SaucerSwap V2 Whitepaper

Concentrated Liquidity Market Maker on the Hedera Network

Matthew DeLorenzo Vae Vecturne

October 25, 2023

Abstract

SaucerSwap V2 is a decentralized exchange protocol on the Hedera network, enabling concentrated liquidity provision for increased capital efficiency compared to SaucerSwap V1. Hedera's architecture separating cryptocurrency tokens and smart contracts presented unique challenges and opportunities described herein.

1 Introduction

SaucerSwap V2 is a set of smart contracts that create an automated market maker (AMM), a protocol that facilitates peer-to-peer market making and swapping of cryptocurrency tokens on the Hedera network. It is a modified fork of Uniswap V3 that concentrates liquidity by "bounding" it within arbitrary price ranges [1], enabling increased capital efficiency compared to SaucerSwap V1.

On the Hedera network, cryptocurrency tokens exist independently from smart contracts on the Hedera Token Service (HTS). [2] Smart contracts are stored and interacted with via the Hedera Smart Contract Service (HSCS). [3] The Hedera mainnet launch of Smart Contracts 2.0 enabled ERC-20 and ERC-721 standard token operations on HTS tokens by smart contracts on HSCS, resulting in a growing decentralized finance (DeFi) ecosystem on Hedera. [4]

2 Modifications to Uniswap V3

The SaucerSwap V2 fork of Uniswap V3 core and periphery repositories were left largely unchanged to retain composability and familiarity for DeFi developers. [5] The few modifications made focused on compatibility with the Hedera ecosystem (HTS and HSCS). Due to the low cost and high throughput offerings of Hedera, attention was paid to mitigate arbitrary contract state expansion and contract creation. [6]

2.1 Pool Creation Fees and Mint Fees

The Hedera network plans to charge rent fees on smart contracts [7], payable in the native token hbar, as a function of the number of key/value pairs of storage in smart contracts. Smart contract developers may define an `autoRenewAccountId` for one or more smart contracts, to be periodically debited by the network to pay their rent costs. SaucerSwap uses one `autoRenewAccountId` (denoted `rentPayer`) to pay for all smart contracts in the SaucerSwap ecosystem. SaucerSwap V2 discourages ordinarily inexpensive contract creation and contract state expansion by charging fees to create new liquidity pool contracts and to mint liquidity positions in pool contracts.

A pool creation fee is defined in terms of US dollars and assessed in hbar in the `createPool` function in `UniswapV3Factory.sol`, using an hbar exchange rate precompile contract. A storage variable `poolCreateFee` is set by the factory owner account to raise or lower the required fee in units of tinycents (1 cent USD = 1e8 tinycents). The precompile contract `address(0x168)` is called to convert `msg.value` in tinybars to tinycents at the current fair market value, and must be greater than or equal to `poolCreateFee` to create a new pool.

A position mint fee is assessed similarly in the `mint` function in `UniswapV3Pool.sol`. SaucerSwap V2 charges a fee to mint positions because it expands smart contract state in the pool contract by adding a storage struct to a mapping with the minted position's data. Like the pool creation fee, the position mint fee is a storage variable in `UniswapV3Factory.sol`, in units of tinycents. The mint fee is paid to the pool contract and sent to the `rentPayer` by the factory owner by calling `collectProtocol`.

2.2 Non-Fungible Tokens as Liquidity Positions

SaucerSwap V2 uses HTS non-fungible tokens (NFTs) as an option for representing liquidity positions. Instead of a liquidity provider keeping track of a data struct to manage a position in a pool contract, a smart contract, `NonfungiblePositionManager.sol`, mints the position for the liquidity provider and issues an NFT. The NFT functions as a proof of ownership of the underlying position and is used to manage the position size, collect swap fees, etc. This is advantageous because it consolidates position struct information in a central location and allows liquidity positions to change owner via a `HAPI Crypto Transfer`.

The contract `NonfungiblePositionManager.sol` is set as the NFT's supply key and treasury key, which allows it to control the NFT's minting and burning on HTS. The NFT's metadata consists of a SaucerSwap Labs-owned base URL, appended with the 7 byte hex-encoded serial number of the NFT. An event is emitted when a new NFT serial number is minted, prompting the creation of an image file by SaucerSwap Labs. The image file contains artwork with overlaid position information, hosted at the full URL of the NFT's metadata field.

`NonfungiblePositionManager.sol` only utilizes the NFT serial number for its `positions` mapping. In the extreme example of SaucerSwap Labs losing control over the rights or functionality of the URL, liquidity position NFTs would still be redeemable by the owner or approved spender of the NFT.

2.3 Token Association

Hedera discourages token balance mapping expansion by requiring account and contract entities be associated to tokens before taking custody. `UniswapV3Pool.sol` associates both reserve tokens to itself in the constructor. The abstract contract `PeripheryImmutableState.sol`, inherited by most periphery contracts, associates wrapped hbar (WHBAR) in its constructor. Since these can be handled by the contracts themselves, it prevents the need for external input to associate tokens.

`SwapRouter.sol` custodies intermediate tokens for multi-hop routes, and `QuoterV2.sol` custodies tokens in simulated swap transactions; both contracts require an external call to associate tokens. These contracts inherit `Ownable.sol` [9], allowing `owner` account to associate tokens to each contract. Token association is the only functional role granted to `owner` for `SwapRouter.sol` and `QuoterV2.sol`.

References

[1] Hayden Adams et al, Uniswap V3 Core White Paper. https://uniswap.org/whitepaper-v3.pdf

[2] Hedera Hashgraph LLC, Tokenization on Hedera. https://hedera.com/hh_tokenization-whitepaper_v2_20210101.pdf

[3] Hedera Hashgraph LLC, Hedera Documentation. https://docs.hedera.com/hedera/

[4] Gehrig Kunz, Smart Contracts 2.0: Live on Mainnet. https://hedera.com/blog/smart-contracts-2-0-live-on-mainnet

[5] Uniswap Labs GitHub Repository. https://github.com/Uniswap

[6] Leemon Baird and Atul Luykx, The Hashgraph Protocol: Efficient Asynchronous BFT for High-Throughput Distributed Ledgers. https://hedera.com/hh-ieee_coins_paper-200516.pdf

[7] Hedera Hashgraph LLC, Smart Contract Rent on Hedera. https://docs.hedera.com/hedera/core-concepts/smart-contracts/smart-contract-rent

[8] Uniswap Labs, Getting a Quote. https://docs.uniswap.org/sdk/v3/guides/quoting

[9] OpenZeppelin GitHub Repository. https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol

DISCLAIMER