Eventum: Platform for Decentralized Real-World Data Feeds

Martin Mikeln, Luka Perović

eventum.network

January 18th, 2018

v2.0

Abstract

The world has become data-driven, and access to real-time information, which often comes via external data feeds (i.e. APIs), is crucial. The problem is that enormous amount of real-world data all around us is never captured and turned into a real-time API. We present the design and implementation of a new platform, called Eventum, where the truth is sourced from the crowd and then automatically turned into a real-time API. Anyone in the world can get paid for what they see and experience, while developers get any data they want more cheaply and reliably. Eventum utilizes the wisdom of the crowd principle in a decentralized network of data providing and validation nodes with blockchain as a court system. We aim to offer a much-needed layer for secure, reliable and cheap data feeds which do not rely on a single source of truth.

Keywords: data feeds, real-time, API, wisdom of crowds, decentralized network, blockchain, consensus, smart contracts, Ethereum, validation nodes, Swarm

1. Problem

Data is the gold of the 21st century [1][2][3][4]. To survive and grow, modern companies need to be data-driven and have access to real-time information, which often comes via external data feeds (i.e., APIs).

The problem is that there is an enormous amount of real-world data all around us that can not be captured and turned into a real-time API. Whether we are shopping for a new pair of jeans, taking a selfie at a concert, reading the latest news online, watching a football game from our couch or scrolling through Twitter's newsfeed, we have access to invaluable amounts of data that is never used and impossible to monetize.

Moreover, the data that is available through data feeds is often unreliable and unverified, because it relies on a single data source susceptible to a high rate of errors and failures. Notable examples include: (1) the fake news detection systems, where it is impossible for a single human or an AI system to detect dubious news and hoaxes reliably; (2) a real-time content moderation system, where a comment that may seem appropriate to someone could be flagged as racist by someone else; (3) a reporter’s one-sided perspective of real-world events, such as concerts, tournaments, global conflicts, riots and protests.

Nearly all existing real-world data APIs are tightly controlled by a single company, which has complete control over the data and the prices. This means significant margins are included in the prices and the safety and integrity of data cannot be guaranteed. A good example are sports data APIs, which can exceed hundreds of thousands of dollars for one sport per season [5].

This presents a multi-billion dollar [6][7][8] gap, where on the one side there is enormous demand for cheap and reliable data feeds from every modern company, and on the other side there are more than 2 billion people with smartphones [9] and easy access to a significant part of this data, with no way of providing and monetizing it.

2. Solution

A solution to the problems outlined above is Eventum. Eventum is a decentralized platform where the truth is sourced from the crowd and then automatically turned into a real-time API. Reporters get paid for what they see and experience, while developers get any data they want more cheaply and reliably.

Wisdom of crowds [10][11] principle is used to determine the truth from multiple data sources (i.e. people reporting the data) instead of a single data source, which removes bias and single point of failure. This process is done in real-time by a decentralized network of reporting and validation nodes in the Eventum system. Asymmetrical cryptography and blockchain solutions, Ethereum [12], Swarm [13] and Whisper [14], are also used to provide a secure and fair system with free market economics.

Eventum is an open-sourced, extensible platform with an easy-to-use API interface and flexible SDK. Developers can get any real-world data in the form of an API while also being sure the data is verified, i.e. a result of consensus, for which rules are set by the developer himself. The data can not be stored or controlled by a 3rd party, and all funds are securely stored on the blockchain. Moreover, the price is only governed by free market economy, which means there are zero fees and margins on the price of data.

Our time is valuable and our observations and cognitive capabilities are far beyond what any artificial intelligence system is capable of. So why not make use of that and monetize what you see, do and experience. When you become a data provider, you are providing information via a desktop or mobile app and getting a reward if you are part of the consensus (i.e. agreeing with the majority of other reporters). You can also help the network and earn passive income by running a validation node which functions as just-in-time masternode.

3. How it Works

Eventum's system is a bridge between data providers and developers. Data providers have access to data that is valuable, which they then report via a mobile/desktop app to the validation nodes in the Eventum system. Validation nodes then wait for a consensus to be reached (i.e. multiple data providers send the same information) and then send this data to the developer in the form of a real-time API. The developer locks a reward in a smart contract, which is then split and given to data providers that were part of the consensus. The reward is split proportionally to the speed at which the data was provided, so the real-time nature of the API is incentivized.

