Neon EVM

Ethereum Smart Contracts Scaled by Solana
(version 1.5, May 18, 2023)

Neon EVM is an Ethereum Virtual Machine on Solana that allows developers to scale Ethereum dApps by using Solana as the settlement layer.

Abstract

Ethereum is the dominant blockchain protocol that supports smart contracts. However, transaction speed and the cost of transactions can constrain the performance and potentiality of dApps. While Solana is one of the most technically advanced blockchains, offering low gas fees and high throughput of transactions due to its technological innovations.

This paper introduces Neon EVM, a protocol that allows Ethereum-like transactions to be processed on Solana according to Ethereum rules. In addition to providing access to the growing Solana market, Neon EVM allows Ethereum dApps to take full advantage of the functionality native to Solana, including parallel execution of transactions. As such, Neon EVM allows dApps to operate with the low gas fees, high transaction speed, and high throughput of Solana.

Neon EVM is built as a smart contract on Solana. A transaction request is sent to an intermediary proxy server, Neon Proxy, which wraps Ethereum-like transactions into Solana transactions and sends them to Neon EVM for parallel execution by Solana.

To enable the parallel execution of smart contracts, Neon EVM implements several strategies. For example, each contract keeps its data in its own Solana storage, and account balances used to pay for Neon transactions are also separated.

This solution allows any Ethereum application to be run on Solana with minimal reconfiguration of the codebase; this applies to Uniswap, AAVE, Curve, Saddle Finance, etc. Similarly, all the key tools for Ethereum dApps can work on Solana via Neon EVM, including Solidity, MetaMask, Remix, and Truffle. Furthermore, Neon EVM can be updated easily when new Solana or Ethereum functionality is implemented.

Neon EVM provides a unique opportunity to developers who want to unlock access to the liquidity on Solana, enjoy a first-mover advantage and reach new customers on Solana, or who want to scale with the low gas fees and high throughput that Solana provides.

Introduction

Blockchain technology offers superior decentralized asset preservation systems, ledger immutability, and permissionless transparency. In response to these utilities, the Layer 1 (L1) blockchain landscape has supported the emergence of two major players: Ethereum and Solana.

Ethereum has established itself as the dominant blockchain protocol to execute smart contracts. Arguably it offers the most advanced infrastructure for dApp developers and end users. However, Ethereum suffers from significant technical constraints (specifically with regard to its TPS and TTF: transactions per second and time to finality, respectively). Furthermore, transactions within Ethereum are also relatively expensive. To address this, a Layer 2 (L2) scaling ecosystem is emerging to take computation off the Ethereum network, share transaction costs, and instead use the L1 as a secure settlement layer — providing transparency over the outcome of transactions.

Solana also supports scripting, but in contrast to Ethereum, Solana was designed to support high-speed and low-cost transactions. This makes Solana a viable solution for high-throughput, low-value transactions while providing transparency and immutability. As of Q1 2023, Solana handles more transactions than any other L1 blockchain.

This bifurcation of the L1 landscape, while a natural outcome of a free marketplace and the blockchains' technical strengths, negatively impacts the builders in the blockchain space. dApps find themselves specialized and tied to one or other of the major blockchains, unable to access users on the neighboring ecosystem.

This paper presents a solution to these problems. Neon EVM protocol provides dApp builders an opportunity to scale existing Ethereum dApps by using Solana as the L1 settlement layer. Furthermore, this protocol enables the mobility of tokens traditionally tied to the native L1. For the first time, Ethereum native dApps have access to the user base on Solana.

Not only does Neon EVM break down the barriers between Ethereum and Solana, but based on initial bench tests, the Neon EVM is demonstrably the fastest EVM currently available. Furthermore, Neon EVM outcompetes existing Ethereum L2 scaling solutions in terms of both speed and cost.

Challenges addressed

Neon EVM protocol offers a solution to several challenges. Broadly, these can be divided into transactional challenges and ecosystem access challenges.

Transactional challenges

Ethereum remains the dominant blockchain protocol linked to smart contracts — providing the most sophisticated infrastructure for application development. However, Ethereum's throughput is limited, with a recorded maximum of 58 TPS, at the time of writing, and sidechains such as Polygon recording a maximum of 470 TPS.

The technical constraints faced by Ethereum are based, in part, on how the state is represented and updated. Within Ethereum, the state is represented by a Merkle-Patricia Trie that stores key-value data for all smart contracts. Such smart contracts, commonly written in Solidity, do not have separate references to shared data and contracts' code. This necessitates that smart contracts must be executed in sequence to ensure deterministic behavior. This limits throughput and explains why transaction speeds suffer when the network is busy. Furthermore, as the network gets busier, gas prices increase as transaction senders attempt to outbid each other. This can make transactions very slow and expensive, making Ethereum unviable for certain types of dApps.

