve)除以资产b的储备(reserve)来计算。

$$P_t = \frac{r_t^a}{r_t^b}$$

如果这个价格偏离（超出手续费足够的数额），套利者会和Novaswap进行交易使价格回归正常，所以Novaswap提供的价格趋向于追踪资产在相关市场的价格。这意味着它可以被用作一个近似的价格预言。

Novaswap改进了预言的功能，通过测算和记录每个区块第一笔交易之前的价格（也就是前一个区块最后的价格）。这个价格比一个区块内的价格更难被操纵。如果攻击者提交了一笔交易（transaction）尝试在区块末尾处操纵价格，其他的套利者可以提交另一个交易（transaction）立即进行反向交易。某个矿工（或有足够gas填满整个区块的攻击者）可以操纵区块末尾处的价格，除非他们可以挖出下一个区块，否则他们他们没有特殊的的套利优势。

具体来说，Novaswap追踪每个和合约交互的区块开始处的价格的累加和，来累加价格。每个价格用距离上一个更新价格的区块的时间进行加权，根据区块时间戳。这意思是累加器的值在任意时间（更新后）的值等于合约历史上每秒的现货价格的和。

$$a_t = \sum_{i=1}^{t} p_i$$

要计算从时间 t1到t2的时间加权平均价，一个外部调用者可以检查t1和t2时间的累加器的值，将后值减去前值，再除以期间经过的秒数。（注意，合约本身并不记录历史累加值，调用者必须在时间段开始处调用合约来读取和储存这个值。）

$$p_{t1,t2} = \frac{\sum_{i=t1}^{t2} p_i}{t_2 - t_1} = \frac{\sum_{i=t2}^{t2} p_i - \sum_{i=t1}^{t2} p_i}{t_2 - t_1} = \frac{a_{t_2} - a_{t_1}}{t_2 - t_1}$$

预言的用户可以选择这个区间的起始和结束时间。选择更长的区间可以让攻击者操纵价格的成本更高，虽然这会导致价格变化滞后。

一个复杂之处：我们应该用资产B来计算资产A的价格还是用资产A来计算资产B的价格？虽然用B计算的A的现货价格总是用A计算的B的现货价格的倒数，但是在某一时间段内用B计算的A的均价不等于用A计算的B的均价的倒数。举个例子，如果USD/RNA价格在区块1中是100，在区块2中是300，USD/RNA的均价是200 USD/RNA，但是RNA/USD的均价是1/150 RNA/USD。因为合约无法知道用户想要用哪个资产作为账户单位，所以Novaswap会同时追踪两个价格。

另一个复杂之处：有没有可能某个人发送资产给交易对合约，用来改变它的余额和边际价格，但又不和它交互，因此不会触发价格更新。如果合约简单的检查它自己的余额然后更新预言，攻击者可以在某一个区块中第一次调用合约之前向合约发送资产，从而操纵价格。如果上一次交易的区块是X秒以前，合约会在累加之前错误地把新价格乘以X，即使没有用户用那个价格进行过交易。为了防止这个问题，核心合约在每次交互后缓存它的资金储备，用缓存的资金储备更新价格预言而不用当前资金储备。除了保护价格预言被操纵，这个改动也启用3.2节中描述的合约重新架构。

2.2.1 精度

Solidity不支持非整型数，Novaswap用简单的二进制定点数格式编码和控制价格。具体来说，某一时间的价格存储为UQ112.112格式，意思是在小数点的任意一边都有112位精度，无符号。这些数字的范围是 [0,$2^{112}-1$] ，精度是 $\frac{1}{2^{112}}$ 。

选择UQ112.112格式是由于实用的原因，因为这些数可以被存在uint224中，在256位中剩余的32位空余。储备资金各自存在uint112中，剩余32位存储空间。这些空闲空间被用于之前描述的累加过程。具体来说，储备资金和时间戳存储在至少有一个交易的最近的区块中，mod 232（译者注：取余数）之后可以存进32位空间。另外，虽然任意时间的价格（UQ112.112数字）确保可以储存进224位中，但某段时间的累加值确保能存下。存储A/B和B/A累加价格空间尾部附加的32位用来存连续累加溢出的位。这样设计意味着价格预言只在每一个区块的第一次交易中增加了3次SSTORE操作。

主要的缺点是32位不足以储存时间戳并确保不溢出。事实上32位Unix时间戳的溢出日期是2106年7月2日。为了确保系统在这个日期后以及每 $2^{32}-1$ 秒的间隔能够继续运行，预言简单要求每个间隔至少检查一次价格（大约136年一次）。这是由于累加的核心函数（mod取余运算）是溢出安全的，意思是预言用溢出算法计算差值，跨越溢出区间的交易可以被合理计算。