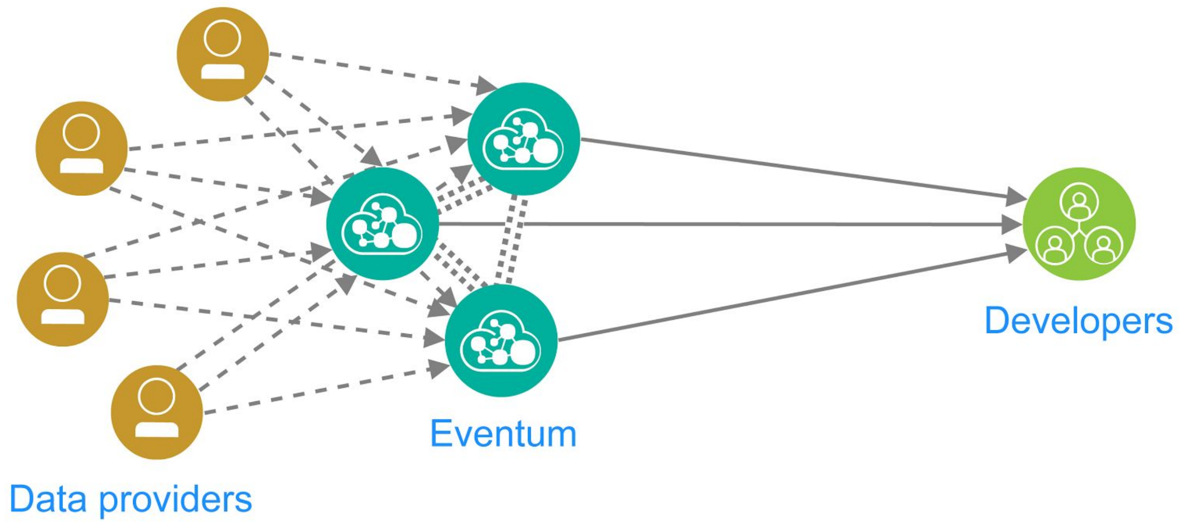

Figure 1. Data flow through the Eventum network

4. Architectural Overview

Eventum is a platform built on top of a decentralized network of reporting and validation nodes with blockchain as a court system [15]. The platform consists of 3 layers: the core layer, the services layer and the application layer. Both the decentralized network of nodes, which provides the real-time speed, and blockchain, which provides the security and payment solution, are tightly integrated into all 3 layers.