An Ethereum-native solution to these problems is L2 scaling. L2 scaling technologies take transactions off the L1 Ethereum Mainnet and submit them in bundles. Thus far, the preferred family of L2 scaling solutions, Rollups, have effectively reduced gas fees. Reports claim that the reductions are up to 100x; more typically, the reductions are in the region of 10x.

In contrast, Solana is designed to support massive scaling of decentralized applications, with a theoretical maximum throughput of more than 50,000 TPS with 5184 TPS as its maximum recorded live on Mainnet. Among Solana's innovations is its Proof-of-Stake consensus system that's reinforced via a Proof-of-History protocol. Solana provides a transaction parallelization technology that optimizes resources and ensures that it can scale horizontally across CPUs and SSDs, and an optimized mempool system that speeds up throughput.

This allows Solana to outcompete both L1 and L2 solutions. For example, at the time of writing, the average transaction fee for sending ETH on the Ethereum network was ~$2. In the same period, the L2 ecosystem were able to offer between $0.5–0.1, i.e. 4–20x savings. By comparison, Solana's average transaction fees stand at ~0.00001 SOL ($0.0002) per transaction, i.e. 10,000x savings. Initial tests of Neon EVM demonstrate this in practice, as shown by load testing, and transaction cost tests (Tables 1–3).

Table 1: Dollar equivalent cost for a transfer of Neon

Transaction	Tokens	Cost in lamports	$ Equivalent
Transfer	NEON	15,000	0.0003165*

* Sol price of $21.10: Transfer cost = 15'000 * 21.1 / 1'000'000'000 = $0.0003165

Table 2: Dollar equivalent cost for "Uniswap V2" dApp

Action	Fee in NEON	Accounts	Transactions	$ Equivalent*
Token approval	0.142028	9	1	0.035507
Swap	0.00092994	18	1	0.000232485

*NEON price estimated at $0.25

Table 3: Dollar equivalent cost report for Aave

Transaction	Fee in NEON	Accounts	Transactions	$ Equivalent*
Token approve	0.141784	9	1	0.035446
Deposit to lending pool	0.468461	29	1	0.117115
Borrow from lending pool	0.308169	37	26	0.0770424
Repay	0.0032492	31	1	0.000812301
Flashloan	0.28264	27	1	0.07066
Withdraw	0.0032492	30	1	0.000812301
Liquidation	0.192379	51	53	0.0480947

*NEON price estimated at $0.25

Ecosystem access challenges

Due to the fact that Ethereum and Solana have different transactional structures by design, there has never been a meaningful solution that allows Ethereum dApp developers to take advantage of the transactional innovations offered by Solana.

This means that just as mobile app developers had to develop for both Android and iOS to access both markets, till now, Ethereum dApp developers have had to custom develop to access another L1 market they wanted to target. Due to having specialized in the popular programming language Solidity, and becoming familiar with Ethereum tooling, many dApp developers don't have the ability to adapt their dApps to comply with Solana's different technical and transactional requirements. Neon EVM protocol changes this.

In the mobile app space, Xamarin, Flutter, and other tools provide cross-platform mobile app development. Similarly, Neon EVM brings dApp developers a robust cross-L1 blockchain development solution: allowing them to bring their dApps from Ethereum to Solana with minimal reconfiguration of the codebase.

This opens up Solana's user base to Ethereum dApps.

Neon EVM architecture

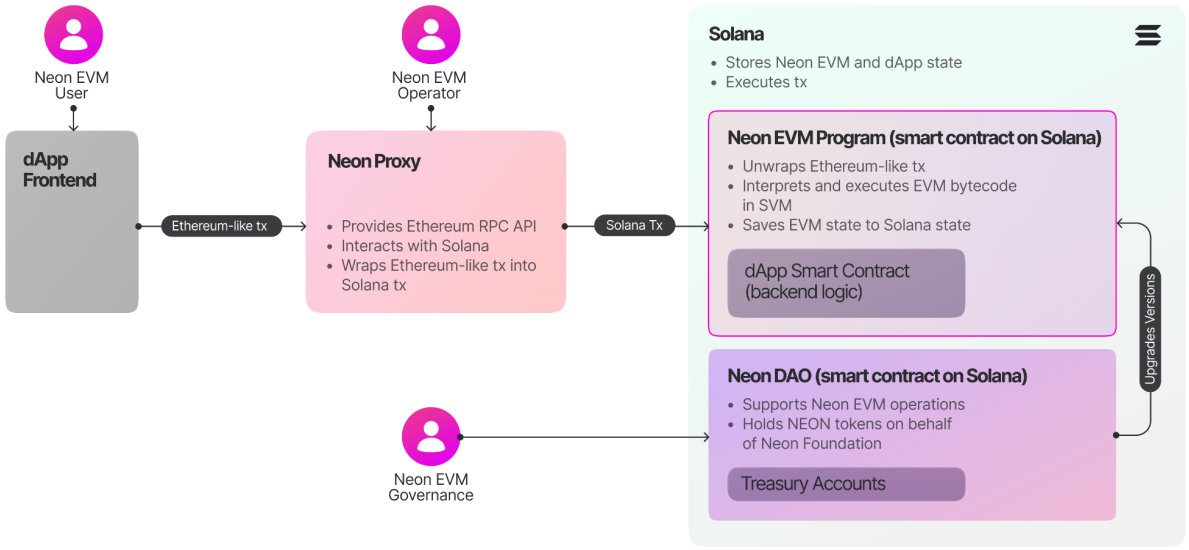
The following section provides a high-level overview of the technical solution that Neon EVM offers.

