IBetYou

Decentralized betting protocol that enables users to bet and earn incentives through participation

Abstract:

This project is an experimental customer-facing solution in bringing Ethereum or, more specifically, smart contracts, advantages to the average internet user and explaining its benefits straightforwardly. There are multiple layers of the solution that will be obvious to the user as he progresses through the system. The idea is to slowly lead the user through the concepts like smart contracts, lending protocols, prediction markets, decentralized dispute resolution without first trying to explain it to them. Our hope is they will understand them intuitively through examples and the end value they produce.

The flow will be as follows :

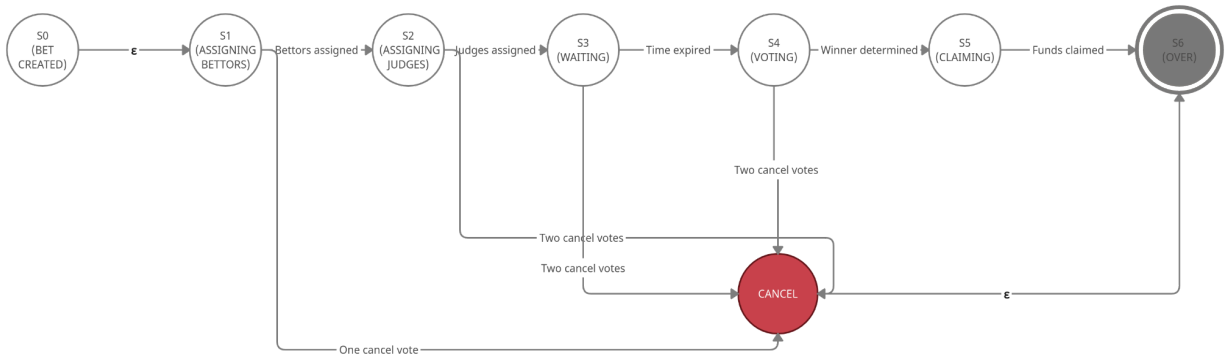
1. the user (bettor) decides to initiate a bet with someone he knows and with whom he agreed on the details offline.
2. The bettor comes to IBY to formalize it using the service, inputting the email of the friend (counter bettor), the content of the bet “I.e I bet your new year's resolution to go to gym three times a week will last until 15th of February” if you loose you pay for the dinner (equivalent of 0.3 ETH).
3. Then the user (bettor) appoints an arbitrator (a person who makes sure the bet outcome really happened) - (bettor judge)
4. The bettor locks in 0.3ETH in the contract and sends the bet to his friend (counter-bettor)
5. In turn the friend (counter bettor) receives the email with the challenge where he (the counter bettor) will appoint his arbitrator (counter bettor judge) and lock in the equivalent amount (i.e 0.3 ETH) in the smart contract.
6. While the funds are locked in the smart contract they would be deposited to an interest bearing protocol (i.e mStable, AAVE, Compound) where they will generate yield for the duration of the bet.
7. Once the bet is finalized and arbitrators have decided who won, the winner would receive the full amount of the bet (0.3 ETH initially deposited for bet initiation and 0.3 ETH for the other side, plus a portion of the yield generated during the locked period).
8. The yield would be distributed equally between the arbitrators and us.

In case of a dispute or the inability to reach a conclusion by the appointed arbitrators we will appoint that decision to Kleros Court or a similar solution for the final decision.

Motivation:

We are still in early days of blockchain where users often have to be quite technical to use any part of the stack. We acknowledge the existence of prediction markets and other complex systems that serve the same purpose but we believe that having a Dapp that tries to solve a problem of simple personal betting in a decentralized manner with UX being as simple and familiar as possible still represents a great obstacle but also an opportunity for the space in general. Our goal in the first year of the project is to reach at least $5M TLV in the platform. We believe this will prove that the game has crossed the UX hurdles and it is ready for the next level of development. This whitepaper explains ways we have addressed technical problems and the way it's being implemented and the future that awaits us. If you want to bet with us join us on https://ibetyou.me/.

Technical explanation:



Bet finite state machine

State description

S0 - Initial state (Bet Created)
S1 - Assigning bettors
S2 - Assigning judges
S3 - Bet waiting
S4 - Voting stage
S5 - Claiming
S6 - Over

State transitions

S0 -> S1 - ε-transition

S1 -> S2 - Both bettor and counter bettor participated
S2 -> S3 - Both judges participated
S3 -> S4 - Waiting time has expired
S4 -> S5 - Winner decided by having most of all votes cast
S5 -> S6 - Every eligible participant has claimed funds
CANCEL -> S6 - ε-transition

**Every state except S5 and S6 can transition to CANCEL state if number of required cancel votes equals number of bettors present.