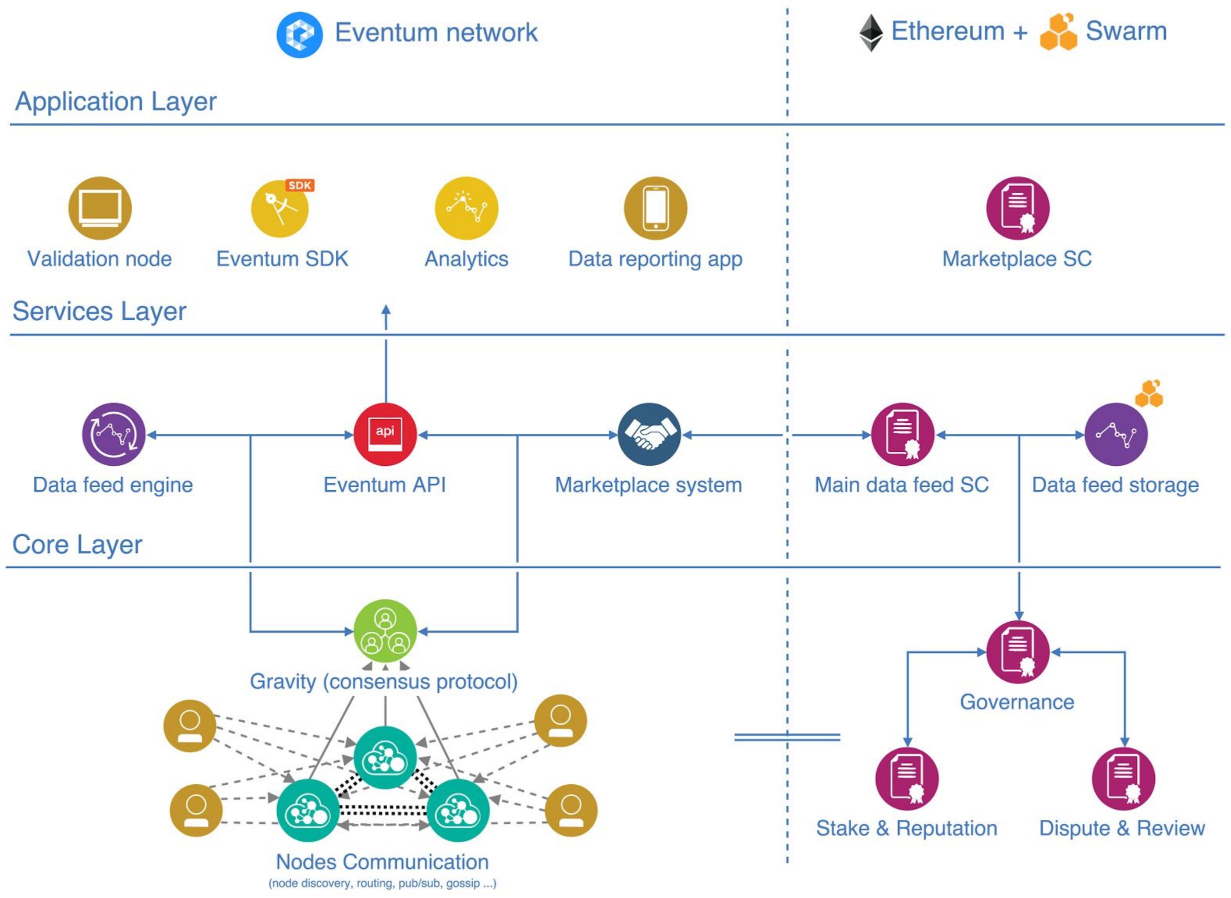

Figure 2. 3-layer architecture with decentralized Eventum network + Ethereum and Swarm

The blockchain is necessary for the platform to be safe and fair. It is necessary for the free market economy to work, for funds to be safely stored and distributed, for the dispute mechanism to be transparent, for the reputation system to function and for the consensus rules to be safely stored. The blockchain is also responsible for decentralized governance, which is built in the core layer and is overseeing all the changes to the system. Eventum is blockchain agnostic and can be used with any blockchain that supports smart contracts, such as Ethereum, Neo [16] and Lisk [17]. The Alpha version is currently based on Ethereum smart contracts.

While all the sensitive information is stored on the blockchain, the real-time nature of Eventum requires hundreds of thousands or even millions of transactions per second. This is not possible on the blockchain itself, even with the scaling solutions like sidechains [18] and sharding [19]. That is why data reporting, consensus building, timing and reward calculations, discovery protocols and routing is done off-chain in a decentralized network of Eventum nodes. All the data is encrypted and nodes are always required to agree on the data they receive and send (i.e. by reaching intermediate consensuses) so that malicious nodes are quickly detected and ignored.

5. Nodes

5.1. Validation Nodes

Validation nodes represent the core part of the decentralized Eventum network and are responsible for capturing the data, building the consensus and delivering the data to developers. Validation nodes earn fees for every piece of data that is correctly processed, while also being involved in the governance of the system, where each node's vote is weighted by its reputation. They are required to stake a certain number of EVT tokens for the duration of the market (including the dispute time) they are validating and thus act as just-in-time masternodes. They are lightweight and they efficiently run on mobile devices.

To prevent malicious actions, validation nodes are selected randomly just before the market is created, they process encrypted data and are part of the reputation system. Because the token distribution model is fair (personal caps), centralization is also limited.