2.3 协议手续费(Protocol fee)

Novaswap包括0.05%协议手续费，可以打开或关闭，如果打开，手续费会被发送给工厂合约中指定的feeTo地址。

初始时，feeTo没有被设定，不收手续费。一个预先指定的地址feeToSetter可以在Novaswap工厂合约上调用setFeeTo函数，设置feeTo地址。feeToSetter也可以自己调用setFeeToSetter修改feeToSetter地址。

如果feeTo地址被设置，协议会收取5pb的手续费，从流动性提供者的30bp手续费中抽取1/6。交易者将在所有交易上支付0.3%手续费，83.3%的手续费给流动性提供者，16.6手续费给feeTo地址。

总共收集的手续费可以用自从上次手续费收集以来的增长来计算。以下公式给出了t1和t2之间的累加手续费占t2时间资金池中流动性的百分比：

$$f_{1,2}=1-\frac{\sqrt{k_1}}{\sqrt{k_2}}$$

如果fee在时间t1前启用，feeTo地址应该获得1/6的t1到t2时间段内的累加手续费。因此，我们要铸造新的流动性代币给feeTo地址 $\phi * f_{1,2}$，其中 $\phi = \frac{1}{6}$ 。

我们要选择一个sm满足以下关系，其中s1是t1时刻的流通份额（outstanding shares）总量：

$$\frac{s_m}{s_m + s_1} = \phi * f_{1,2}$$

经过变换，将替换为后，解得

$$s_m = \frac{\sqrt{k_2} - \sqrt{k_1}}{\frac{1}{\phi} * \sqrt{k_2} + \sqrt{k_1}} * s_1$$

设，得到以下公式

$$s_m = \frac{\sqrt{k_2} - \sqrt{k_1}}{5 * \sqrt{k_2} + \sqrt{k_1}} * s_1$$

假设初始存款人存了100USDT和1RNA在交易对中，收到10份额。一段时间后（如果没有其他存款人参与）他们把钱转出，这时交易对有96USDT和1.5RNA，用上面得公式可以得出：

$$s_m = 0.0286$$

2.4 资金池份额的meta交易(Meta transactions for pool shares)

Novaswap交易对铸造得资金池份额原生支持meta转账。这意思是用户可以用签名授权一个他们的资金池份额的转账，而不用从他们的地址进行链上转账。任何人都可以调用permit函数来以用户的名义发送签名，支付gas手续费并在同一个转账中执行其他操作。

3 其他修改

3.1 Solidity

Novaswap是使用Solidity实现的。

3.2 合约重新架构

Novaswap在设计时优先考虑的一个方面是最小化受攻击表面积和核心交易对合约的复杂度，交易对合约储存了流动性提供者的资产。这个合约中的任何bug都可能是灾难性的，因为数百万美元的流动性可能会被窃取或冻结。

在评估核心合约的安全性时，最重要的问题时它是否保护流动性提供者以防他们的资产被窃取或冻结。除了基本的允许资产在资金池中互换的功能，任何支持或保护交易者的特性，都可以在路由合约中处理。

事实上，甚至一部分的互换功能也可以被抽出放到路由合约中。之前提到过，Novaswap储存有每种资产最近一次记录的余额（为了防止预言机制被操纵）。新的架构利用了这一点来进一步简化Novaswap合约。

Novaswap中，卖方在调用互换函数之前发送资产到核心合约。然后合约通过比较上次记录的余额和最新余额来计算它收到了多少资产。这意味着核心合约对交易者转账的方式是不可知的。除了transferFrom以外，也可能是元交易（meta transaction），或未来任何其他的授权ERC20代币转账的机制。

3.2.1 sync()和skim()函数

为了防止特殊实现用来修改交易对合约余额的代币，并且更优雅地处理总发行量超过的代币，Novaswap有两个救援函数：sync()和skim()。

sync()作用是在代币异步地减少交易对的余额时的恢复机制。这种情况下，交易会收到次优的汇率，如果没有流动性提供者作出反应，交易对会卡住。sync()的作用是设置合约的储备金为合约当前的余额，提供一种稍微优雅一点的恢复机制。

skim()作用是在发送到代币的数量溢出了uint112大小的储备金存储空间时的恢复机制，否则交易会失败。skim()函数允许用户将提出交易对当前余额和的差值大于0时，将差值提出到调用者的地址。