Finite state machine flow description

Bet enters state S0 after it has been created. From S0 it goes to S1 after both bettors have accepted a bet. State S2 is reached when both bettor and counter bettor have joined. State S3 is reached after both judges join. State S3 is the “waiting state”. The bet exits state S3 when voting time limit* is reached. In state S4 judges are allowed to vote. The bet stays in state S4 until one of the bettors has more than half of the maximum votes. Bet stays in state S5 until all eligible parties claim their rewards. After that, final state S6 is reached.

*Judges cannot vote until a certain time limit is reached.

Describing the mechanism of yield creation

IBY by itself presents a leap forward in better blockchain UX on so many levels and one of the advantages is bringing non crypto native users and their liquidity on the chain. But once we have that liquidity what are we going to do with it?

We plan for IBY protocol to be one of the biggest liquidity providers on the Quickswap where we “forward” user stakes from their bets and exchange them into ma tokens (maUSDC, maUSDT, maAAVE etc.) and only when the specific bet is concluded we return ma tokens to original tokens with the increased amount. Surplus is then directed towards IBY governance to decide what to do with it. Set of governance proposals will decide what is done with the surplus (yield). Users holding IBY tokens will earn yield without even having to claim it and that will make IBY to be worth more during time. However the exact details of this mechanic will be decided with a community vote via DAO.

Describing the mechanism of yield creation (Notice: technical stuff!)

Here we are going to describe how yield is generated for IBY protocol.
First we define which tokens we want to interact with. In this list are maUSDC, USDC and MATIC (equivalent to ETH on Matic network). Beside that we also define quickswap router address that we want to interact with called IQuickSwapRouter. After we've established basics let's deep dive into mechanics of actual swapping. In this explainer we will cover case where user deposits Matic tokens into betting contract.

```
contract Exchange is ReentrancyGuard {
    address public constant maUSDC =
        address(0x9719d867A500Ef117cC201206B8ab51e794d3F82);
    address public constant USDC =
        address(0x2791Bca1f2de4661ED88A30C99A7a9449Aa84174);
    address public constant MATIC =
        address(0x0000000000000000000000000000000000001010);

    IQuickSwapRouter02 router =
        IQuickSwapRouter02(0xa5E0829CaCEd8fFDD4De3c43696c57F7D7A678ff);
```

Image 1. Quickswap router and tokens definitions (addresses)

After we've defined everything let's deep dive into swapMaticForMaUSDC.
First we ask Quickswap to swap exactETHForTokens where we say, let's swap ETH to USDC and because we are on Quickswap and not Uniswap that actually means that we are swapping Matic to USDC. After that has been done we check our contract wallets USDC balance and continue to swap USDC for maUSDC. If that is all done successfully then we hold that balance of maUSDC and hold it as it increases in value over time.

```
/**
 * @notice Swaps MATIC(ETH) tokens for maUSDC tokens
 * @param _unixTime If transaction is not mined and unixTime has expire, transaction will revert
 */
function swapMaticForMaUSDC(uint256 _unixTime) public payable nonReentrant {
    router.swapExactETHForTokens{value: msg.value}(
        0,
        _createPath(router.WETH(), USDC),
        address(this),
        _unixTime
    );

    uint256 USDCAmount = _getTokenBalance(address(this), USDC);
    IERC20(USDC).approve(address(router), USDCAmount);
    router.swapExactTokensForTokens(
        USDCAmount,
        0,
        _createPath(USDC, maUSDC),
        msg.sender,
        _unixTime
    );
}
```

Image 2. Swapping Matic for maUSDC

After the bet finishes we have to payout the winner the full amount of deposited bet (bettor + counter bettor). To do that we exchange maUSDC tokens that we have on the wallet and swap them to USDC. After that we swap them for ETH (we are on Matic so this means we are actually transferring them for Matic tokens). And we will see in the end that we have more than what we had on the start.