Communication with other nodes is a crucial part of the system and is based on an enhanced version of gossip algorithm [20], where consensus needs to be reached in multiple steps of the protocol. A similar mechanic is also used for the node discovery and routing, while Swarm and smart contracts are used for storing other data (e.g. nodes list, current data feeds, consensus rules, categories and descriptions). When a stable version of Whisper (communication protocol for DApps) will be implemented, all contract to contract communications will be done this way.

5.2. Data Providing Nodes

Data reporting nodes have a limited amount of blockchain communication and unidirectional communication to validation nodes. They are only responsible for providing the data to the system.

6. Smart Contracts

Smart contracts are responsible for the safety and fairness of Eventum and are used for staking, holding the funds, reputation system, dispute, review mechanism, and governance. At the creation of the market, the developer locks in his reward together with the hash of the consensus rules (which are saved on Swarm). After the selection process, first, the validation nodes are added to the contract and then the data providers. Another set of contracts is used for keeping and updating the reputation score of all the nodes in the system and another set for the dispute and review mechanism. Changes to the core layer of Eventum's architecture are done via a decentralized voting system implemented in the set of contracts responsible for the governance of the system.

7. Authentication

While Eventum does not enforce any authentication apart from the private/public key associated with each node, it does support authentication identifiers, which can be used throughout the system. This means that authentication can be required for some (or all) parts of the system, which can be especially useful for guaranteeing uniqueness of the votes from the data providers. Authentication can also be useful when a special set of data providers are required (e.g. only doctors can join a specific market). If possible, a decentralized authentication and identity system (e.g. Civic [21]) should be used.

If an authentication identifier is used, its hash (which represents the person and its linked "roles") is used to generate new Eventum identity. This information is linked together with a node's address, public cryptographic key and other information needed for effective communication inside the system and written to a public nodes list (which is distributed to all the nodes in the system). The integrity of data can be independently verified by each client simply by comparing its node list with the list from other clients. The identity of users can also be verified by doing a back check of the public identifier with the identity platform. Automatic identity and integrity checks are built-in to the system and every information change (e.g. new node's location) can only be performed by the owner of the public key. The change itself is propagated via gossip algorithm to other clients.

8. Market Creation

To receive data from Eventum, a developer first needs to define what kind of data he wants, what are the rules for the consensus and how much he is willing to pay for it. This process is called market generation. Each created market can consist of one or more data feeds, where each data feed can be of a single-event type or multiple-event type. Data feed consists of one or more data fields that represent a single report or a decision that needs to be made by data providers.

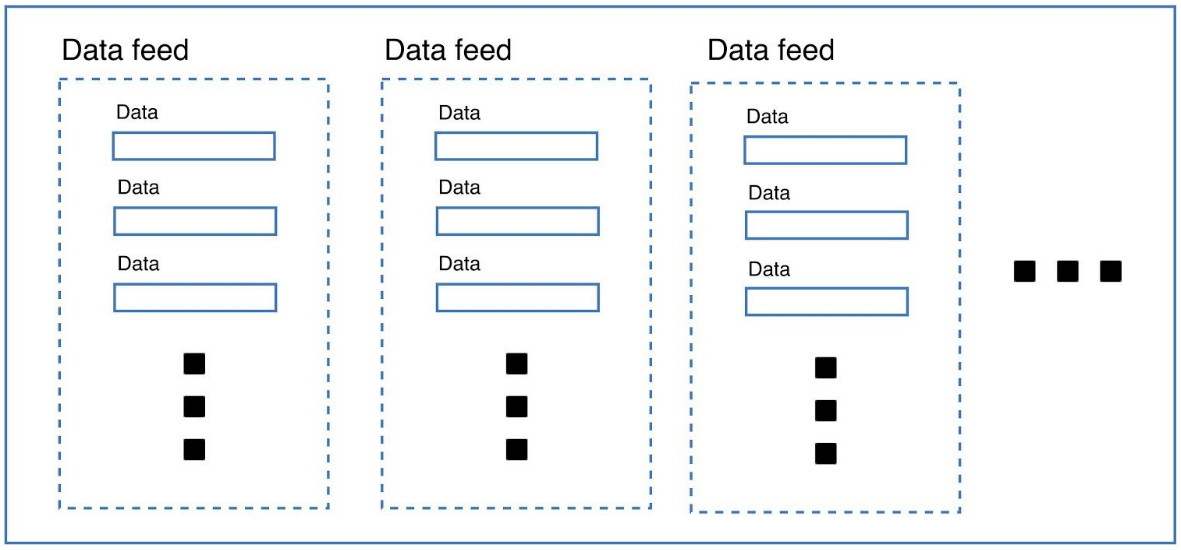

Figure 3. Each market can contain multiple data feeds on which consensuses need to be reached. Each consensus is reached on a collection of data fields.

