FansUnite Protocol

R. Cimoszko*, M. Miorelli, and S. Rothwell

Friday 11th May, 2018
version 1.1

Abstract

We introduce a protocol for sports wagering of any ERC20 token using the Ethereum blockchain and InterPlanetary File System (IPFS). The resolution of wagers and the provision of event data is crowd-sourced by a decentralized network of oracles incentivised to provide low-latency and accurate sports data. The FansUnite protocol will serve as the building blocks for decentralized applications (DApps) that span traditional bookmaking, betting exchanges, and a variety of other services reliant on sports data. These DApps will be free to charge fees, utilize their own tokens for betting, and be able to set market prices based on full market information transparently available on the blockchain. The FansUnite protocol will provide the infrastructure to allow for parties to engage in sports wagering in a completely trustless and distributed fashion without the need for traditional intermediaries.

*ryan@fansunite.io

Contents

1 FansUnite: The Protocol for Betting ERC20 Tokens

In the sports betting industry, where bettors are primarily concerned with competitive pricing and having their transactions safely processed, trust is paramount. These user concerns have led to a gradual consolidation of betting transaction volume with a few established operators. These operators are able to leverage their disproportionate share of market information to efficiently price markets and further entrench their position as market leaders. This consolidation has several negative consequences for consumers, including reduced choice for bettors, discouraged new market entrants and stifled innovation.

Blockchains are able to facilitate trust in financial transactions between arms length parties. The advent of smart contracts made it simple to hold funds from various parties in escrow and disburse them programmatically based on a given outcome. These developments will soon transition the gaming industry from trust in operators to trust in the programmatic execution of code; however, one chief obstacle remains in the way of a disintermediated sports betting solution on Ethereum - the handling of real-world information.

The Ethereum Virtual Machine (EVM) is effectively a walled garden; a closed ecosystem in which all operations are controlled by the ecosystem operator, with no access to external data. Smart contracts have no ability to access information from the outside world without the use of trusted intermediaries known as Oracles. This represents a significant hurdle to decentralized applications (DApps) in sports betting, as sporting event data must somehow be ported on-chain.

Numerous DApps have attempted to create sports betting tokens on Ethereum, but none have provided a solution that successfully disintermediate sports betting from relying on centralized external data providers or balances access to market information. These projects all propose proprietary and application-specific implementations that fragment network effects and duplicate developer efforts. These platforms each independently post event and market information to the Ethereum blockchain, resulting in a large duplication of similar requests that clogs the network.

FansUnite solves these observed deficiencies with a free and token-agnostic protocol for decentralized sports betting. We propose a solution that allows for applications to utilize and accept bets in their own currencies, while still leveraging a public library of smart contracts, standardized data structures, and a network of independently operated oracles incentivized to provide low-latency sports data. Furthermore, we achieve a efficient marketplace by providing developer tools to aggregate market information of all DApps on the protocol. The protocol provides equal access to market information and encourages competitive pricing amongst operators. As such, the protocol encourages new market entrants who are better able to predict markets and offer lower fees. This protocol is a necessary step towards unifying developer efforts behind a community governed and disintermediated sports betting solution on the Ethereum blockchain.

2 Core Components

The FansUnite Protocol is comprised of the following:

- Solidity Smart Contracts
- InterPlanetary File System (IPFS)
- Developer Tools - FansUnite.js JavaScript Library
- Event listing and Oracle management DApp

2.1 Smart Contracts

The primary feature of the FansUnite protocol is the use of the blockchain to facilitate decentralized betting between two parties that is secure, transparent and fair. The Ethereum blockchain is used as a distributed database to store the required information used to settle a bet. Smart contracts are also used to escrow enough tokens to cover each party's risk, guaranteeing sufficient tokens for all winning parties can be paid out.

If blockchain-based betting is to be fully decentralized, a single point of truth cannot be used to provide the outcome of an event. The FansUnite Protocol will utilize a network of independently operated oracles to provide the data that will ultimately grade a bet. These oracles will be incentivised and rewarded for providing fast and accurate data, with the reward based on the betting volume of that specific event. These oracles will, through smart contracts, stake FAN tokens, earning or losing reputation dependent on whether their data is ultimately correct or not.