Neon EVM interacts directly with Solana nodes to carry out transactions on Solana. This makes Solana's throughput, swift block speeds, and low gas prices available to Ethereum contracts. As shown in the following diagram, the service is made up of three main components:

- **Neon EVM Program**
- **Neon Proxy**
- **Neon DAO**

Participants in the system include:

- **Neon EVM user**: any user who has an account in Neon EVM with a balance in NEON, ERC-compliant tokens (e.g. ERC-20, ERC-721, or ERC-1155)
- **Neon EVM Operator**: A Neon Proxy Operator. Operators run Neon Proxy and provide a public endpoint to accept Neon transactions. Each Neon Proxy Operator charges a fee for Neon transaction execution. The fee should allow the Operator to cover hardware costs, transaction execution costs, and bring a profit.
- **dApp:** a Neon EVM client, i.e. any application that has an EVM (Solidity, Vyper, etc.) bytecode contract loaded into Neon EVM on Solana
- **Neon Transaction (tx)**: an Ethereum-like transaction formed and signed according to Ethereum rules



Neon EVM Program

The Solana blockchain has its own Virtual Machine that's based on the Berkeley Packet Filter (BPF). BPF bytecode was originally designed for fast execution: Solana Virtual Machine (SVM) supports just-in-time compilation for BPF bytecode — significantly increasing the speed of execution of BPF contracts.

Neon EVM Program is, like all Solana smart contracts, written in Rust and compiled into the Berkeley Packet Filter bytecode of an SVM. This allows Neon EVM to take full advantage of Solana functionality, including parallel execution of transactions. Furthermore, Neon EVM Program is easy to update regardless of Solana's hard forks.

Neon EVM performs the following functions:

- Uploads EVM contracts (built by Solidity or Vyper compilers) to individual Solana accounts
- Verifies signatures according to Ethereum rules before executing transactions on Solana
- Executes Neon transactions: where necessary, in an iterative manner taking into account Solana resource constraints to ensure finalization of transactions
- Calculates gas consumption in SOL (lamports) and NEON
- Receives and passes payment in NEON token from the user to the Neon EVM Operator for the gas consumed and fees
- Transfers fees in SOL tokens from the Neon EVM Operator account for the execution of Neon transactions in the governance pool

Neon Proxy

Neon Proxy provides an intermediary between Neon EVM clients and Neon EVM in the form of an Ethereum-like Web3 API. Neon Proxy is free software, provided as a containerized solution, and available to anyone.

To enhance redundancy and create an independent market, Neon Proxy is run by independent Operators to create a standard user flow. Neon Proxy packages the transaction into a Solana transaction to facilitate the seamless execution of Ethereum-like transactions on Solana. For Neon's beta phase, a pool of Operators has been selected based on their technical capabilities. Only this select pool may handle the iteration of transactions. See the section "Iterative execution of Neon transactions" for more information.

dApps are able to choose an Operator based on competitive attributes such as prices for transactions.