Eventum's flexibility offers developers to keep their data feeds private (i.e. only they can unlock the data with the private key), to open them to everyone (i.e. they pay for the data but share the APIs output for free with everyone) or to share the data feed with other developers and charge an access fee. Sharing the data reduces duplicated data feeds and saves nodes' resources.

A significant obstacle in the development space is learning new data feed/API specifications and tailoring the existing systems to receive the data in the specified format. If the system needs to receive data from multiple APIs, the solution is usually a complex architecture which is prone to errors, inefficient and with long development cycles.

To solve this problem, Eventum uses a small set of predefined fields and leaves everything else to the developer. The market definition is simply a JSON specification of the consensus rules and data fields, while the data received from validation nodes comes directly to the webhook defined by the developer (and the structure is excactly as it was defined in the market definition). This means that developers only need to define data types and select consensus rules. A market definition can also be generated via a simple UI, which further simplifies the process.

Each data feed needs to be defined by one or more supported data types for providers to successfully reach a consensus. If multiple data fields are used, a consensus is only reached if all of them are identical. Basic supported data types are boolean, integer, float and string (case-insensitive). Developers can also use categorical data type which can hold a fixed number of basic data types and where the result is accepted when one or more of this data types have a value.

Basic data types:

`boolean`: the type of the built-in values True and False. Useful in conditional expressions, and anywhere else you want to represent the truth or falsity of some condition.
`integer`: integers with at least 32 bits of precision
`float`: floating-point numbers, equivalent to C doubles
string: case-insensitive string
categorical: a value that holds one or more (fixed) basic data types. At least one of the values needs to have a value.

9. Consensus Rules

The second set of data that needs to be included - and is, of course, the integral part of the system - contains the consensus rules and reporting parameters. Eventum has a built-in algorithm that always calculates and recommends the best setting for the parameters described below, based on current market statistics and type of data that is being reported, but the parameters can always be adjusted manually.

Some of the parameters are:

`min_people`: minimum number of people that must join the data feed,
`max_people`: maximum number of people that can join the data feed,
`min_consensus`: minimum number of people that must form a consensus,
`consensus_rule`: percentage of votes that needs to be the same = form a majority,

`consensus_mode`: set to "confirmations" or "final"*,
`reward`: the total reward for all participants that formed the consensus,
`min_validation_nodes`: minimum number of validation nodes that receive and process data from providers,
`start_dispute_time`: time spent after consensus was reached when information about the result is sent to a subset (defined by Z) of people that did not reach consensus,
`max_dispute_time`: time slot in which people have to open a dispute,
`market_join_bonus`: bonus that scales reward up for people that joined the data feed early,
`identity_hash`: authentication identifier which can indirectly impose limits for people joining the data feed, such as country, age, gender and reputation level
`K`: stake factor used in stake calculation (C/n*K) which can be paid by data providers when joining the data feed,
`category/category2/category3`: three tiers of categories selected for the data feed which influences data feed discoverability,
`summary/info/rules`: different identifiers that explains the information and rules about the data feed,
`Z`: cross validation factor which defines the portion of data providers that were not part of the consensus and will receive the result of consensus so a dispute can be opened immediately,
`time_to_join`: time until users can join the market,
`start_event`: defines the start of the reporting**,
`stop_voting`: time by which reporting should stop: this could be set by a percentage of voters (which needs to be higher than min_consensus) or by an absolute timestamp***

*By setting consensus_mode to "confirmations" (default), the developer can choose to get results immediately after the consensus is reached, while also receiving “confirmations” of the consensus (and possibly new consensus that replaces the old one if the majority changes). If consensus_mode is set to "final", he will only receive the final state when stop_voting is reached. The first option is set by default because consensuses changes are very rare and the speed with which the consensus is delivered is much faster in the default option.