A dispute system will also be developed using smart contracts. This will allow bettors to challenge the result provided by the oracle network if

they believe an event was resolved incorrectly. FAN Tokens are required to be staked to initiate a dispute. For more information see Section 3.5.3.

2.2 InterPlanetary File System (IPFS)

InterPlanetary File System (IPFS) is a peer-to-peer distributed file system. Storage on IPFS is significantly cheaper than storing on the the Ethereum blockchain and allows the protocol to record metadata about participants, events and leagues that are not critical to grading a bet. Data on IPFS is referenced by a unique cryptographic hash that will be stored on the blockchain. The additional metadata will be a resource for developers to create content rich DApps.

2.3 Developer Tools - FansUnite.js JavaScript Library

A JavaScript library (FansUnite.js) is being created which is an abstraction layer to easily interact with the underlying smart contracts. The JavaScript library will aid in fostering and accelerating adoption of the protocol.

2.4 Event Listing and Oracle Network DApp

The large diverse oracle network will allow for a fully decentralized system. To facilitate initial adoption, a DApp is being released that will let anyone easily create an oracle, manually stake, provide data and earn a percentage of ERC20 tokens that is wagered through the protocol. Additionally, this DApp will list all the created events/markets and allow for anyone to create new events/markets.

3 Technical Procedures

3.1 Event and Custom Market Creation

Every bet must reference one or more events (or a custom market) as the outcome of events is used for grading the bets. The event schema has been designed to accommodate all sporting events and is defined in the Table 1.

Critical parameters required to grade all the various bet types are recorded on the smart contract. Additional parameters that help describe the event are stored using IPFS and cryptographically linked to the event on the smart contract.

Events can be created by sending transactions directly to the smart contract. However, to drive initial adoption and reduce friction for operators and non-technical users, a DApp will be released utilizing our abstraction layer (FansUnite.js), that will allow for easy interaction with the smart contracts.

3.1.1 Minimizing Duplicate Events

Operators are encouraged to use events that have already been created, thus reducing their cost while simultaneously minimizing strain on the Ethereum network. The event listing DApp will display all events already created, helping operators minimize the likelihood of creating duplicate events. The long term goal is for all parties on the protocol (bettors, operators, & oracles) to always agree on the exact same event. Additional functionality will be built into the event smart contracts to reduce the likelihood of duplicate events. This additional functionality will be achieved by creating a unique ID hashing specific parameters such as participant IDs and event start time and preventing a duplication of that ID from being stored.

3.1.2 Custom Markets

In order to satisfy all markets that a traditional bookmaker offers, the protocol must allow for custom markets to be created. Custom markets traditionally will fall into one of the following categories:

- Binary outcome - Whether a given outcome does or does not occur; and
- Multiple outcomes - Which outcome out of a list of outcomes occurs.

Custom markets can be associated with sporting events, but it is not necessary. The creator of the custom market will also have to provide the possible outcomes.

3.2 Bet Submission

Decentralized bookmaking shares many characteristics with decentralized trading. As such, components of the FansUnite protocol are inspired by the 0x project. Bookmakers offer betting odds on a plethora of markets and events, with odds that constantly change depending on the amount that is wagered on each side. With the current state of Ethereum, storing these changing odds on the blockchain is costly and inefficient. Inspired by 0x, a messaging system using off-chain relaying and on-chain submission/settlement will be implemented that will allow for bookmakers to offer traditional markets and their associated dynamic odds and betting exchanges to display all offers prior to submitting them to the blockchain. All bets are ultimately

Table 1: Event Schema Definition

Parameter	Data Type	Constraint	Description
participantIds	bytes32 (array)	required	List of all participants IDs competing in the event
minPeriods	uint	required	The minimum number of periods of an event
maxPeriods	uint	required	The maximum number of periods of an event
startTime	uint	required	Time and date of when the event begins
extraPeriodsBeginAt	uint	optional	The period at which extra time / overtime begins
shootout	boolean	required	An indicator for whether the event can potentially end in a shootout if the score is equal after the maxPeriods
draw	boolean	required	An indicator for whether an event can potentially end in a draw
homeParticipant	bytes32	optional	The participant id of the home team
leagueId	bytes32	optional	The ID of the league of which the event is part of
sportId	bytes32	optional	The ID of the sport of which the event is part of

submitted to the betting smart contract and tokens are automatically escrowed.

