

什么是DEX?

- 交易所：资产买卖金融市场，证券(股票)交易所、期货交易所、大宗商品交易所...
- DEX：Decentralize Exchange 去中心化交易所（Token 兑换 Token）
- 两类：订单簿（撮合）、兑换池
- 订单簿：以0x协议为代表，链下订单撮合和链上结算
- 兑换池：以 Uniswap 为代表，自动做市商协议

UNISWAP DEMO

Token 交易：<https://app.uniswap.org/#/swap>

提供流动性：<https://app.uniswap.org/#/add/v2/>

查看流动性：<https://app.uniswap.org/#/pools/v2>

Uniswap

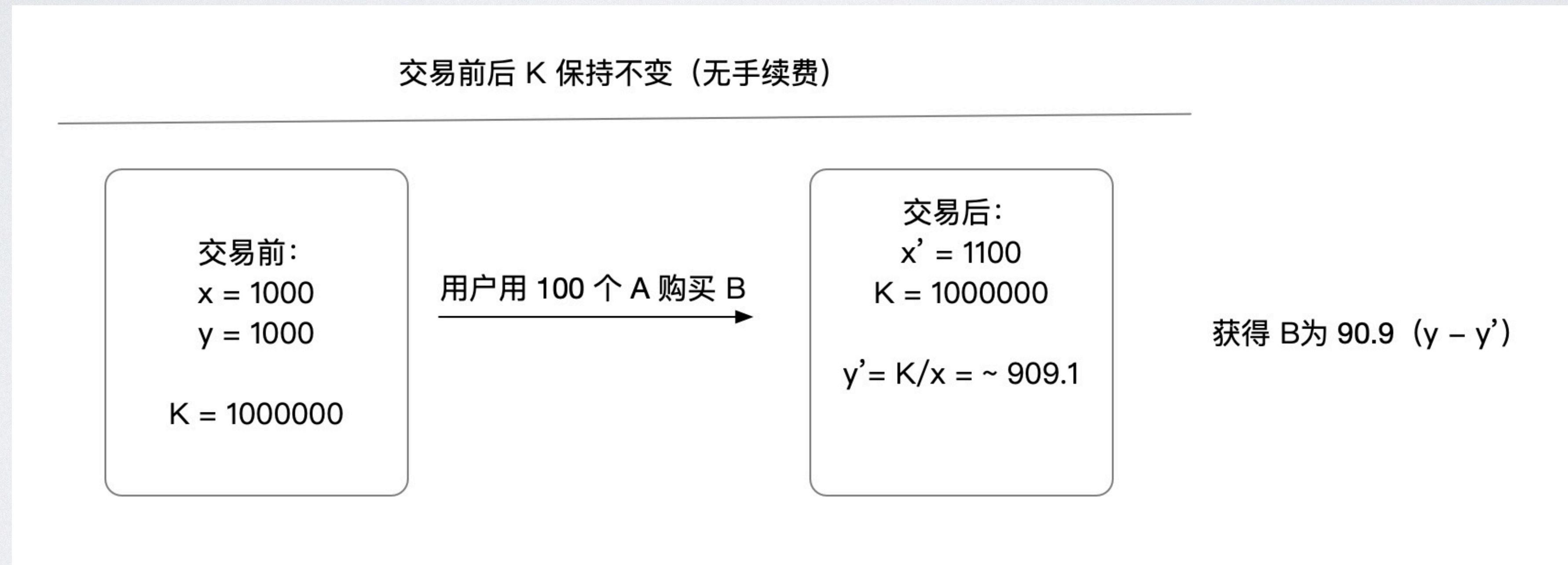
- AMM协议： AutoMated Market Making
 - AutoMate(d)：自动，没有中间机构进行资金交易
 - Market Making: 做市商（保证订单得以执行），流动性提供者（LP: liquidity providers）
 - 流动性指的是如何快速和无缝地购买或出售一项资产
 - LP 是提供资产的人以实现快速交易。
 - 常量乘积模型： $K = x * y$

Uniswap

- 常量乘积模型： $K = x * y$
 - AMM 的执行引擎，没有价格预言机，价格用公式推导
 - x : token0 的储备量 (reserve0)
 - y : token1 的储备量 (reserve1)
 - 提供流动性：
 - 转入token0、token1，增加reserve0、reserve1，拿到流动性凭证 $= \sqrt{x * y}$
 - 兑换时， K 保持不变
 - 减少reserve0，就必须增加reserve1
 - 减少reserve1，就必须增加reserve0
 - 移除流动性
 - 通过流动性凭证，撤出token0、token1

Uniswap

- 常量乘积模型： $K = x * y$



价格滑点 (slippage)：一次交易使价格改变的程度，单笔交易量越大对价格的影响越大

Uniswap

交易前后 K 保持不变（扣除0.3%费）

交易前：

$$x = 1000$$

$$y = 1000$$

$$K = 1000000$$

用户用 100 个 A 购买 B
用于交易的A 为 99.7

交易后：

$$x' = (1000 + 99.7) + 0.3$$

$$y' = K / (1000 + 99.7) = 909.33$$

$$K = 1000000$$

$$K' = 1000263$$

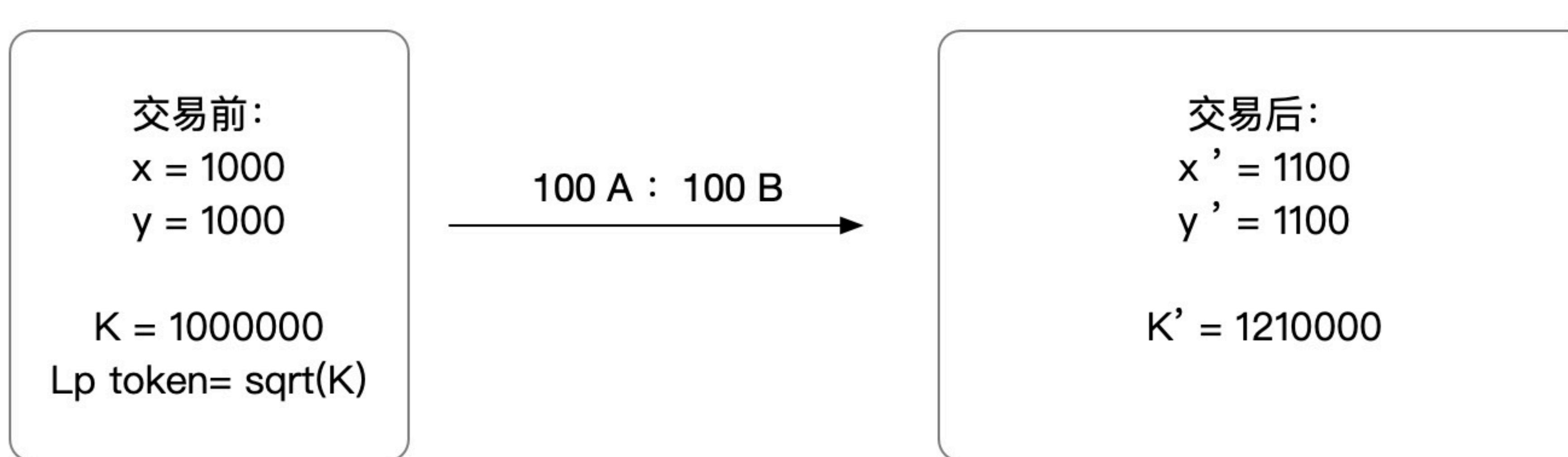
获得 B: 90.67 ($y - y'$)

$$x \cdot y = (x + \Delta x)(y - \Delta y) = k$$

$$\Delta y = y - \frac{x \cdot y}{x + \Delta x} = \frac{\Delta xy}{x + \Delta x}$$