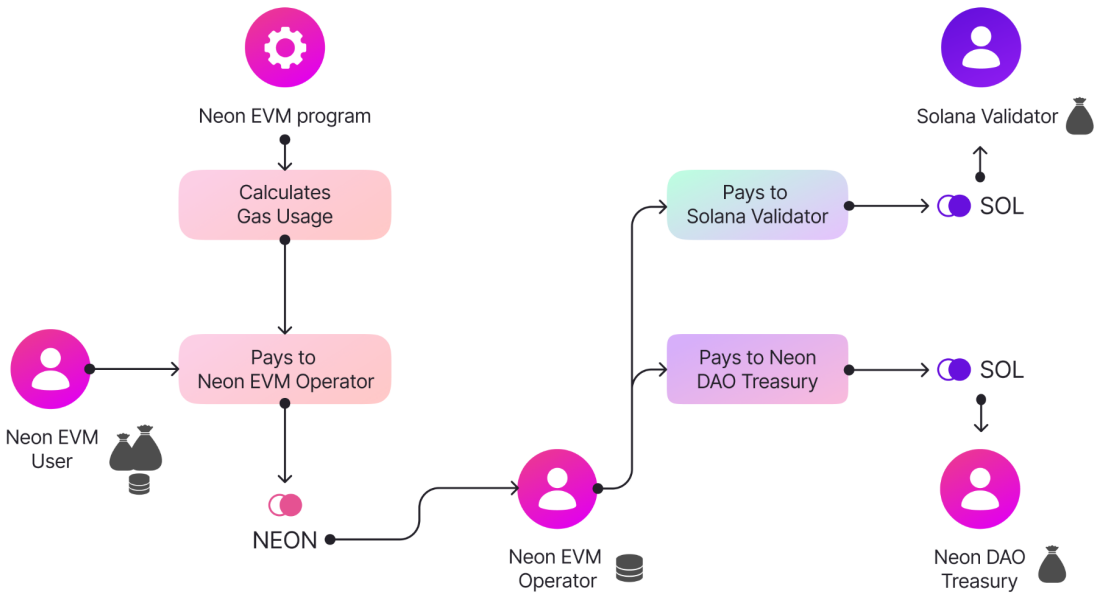
Alternatively, as per the Neon EVM's Roadmap, a Neon EVM client will be able to independently deploy a Neon Proxy and execute Neon transactions on Solana with Neon EVM without using the services of a Neon Proxy Operator (see the section "Decentralization" for further details). In this case, the dApp has to cover payment in SOL tokens, as per the Neon EVM economic rules presented in the following section.

Neon DAO

Neon DAO is a decentralized Neon EVM governance layer that's executed as an SPL Governance program. It exists as a series of contracts deployed on Solana, and receives fees for its services. DAO community participants, i.e. NEON holders, utilize a web interface to raise, and vote on, proposals that impact the Neon EVM using an SPL governance program with add-ins.

Neon DAO is responsible for managing the design and function of Neon EVM. This includes setting up Neon EVM parameters and agreeing on redesign and updates to the Neon EVM Program. For example, should incentivization strategies require modification, e.g. variables such as the size of the deposit required from Neon Proxy Operators who undertake iterative Neon transactions, the DAO governs the change process.

Neon EVM economy



The Neon EVM economy is fee-based. The Neon token, NEON, is a utility token with 2 functions (as per the figure above):

1. **Payment of gas fees**
   The Neon Proxy Operator accepts payment from the user in NEON tokens to pay the gas fees required for transaction execution.
2. **Governance**
   Owners of NEON may engage in the Neon DAO governance activities.

Payment for Neon transactions

A Neon transaction is formed according to Ethereum's standards. Each transaction provides data that accommodates the payment for Neon transactions. As per the Ethereum legacy transaction type, gas data is specified by two properties: `gas_price` and `gas_limit`.

Within an Ethereum transaction, these specify what the user is willing to pay for the gas spent on the transaction to the miner that produces a block. Instead of paying miners, this fee is taken in NEON to pay the entity that submits the transaction on Solana. Typically this is a Neon Proxy Operator (see the section on "Neon EVM: future development" for alternative options).

The cost of Neon transaction execution in Neon EVM is calculated according to Solana rules for Solana transaction execution. This enables the cost of gas in Neon EVM to be significantly lower than on Ethereum or an Ethereum-native L2, as it's based on Solana's gas price.

The Solana gas price is determined by fee parameters that include the number of compute units required and the number of signatures a transaction contains. By implementing Neon EVM to transact on Solana, the number of signatures is always set to 1: that of the Neon EVM Operator in charge of the transaction.

The fees paid in SOL tokens by the Neon EVM Operator fall into two categories:

- The fee to the Solana leader for executing the Solana transaction
- A fee to the Neon EVM treasury

If a Solana account is created as an outcome of the transaction: the fee also includes the lamports required to comply with rent exemption. For example, a Neon transaction creates a new Solana account to transfer SPL tokens, such as NEON tokens, to new Neon users.

In the case that the transaction is parsed iteratively, the Neon EVM Operator pays fees in SOL tokens for each iteration (see the section on "Iterative execution of Neon transactions" for more information).