**start_event could be dynamic and is by default defined with a median of timestamps from consensus votes. If stop_voting is defined with a timestamp, consensus can be reached and only then a stop_voting rule can be evaluated. If stop_voting prevents some votes to be included in the consensus, the consensus is not reached, and market ends; otherwise, the consensus is reached, and data is pushed to the developer.

The main parameters that define the consensus are min_consensus and consensus_rule. A consensus is reached when at least min_consensus of data providers which also needs to represent consensus_rule of the votes report the same data.

Example: `min_consensus = 10`, `consensus_rule = 2/3` -> Consensus is reached if 14 out of 20 votes report the same data, because `14 > 2/3 of 20 && 14 > 10`.

```
{
  "datafeeds":[
    {
      "name": "UCL Final - Ronaldo scores",
    "description": "Report when Ronaldo scores its first goal in UCL Final",
      "category1" : "Sports",
      "category2" : "Football",
      "category3" : "UCL",
      "min_people": 20,
      "max_people": 200,
      "min_consensus": 20,
      "consensus_rule": 0.66,
      "reward_amount": 0.03,
      "reward_currency": "ETH",
      "stake_K": 1,
      "cross_validate_Z": 0.1,
      "data_type": "boolean",
      "default_value": "None",
      "min_validation_nodes": 4,
      "start_dispute_time": 7200,
      "max_dispute_time": 86400,
      "market_join_bonus": 1.1,
      "identity_hash": "None",
      "reporting_timestamp": 1502944665,
      "time-to-join": 1502344665,
      "webhook_url": "https://example.com/eventum_api/ronaldo_scores"
      ....
    },
    {
      "name": "UCL Final - Yellow Card",
      ...
    }
  ]
}
```

Figure 4. Market definition

10. Reward Mechanism

Data providers report the "right" information only if they are rewarded when they provide the same information as the majority of voters. The reward is provided by the developer who pays for the verified, secure and real-time data he gets from the Eventum system.

Reward is locked in the smart contract when the market is created, together with the data feed parameters and consensus rules. The amount of the reward could be of any size but only competitive offers will get picked on the free market which is governed only by the supply and demand. Because figuring out which price is right is hard, Eventum automatically calculates the optimal price for the market from the data feed parameters, consensus rules and current prices in the system. This estimate can be re-adjusted to enter the ecosystem with more (or less) aggressive pricing.

Each data provider that is part of the consensus will on average receive a share of the reward equal to C/n, where n is the number of the data providers in the consensus. But because the reward is split proportionately to the speed by which the data was provided, provider's share is higher or lower than C/n. The reward's shares are distributed with an exponentially decreasing function. This incentivizes the real-time nature of the Eventum.

To stop people from guessing the outcome they need to report (and thus getting a bigger share of the reward for being "faster"), developers must define data to be reported with unique properties.

Example: Reporting a yellow card at the football match can be done at any time, so a better option is to request a yellow card event and the minute at which yellow card was assigned. An even better option is to report the footballer who received the card to.

11. Staking (Data Providers)

Providers can be free to join any market, they can be required to have a certain level of reputation score, they can be forced to stake a fixed amount of tokens or a hybrid approach can be used, where staking is only necessary when reputation score falls below the threshold. Same mechanics can be used to limit voting itself.

In both cases, if data providers act maliciously or vote differently than a majority, their stakes go to the reward pool.

If the staking mechanic is used, the size of the stake is `C/n*K`, where `C` is the total reward set by the developer, `n` is the current number of people that joined the data feed so far and `K` is the stake factor set by the developer (usually between 1 and 5).

To reduce timing speculations and market manipulations, developers can utilize the `market_join_bonus` parameter. It is a scaling factor for the reward that each user gets if they join the market before `min_people` is reached. `min_people` setting as well as inverse proportionality of the stakes and number of data providers (small number of providers pay higher stakes), prevents groups of people taking advantage of almost empty data feeds and trying to "force" majority without consequences.