3.2.1 Messaging System

The messaging format outlined in Table 2 allows for any ERC20 token to be wagered and is applicable for both traditional bookmaking (such as Bet365) and betting exchanges (such as BetFair). For any bet to occur, it requires a **Backer** and a **Layer**. In traditional bookmaking the customer is the Backer (bets that an outcome will occur) and the bookmaker is the Layer (bets that the outcome will not occur). Betting exchanges allow for the opportunity for anyone to be the Layer or Backer. The ability to service both use cases is important for the long term success and adoption of trustless decentralized sports betting.

Traditional Bookmaking

The process for a bookmaker to accept a bet is outlined below:

1. Backer (bettor) and Layer (bookmaker) approves the betting contract to access their balance of the token they want to wager.
2. Backer creates a message (message format shown in Table 3.2) and signs the order with their private key.
3. Backer broadcasts the message to the Layer.
4. Layer receives the message and confirms that the parameters are consistent with their offerings.
5. If the parameters match, then the Layer submits the message to the betting contract.

This approach allows bookmakers to offer a traditional experience with each bet ultimately being submitted and settled on the blockchain. Gas fees will be incurred by the bookmaker, thus not requiring the customer to pay additional gas fees and hold Ether (ETH).

Betting Exchanges

The process for a betting exchange to facilitate a bet is outlined below:

1. The Betting Exchange cites a fee schedule and the address they will use to collect transaction fees.
2. Backer creates a bet, sets the backerFee and layerFee to values that satisfy the Betting Exchange's fee schedule, sets the feeRecipient to the Betting Exchanges desired receiving address and signs the bet with their private key.
3. Backer transmits the signed bet to the Betting Exchange.
4. The Betting Exchange receives the bet, checks that the bet is valid and that it provides the required fees. If the bet is invalid and does not meet the Betting Exchange's requirements, the bet is rejected. If it is satisfactory, the Betting Exchange posts the bet for layers to accept.
5. Layers receive an updated version of all available bets.
6. Layer matches the Backer's bet by submitting it to the betting smart contract.

Table 2: Bet Messaging Format

Parameter	Data Type	Constraint	Description
backer	address	required	Ethereum address of backer
layer	address	required	Ethereum address of layer
backerToken	address	required	Ethereum Address of the ERC20 token that the backer is staking
layerToken	address	required	Ethereum Address of the ERC20 token that the layer is staking
feeRecipient	address	optional	Ethereum Address of Betting Exchange that is charging the transaction fee
betContractAddress	address	required	The betting smart contract address
backerFee	uint	optional	How many FAN tokens the backer will pay the feeRecipient (betting exchange)
layerFee	uint	optional	How many FAN tokens the layer will pay the feeRecipient (betting exchange)
backerTokenStake	unit	required	How many ERC20 tokens the backer is staking
layerTokenStake	unit	required	How many ERC20 tokens the layer is staking
period	unit	required	There period at which the bet ends at, such as 1st period, 1st half, or full game
eventId	bytes32	required	Event ID hash
betType	uint	required	Type of bet, such as moneyline, spread, total points or team total points
participantId	uint	optional	Participant ID (required for moneyline, spread and team total bet types)
points	uint	optional	Required for total points and team total point bet types
overUnder	uint	optional	1 = over, 0 = under
expiration	uint	required	Time at which the offer expires.
resultIfDraw	uint	required	Identifies whether a draw results in a win, loss or push.
V	uint8	required	Elliptic Curve Digital Signature Algorithm (ECDSA) signature of the above arguments
R	bytes32		
S	bytes32		

This approach allows licensed operators to setup a betting exchange and earn fees with a low barrier to entry. The betting exchanges can compete by setting their own fee schedule and developing UI that can attract customers.