3.3 处理非标准和非常规代币

ERC20标准要求transfer()函数和transferFrom()函数返回布尔值表示调用的成功或失败。一些代币对这两个函数的实现没有返回值。

Novaswap用不同的方式处理非标准的代币实现。具体来说，如果一次transfer()调用没有返回值，Novaswap把它转换为成功而非失败。这个改动不应该影响任何遵从ERC20协议的代币（因为那些代币中transfer()总是有返回值）。

某些ERC20代币违反了这个假设，包括支持ERC777协议的"hooks"的代币。为了完全支持这些代币，Novaswap包含了一个“锁”，直接防止重入所有公共的修改状态的函数。

3.4 初始化流动性代币供给

当新的流动性提供者向现有的Novaswap交易对中存代币时，计算铸造的流动性代币数量基于现有的代币数量：

$$s_{minted} = \frac{x_{deposited}}{x_{starting}} * s_{starting}$$

如果是第一个存款人呢？在 $x_{starting}$ 为0的情况下，这个公式不能用。

Novaswap设初始份额供给等于存入地RNA数量（以Wei计）。这有一定的合理价值，因为如果初始流动性是在合理价格存入的，那么1流动性份额（和RNA一样是18位小数精度代币）大约值2RNA。

但是这意味着流动性资金池份额的价值依赖于初始存入的比例，这完全可能是任意值，尤其是因为没有任何比例可以反应真实价格的保证。另外，Novaswap支持任意交易对，有许多交易对根本不包含RNA。

相反，Novaswap初始铸造份额等于存入代币数量的几何平均值：

$$s_{minted} = \sqrt{x_{deposited} * y_{deposited}}$$

这个公式确保了流动性资金池份额的价值在任意时间和在本质上和初始存入的比例无关。举个例子，假设当前1 ABC的价格是100XYZ。如果初始存款是2 ABC和200 XYZ（比例1:100），存款人会获得份额。这些份额现在应该任然值2 ABC和200 XYZ，加上累加手续费。

如果初始存款是2 ABC和800 XYZ（1：400比例），存款人会收到资金池份额。

以上公式确保了流动性资金池不会少于资金池中储备金额的几何平均值。但是，流动性资金池份额的价值随时间增长是可能的，通过累加交易手续费或者向流动性资金池“捐款”。理论上，这会导致一种极端情况，最小的流动性份额数量（1e-18份额）过于值钱，以至于无法位小流动性提供者提供任何流动性。

为了减轻这种情况，Novaswap销毁第一次铸造的1e-15资金池份额，发送到全零地址而不是铸造者。这对任何代币交易对都是微不足道的。但是这显著地提高了上述攻击地成本。为了提高流动性资金池份额价值到100美元，攻击者需要捐献100000美元到资金池总，这会被作为流动性永久锁定。

3.5 包装RNA

GeneChain原生资产RNA的转账接口和ERC20交互用的标准接口不同，使用了一种标准的“包装的RNA”代币，WRNA。原生RNA需要包装后才能在Novaswap上交易。

3.6 确定交易对地址

Novaswap交易对是通过单一的工厂合约进行实例化的。Novaswap使用GeneChain新的CREATE2运算码来创建确定地址的交易对合约，这意味着可以在链下计算某个交易对的地址，不用查看GeneChain区块链的状态。

3.7 最大代币余额

为了高效地实现预言机制，Novaswap只支持最高储备余额 $2^{112}-1$。这个数字足以支持总发行量超过一千万亿的18位小数精度的代币。

如果储备余额超过了，任何swap函数调用都会失败（由于_update()函数中的检查逻辑）。要从这个状态中恢复，任何用户都可以调用skim()函数从流动性池中删除超出的资产。

4 NOVA 代币

4.1 分配

`NOVA` 是 `Novaswap` 协议代币。总发行量1亿枚。其中：

1. 20%分配给投资者。在上线前释放，将会采用预售10%，空投10%的形式下发。
2. 20%分配给技术团队，4年释放，每季度释放其中的1/16。
3. 60% 作为流动性挖矿奖励池

- 每个待上线swap的货币对，都设有奖励计划，指定周期、指定总释放奖励数，奖励给提供流动性的用户，按质押资产的比例分配
- 上线swap货币对的计划，提前公告
- 挖矿奖励池每年增发2%，即120万枚

4.2 流动性挖矿

在 `Novaswap` 上线一周后，我们会开启流动性挖矿，奖励NOVA代币给交易对提供流动性的用户。

4.3 治理

NOVA代币持有者享有对社区发展决定的权力，通过治理功能进行投票来实现：

- 持有(代理)NOVA总发行量1%以上的地址，可以发起提议
- 提议获得赞成票大于NOVA总发行量4%，即通过
- 投票阶段持续7天
- 如果投票通过，2天后可以执行