stratosph.com
contact@stratosph.com

Decentralized Finance (DeFi)

Part I

January 2021

Table of contents

Introduction

A relatively young crypto sub-sector, DeFi has been building on the premise of a truly P2P financial services without trusted third parties. Philosophically it is the natural extension of Bitcoin from an operational perspective. Technically it's enabled thanks to the smart contracts - financial logics implemented on-chain with the virtual machine executing the state transition functions. Smart contracts are simple programs which define a sequence of instructions to move an account on a blockchain from one state to the next. More complicated business logic can be achieved by a combination of smart contracts purposely built.

Ideally, the DeFi protocols are natively on-chain. This means the whole program is coded and executed internally on a public permissionless blockchain, mined by the miners, validated by the nodes, without intervention or verification by any third parties. This should differ from other enterprise blockchain applications where the blockchain is a mere database where the controlling entity has full permission to alter the blockchain and often business logics are done externally.

So far, the key pillars of DeFi, as it's the case in other financial market's structure, are: lending/borrowing platforms, decentralized exchanges (DEXes), derivative assets issuance and trading, etc.

DeFi is a big deal in crypto. After nearly a decade, beside Bitcoin, it's the second significant product-market fit of the blockchain as a technology. To date, the most popular blockchains for the DeFi protocols to build on is Ethereum, although other chains are emerging, such as Polkadot, Solana, Binance Smart Chain (BSC), etc. Cross-chain technology is also progressing, for example the Cosmos ecosystem.

To offer you an overview on DeFi, we will adopt an approach from an engineering design point of view: we won't present a list of the current protocols and their features as customarily done but we will explain the technical hurdles which have led to the solutions and shaped the landscape.

Current smart contract limitations

Due to the current blockchain architecture, on-chain transactions are costly both in terms of storage and computing power. After all, all the miners have to replicate the same state transition and the nodes to verify its validity.
One can list several architectural constraints to the smart contracts:

- Blocktime: is not instantaneous. In Bitcoin, there is a block every 10 minutes. In Ethereum, every 15 seconds. There are other chains with faster block time however there are other issues in security, decentralization and adoption.
- Blockspace: is scarce. Increasing the block size doesn't seem to be a solution, evidence being that the market has rejected it (such as the case of the big-block Bitcoin forks like BCH or BSV). This limits the admissible amount of on-chain data storage.
- Algorithmic complexity: the virtual machines usually have limitations on what it can process. There is hardly as much freedom in on-chain programming as in standard environments. Maths operation, pseudo-randomness generation, sorting, NP-complete problems are very hard to implement efficiently.
- Gas: on-chain execution is paid via gas. More complex functions require more gas to be executed. In congestion time the gas cost could become prohibitive to both small and large players.
- Sequential execution: on-chain transactions are executed in sequential order, as by construction the nodes have to agree on the latest state of the chain before it can change further. Parallel execution for smart contracts can be made possible only via changes in the base blockchain layer and can't be done easily by pure engineering at the smart contract layer.
- Transaction triggering: a smart contract is executed only when it's called, either by a human address or by another smart contract. So when it needs to be called, incentives are paid to the caller to trigger the transition (see discussion about the Keepers below).

Smart contracts programming

In summary, preferably the smart contracts should only be used to execute algorithms with complexity O(1). As such, when one thinks about blockchain programming, the accounts

should be handled separately with no dependency on the others, or they are pooled all together to be treated as one single big account.
That's the reason the concept of pool is extremely prevalent in DeFi.

We will now discuss different building blocks of the current DeFi.

Lending

Lending is the starting point. One would like to build a protocol which allows the participants to lend and borrow from each other, again without supervision.
There are two important considerations. First, one would need an interest rates model to determine the lending rates (akin to a yield curve). Second, collateral management: when the borrowed amount exceeds the collateral, the latter would need to be liquidated to cover the loan. This happens because in crypto both the loan and the collateral can be floating and market volatility can render the borrower insolvent.
Fortunately, the stablecoins (e.g. USDT or USDC) are already on-chain, which means the problem of porting USD to the blockchain has already been solved some years back.

Pooling of borrower and lender

An order book based lending market (a la Bitfinex) is not practical on-chain. One must pool the participants into a borrower and a lender pool. The lenders deposit available funds into the lending pool, from which the borrowers withdraw the funds.

Interest rate model

The rate would be higher when there are more borrowers than lenders. Or one can see it as when the lending pool is more depleted. Based on the remaining available funds to lend one can design an interest rate curve to incentivize replenishment of the lending pool.
Arguably it seems far from a truly free market mechanism, nonetheless it solves a design problem coming from the limitations of the smart contracts (as discussed above).
This is the solution adopted by Compound and Aave.

Collateral liquidation

Collateral liquidation implies many other problems.

First, when to liquidate, i.e. the smart contracts would have to know the current price of the assets to determine under-collateralization. This is known as the oracle problem (discussed below) as smart contracts are purely on-chain logics and a priori have no knowledge of the external world.
Second, who liquidates the bankrupt accounts. As noted above, the smart contracts can't call themselves, so one needs external keepers to trigger the liquidation contract on the endangered accounts.
Third, where to liquidate: since things happen on-chain, it's impossible to use centralized exchanges (such as Binance or Coinbase) to liquidate the funds. One needs to build decentralized exchanges (DEXes) - that live and function only on the blockchains - to support the liquidation activities.

At this point one can see that the oracles, the keepers and the DEXes are inevitable infrastructural building blocks in DeFi. They respond to financial purposes but their existence arises rather from engineering problems.

These considerations have informed the design choices and the functioning of the popular lending platforms.

Compound: https://compound.finance/
Aave: https://aave.com/

Keepers

Initially done by individuals either manually or using bots. To be a keeper one would need some capital to take over the liquidated account and pay for the transaction fee, and some smart contract literacy to be able to call them (either from GUI like Etherscan or building their own bots). The operations have been creating gas wars as the keepers compete by front-running each other to take hold of the more profitable accounts.

Recently the emergence of the keeper protocols helps optimizing capital efficiency and mitigating gas wars. For example, in KeeperDAO, the protocol pools participants' assets which will then be readily borrowable by the keepers who are specialized in running the liquidations. The pool participants and the keepers share the rewards. The keepers won't need capital individually because they can borrow from the pool, and they can work together towards a more collaborative gas scheme as they eventually share the profits.

KeeperDAO: https://keeperdao.com/

Oracle problem

The problem is how to bring off-chain data to on-chain contracts in a secured and trust-less manner. Off-chain data includes but is not limited to price data, volatility data, weather data, prediction market outcomes etc.
The problem is complex and a deep dive on oracles is not the purpose of this note. The current solution involves a mixture of infrastructural design (decentralized nodes, signed messages) and incentivization via tokens (slashing collateral of dishonest nodes, paying punctuality and accuracy, etc).

Prominent oracle protocol is Chainlink. Newer protocols are catching up, such as Band, Dia, Paralink (being built on Polkadot).

Chainlink: https://chain.link/

In Part II which is coming shortly, we will talk about the DEXes and some basic elements of the mathematics behind it.