Uniswap v2 核心

Hayden Adams	Noah Zinsmeister	Dan Robinson
hayden@uniswap.org	noah@uniswap.org	dan@paradigm.xyz

2020年3月

摘要

这篇技术白皮书解释了Uniswap v2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，随后在转账中支付的快速互换（flash swaps）功能，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Uniswap v2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

Uniswap v1是一个以太坊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（constant product fomular）。每个Uniswap v1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（译者注：0.3%）的交易手续费给流动性提供者。合约是不能升级的。

Uniswap v2基于同样的公式实现，添加了几个特性。最重要的一个是支持ERC20代币/ERC20代币交易对，而不像v1只支持ERC20代币/ETH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。这允许其他以太坊合约计算两个资产在任意时间区间上的TWAP价格。最后，它启用了快速互换（flash swaps）功能，用户可以自由地接收资产并把他们用在链上的其他地方，只要在转账的最后支付（或返还）即可。

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。

Uniswap v2修复了一些v1中的小问题，也在实现中重新设计了架构，减少了Uniswap的受攻击面，通过把逻辑最小化放在核心合约（core contract）中，这个核心合约维持了流动性提供者的资金，从而使系统变得更容易升级。

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Uniswap v2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）。

译者注：

合约代码：https://github.com/Uniswap/uniswap-v2-core

Uniswap v1 白皮书：https://hackmd.io/@HaydenAdams/HJ9jLsfTz#%F0%9F%A6%84-Uniswap-Whitepaper

2 新特性

2.1 ERC20交易对

Uniswap v1使用ETH作为过渡货币（bridge currency），每个交易对都包含ETH。这使得路由更简单（ABC和XYZ之间的每笔交易都通过ETH/ABC交易对和ETH/XYZ交易对进行），并且减少了流动性的分散。

但是，这个规则给流动性提供者增加了可观的成本。所有流动资金提供者都有ETH的敞口，并且由于其他资产相对于ETH的价格变化而遭受无常损失（impermanent loss）。当两个资产ABC和XYZ相关时（例如，如果它们都是锚定美元的稳定币），则在Uniswap交易对ABC/XYZ上的流动性提供者遭受的无常损失会比ABC/ETH或XYZ/ETH交易对少。

使用ETH作为强制性过渡货币也会给交易者带来成本。交易者必须支付的费用是直接购买ABC/XYZ交易对的费用的两倍，因此他们遭受两次滑点。

Uniswap v2允许流动性提供商为任意两个ERC-20创建交易对合约。

任意ERC-20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（通过链下或通过链上路由器或聚合器进行）。

2.2 价格预言(Price oracle)

Uniswap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的储备除以资产b的储备来计算。

$$p_t = \frac{r_t^a}{r_t^b} \tag{1}$$

如果这个价格偏离（超出手续费足够的数额），套利者会和Uniswap进行交易使价格回归正常，所以Uniswap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。

但是用Uniswap v1作为链上价格预言是不安全的，因为它非常容易被操纵。假设一个其他的合约用当前的ETH-DAI的价格交割某个衍生品，操纵价格的攻击者会从ETH-DAI交易对买入ETH，在衍生品上触发结算（引起失真的结算价），然后再把ETH卖给交易对，用真实价格反向交易。这甚至可以用一个原子的转账来完成，或者通过一个控制区块内交易（transaction）顺序的矿工。

Uniswap v2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，Uniswap v2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。

$$a_t = \sum_{i=1}^{t} p_i \tag{2}$$

要计算从时间t_1到t_2的TWAP价格，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。）

$$p_{t_1,t_2} = \frac{\sum_{i=t_1}^{t2} p_i}{t_2 - t_1} = \frac{\sum_{i=1}^{t_2} p_i - \sum_{i=1}^{t_1} p_i}{t_2 - t_1} = \frac{a_{t_2} - a_{t_1}}{t_2 - t_1} \tag{3}$$

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/ETH价格在区块1中是100，在区块2中是300，USD/ETH的均价是200 USD/ETH，但是ETH/USD的均价是1/150 ETH/USD（译者注：他们的均价不是倒数关系）。因为合约无法知道用户想要用哪个资产作为账户单位，Uniswap v2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没有一等的非整型数的数据结构的支持，Uniswap v2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 2^{32}（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（当前花费15000gas）。

