Hello, Aave (and StarkNet) community members!

Today finally we can announce that Phase 1 of the Aave <> StarkNet project is completed on https://github.com/aave-starknet-project/aave-starknet-bridge

For previous context on the project Request for Approval. Aave <> Starkware Phase I

What has been finally done?

The initial scope of the project was to create an infrastructure of smart contracts to allow users and other applications to deposit on Aave v2 Ethereum, but from StarkNet, without transacting on Ethereum.

Even if some implementation aspects have changed since the initial architecture phase (more details HERE) the final result is still the one expected.

The system allows the following:

- Liquidity providers can “bridge” their Aave v2 Ethereum aTokens via the Aave <> StarkNet bridge to/from StarkNet.

While being on StarkNet, the representation of the aTokens there keeps accruing the yield of Aave on Ethereum, but also allows the liquidity provider to for example “sell” them to Starknet users for a premium, or even use them in one of the Starknet DeFi applications that are continuously appearing.

- StarkNet users (mainly those holding assets on the network) are able to “buy” the StarkNet aToken representations, not being submitted to transaction costs of Ethereum

. And by holding them in their wallet, they accrue yield.

- If any AAVE rewards program would be active on Aave v2 Ethereum, the system allows holders of aTokens on StarkNet to accrue those AAVE rewards over time

, and claim AAVE on Ethereum, whenever they decide to.

Goals achieved with the project

The goals we think are achieved by having this Phase I completed are:

- Enable infrastructure to increase the user base of Aave, by tapping into users and applications native to StarkNet.
- Create new monetization models, in this case for liquidity providers, and provide them with an adequately developed base infrastructure.
- Factually expand Aave (even if in a controlled way and partially for now) to a new and promising network like StarkNet, following the approach of bringing continuous innovation to the table.
- Experiment/understand how suitable is technically to have Aave software on a non-EVM network.
- Test out a new model of development contribution on Aave, with multiple parties not connected before coming together in a project for the community.

How to interact with the system?

This Phase I was conceived as infrastructure development, with user interfaces being out of scope. The rationale of this is to not be opinionated on how the bridge should be used and to remove any type of centralized dependency.

All actions using the Aave <> StarkNet bridge system need to be done via the following smart contracts:

IMPORTANT. The contracts will not be operative until an upcoming Aave governance proposal will activate the system.

- Bridge.sol [github]
- deposit(address l1AToken, uint256 l2Recipient, uint256 amount, uint16 referralCode, bool fromUnderlyingAsset)
- L1 users can call this function to deposit aTokens or their underlying asset to the Bridge contract. Doing this will mint StarkNet aToken representations on the StarkNet side for user l2Recipient

.

- L1 users can call this function to deposit aTokens or their underlying asset to the Bridge contract. Doing this will mint StarkNet aToken representations on the StarkNet side for user l2Recipient

.

- withdraw(address l1AToken, uint256 l2Sender, address recipient, uint256 staticAmount, uint256 l2RewardsIndex, bool fromUnderlyingAsset)
- L1 users can call this function to actually withdraw aTokens or their underlying asset from the Bridge contract, after initiating this withdrawal from the StarkNet side.

- L1 users can call this function to actually withdraw aTokens or their underlying asset from the Bridge contract, after initiating this withdrawal from the StarkNet side.

- receiveRewards(uint256 l2sender, address recipient, uint256 amount)
- L1 users can call this function to actually receive reward tokens accrued on StarkNet side, after calling the bridge_rewards

function on the StarkNet side.

- L1 users can call this function to actually receive reward tokens accrued on StarkNet side, after calling the bridge_rewards

function on the StarkNet side.

- deposit(address l1AToken, uint256 l2Recipient, uint256 amount, uint16 referralCode, bool fromUnderlyingAsset)
- L1 users can call this function to deposit aTokens or their underlying asset to the Bridge contract. Doing this will mint StarkNet aToken representations on the StarkNet side for user l2Recipient

.