3.2.2 Token Escrow Calculation

The number of tokens that is required to be escrowed into the smart contract when a bet is made is equivalent to the exposure that that address is subject to. For a sports book operator who often lays both sides of a bet, it is not necessary to escrow tokens to match every wager. As long as they have escrowed enough tokens to cover their exposure, all parties are guaranteed to be paid out. Exposure can be defined as

$$E = \mathcal{P}_{\phi_{\max}} - \sum_{\phi \neq \phi_{\max}} \mathcal{B}_{\phi}, \tag{1}$$

where ϕ represents the set of possible outcomes, e.g. $\phi \in \{\text{win}, \text{draw}, \text{lose}\}$, and $\mathcal{B}_{\phi}$ and $\mathcal{P}_{\phi}$ are defined as

$$\mathcal{B}_{\phi} = \sum_{i=1}^{N_B} \delta_{\phi_i \phi} x_i, \tag{2}$$

$$\mathcal{P}_{\phi} = \sum_{i=1}^{N_B} \delta_{\phi_i \phi} \left(D_{\phi} - 1\right) x_i, \tag{3}$$

and are the betting volume and the total payout for outcome ϕ, respectively. Equations (2) and (3) depend on the bet amount x_i bet on the possible outcome ϕ_i by a bettor i and from the odds D_{ϕ} of the possible outcome ϕ. The total number of bettors is denoted as N_B and $\phi_{\max}$ is the outcome with the maximum payout, i.e.,

$$\phi_{\max} = \underset{\phi}{\operatorname{argmax}}\, \mathcal{P}. \tag{4}$$

A running total of each party's exposure will be calculated and recorded on the smart contracts. If a party is attempting to lay a bet and the total tokens escrowed into the betting contract is below their total exposure, the additional tokens needed will automatically be escrowed from their wallet. If their wallet does not have the required tokens or allowance, then the bet will not be accepted and any actions associated with that account is halted.

3.2.3 Partial Matching

With betting exchanges, bettors may not always want to lay/back the entire offer which would result in offers being partially matched. The ability to allow for partial matching is a critical function for betting exchanges. The FansUnite Protocol allows for partial matching by requiring the addition of a parameter to the smart contract which indicates the desired amount that the layer is wanting to match (layerTokenAmount). The betting smart contract will automatically update the amount of tokens remaining and allow for multiple offers to be matched until the offer has been satisfied or expired.

Additional functionality common with betting exchanges is the ability to cancel the remainder of a partially matched bet and the ability to change the offering odds to attract layers to match the remaining stake. Both these functionalities will be implemented in the betting smart contract. The original backer will have to send a transaction to the smart contract indicating they want to cancel or change the odds on the remaining offer and will be required to pay the appropriate gas.

3.2.4 KYC/AML Implementation

Although we feel strongly about full decentralization, we also believe there is a social responsibility to prevent underage gambling, money laundering, gambling addictions and adhere to gambling laws in each jurisdiction. Know Your Customer (KYC) and Anti-Money Laundering (AML) will be implemented into the protocol. There are many identity protocols that are under development, such as uPort or Civic, and depending on the maturity of their technology at the time of implementation, we will decide which solution makes sense. Properly validated addresses will have to be whitelisted on the betting contract in order for it to accept any transactions.

3.3 Event Resolution

A network of incentivized oracles are employed to provide the requisite event data for market resolution. To foster initial adoption, a DApp is being created which allows anyone to become an oracle and manually input event data for an event. Tech savvy developers can send event data directly to the smart contract or use our abstraction layer to programmatically send their results. The format of the event data depends on the parameters initially set for the event.

3.3.1 Oracle Staking and Reward

Oracles stake FAN tokens to be eligible to provide event data that is ultimately used to grade wagers. If an oracle provides data that is part

of the consensus and the data is deemed as the correct result, the FAN tokens staked by the oracle is returned and the oracle earns a reward proportional to the amount they staked, their reputation and the timeliness of the data provided. Oracles that provide data that is not part of the consensus and ultimately deemed as incorrect, will lose their FAN token stake and their reputability score will decrease. The tokens that were staked will be transferred to the Contingency Reserve Fund which is used as a final failsafe to pay out wagers that were graded incorrectly (see Section 3.5.2). The risk of losing tokens incentivises the oracles to act appropriately and provide accurate results. As betting volume increases, the total number of oracles and the number of tokens staked will increase and the network will become more distributed.