12. Selection of Validation Nodes

When all the parameters are selected, the JSON is saved on the Swarm, while its hash and the reward gets locked to a smart contract on Ethereum. When `max_people` is reached or `time_to_join` passes, validation nodes are selected in just-in-time fashion and added to the smart contract. Because the validation nodes are not known in advance, neither developer nor data providers can influence them. Validation nodes are selected randomly (anyone has equal chance to be selected) via a blockchain function which is based on the last block hash. This is a secure method for random number generation in the Eventum' case because value of the market is usually far lower than the mining reward, a market is a short-lived thing, usually only lasting over a timespan of few blocks and because data providing nodes are not yet joined and so the market could not even reach min_people.

```
contract select_validation_node {
    /* Select a random validation node with an ID between 0 and 1000 */
    function randomNode(uint seed) constant returns (uint randomNumber) {
        return(uint(sha3(block.blockhash(block.number-1), seed ))%1000);
    }
 }
```

Figure 5. random node selection function in solidity

When validation node is selected, it needs to join the market in a specified time slot; otherwise, a search for a new node begins, until `min_validation_nodes` is reached. Data about the new market is then propagated through the network and notifications are sent to all the subscribed data providers.

13. Joining the Data Feed

Data providers can discover new markets via the built-in push notifications system if they are subscribed to certain developers, topics or categories. They can also search and filter for specific data types, events, names, dates and locations. Because the price is one of the most important factors, 2 numbers are shown to data providers: the current average reward and minimum average reward. The current average reward is equal to the average reward they will receive if no one else joins the market. Minimum reward is the average reward they will receive if the maximum number of people join the market (this is set by the developer, and it is shown to the reporter). In any case, these numbers are only an average, because the rewards are assigned proportionally to the speed with which the data is reported, so their reward can be higher (if they are faster than the average reporter) or lower (if they are slower then the average reporter). This adds to the fairness of the system, and it is an excellent initiative for the reporters to act and report fast while still giving them a good estimate of the reward they will receive.

If the staking is turned on and data provider joins before the minimum amount of people is reached as specified in `min_people` he pays `C/min_people*K` otherwise he pays `C/n*K`. The stake is returned to him after the event has ended and dispute time has passed (`start_dispute_time + max_dispute_time`) without disputes being opened or until all disputes are resolved. If he does not send the data that majority agreed upon or he tries to manipulate the market in some other way, his stake is taken and put into reward pool, and his reputation is lowered.

Data providers have time to join the data feed and send their stake until `time_to_join` passes. They also need to pass any criteria, as specified via the `identity_hash`. If they join before `min_people` is reached, their base reward level gets raised by `market_join_bonus` amount. Because joining the market means adding itself to the smart contract, blockchain transaction's time needs to be taken into account.

14. Reporting Data

The event can have a defined start time (defined via the timestamp saved in `start_event` parameter), or it can be undefined, and data providers report the information when they have it. In the latter version, start_event is defined with a median of timestamps from consensus votes. With developer's public key encrypted data is sent to validation nodes, which can now in a completely decentralized manner compare results and vote on what majority thinks is the right data. They also have the complete list of all the participating data providers (which they can always double check in the smart contract without any fees) along with the reported data, so they can check if only registered people are broadcasting data to them.

If the ratio between data providers and validation nodes is not balanced (e.g. a considerably higher number of data providers), data providers (`n`) chose a subset of few random validation nodes (i.e. `I = 3`) from all nodes (`m`) to which they send their vote. Those nodes then use gossip algorithm to tell each other of the votes they know about. They accept the vote as the truth, when they receive it from the majority of other validation nodes. If initially selected nodes do not respond, data providers choose new ones.
Validation nodes can lower amount of connection from data providers to them from `n` to each validation node to `(n/m)*I` in the best case, which decreases the load on validation nodes a lot. Gossip algorithm is `O(log n)` fast, so information travels extremely fast for low `m` (which will usually be the case). This `I` reduction, of course, lowers accuracy of average latency calculation (especially if one or more of the initial I validation nodes is not available and data provider needs to send his vote to a new node(s)).