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 快速互换(Flash Swaps)

Uniswap v1中，用户用XYZ买ABC需要发送XYZ到合约，然后才能收到ABC。如果用户需要ABC为了获取他们支付的XYZ，这种方式是不方便的。举个例子，用户可能在其他合约中用ABC买XYZ，为了对冲Uniswap上的价格，或者他们可能在Maker或Compound上卖出抵押品平仓然后返还给Uniswap。

Uniswap v2添加了一个新的特性，允许用户在支付前接收和使用资产，只要他们在同一个原子的转账中完成支付。swap函数调用一个可选的用户指定的回调合约，在这之间转出用户请求的代币并且强制确保不变。一旦回调完成，合约检查新余额并且确保满足不变（在经过支付手续费调整后）。如果合约没有足够的资金，它会回退整个交易。

用户也可以用同样的代币返还给Uniswap资金池而不完成互换。这高效地让任何人从Uniswap资金池中快速借取任何资产（Uniswap收取同样的千分之三的交易手续费）。

2.4 协议手续费(Protocol fee)

Uniswap v2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Uniswap v2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（译者注：k是常数乘积，可以看v1白皮书）的增长来计算（也就是 ）。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：

$$f_{1,2} = 1 - \frac{\sqrt{k_1}}{\sqrt{k_2}} \tag{4}$$

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$\phi \cdot f_{1,2}$，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：

$$\frac{s_m}{s_m + s_1} = \phi \cdot f_{1,2} \tag{5}$$

经过变换，将$f_{1,2}$替换为后，解得s_m

$$s_m = \frac{\sqrt{k_2} - \sqrt{k_1}}{(\frac{1}{\phi} - 1) \cdot \sqrt{k_2} + \sqrt{k_1}} \cdot s_1 \tag{6}$$

设$\phi = \frac{1}{6}$，得到以下公式

$$s_m = \frac{\sqrt{k_2} - \sqrt{k_1}}{5 \cdot \sqrt{k_2} + \sqrt{k_1}} \cdot s_1 \tag{7}$$

假设初始存款人存了100DAI和1ETH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96DAI和1.5ETH，用上面得公式可以得出：

$$s_m = \frac{\sqrt{1.5 \cdot 96} - \sqrt{1 \cdot 100}}{5 \cdot \sqrt{1.5 \cdot 96} + \sqrt{1 \cdot 100}} \cdot 10 \approx 0.0286 \tag{8}$$

2.5 资金池份额的元交易(Meta transactions for pool shares)

Uniswap v2交易对铸造得资金池份额原生支持元转账。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

Uniswap v1是用Vyper实现的，一种类似于Python的智能合约预言。Uniswap v2是用更广泛使用的Solidity实现的，因为它在开发的时候需要一些Vyper中不可用的功能（比如翻译非标准ERC20代币的返回值，通过内联汇编访问新的字节码chanid）。

3.2 合约重新架构

Uniswap v2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，Uniswap v2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Uniswap v1合约。

Uniswap v2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 手续费调整

Uniswap v1通过转入合约的代币数量，在保持常数乘积不变之前收取交易手续费。合约强制确保了以下公式：

$$(x_1 - 0.003 \cdot x_{in}) \cdot y_1 >= x_0 \cdot y_0 \tag{9}$$

使用flash swaps时，Uniswap v2引入了xin和yin可以同时为非零的可能性（当用户想要返还同样的资产，而不是互换时）。为了处理这种情况，同时正确地收取手续费，合约强制确保：

$$(x_1 - 0.003 \cdot x_{in}) \cdot (y_1 - 0.003 \cdot y_{in}) >= x_0 \cdot y_0 \tag{10}$$

为了简化链上计算，两边同时乘以1000000，得到：

$$(1000 \cdot x_1 - 3 \cdot x_{in}) \cdot (1000 \cdot y_1 - 3 \cdot y_{in}) >= 1000000 \cdot x_0 \cdot y_0 \tag{11}$$