It is not clear what level of incentivization grows the network efficiently. Too much incentivization adds unnecessary tax to the Ethereum network, while too little incentive might result in a quorum that is not resistant to false-positives.

Our approach is to optimize the incentivization so to maximize the average and minimize the variance of the reputabilities distribution of the whole pool of oracles. For this purpose, to each oracle will be associated a reputability which reflects the accuracy of the information provided by the oracle with respect to the quorum in past resolutions. When participating in the resolution of an event, an oracle i will have to stake an amount of tokens x_i inversely proportional to its reputability probability P_i, i.e.,

$$x_i = \frac{r}{P_i^\kappa}, \tag{5}$$

where r is the reward for providing correct information and κ is a numerical coefficient which tunes the penalization level for the low reputation oracles. The reward amount r will be determined once the betting volume V for the event is known as

$$r = \frac{\beta V}{N_o}, \tag{6}$$

with N_o the number of oracles and β representing the percentage of the total betting volume rewarded to the pool of N_o oracles. From Eqs. (5) and (6) it is easy to see the ROI for an oracle providing the correct information is equivalent to its reputability percentage. Since the reward amount is fixed by Eq. (6) for an event, low reputability oracles will be required to stake more.

The number of oracles N_o in the quorum is selected using the Kolmogorov-Smirnov (KS) test between a sample from the oracles reputability distribution and the whole distribution itself. To show how this works in practice, in the left panel of Figure 1 we assume the reputability distribution to be normal in the interval $[0, 1]$ with mean 0.95 and variance 0.3.

In the right panel we plot the KS statistic between the full reputability probability distribution (left panel) and distributions of varying sample size N_o. The band represents the 95% CI for the KS statistic and has been obtained using repeated simulations. A very low KS statistic means we cannot confidently reject the null hypothesis, i.e., the hypothesis the two distributions are the same. From Figure 1 we see that the KS statistic converges as we increase the sample size. Although the more oracles, the better, at $N_o \approx 100$ we already have a reasonable convergence and the p-value is large enough so that we cannot confidently reject the null hypothesis. Based on the results of Figure 1, for the next simulations we choose $N_o = 100$ as the minimum number of oracles.

Once the number of oracles N_o and the volume of the event are set, the stake amounts are determined using Eqs. (5) and (6). To investigate the possible ROI of an oracle, consider that the total betting volume for March Madness in 2017 was estimated to be around 10 billion USD, meaning roughly 140 million USD per game. Considering that the last games probably had more volume than the first, we can assume a varying betting volume per game between 100-200 million USD. Setting the amount of rewards to be split between the oracles to be of 0.25% of the total betting volume of the game, and setting $\kappa = 2$ and $N_o = 100$, in Figure 2 we show the ROI in USD for oracles with different reputabilities. The curves have been obtained via Monte Carlo simulations and the bands represent the 95% CI around the mean.

Figure 2 shows how the oracles are incentivized to maintain high reputability scores: oracles with low reputation will, in the long run, have lower, if not negative, ROIs. For oracles with high reputability the variance around the mean ROI is minimized and a steady gain is realized in the long run.

Time Incentivisation Rewards

To ensure timeliness in the resolution of the events and grading of bets, each oracle will receive a bonus reward b_i dependent on its response time t_i. If we denote the total time bonus reward to be distributed between the oracles with B_T, then we have to satisfy the constraint