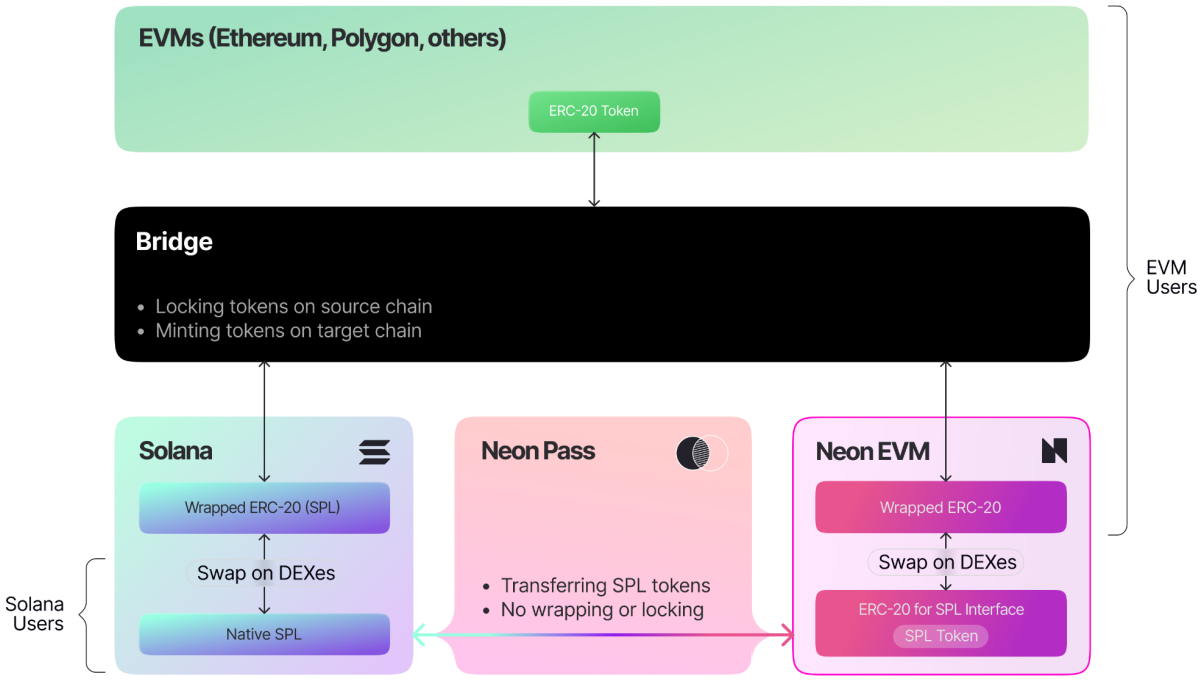
Transacting with tokens

Neon EVM allows dApps to work with ERC-20 tokens on Solana. Two variants of ERC-20 are available:

1. ERC-20 tokens

ERC-20 tokens are minted on EVM-compatible chains. Neon EVM works with both Neon EVM-native tokens, i.e. those minted via the Neon EVM, and with wrapped tokens bridged from other EVM chains.

Bridging tokens



Neon EVM allows users to bridge and dApps to use ERC-20 assets from Ethereum and EVM-compatible chains to Neon EVM, as per the preceding image. A bridge is an EVM third-party solution (independent from Neon) with its own operators. Neon EVM has a bridge integration established for the beta phase launch.

2. ERC-20-for-SPL token

The ERC-20-for-SPL solution enables dApps to utilize native Solana SPL tokens via the standard ERC-20 interface. This is a specific smart contract that can be created in Neon EVM to accommodate each existing SPL token on Solana to provide access to it from dApps. In order to be able to use the SPL token from your Solana account balance, it must be transferred to your NeonEVM account via NeonPass (see the following section, NeonPass: moving tokens between Neon EVM and Solana).

Two contracts are available on Neon EVM that enable SPL tokens to be transacted as ERC-20 compliant:

2.1 ERC-20-for-SPL

The ERC-20-for-SPL contract works with a precompiled contract within Neon EVM which can call the SPL token program. This enables you to utilize existing SPL tokens e.g. SOL or NEON, as wSOL or wNEON, respectively, via the ERC-20 interface, i.e. this contract assigns the ERC-20 standard to the token.

2.2 ERC-20-for-SPL-Mintable

The ERC-20-for-SPL-Mintable contract has two additional methods that enable you to use the Neon EVM to mint a new SPL token and wrap it as ERC-20-compatible. This contract creates a new SPL token using Solana's Token Program and provides mint and freeze authority to the Neon account specified in the constructor.

NeonPass: moving tokens between Neon EVM and Solana

NeonPass is a crucial tool for integrating Neon with Solana as natively as possible. From a non-technical user's perspective, NeonPass is an intuitive interface that enables users to move SPL tokens between Neon EVM and Solana in either direction. The user connects a Solana wallet, such as Phantom, and an EVM wallet, such as MetaMask, to undertake a direct transfer. Once the token is transferred, it may then be used by dApps via the ERC-20 standard interface.

NeonPass is *not* a bridge connecting two separate blockchain ecosystems. Rather, it is a two-way transfer tool for moving assets in and out of the Neon EVM. Under the hood, NeonPass functions by transferring SPL tokens from standard Solana "Associated Token Accounts" to Neon EVM Token Accounts "packed" in an ERC-20-for-SPL interface within Neon EVM.