Uniswap

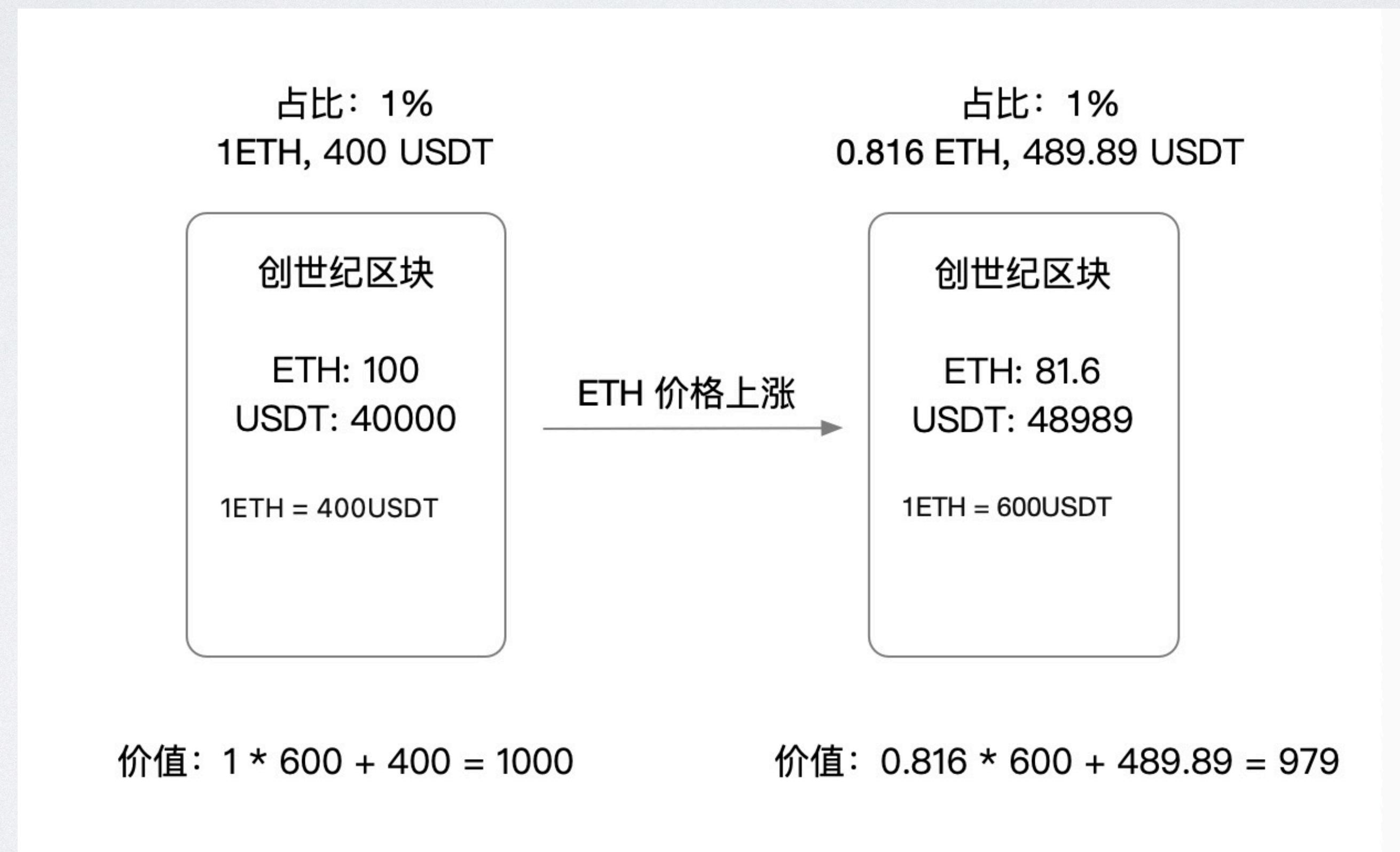
添加流动性 增加 K



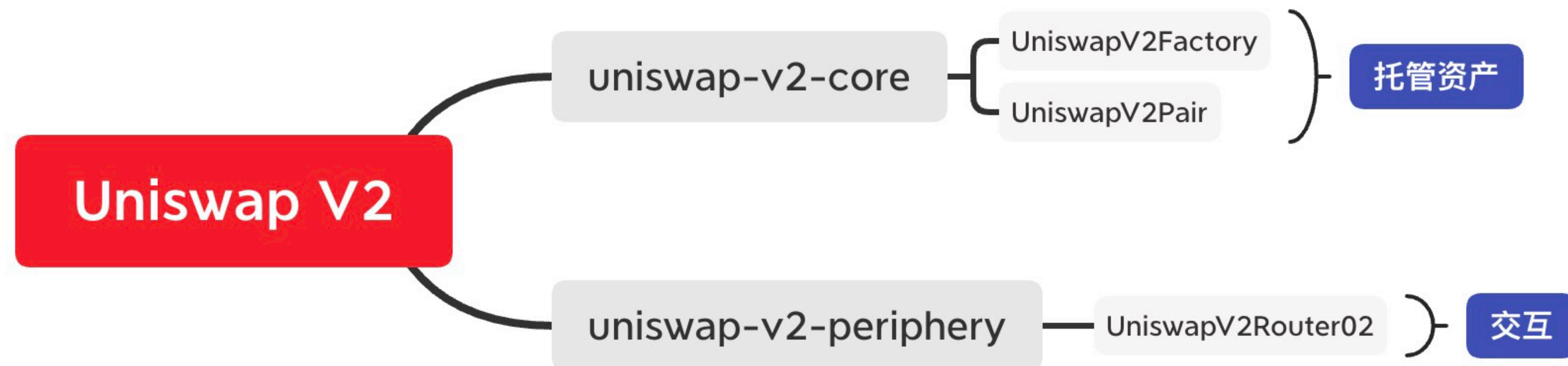
获得 lp token: 100

Uniswap – 无常损失

- 流动性提供者无常损失：一对代币存入Uniswap后，如果一种代币以另一种进行计价的价格上升，在价格上升后取出，总价格比原价值低一些，低的部分就是损失。
- 思考：为什么会造成损失？