$$\sum_{i=1}^{N_o} b_i = \alpha \sum_{i=1}^{N_o} f(t_i) = B_T,$$

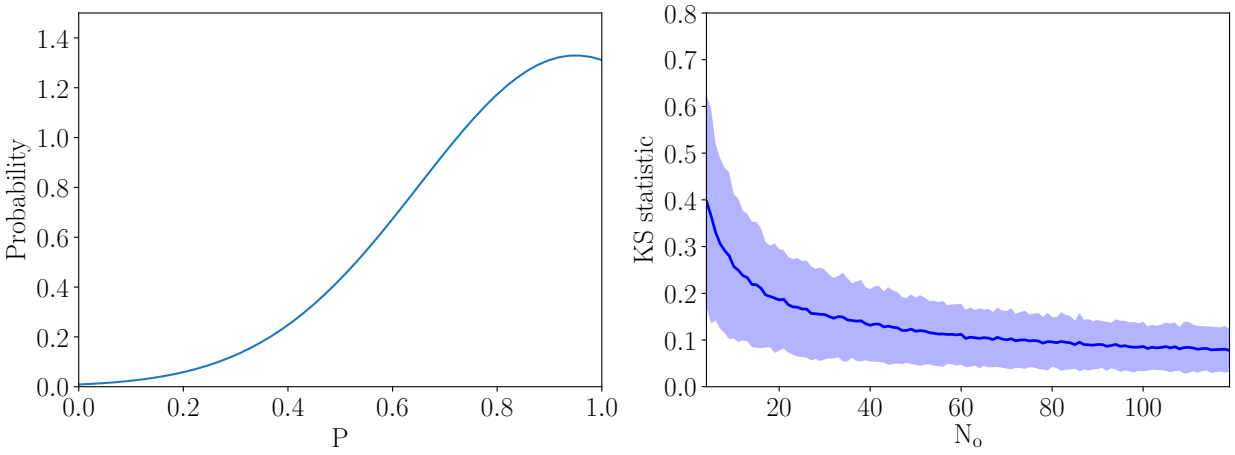

Figure 1: *Left panel* – Assumed reputability probability distribution for the whole oracles'pool. *Right panel* – KS statistic between the probability distribution in the left panel and different distributions sampled randomly from the latter with varying sample size N_o. The colored band represents 95% CI around the KS statistic value.

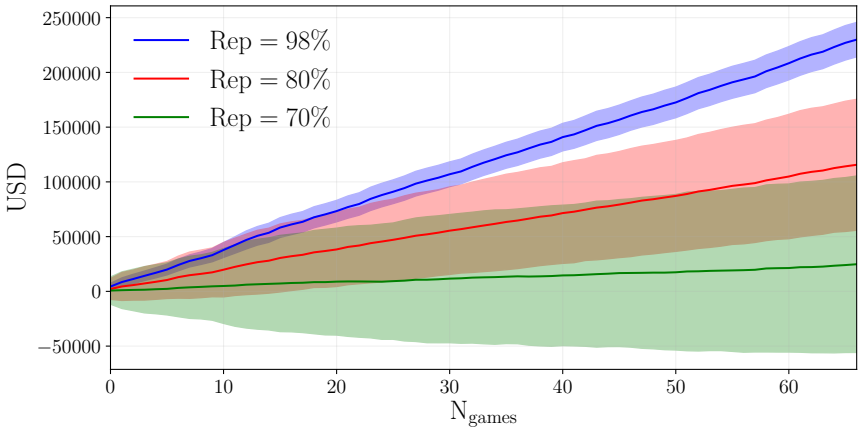

Figure 2: Possible ROI for stakers with different reputability as a function the number of games served as oracles. The solid lines represent averages, while the bands are the 95% CI.

where we assumed $b_i = \alpha f(t_i)$. The time bonus for an oracle i is then found to be

$$b_i = \frac{f(t_i)}{\sum_{i=1}^{N_o} f(t_i)} B_T,$$

where different functional forms of $f(t_i)$ can be used to tune the proportion of reward between fast and slow oracles (e.g., one can use a decreasing exponential or an inverted logistic function).

3.3.2 Oracle Consensus

Consensus is not determined solely via majority vote. The oracles' votes will be weighted by their reputability score. This prevents a batch of M particularly bad oracles to prevail on a smaller batch of $N < M$ of very good ones. If we have two possible outcomes $\mathrm{o} = A, B$, and oracles vote for either $\mathrm{o} = A$ or $\mathrm{o} = B$, a consensus score is calculated for both groups of oracles as

$$P_{\mathrm{o}} = \frac{\sum_{i=1}^{N_o} \delta_{\mathrm{o},\mathrm{o}_i} P_i}{\sum_{i=1}^{N_o} P_i}. \tag{7}$$

The consensus is then reached if any of the two scores is more than a threshold γ which can be set, for example, at 80%.