Neon EVM ERC-20-for-SPL Token Accounts are specialized Solana accounts instantiated in the Neon ecosystem. These accounts can interact with Solidity dApps and are similar in structure to Associated Token Accounts in the broader Solana environment. They store tokens associated with a user's Neon EVM-facing MetaMask wallet.

It's important to recognize that NeonPass sends the original SPL asset from the source to the target address. This "transfer" feature differs greatly from the "bridging" functionality of standard blockchain bridges (bridges lock the original assets in smart contracts and mint "synthetic" assets in the destination blockchain ecosystems).

NeonPass is the revolving door of liquidity between Solana and Neon EVM. It equips users with an easy-to-use tool for:

- Sending permitted SPL tokens, including NEON, into Neon EVM ERC-20-for-SPL Token Accounts to use in dApps as well as to pay for Neon transactions (in NEONs). Permitted SPL tokens, excluding NEON, have a corresponding ERC-20-for-SPL smart contract deployed in Neon.
- Withdrawing SPL tokens from Neon EVM ERC-20-for-SPL Token Accounts back to Solana

Neon EVM features

Neon EVM offers the functionality of Ethereum's transaction standards with the speed, cost, and security of Solana's L1. It does so while offering the security and finality required by dApps, including alternative flows to present opportunities for self-autonomy. The following section goes deeper into such technological features of Neon EVM.

Implicit Neon EVM security

The security of the Neon EVM protocol is enforced in the following ways:

- Neon EVM smart contract code is audited regularly and available for anyone to review
- Neon transactions broadcast to the SVM are validated by independent Solana validators
- Decentralized Neon EVM governance is responsible for the updates to the Neon EVM contract

Independence of operations within Neon EVM

Neon EVM ensures the independence of its operations by providing open access to its infrastructure to anyone who is willing to, and capable of, running Neon Proxy. Moreover, Neon Proxy can be replaced with a client library by any Neon EVM client. See the section "Decentralization" for more information.

Furthermore, the transactions received by Neon EVM can't be discriminated against because they don't have any attributes that determine their priority. The unchangeable nonce and user signature fields verified by Neon EVM guarantee consistent execution of Neon transactions and protect from re-execution.

Preconditions for the execution of Neon transactions

As with other Solana-native dApps, to enable the execution of Neon transactions, the user must grant Neon EVM access to their accounts. Neon EVM then acts on behalf of the user for user transactions such as transfers, swaps, or other operations.

To prevent forgery and unlawful operations, the following fields are checked and verified before a transaction is broadcast to the SVM:

- The nonce field has a unique transaction index, which is verified by Neon EVM's smart contract. This makes double-spend attacks impossible.
- The signature field is formed according to the Ethereum rules and verified by the Neon EVM smart contract. Note: the procedures for validating the Solana signature and the Neon signature are different and are implemented using different algorithms.

Transfer of the native token

In a similar way to how a transfer is handled in a traditional EVM, the 'value' property allows the transaction to define the amount of the native token to be transferred. Within Neon EVM, dApps can transfer the native token of the platform, NEON, to this amount.

Parallel execution of Neon transactions on Solana

Most blockchains process transactions in a single thread, meaning that the blockchain state is modified by one contract at a time. In contrast, Solana can process tens of thousands of contracts in parallel using as many cores as are available on a Solana node. This functionality, known as Sealevel, greatly increases throughput.

Neon transactions are executed by Solana as native transactions in parallel while restricting access to shared data from the Solana state. However, in some cases, a Neon transaction requires more resources than Solana allocates for one transaction. In this case, the Neon EVM executes the transaction iteratively, and the extended mode of restricting access to shared data in the Solana state is used. See the section on "Iterative execution of Neon transactions" for more information.

Parallel processing is possible because Solana transactions describe all the states a transaction will read or write while executing. This prevents transactions from overlapping, which allows independent transactions and those that are reading the same state to be executed concurrently.

To ensure the parallel execution of Solana transactions, Solana requires a list of all Solana accounts involved in a transaction. If there is a call to a Solana account that's not specified in the header of the Solana transaction, the algorithm aborts the execution with an error.