Uniswap – 代码分析



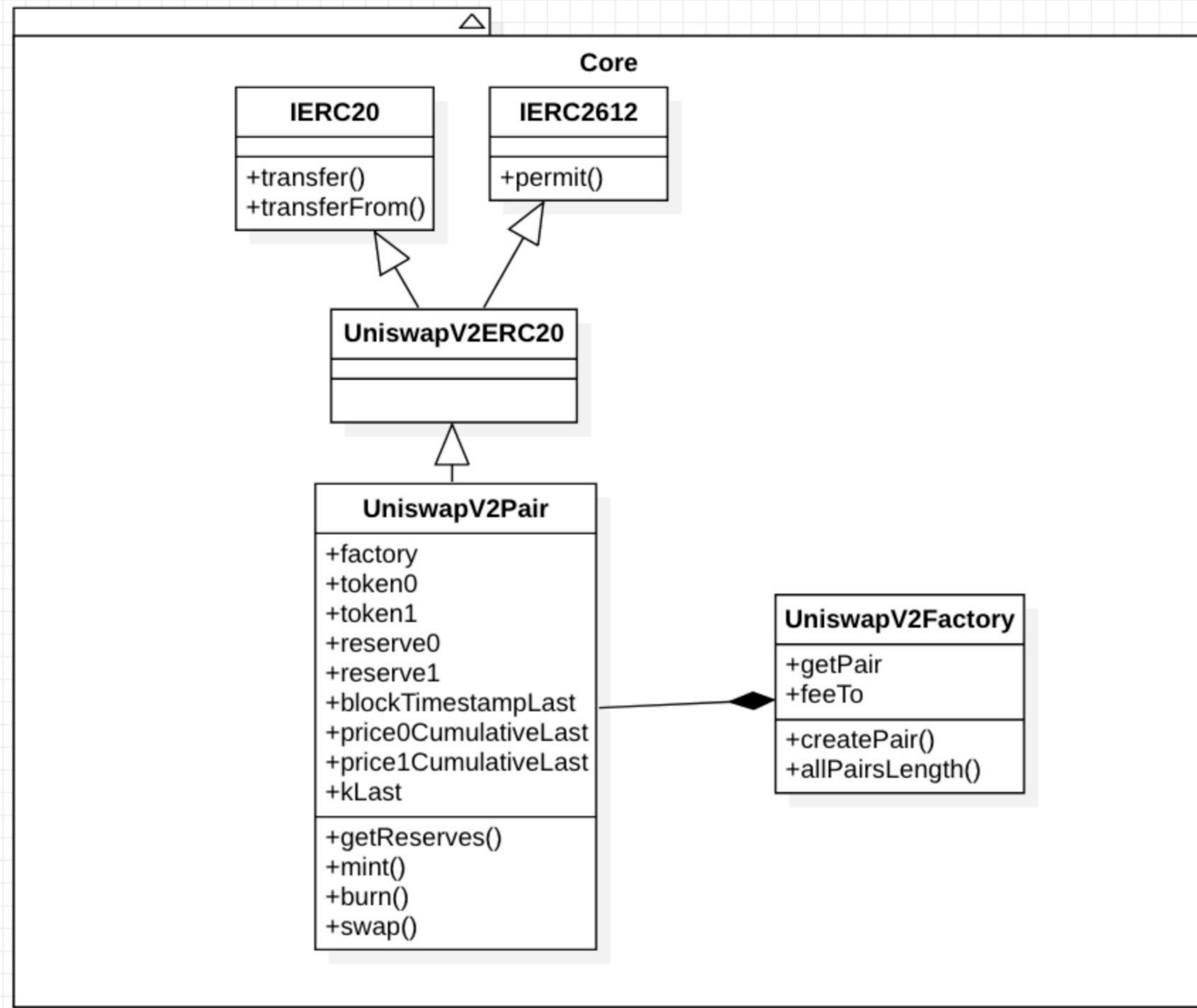
<https://github.com/Uniswap/v2-core>
<https://github.com/xilibi2003/v2-core>

<https://github.com/Uniswap/v2-periphery>
<https://github.com/xilibi2003/v2-periphery>

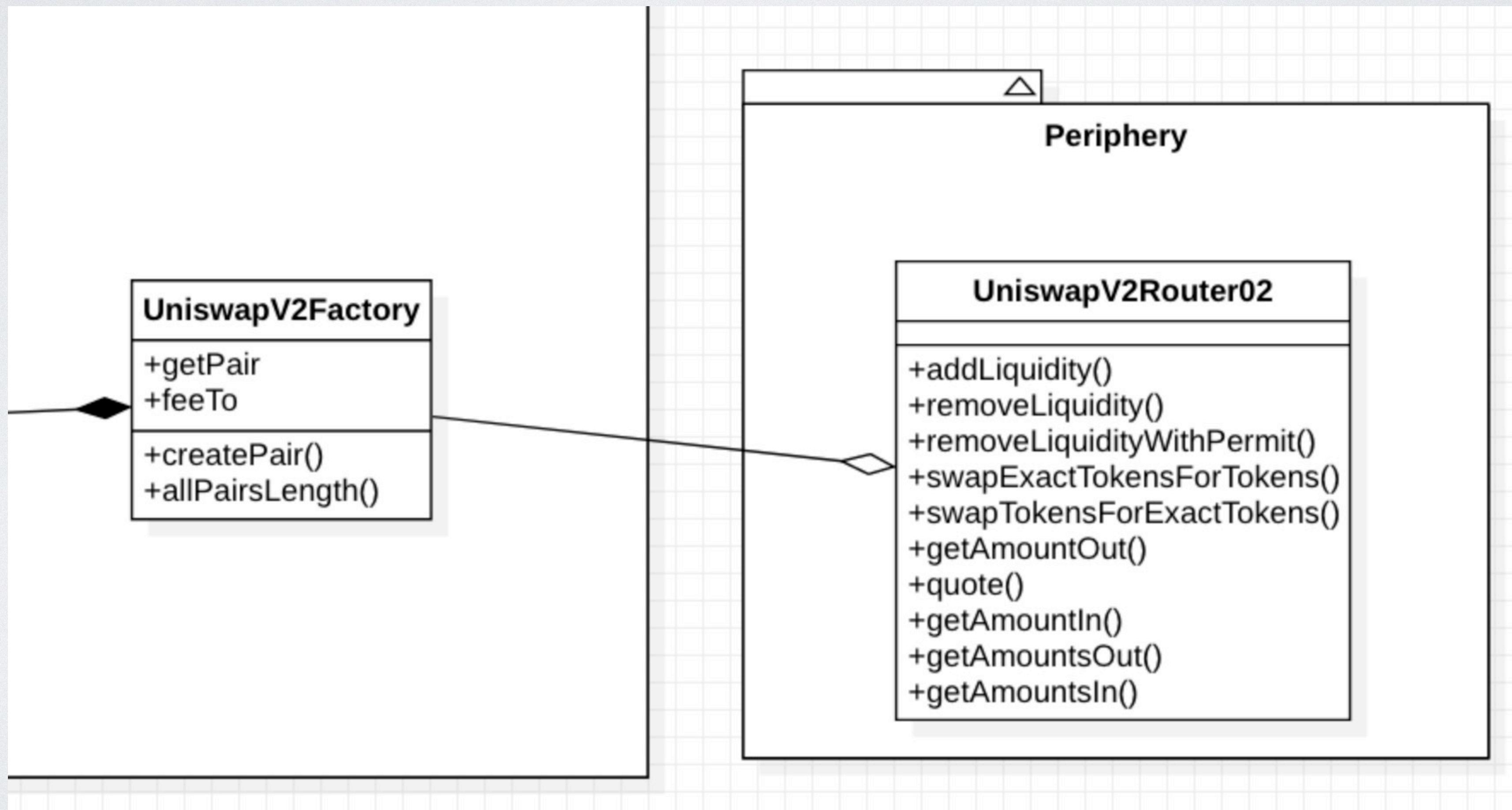
WETH

- WETH 是 Ether 的ERC20包装(Wrap-ETH)
- | WETH == | ETH
- WETH 合约 (WETH9)
 - <https://etherscan.io/token/0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2#code>
 - deposit()
 - withdraw()