3.3.3 Oracle Reputation

To each oracle will be associated a reputability. The reputability probability P_i for an oracle i is calculated as the ratio between the number of correct information C_i provided by the oracle itself and the total number of times served as an oracle N_i. For the calculation of this ratio, only a limited number N_i of the most recent previous resolutions will be considered and P_i will be updated after each resolution. To incentivize oracles to provide correct information, incorrect information I_i will be weighted more in the reputability calculation. Specifically, the reputability probability can be calculated via

$$P_i = \frac{C_i}{N_i} = \frac{N_i - \gamma I_i}{N_i},$$

where $\gamma \geq 1$ is a weight which tunes how much the reputability of incorrect oracles is penalized.

3.4 Bet Grading

The betting smart contract records the following for each address:

- Token Balance - The total number of tokens that the account has escrowed and/or won; and
- Exposure - The total number of tokens that the account is potentially liable for (see Section 3.2.2);

Once consensus has been reached by the selection of oracles, the outcome of the event is determined and the event will be flagged as resolved. Parties will then be able to send a transaction to the smart contract that would grade their bet and update the token balance and exposure for the two parties involved in that bet. The smart contract will contain all the logic to correctly determine the rightful winner based on the bet type, event parameters and the data provided by the oracle consensus.

3.4.1 Bet Resolution Library

Bets, in their simplest form, fall into one of the following three types:

- Boolean - The outcome occurs or does not occur;
- Differential - The difference between two values and whether it was greater or less than; and
- Totals - The summation of values and whether it was over or under.

Common smart contract libraries will be used to perform routine operations such as grading wagers. The use of only one Bet Resolution smart contract library ensures that bets are graded identically and transparently across the entire protocol. Library deployments will be structured in a way that maintains the ability to upgrade them. This is important as Solidity is still a relatively immature language and future solutions may be more efficient. Additionally, maintaining the ability to upgrade will allow for more sophisticated bet types to be implemented as changes in sports betting are inevitable. Deployment of new libraries will be governed by token holders as discussed in Section 4.1.1.

3.5 Incorrectly Graded Wagers

The FansUnite Protocol allows parties to grade bets and update token balances at the moment oracles reach consensus. This approach emphasizes speed of bet resolution, which is critical for sports bettors. However, there will be scenarios when the results for an event are wrongfully reported and bets are incorrectly graded. A dispute process will be used to handle incorrectly graded bets. Parties will be subject to a withdrawal allowance that will reduce the severity of bad actors and a Contingency Reserve Fund will be used as a last resort fail-safe to guarantee rightful winners are made whole.

3.5.1 Dispute Process

Betting parties have a window of 48 hours to initiate a dispute. After this 48 hour window, the event will be deemed final and the oracles stakes will be returned. To initiate a dispute, the disputer or disputers must suggest a result they believe to be correct and stake a total amount of FAN tokens equal to the current oracle reward for that event. Any bets that have not yet been graded on that event will be temporarily halted from being graded until the dispute has been resolved to prevent others from wrongfully claiming bets. Once a dispute has been initiated, a new selection of oracles will be notified to provide results for the event. The staked FAN tokens from the disputers is used as a potential reward for the new selection of oracles. Consensus is reached using the same methodology described in Section 3.2.

If the consensus from the new selection of oracles is equal to the suggested result from the disputers, the dispute will be deemed successful and the following will occur:

- Disputers stakes are returned.
- Reward for the event is distributed to the new selection of oracles.
- The initial oracle selection will lose their stakes for providing incorrect results. These stakes are transferred to the Contingency Reserve Fund.
- The initial oracles will lose reputation.
- Parties that were involved with the incorrectly graded bets will be able to regrade their bets which will update their token balances and exposure.