Example:
In February 2021 on our test contract each side deposited 1 Matic token, so it's a total of 2 Matic tokens. Those 2 tokens were exchanged for maUSDC tokens following the path above and after that we left them there for 2days. After the bet was finished smart contract automatically converted that maUSDC to Matic and now we had 2.3 Matic which means we earned a yield of 0.3 Matic as simple as that.

```
/**
 * @notice Swaps maUSDC tokens for MATIC(ETH) tokens
 * @param _unixTime If transaction is not mined and unixTime has expire, transaction will revert
 */
function swapMaUSDCForMatic(uint256 _unixTime) public nonReentrant {
    uint256 maUSDCAmount = _getTokenBalance(msg.sender, maUSDC);
    IERC20(maUSDC).transferFrom(msg.sender, address(this), maUSDCAmount);
    IERC20(maUSDC).approve(address(router), maUSDCAmount);
    router.swapExactTokensForTokens(
        maUSDCAmount,
        0,
        _createPath(maUSDC, USDC),
        address(this),
        _unixTime
    );

    uint256 USDCAmount = _getTokenBalance(address(this), USDC);
    IERC20(USDC).approve(address(router), USDCAmount);
    router.swapExactTokensForETH(
        USDCAmount,
        0,
        _createPath(USDC, router.WETH()),
        msg.sender,
        _unixTime
    );
}
```

Image 3. Swapping maUSDC for Matic

This is just a start as multiple more advanced strategies can be created where not only swap can be done, but also providing liquidity on Quickswap, integrating with Compound etc.

Creating a mainstream appeal

Gasless experience, wallet abstraction, easier onboarding

We will eventually want to reach the broad mainstream but in order to do that we also need to rethink the UX approach, especially with gas prices at all time high. The idea is to abstract gas fees from users so he/she doesn't even know they exist by integrating Biconomy to achieve gasless experience. Other UX problems that need solving are creating an experience where wallets are abstracted and user flow is much smoother than your usual metamask powered dApp. Ideally we would have an experience where users login with their twitter, facebook or any other social login or username and password even abstracting usage of wallets. There will be two modes : normal and advanced mode. In normal mode you wouldn't need to know how to use crypto wallets, save seed phrases and private keys. Using Arkane network to seamlessly onboard users, create a profile like they are opening an account on a social media website, buy

crypto with your credit card, interact with the protocol without even having to know what wallet is and everything is still on chain, gas free and mobile friendly.

IBY DAO - the future

Ultimate goal with all of this is completely decentralized protocol that will earn it's holders yield for just holding the IBY token. Where DAO will decide on every important governance decision, what to develop in the future and additionally adding advanced farming strategies and integrations with protocols like Compound, Alpha Hommora and similar future opportunities on top of the AAVE integration.

IBY governance mechanics

The proportion of active voters on any given blockchain project or protocol is fairly low - usually hovering in the single digit percentages - the so called voters apathy. This means that either a "loud minority" controls the protocol or a quorum cannot be reached and proposals are never voted on.

To circumvent these issues, we need a mechanism of representing so-called "passive token holders". These are people who are, for one reason or another, not interested in actively participating in the development and governance of the IBY protocol - but hold IBY tokens.

Council

The voters' apathy will be solved using a mechanism of elected council members. The council consists of an odd number of council members. The initial number of council members are set to be seven. The council can grow in size, but never exceed 33

Elections

Council members are elected through a series of log-scaled, liquid democracy voting mechanisms.

Continuous Liquid Democracy

Liquid democracy is a form of delegative democracy whereby an electorate has the option of vesting voting power in delegates as well as voting directly themselves. The electorate which vests power in delegates does so on a continuous basis.

The candidates with the most IBY staked to their name are automatically elected to the council.

Since IBY holders can either re-stake and revoke their vote constantly - every block is a “mini election” and a new prospective candidate can “rise through the ranks” quickly - if decided so by the community. The votes of IBY holders which do not participate in the election mechanism are not counted towards the final tally of the votes.

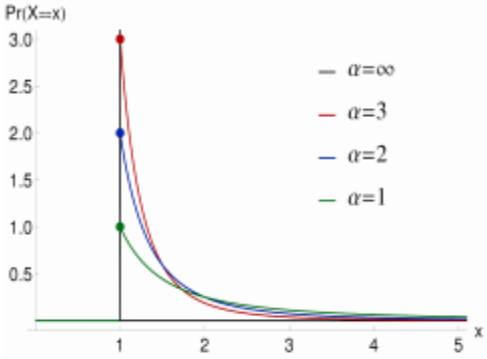
Phragmén Voting

The IBY Governance council is chosen by means of a Phragmen Election - quite well described on the following link: https://wiki.polkadot.network/docs/en/learn-phragmen

Anti-Pareto Scaling

To ensure that the governance council is not dominated by “superstar” council members who are very hard to outvote and thus have indiscriminate control over the protocol - a system of “Anti-Pareto Scaling” is employed.