- L1 users can call this function to deposit aTokens or their underlying asset to the Bridge contract. Doing this will mint StarkNet aToken representations on the StarkNet side for user l2Recipient

.

- withdraw(address l1AToken, uint256 l2Sender, address recipient, uint256 staticAmount, uint256 l2RewardsIndex, bool fromUnderlyingAsset)
- L1 users can call this function to actually withdraw aTokens or their underlying asset from the Bridge contract, after initiating this withdrawal from the StarkNet side.

- L1 users can call this function to actually withdraw aTokens or their underlying asset from the Bridge contract, after initiating this withdrawal from the StarkNet side.

- receiveRewards(uint256 l2sender, address recipient, uint256 amount)
- L1 users can call this function to actually receive reward tokens accrued on StarkNet side, after calling the bridge_rewards

function on the StarkNet side.

- L1 users can call this function to actually receive reward tokens accrued on StarkNet side, after calling the bridge_rewards

function on the StarkNet side.

- bridge.cairo [github]
- initiate_withdraw(l2_token : felt, l1_recipient : felt, amount : Uint256)
- L2 users can call this function to bridge back StarkNet aToken representations to Ethereum. L1 user l1_recipient

should then call the function withdraw

on the L1 Bridge contract to actually withdraw tokens.

- L2 users can call this function to bridge back StarkNet aToken representations to Ethereum. L1 user l1_recipient

should then call the function withdraw

on the L1 Bridge contract to actually withdraw tokens.

- bridge_rewards(l1_recipient : felt, amount : Uint256)

- L2 users can call this function to bridge back accrued rewards to Ethereum. L1 user l1_recipient should then call the function receiveRewards

on the L1 Bridge contract to actually receive reward tokens.

- L2 users can call this function to bridge back accrued rewards to Ethereum. L1 user l1_recipient should then call the function receiveRewards

on the L1 Bridge contract to actually receive reward tokens.

- initiate_withdraw(l2_token : felt, l1_recipient : felt, amount : Uint256)
- L2 users can call this function to bridge back StarkNet aToken representations to Ethereum. L1 user l1_recipient

should then call the function withdraw

on the L1 Bridge contract to actually withdraw tokens.

- L2 users can call this function to bridge back StarkNet aToken representations to Ethereum. L1 user l1_recipient

should then call the function withdraw

on the L1 Bridge contract to actually withdraw tokens.

- bridge_rewards(l1_recipient : felt, amount : Uint256)
- L2 users can call this function to bridge back accrued rewards to Ethereum. L1 user l1_recipient should then call the function receiveRewards

on the L1 Bridge contract to actually receive reward tokens.

- L2 users can call this function to bridge back accrued rewards to Ethereum. L1 user l1_recipient should then call the function receiveRewards

on the L1 Bridge contract to actually receive reward tokens.

- static_a_token.cairo [github]
- claim_rewards(recipient : felt)
- L2 users can call this function to mint StarkNet reward tokens they have accrued.
- L2 users can call this function to mint StarkNet reward tokens they have accrued.
- ERC-20 like functions
- claim_rewards(recipient : felt)
- L2 users can call this function to mint StarkNet reward tokens they have accrued.
- L2 users can call this function to mint StarkNet reward tokens they have accrued.
- ERC-20 like functions

A more extensive list of the smart contracts' API can be found on the official repository of the project https://github.com/aave-starknet-project/aave-starknet-bridge

It is also possible to interact programmatically/integrate applications via a TS library produced within this scope https://github.com/aave-starknet-project/starknet-aave-bridge-js.

Access control and protections

- The Aave <> StarkNet bridge (both Ethereum and Starknet sides) will be fully controlled by the Aave Governance smart contracts on Ethereum, whenever the governance decides so

. Control of the StarkNet components is exercised via a simplified cross-chain governance set of contracts, partially using a Cairo smart contract used by MakerDAO on a similar project (props!).