If the consensus from the new selection of oracles equals the same results from the original consensus, the dispute will be deemed as unsuccessful and the following will occur:

- Disputers stakes are distributed to the new oracles as a reward.
- Reward from the original oracle selection remains.
- Original oracle selection does not lose any reputation.
- Disputers lose reputation.

Alternatively, if the consensus from the new selection of oracles does not equal the results from the original consensus and also does not match the result suggested by the disputers, the dispute will also be deemed as unsuccessful. The result from the new selection of oracles will be considered the correct result and will be used to grade the wagers.

3.5.2 Withdrawal Allowance

Once the event has exceeded the 48 hour dispute time-frame, the entire winnings/initial stake can be withdrawn from the betting smart contract. Prior to the 48 hour dispute time-frame, the number of tokens that can be withdrawn from a winning bet is calculated using the function:

$$f(t) = \frac{1}{1 + e^{-\lambda(t - t_{1/2})}}, \tag{8}$$

where the parameter λ controls the steepness of the curve and $t_{1/2}$ is the time at which half of the winning can be withdrawn.

Although the bettor is unable to withdraw the entire amount of winning tokens prior to the 48-hour dispute time-frame, they can still continue to bet their winnings. If at any time a dispute arises, the party will be subject to additional exposure and will be required to have a token balance that is greater than or equal to their exposure. The betting smart contract will automatically escrow additional tokens if the the owner has provided the required allowance and that the owner also has enough tokens in their wallet. A party must always have a token balance greater than their exposure in order to participate in the network.

3.5.3 Contingency Reserve Fund (CRF)

There is the case that a party withdrew a portion of tokens from a bet that was disputed and ultimately regraded and the party is unwilling or unable to escrow additional tokens to cover their exposure. A Contingency Reserve Fund (CRF) will be used to make wrongful parties whole in this scenario. The CRF will be initially seeded and further contributed to by loss stakes from oracles that provide incorrect data.

4 FAN Token

The FAN token will serve the following purposes:

- Decentralized governance on protocol decisions;
- Betting exchange transaction fees;
- The required token for staking by oracles;
- Resolution dispute staking;
- Contingency reserve fund; and

- Incentivise data oracles to provide accurate and low latency sports results.

4.1 Governance

Decentralized governance is an integral part of the longevity and success of the protocol. Token holders will have a proportional vote on development updates.

4.1.1 Protocol Upgrades

Periodic upgrades and improvements to the smart contracts powering the underlying logic of the protocol are necessary. Ethereum smart contracts are immutable and once deployed, the code cannot be changed. However, several smart contracts can be combined and structured in a way that allows for the underlying logic to be upgraded. Upgrading must be done in a fair and transparent way that doesn't negatively impact stakeholders. As such, a voting system will be implemented to decide how and when new code will be implemented and deployed.

On-chain decentralized governance is still a heavily researched subject. While governance is an important component of the protocol, it is also not part of our core business and we will lean on the expertise of a partner organization such as Aragon or Harbour for proper implementation.

Community governance will be undertaken in phases with a gradual roll-out of functionality to the Community. These phases will be announced at a later date.

4.2 Betting Exchange Transaction Fees

The protocol allows for DApps, specifically betting exchanges, to charge fees in FAN tokens. DApps can cite their own fee schedule using any sort of method they choose, such as a flat rate or a rate dependent on the volume wagered. Although the protocol allows for fees, betting exchange DApps aren't required to charge transaction fees.

4.3 Resolution Dispute Staking

FAN tokens are required to be staked to initiate a dispute if one of the wagering parties believe that the oracle network reached an incorrect consensus. Staking is used for a potential reward for oracles/arbiters who provided the correct data and for the prevention of frivolous disputes.

4.4 Contingency Reserve Fund

FAN tokens are accumulated in a Contingency Reserve Fund (CRF) to pay out successful disputes where one party decides not to return wrongfully claimed tokens. The reputation system is used to deter bad actors while the CRF is only used as a fail-safe.

4.5 Oracle Staking

Oracles are required to stake FAN tokens to be eligible to provide event data that is ultimately used to grade wagers. Oracles earn a reward proportional to the amount of FAN tokens they staked, their reputation and the timeliness of the data provided. See Section 3.3.1 for more details.