There has been concern recently about whether or not the “composability” property of Ethereum - basically, the ability of different applications to easily talk to each other - will be preserved in an eth2 cross shard context. This post argues that, yes, it largely will be.

What does sharding change?

Transactions within a shard could happen as before. Transactions between shards can still happen, and happen quickly, but they would be asynchronous, using receipts. See https://github.com/ethereum/wiki/wiki/Sharding-FAQ#how-can-we-facilitate-cross-shard-communication for more information. In general, workflows of the form “do something here that will soon have an effect over there” will be easy; workflows of the form “do something here, then do something over there, then do more things here based on the results of things over there, all atomically within a single transaction” will not be supported. Doing things of that form would generally require first “yanking” the contract from the “over there” shard to the “here” shard and then performing the entire operation synchronously on one shard. However, as we can see from examples below, most use cases would not be significantly disrupted or could be trivially rewritten to survive in a cross-sharded model.

Tokens

The ERC20 standard would need to be modified. Tokens would be able to exist on all shards, and seamlessly move from one shard to another just like ETH. This can be done with receipts, in the same way that ETH is moved from one shard to another, we can move tokens from one shard to another. There are no fundamental difficulties here.

Composability example 1: Uniswap <-> Tokens

Nearly all DeFi applications are uses of composability, because tokens are a type of application and so any DeFi application that uses tokens is an application that interacts with another application. Let us look at Uniswap as an example. In Uniswap, a user sends some quantity of token A to a Uniswap contract, which sends some quantity of token token B back to the user. Uniswap requires strict dependency between all transactions that interact with it: the Nth transaction must be aware of the output of the N-1’th transaction, because this is how the price updating algorithm works.

Hence, the Uniswap contract would need to live on a single shard (there are designs for multi-shard Uniswap, but they are more complex). Users looking to trade would perform a 2-step procedure:

1. Send their token A to the shard that Uniswap is on.
2. Perform a trade with Uniswap as before (the transaction doing this would be combined with the transaction “claiming” the receipt from step (1), so it’s a single step)
3. [Optional] If desired, move the token B that Uniswap gave them to some other shard.

Composability example 2: Lending on Compound (including cDAI etc)

Compound could also exist on a single shard (if Compound gets too popular, different instances

of Compound representing different pairs of tokens can be put on different shards). Users with a token would move their token over to the shard the particular Compound instance is on, and (create | fill | bite) a leverage position as before.

Composability example 3: tokens inside L2 scaling solutions (Rollup, Plasma…)

Move your tokens to the shard the L2 scaling solution has a contract on. Deposit into the contract. Done.

Composability example 4: rDAI, gDAI, etc

Move your DAI into the [insert DAI flavor here] contract. Take [insert DAI flavor here] out, and move it to whatever shard you want. The [insert DAI flavor here] contract itself could just sit on the same shard as the Compound instance for DAI for convenience.

Composability example 5: Set Protocol

Move your tokens into the shard that the set protocol contract is on (different instances could be in different shards as in Compound). Send them into the set protocol contract, get out a set token, move the set token to whatever shard you want.

Composability example 6: oracles

Synchronous

cross-shard transactions are not supported, and so the “call a contract and immediately see what the answer is” workflow would NOT work. Instead, however, you could simply provide a Merkle proof showing the value in the state of the contract on the other side in the previous block (or in the most recent block for which the application’s shard is aware of the oracle contract’s shard’s state root).

Composability example 7: non-fungible assets and markets

Non-fungible assets including NFTs, in-game assets, ENS names, MakerDAO CDPs, Compound positions, etc,can be "yanked" to other shards, where they can interact with other applications (eg. atomic swap markets, auctions) seamlessly as before.

Overlay tools (eg. Instadapp)

In general, overlay tools that use specialized smart contracts to interact with dapps would need to create contracts for each function that they support, that users could yank to a desired shard, and then use on that shard to perform any needed functionality as before.