The problem of skewed Pareto distributions is seen in many blockchain projects - a small minority of the participants control the vast majority of the network.



In the left graph , we can see that the amount of inequality in the system can be expressed as the variable α in Generalized Pareto Distribution function.

Anti-Pareto Scaling aims to reduce α to a pre-decided number - by applying an inverse logarithmic function to the weighted votes.

By choosing the factor α, we can choose exactly how distributed we want our system to be. When a system is in an equilibrium - the application of the log scale function has no effect. It's only when the system gets too centralized or too distributed, that the Anti-Pareto mechanism kicks in.

The Initial proposal for α is 1.14. This would reduce the Generalized Pareto Distribution to the Pareto Principle - often found in nature and known as the "80/20 Rule".

The best way to determine the α will most likely be through running simulations and having the community decide before the DAO launch.

Voting for proposals

The council and IBY token holders can vote on proposals. The proposals can vary from very specific actions - such as network fee amounts, liquidation ratios and similar - to very broad changes - such as changing the IBY Protocol contracts themselves.

Weighted Voting

The council members vote by acting as a proxy for the IBY holders that staked their tokens to them. The amount of voting power they wield is equal to the amount of IBY staked when voting for them - adjusted by the Anti-Pareto Scaling function

The IBY holders who did not vote for council members vote independently with their IBY tokens. The proposal is accepted with a simple majority vote (50% + 1 vote).

Submitting and Accepting Proposals

Each IBY holder can submit a proposal for a change by staking a set minimum amount of IBY tokens into a proposal contract.
The minimum proposal stake can be set by a Governance Vote.

Other members of the community can stake their IBY into the proposal vote - the proposal with the biggest stake in that voting cycle is selected to be voted on. Council members can also submit proposals. Council proposals don't require staking tokens and are accepted automatically.

Voting Schedule

Voting is done, at max, three-at-a-time - meaning there are never more than two proposals being voted on at a time. Maximum of one proposal comes from the public and a maximum of one from the council. Council proposals are put in a queue and voted in order of proposing (earlier proposals being voted on earlier).
A new proposal is voted on once a week.

Automated Token Buyback Mechanism

Even though stakes on the IBY protocol are deployed in multiple tokens, the value they accrue should be expressed directly through the value of IBY. This is why part of the yield (the exact amount to be decided before the DAO launch) that is earned in any of the tokens like Matic, Quick, USDC or any other token - is deposited into an Automated Token Buyback Contract. The contract has a list of approved DEXes - voted on by a Governance proposal through which it swaps the deposited token for IBY.

Those IBY tokens are then moved to the Treasury. This creates a constant buy pressure for IBY token and reduces the circulating supply of IBY - ensuring that the value of the IBY token will have a positive price pressure exactly correlated to the activity of the IBY protocol. This method is taken from the following article as a better alternative to token burning:
https://www.placeholder.vc/blog/2020/9/17/stop-burning-tokens-buyback-and-make-instead

The Treasury

The IBY Tokens in the treasury are the common good of all IBY holders. The funds from the treasury can only be assigned by a super-majority vote.

Anyone can apply to receive funds from the treasury. As issuing tokens from the treasury will inevitably increase the circulating supply of the IBY token and create a sell pressure - the funds must be used responsibly. This is why a strict super-majority of 2⁄3 of participants must approve the spending of funds from the Treasury.

Sinkhole

The Treasury has an attached “Sinkhole” contract - where funds can be sent to be removed from the circulating supply forever. Once tokens are sent to the Sinkhole contract, they are being moved from circulation through the use of an Exponential Decay Function (https://en.wikipedia.org/wiki/Exponential_decay).

This means that after its made, a decision to burn tokens can be reverted, but the amount which can be recovered is exponentially reduced with time.

IBY Token Technical Specification & Distribution

ERC20

I Bet You’s IBY is an ERC20 token as defined in https://eips.ethereum.org/EIPS/eip-20

Locking

A certain amount of IBY tokens will be locked at launch. These tokens will belong to advisors and founders. They will be unlocked through a “cliff and linear unlock” mechanism where:

● CLIFF - a set amount of blocks where tokens are completely locked. No tokens can be unlocked during this period

● LINEAR UNLOCK - an interval from starting block Bs to end block Bd through which the tokens are unlocked on a linear schedule - block by block.

Permissionless

The tokens must be permissionless - they cannot require any special approval from an “admin” account or have any option of “freezing”.

TODO:

Farming strategies
Compound, Alpha Homorra integration, xchain

Authors:
Edi Sinovčić Luka Sučić Marko Ivanković