3.2.2 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2^{112}的代币，Uniswap v2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$2^{112} - 1$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，比如泰达币（USDT）和币安币（BNB）。Uniswap v1将这种不标准的函数返回值转换成false，表示转账不成功，并且回滚交易，导致转账失败。

Uniswap v2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，Uniswap v2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）。

Uniswap v1此外假设调用transfer()和transferFrom()不能触发Uniswap交易对合约的重入调用。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，Uniswap v2包含了一个“锁”，直接防止重入所有公共的修改状态的函数。这也保护防止了在快速互换（flash swaps）中从用户定义的回调函数重入，如2.3节所描述的那样。

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Uniswap交易对中存代币时，计算铸造的流动性代币（译者注：流动性代币需要看Uniswap v1白皮书）数量基于现有的代币数量：

$$s_{minted} = \frac{x_{deposited}}{x_{starting}} \cdot s_{starting} \tag{12}$$

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。

Uniswap v1设初始份额供给等于存入地ETH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ETH一样是18位小数精度代币）大约值2ETH。

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，Uniswap v2支持任意交易对，有许多交易对根本不包含ETH。

相反，Uniswap v2初始铸造份额等于存入代币数量的几何平均值：

$$s_{minted} = \sqrt{x_{deposited} \cdot y_{deposited}} \tag{13}$$

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$\sqrt{2 \cdot 200} = 20$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$\sqrt{2 \cdot 800} = 40$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，Uniswap v2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ETH

以太坊原生资产ETH的转账接口和ERC20交互用的标准接口不同。结果，以太坊上许多其他的协议不支持ETH，而是使用了一种标准的“包装的ETH”代币，WETH。

Uniswap v1是一个例外。因为每个Uniswap v1交易对包含了ETH作为其中一种资产，所以它可以直接处理ETH，而且更高效地使用gas。

因为Uniswap v2支持任意ERC20交易对，它现在不再支持无包装的ETH。如果添加这个特性需要两倍的核心代码，并且产生ETH和WETH流动性分散的风险。原生ETH需要包装后才能在Uniswap v2上交易。

3.6 确定交易对地址

和Uniswap v1一样，Uniswap v2交易对也是通过单一的工厂合约进行实例化的。在Uniswap v1中，交易对合约用CREATE运算码进行创建，这意味着合约地址依赖于交易对创建的顺序。Uniswap v2使用以太坊新的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，不用查看以太坊区块链的状态。

3.7 最大代币余额

为了高效地实现预言机制，Uniswap v2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$2^{112} - 1$，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。

引用

- [1] Hayden Adams. 2018. url: https://hackmd.io/@477aQ9OrQTCbVR3fq1Qzxg/HJ9jLsfTz?type=view.
- [2] Guillermo Angeris et al. An analysis of Uniswap markets. 2019. arXiv: 1911.03380[q-fin.TR].

- [3] samczsun. Taking undercollateralized loans for fun and for profit. Sept. 2019. url:https://samczsun.com/taking-undercollateralized-loans-for-fun-and-for-profit/.
- [4] Fabian Vogelsteller and Vitalik Buterin. Nov. 2015. url: https://eips.ethereum.org/EIPS/eip-20.
- [5] Jordi Baylina Jacques Dafflon and Thomas Shababi. EIP 777: ERC777 Token Standard.Nov. 2017. url: https://eips.ethereum.org/EIPS/eip-777.
- [6] Radar. WTF is WETH? url: https://weth.io/.
- [7] Uniswap.info. Wrapped Ether (WETH). url: https://uniswap.info/token/0xc02aaa39b223fe8d0a0e5c4f27e
- [8] Vitalik Buterin. EIP 1014

4 声明

这篇论文的目的仅作为一般信息。不构成买卖任何投资产品的投资建议或推荐或请求，也不应该被用来评估任何投资决策的业绩，不应该被用于会计、法律或税务建议或投资推荐。本文反映了作者当前的观点，不代表Paradigm和其附属机构，也不一定反映Paradigm、其附属机构以及隶属于Paradigm的个人的观点。本文反映的观点在没有另行更新的情况下可能发生变化。