第5周



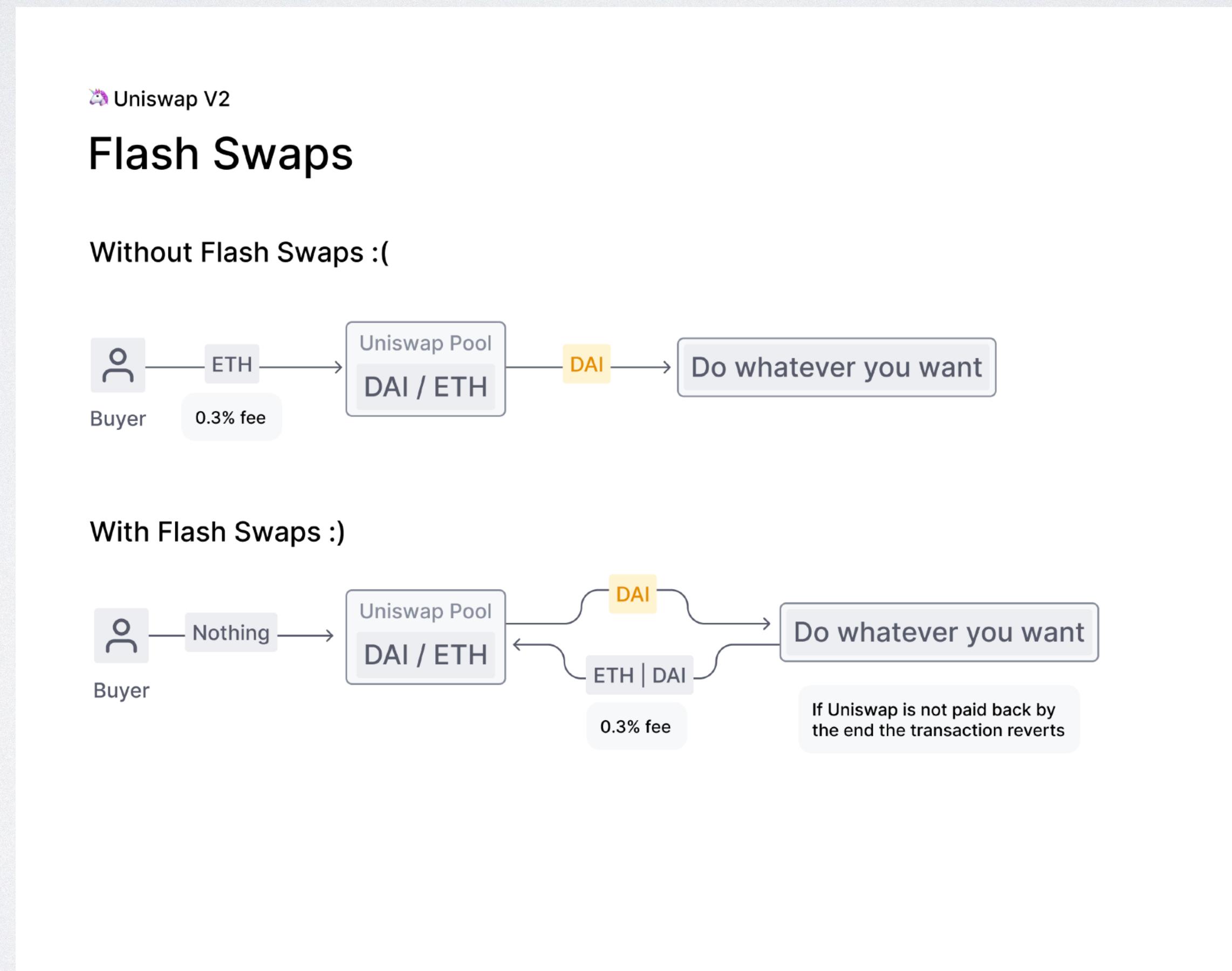
Uniswap – 合约结构



Uniswap 集成使用

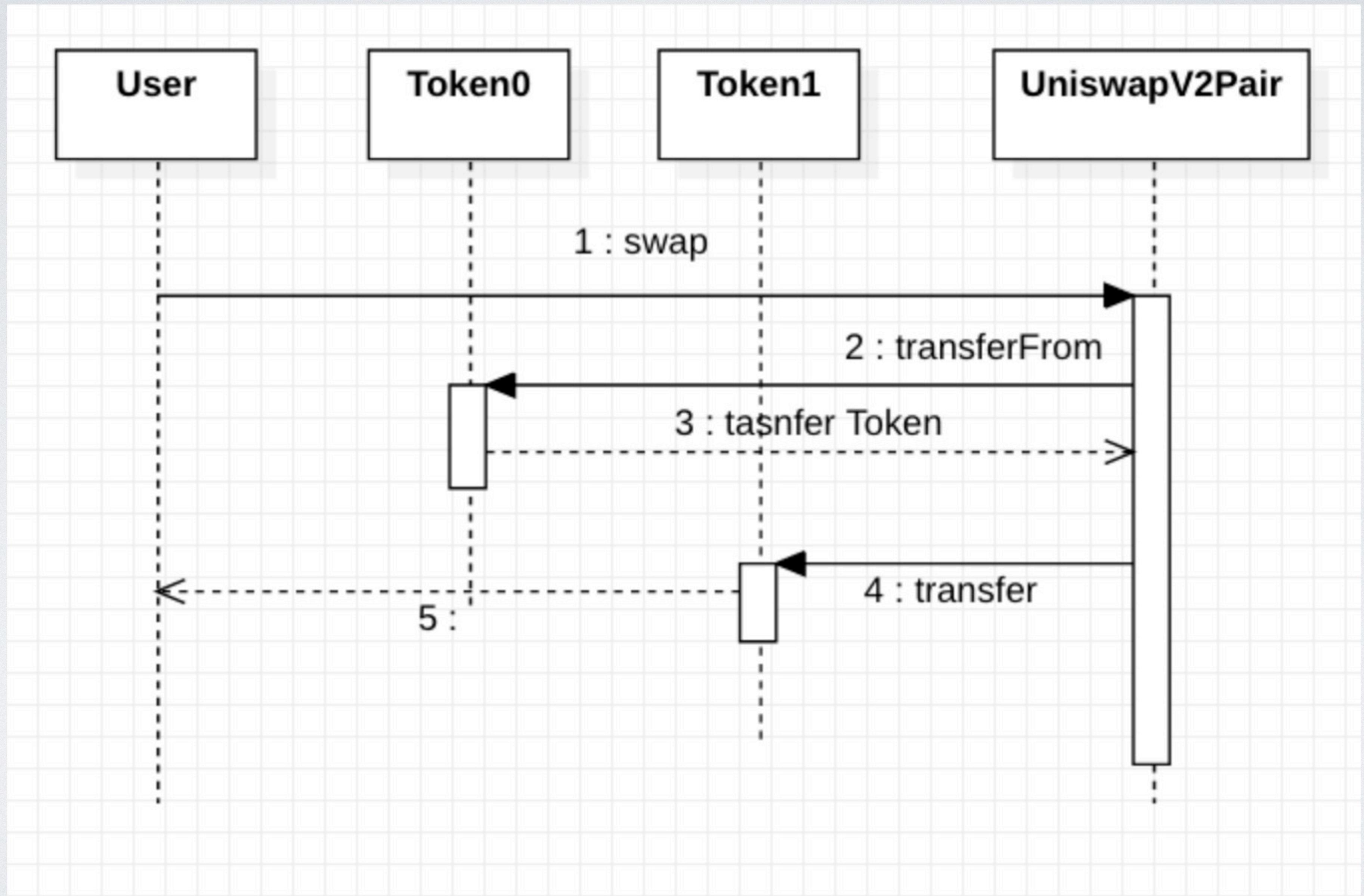
- 交易平台（通过 Router 与其他币进行交易）
- 价格预言机（获取时间加权平均价格）
- 闪电兑（flash swap）