The procedure for the parallel execution of Neon transactions consists of the following steps:

1. Neon Proxy, which has a built-in EVM similar to Neon EVM, receives a Neon transaction from the user.
2. Neon Proxy performs a test launch of the Neon transaction, calling the public Solana cluster RPC endpoint or its own Solana node for the current state.
3. As a result of the test performed, Neon Proxy receives a complete list of contracts and accounts involved in the Neon transaction.
4. Neon Proxy forms a Solana transaction using a list of Neon contracts and Neon accounts, inside which the Neon transaction is wrapped.
5. Neon Proxy sends the Solana transaction for on-chain execution to the Solana cluster.
6. The Solana cluster gets the Solana transaction and sends it for execution to the leading Solana node.
7. The concurrent transaction processor of the Solana node executes the Solana transactions in parallel, checking the independence of the transactions by verifying the Solana accounts in the header of the Solana transaction.
8. Solana transactions executing the packed Neon transactions are processed in parallel, calling the Neon EVM smart contract in the following manner:
    8.1. Neon EVM smart contract is loaded.
    8.2. EVM (Solidity, Vyper, etc.) bytecode smart contract is loaded. Note that each smart contract executed on Solana has its own independent state.
    8.3. The Neon EVM smart contract executes the Neon transaction by calling the EVM smart contract method.
    8.4. When the Neon transaction is executed on-chain, data from the Solana state is used and changed.
    8.5. At the end of the execution of the Neon transaction, Neon EVM updates the Solana state.

Iterative execution of Neon transactions

The SVM limits the resources allocated to the execution of a single transaction to ensure optimal usage of hardware. Neon EVM introduces the iterative execution of Neon transactions to accommodate this restriction.

An incentivization protocol has been established to ensure that a transaction that must be parsed through multiple iterations is completed. The Neon DAO is responsible for managing this protocol which determines the following:

- The fee paid to the Neon EVM governance pool: see “Neon EVM economy”
- Size of the deposit for the iterative execution of a Neon transaction
- The maximum number of iterations (Mi) per Neon transaction
- The maximum number of waiting blocks (Mn)[1]. After Mn blocks, any other Operator can continue the execution and receive the deposit.

[1] The Operator is given a maximum number of blocks (Mn) that may pass between iterations. It’s necessary to limit the execution time of a transaction because all accounts and contracts involved in this transaction will be blocked for use in other Neon transactions, hence the restriction on the maximum number of waiting blocks (Mn).

The main steps to initialize iterative execution are:

1. Neon EVM transfers a deposit in SOL tokens from the Operator's account to a separate account. This deposit is held to motivate the Operator to complete the transaction/s.
2. a. Neon EVM blocks the Solana accounts to be used in the Neon transaction.
   Or
   b. *Alternatively,* if any Solana accounts are already blocked by another Neon transaction, then the new transaction is queued for execution by Neon Proxy.

Once the transaction is selected for execution, the following processing flow applies:

1. a. If it's the first iteration, then the Neon transaction is loaded to a Solana account.
   - The transaction process is started
   - Payment is received from user for uploading the transaction
   - Stores execution details in Solana storage account

   b. If it's not the first iteration, then:
   - If more than Mn blocks have passed, the execution is passed to another Operator
   - Restore the execution details of the Neon EVM from the storage account
   - Complete the maximum number of EVM steps specified in the Solana transaction
   - Neon EVM calculates gas (measured in lamports spent by the Neon Operator)
   - Neon EVM transfers NEON tokens from the Neon user to the Neon Operator at the end of each iteration

2. If it's not the end of execution:
   - Record the Operator that completed a step in iterative execution
   - Increase the number of completed iterations
   - Store execution details in Solana storage account

3. The conditions to end the iterative execution of a Neon transaction are:
   a. The Neon transaction is completed
      Or
   b. The Neon transaction was canceled. In this case, the unspent deposit is burned. Neon transactions can be canceled:
      - By any Neon EVM Proxy Operator, if Mn blocks haven't passed from the last iteration
      - By the Neon EVM Proxy Operator from the last iterative execution

4. At the end of the iterative execution of a Neon transaction:
   - The result of the Neon transaction is saved into the Solana receipt
   - Data modified by the Neon transaction is saved into the Solana storage

- The Neon Proxy Operator that completed the final iteration receives the deposit
- Neon EVM unlocks accounts that were blocked at the start of the Neon transaction
- Solana's state is updated

Neon EVM: future development

Decentralization

Neon EVM has a high level of decentralization, including decentralized governance by the DAO protocol. As a smart contract on Solana, Neon EVM program inherits the same decentralization level as Solana's L1. Part of Neon EVM's product vision is to encourage further decentralization.

The most centralized component of the Neon EVM protocol is Neon Proxy, which packages the transactions to enable Ethereum-like transactions to be compatible with Solana. In Neon EVM's beta phase, as described in this paper, Neon Proxy is operated by a select group of Neon Proxy Operators. This Operator pool was established on the basis of their technical capabilities. Neon DAO is responsible for maintaining the allowlist of Operators and for removing any that exhibit malicious behavior. Parties interested in becoming an Operator must submit a proposal for consideration by Neon DAO.

The Operator pool ensures that there are multiple publicly-available endpoints that accept the transactions from dApps. However, the size of the Operator pool is directly, and negatively, correlated to the centralization effect. To address this challenge, Neon EVM plans to enable self-autonomy for dApps — allowing them to deploy their own Neon Proxy instance.