- Given the still experimental nature of StarkNet, the bridge has a mechanism of deposit caps, to be set via the Aave governance proposal to activate the bridge

.

Security procedures

There have been two types of security measures applied to the project.

Internal

- Units testing of the smart contract components, both on the Ethereum side (Solidity) and StarkNet side, Cairo.
- Integration testing connecting both sides.

External

3 security firms have been involved in verifying the project:

- Certora

. Formal verification procedure on the L1 side of the bridge, both by the Certora team and by community participants on the Aave <> Certora grants program. A total of 104 rules have been written. The final report and rules can be found at https://github.com/aave-starknet-project/aave-starknet-bridge/tree/main/certora

- Nethermind (security arm)

. Security review of all the Cairo components, considering its dependencies on the Ethereum side. The final report can be found at https://github.com/aave-starknet-project/aave-starknet-bridge/blob/main/audit/nethermind_audit.pdf

- Peckshield

. Security review of all the Ethereum components, in 2 rounds. The report can be found athttps://github.com/aave-starknet-project/aave-starknet-bridge/blob/main/audit/peckshield_audit.pdf

Budget recap

The estimated budget at the start of the project was the following

[

new-budget

1572×636 98.4 KB

](https://europe1.discourse-cdn.com/business20/uploads/aave/original/2X/7/7fc0a6a7abe4bdc0d0b7295ac8bf002d7dd299fa.png)

From that, the final expenses have been the following (spent/projected):

- Salaries: $205’000 / $205’000
- Gas cost: $~1’300 (in ETH) / $30’000
- Audits: $52’000 / $60’000
- Buffer. $0 / $40’000
- Misc: $3’000 (sponsoring StarkNet Amsterdan hackathon)
- Bug bounty. To be returned to the Aave DAO, and keep allowance of the 20’000 for potential allocation.

In summary, approx. $261’300 has been spent out of the $385’000 projected

($335’000 if discounting the bug bounty allocation).

After the settlement of the 50/50% split with Starkware, approx. (as part was taken in ETH, currently depreciated in value) $70’000 will be returned to the Aave collector

.

What can be improved in the future?

Even with a functional bridge, there are always aspects to improve on its implementation, especially taking into account how StarkNet evolves over time, and the new possibilities open. Amongst others, medium-term potential extra developments are:

- Bridging based on storage proofs.

On the initial architecture, the underlying message bridging was designed to be done via the so-called storage proofs, potentially making the system more efficient. Temporary cost limitations on StarkNet led to change to a more “traditional” bridging approach, but this route could be explored in the future (with the current repository including already a proof of concept https://github.com/aave-starknet-project/aave-starknet-bridge/pull/79).

- Expand to non-Ethereum networks.

Aave and its aTokens are not only present on Ethereum, so a pretty straightforward movement would be to deploy the Aave <> StarkNet bridge on other EVM “origin” networks, like for example Polygon, Avalanche, or rollups like Optimism or Arbitrum.

- Aave v3 adaptation.

At the moment, the Aave <> StarkNet bridge support Aave v2 Ethereum aTokens, but it is possible to cover also Aave v3 with relatively simple changes.

Wen Phase II?

The idea of an Aave <> StarkNet collaboration has always been to, at the end of the road, have a full-fledged instance of Aave running on StarkNet.

With Phase I completed successfully as a really good touch-base on how developing an Aave Cairo version would be, the next step will be to propose to the Aave community to approve the (for now) final Phase II.

During the following weeks, a similar proposal as with Phase I will be presented in this forum, for the community to discuss and hopefully approve.

Final steps for Phase I

An Aave Governance proposal will be submitted in the following days, for the Aave community to vote on the activation of the bridge.

The proposal will:

- Deploy the L1 side of the bridge.
- Activate the system, by enabling initially 3 assets: DAI, USDC, and USDT. To follow a responsible approach, deposit ceilings of 30’000 will be set for each one of them. We welcome feedback from the community on both these initial assets to be enabled and the deposit ceilings.