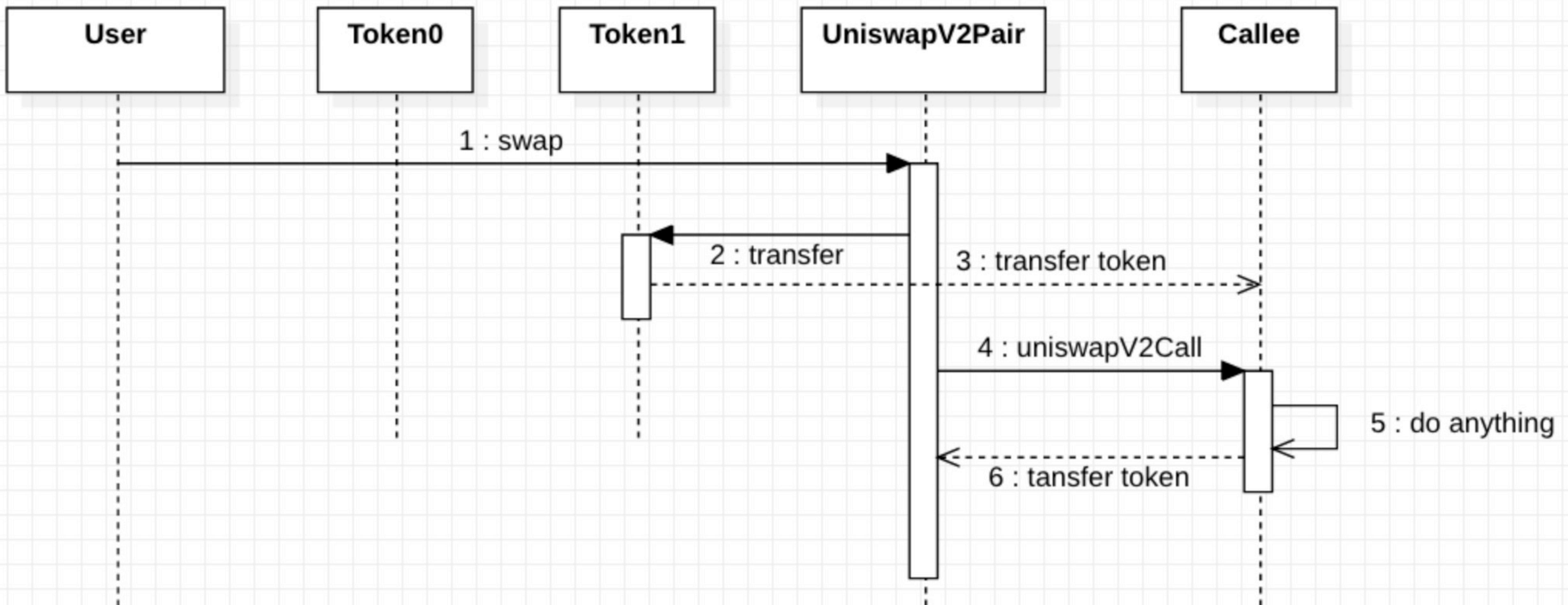
Uniswap 闪电兑



第5周

普通兑换

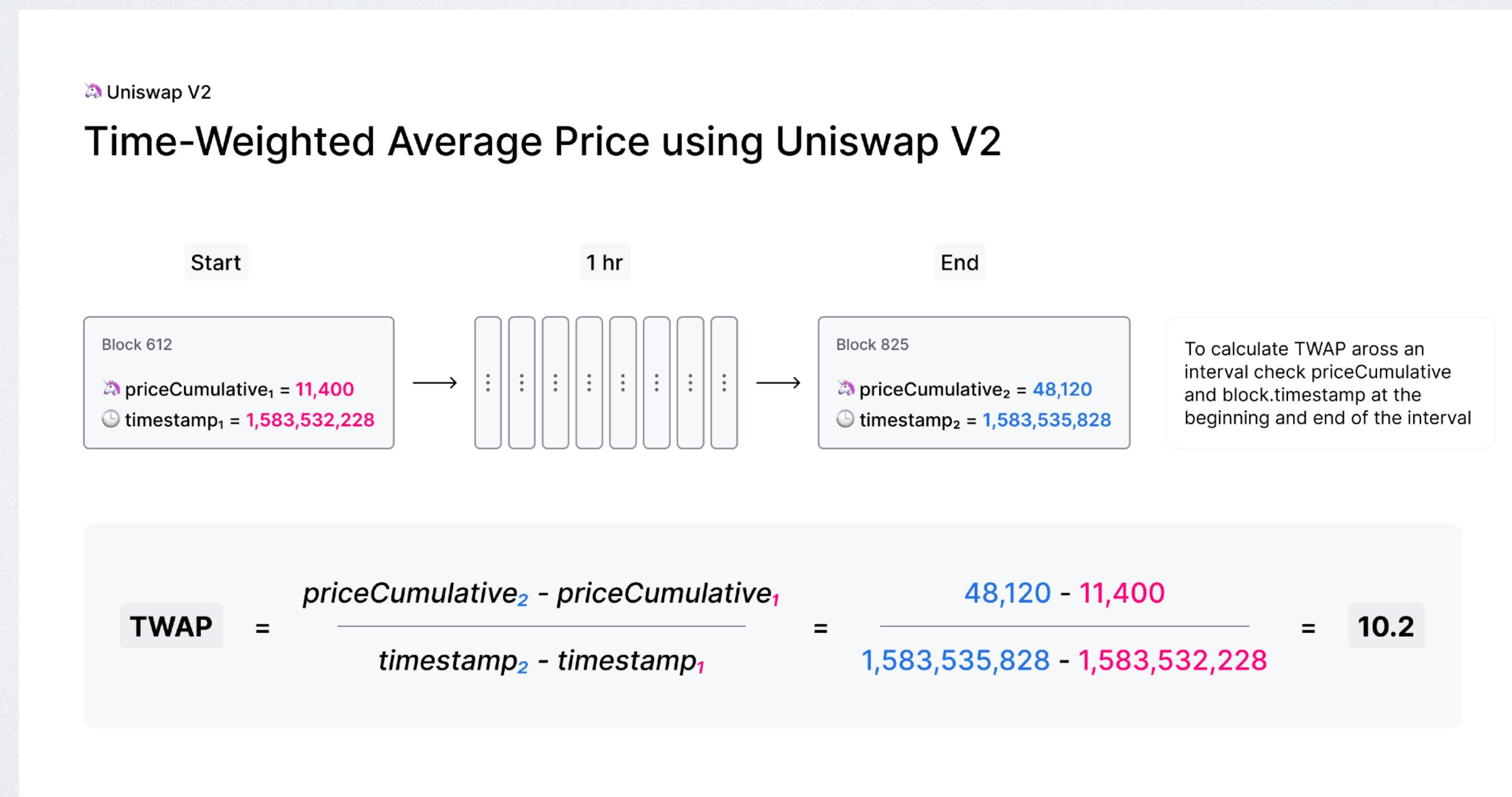




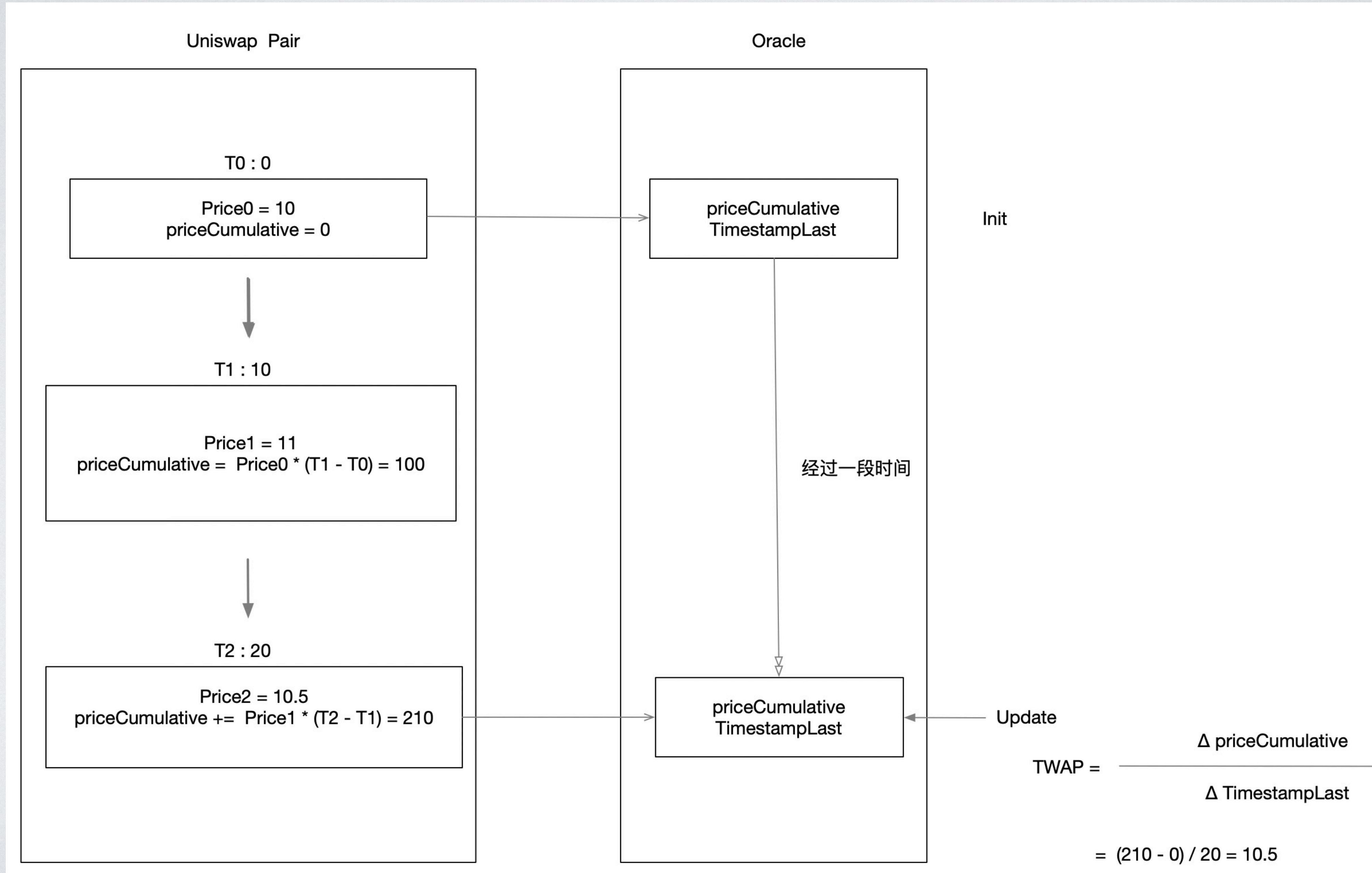
闪电兑换

Uniswap TWAP 价格

- TWAP = Time-Weighted Average Price, 即时间加权平均价格
- `price0CumulativeLast`、`price1CumulativeLast` 记录了 token 的累加价格
 - `priceCumulativeLast1 = priceCumulativeLast1 + price2 * timeElapsed;`