The first step on the roadmap is to enable dApps to deploy Neon Proxy for single transactions. Later, dApps will be invited to autonomously deploy Neon Proxy to handle the more technically challenging iterative transactions.

Optimization

Self-autonomy brings a new level of optimization to dApps beyond that of improving redundancy. It will also allow dApp developers to optimize aspects of Neon Proxy. For example, by default, Neon Proxy performs a test run to obtain a complete list of Neon accounts that are used for the execution of Neon transactions. A test run takes time, and this time could be critical when a transaction must be executed quickly.

Therefore, the roadmap also supports Neon transactions to be executed without a test run. This will require that the Solana transaction is built on the client side (web/mobile) with a Neon

transaction packaged within it. The Solana transaction is then sent directly to a Solana node. Note, this requires a much greater technical aptitude on the part of the developer, including being capable of executing large transactions iteratively.

Solana interoperability

The Neon EVM roadmap also targets ongoing improvements in interoperability with Solana. For example, Neon EVM will provide dApps with the ability to call programs on Solana and vice versa.

Once this functionality is in place, the next step is to provide access to Solana's sophisticated NFT functionality via the Metaplex protocol. This will provide compatibility between ERC-721 and Solana NFT collections — giving dApps access to a rich ecosystem that supports a thriving creators' economy and expanding gameFi arena.

Summary

Neon EVM is a low-friction solution for anyone looking to scale Ethereum dApps on Solana in a developer-friendly manner. To take full advantage of Solana's functionality, Neon EVM is built as a smart contract of Solana — ensuring flexibility in terms of updates to Ethereum or Solana.

Neon EVM is a full Ethereum emulation with gas consumption calculated by Solana rules and iterative execution of large Ethereum EVM contracts. It's fully compatible with all existing Ethereum tools. It enables parallel execution of EVM bytecode (Solidity, Vyper, etc.) contracts on Solana. It also provides access to Solana infrastructure via Ethereum tools, and access to native Solana tokens registered in the SPL token contract through a wrapper interface, which is native to contracts and tools made for Ethereum. Users can pay for the services of Neon EVM in NEON tokens.

Disclaimer

The information outlined in this White Paper may not be exhaustive and does not imply any elements of a contractual relationship. The content of this White Paper, including that of the website https://neonevm.org, is not binding for us and our affiliates and we reserve a right to change, modify, add, or remove portions of this White Paper for any reasons at any time without prior notification. We also reserve a right to annul this White Paper at any time with a prior notification.

This White Paper is designed for general informational purposes only, as a guide to certain of the conceptual considerations associated with the narrow issues it addresses. This White Paper shall not be reproduced, copied, transferred, or otherwise distributed to any third party. While we make efforts to ensure that all information in this White Paper is accurate and up-to-date, any information herein is not professional advice. We also make no representations or warranties regarding the accuracy, reliability, relevance, and completeness of any information herein.

This White Paper is aimed only on the description of Neon EVM without expression of any opinion or promise on its feasibility, investment attractiveness, and (or) relevance. We do not guarantee that the final product will meet your expectations. You should accept all risks related to operations with Neon EVM on your own prior to using the information herein. Information in the White Paper is based on publicly available information and shall not be deemed as a separate investigation. This White Paper does not constitute an investment, legal, tax, regulatory, financial, accounting, or other advice. Nothing in this White Paper shall be deemed to constitute a prospectus of any sort or a solicitation for investment, nor does it in any way pertain to an offering or a solicitation of an offer to buy any securities in any jurisdiction.

Each person who reads this White Paper is reminded that this White Paper has been presented to him/her on the basis that he/she is a person into whose attention the document may be lawfully presented in accordance with the laws of his or her jurisdiction.

Certain statements in this White Paper may constitute forward-looking statements, which are usually identified by the use of words as “is expected”, “believes”, “expects”, “supposedly” “supposes” or similar phrases and statements. Such forward-looking statements or information may be affected by known and unknown risks and uncertainties, which may influence materially on estimates or results implied or expressed in such forward-looking statements. We do not undertake obligations on updating and review of such statements and information. We do not accept any liability for any losses that can be incurred due to use or reliance on such statements. We have not conducted any independent verification in relation to White Paper or any issue reflected herein.

You shall use this White Paper at your own risk. In the absence of express consent, we do not accept any liability for any losses that can be incurred as a result of such usage, including direct and indirect consequences.

This White Paper is not legally binding, does not oblige anyone to execute any agreements or undertake any obligations.
YOU ARE ENTITLED AND ENCOURAGED TO ASK QUESTIONS AND REQUEST NECESSARY INFORMATION. FOR THESE AND ANY OTHER REASONS PLEASE CONTACT US AT info@neonevm.org.