第5周



Q & A

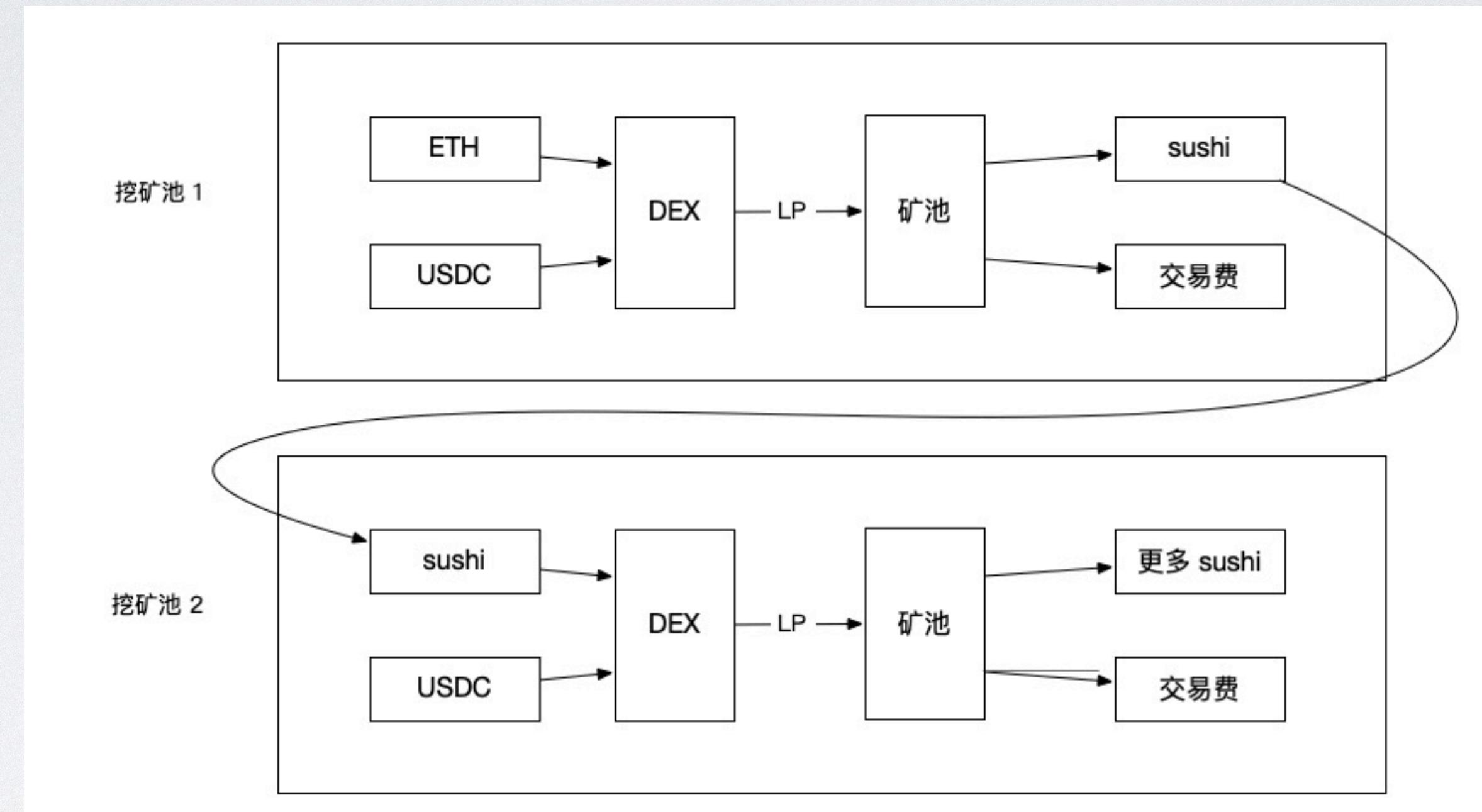
练习题

- 部署自己的 ERC20 合约 MyToken
- 编写合约 MyTokenMarket 实现：
 - AddLiquidity(): 函数内部调用 Uniswap V2Router 添加 MyToken 与 ETH 的流动性
 - buyToken(): 用户可调用该函数实现购买 MyToken

SushiSwap

- Uniswap:
- 提供流动性时才赚取资金池的交易费，撤回流动性不再获得相应的收入
- SushiSwap = Uniswap + 流动性挖矿
 - 流动性挖矿：
 - 1. 为流动性提供者奖励
 - 2. 锁定流动性、弥补流动性损失
 - 公平发行协议币（sushi）：去中心化治理、sushi 代币可持续给参与者带来协议收益

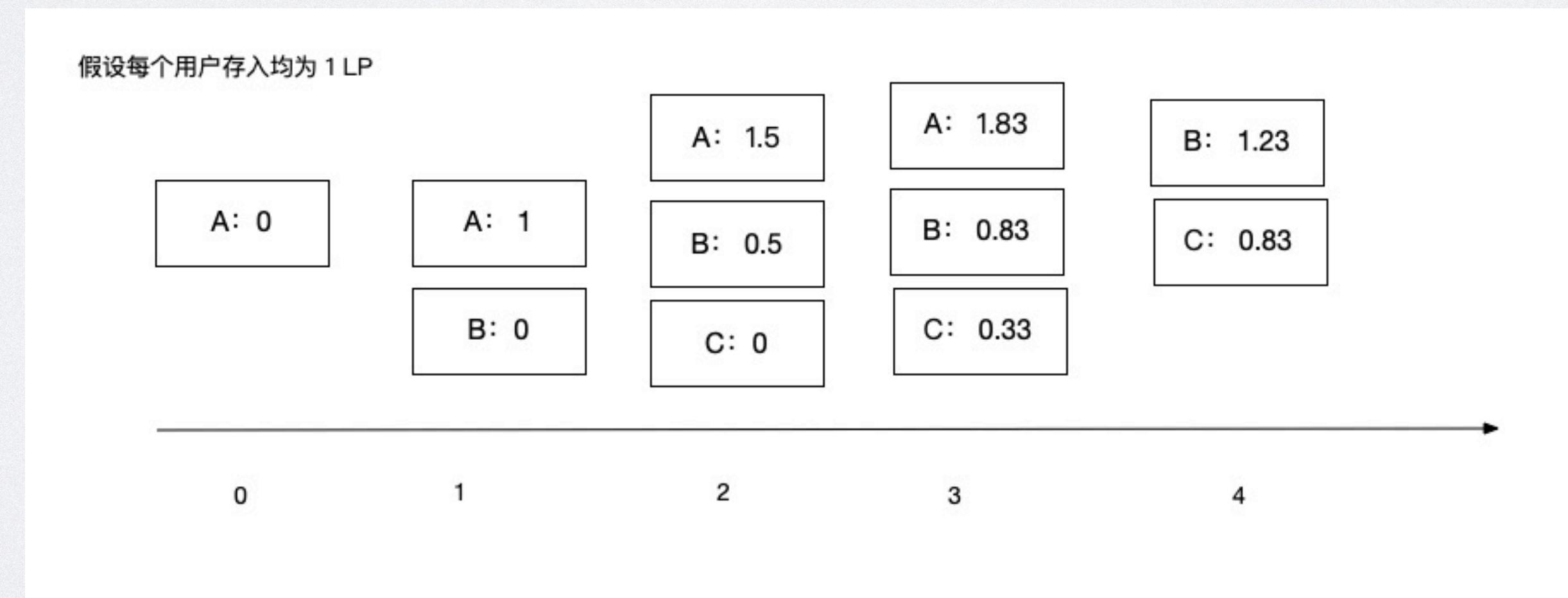
流动性挖矿



- 正向：有效的激励工具，尤其是更高激励的池2，促使人们购买sushi，推高sushi价格，同时提高了挖矿池的收益。促进更多人购买sushi
- 反向：随着挖出的SUSHI越来越多，矿工卖SUSHI，某个时间点，卖出的SUSHI比买入的SUSHI多，SUSHI价格开始下降，收益下跌、抛售、进一步下跌，“矿塌了”。

SushiSwap 挖矿算法

- 设置每个区块可以奖励 sushi Token 数量
- 根据投入的 LP 数量的比例，获取对应的 sushi
- 怎么在有限 gas 下实现？常规方案：



SushiSwap 挖矿算法

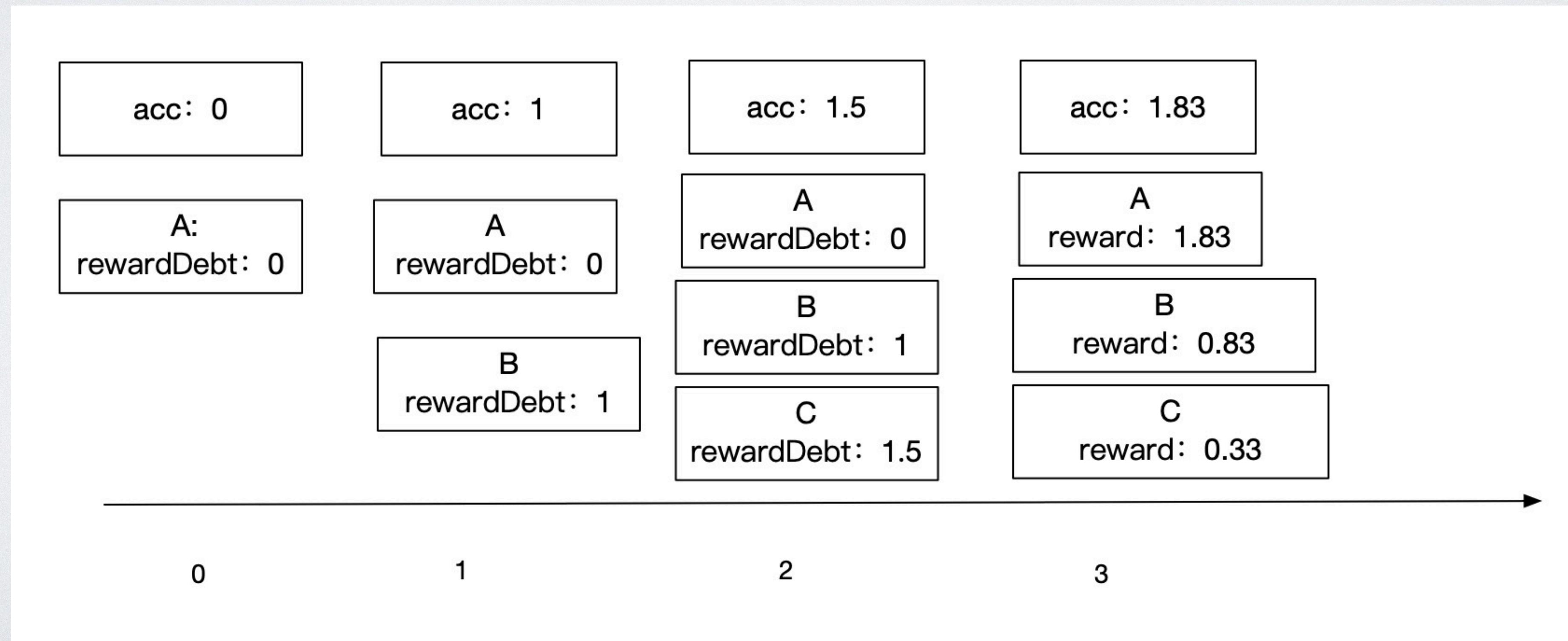
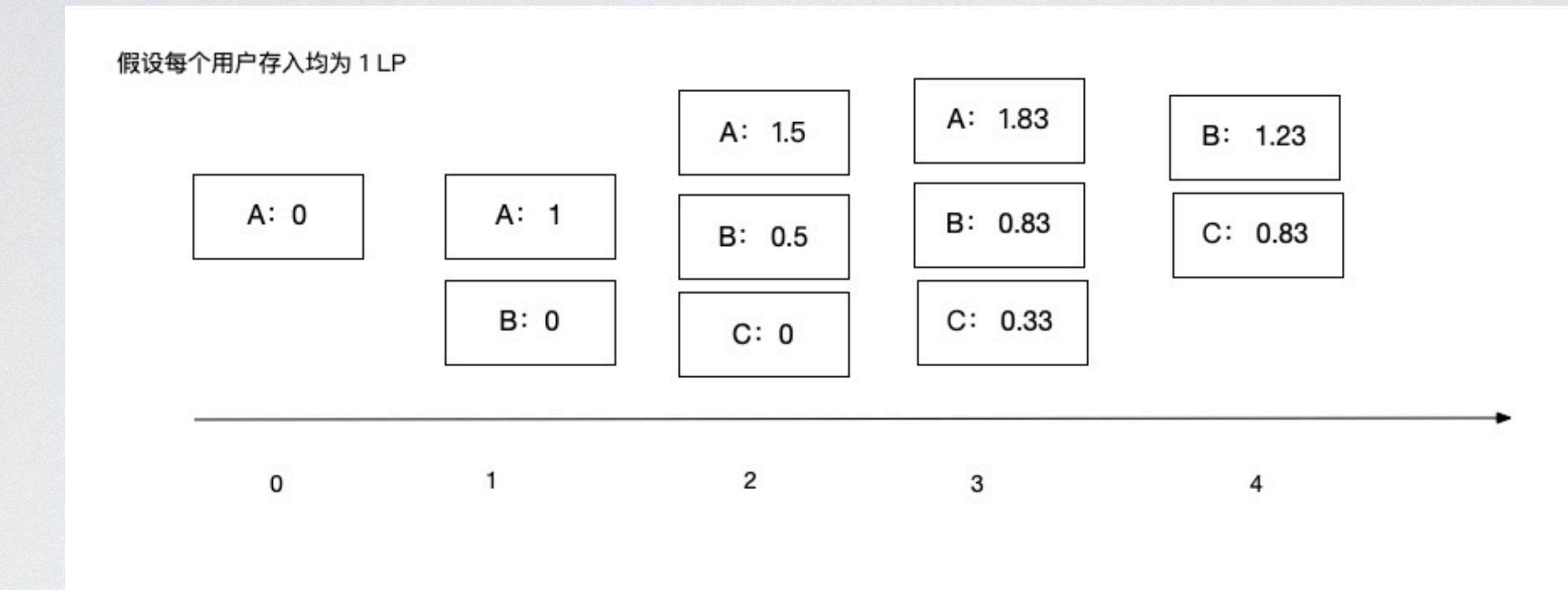
- 全局变量： $accSushiPerShare$, 每个份额可以拿到的**累计奖励**
- 记录进入时刻已奖励部分： $rewardDebt$
- 奖励为： $accSushiPerShare * amount - rewardDebt$

$$accSushiPerShare = R \cdot \sum_{t=0}^t \frac{1}{L(t)}$$

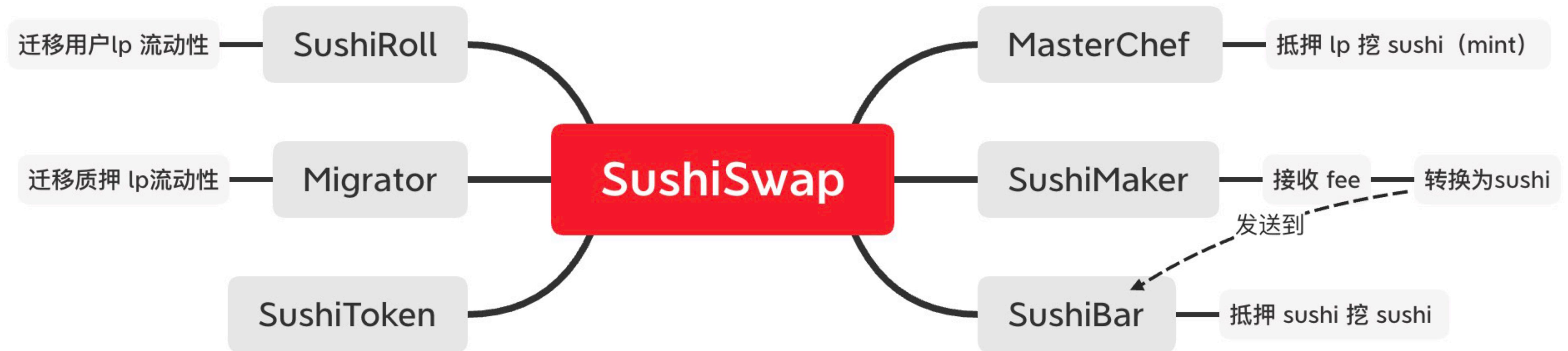
$$rewardDebt = l \cdot R \cdot \sum_{t=0}^{t_0} \frac{1}{L(t)}$$

$$Reward_{alice} = l \cdot (R \cdot \sum_{t=0}^t \frac{1}{L(t)} - R \cdot \sum_{t=0}^{t_0} \frac{1}{L(t)})$$

第5周



Sushiswap – 代码分析



<https://github.com/sushiswap/sushiswap>

Sushiswap – 代码分析

- MasterChef: 流动性挖矿
 - deposit(): 存 LP
 - Withdraw(): 取 LP
 - pendingSushi(): 查看收益
- SushiMaker: LP代币(DEX 交易手续费) 转 sushi
 - convert()
- SushiBar: 质押 sushi, -> xSushi, -> 更多 sushi
 - enter: 质押
 - leave(): 取sushi

<https://github.com/sushiswap/sushiswap>

Q & A

练习题

- 在上一次作业的基础上：
 - 完成代币兑换后，直接质押 MasterChef
 - withdraw()：从 MasterChef 提取 Token 方法