Validation nodes are accepting data as long as they do not get the same data (i.e. same answer/information) from at least `min_consensus` number of data providers which also needs to represent `consensus_rule` of the votes, in other words,
`n[data]> consensus_rule*n>min_consensus`. As soon as they do (and they agree on it among themselves with ⅔ consensus rule), they send the information about the consensus to the developer, along with the list of users that reached this consensus and their latencies, which are both also subject to ⅔ agreement between validation nodes. New data coming in after `min_consensus` is reached is also relayed to the developer and serves as a confirmation of the `min_consensus` decision (e.g. 87% of voters also agrees with the `min_consensus`). If new votes coming in change the `consensus_rule` majority with at least `min_consensus` number of votes, the original consensus is overruled, automatic correction is sent to the developer, and people from original consensus loose stakes and reputation. If `stop_voting` is reached they start replying with `stop_broadcast` and providers stop sending the data. Validation nodes can now call smart contract (in fact, they vote one of them to do this job, others then check) to assign stakes to all the data providers that were part of the consensus. Stakes are assigned after a grace period (e.g. 2 hours after the consensus was made, so the disputes can be opened) and only if a majority of validation nodes assign the same stake structure to the same participants, rewards get assigned, validation nodes get validation fees and developer gets the stake back.

Every data feed can also be set as a multiple-event type. In this case, validation nodes keep track of the round number and they broadcast the `next_round` event to data providers. Multiple events end when either: a consensus is reached on a `stop_voting` (e.g. in a football game people are reporting new fouls until the game ends and they reach a consensus on a "game ends" event which they report), a developer ends the market (in this case if there is an event people are reporting on, everyone gets a reward as if they reached a consensus), `max_rounds` is reached or when `max_time` is reached. So a multiple type event behaves the same as a series of single events with a special mechanic to end the market.

```
{
  "data":[
    {
      "event_id": 10510252,
      "event_name": "Super Bowl LII - Report a touchdown",
      "description": "Touchdown, Super Bowl LII, 4/2/2018",
      "consensus_rules": "cc13fdfc9c5ca32a7bcd2ab7146c07db9fb5c9f34e3b159408f684",
      "contract": "0x24528467dfdabaef3fba7ad5796e39cfa9a99201",
      "category1" : "Sports",
      "event_start": 1517788800,
      "event_end": 1517846400,
      "event_ended_by": 3,
      "event_state": 3,
      "event_nonce": 2,
      "consensus": 1,
      "consensus_reached_in": 5,
      "consensus_value": "86f1d0C35E40DbC027d205C3C1a73727C7d3C4C9511ec6021bd8",
      "consensus_changed": 0,
      "number_of_providers": 24,
      "consensus_size": 19,
      "validation_node": "0xhucc4bd8ee760243dbbf815511ec161d1cd6957d8",
      "average_reputation": 3.45
    }
  ]
}
```

Figure 6. data delivered via the Eventum to the developer

15. Payments and Disputes

Each data provider that is part of the consensus gets its reward, which is equal to `C/n*speed_factor`, where `speed_factor` exponentially adjusts the reward up or down from the average of `C/n`. This means that if data provider is fast enough he can get exponentially more than the average reward. The reward can, of course, be even higher if the data provider joined the data feed early and acquired `market_join_bonus` which is a multiplier added to the `C/n*speed_factor calculation`. If the stake was used, total reward is the difference between the `C/n*speed_factor*market_join_bonus` and the stake that gets returned `(C/n*K)`.

15.1. Reward Pool

If data provider is not part of the majority, he loses his stake and all funds go to the reward pool which is used for rewards issued to nodes who successfully solve the dispute. If the reward pool gets filled over a certain level, the remaining part gets air-dropped to all the participants in the system who have a positive reputation level - this works as a passive income for everyone who participates in the system according to the rules.

15.2. Reputation

The air-dropped reward is proportionally scaled with the reputation level of the recipient. Reputation is gained by following the rules of the system (reporting data with the majority, having a good uptime of the validation node, solving disputes etc.) and lost by playing against the system. Each participant starts with the reputation score of 3.

Low reputation level limits the data feed you are able to join, passive rewards, validation node fees and other benefits. Having low reputation level for a long period of time results in a system-wide flag. The reputation system is built-in to the lowest level of the protocol and is implemented in the smart contracts and Swarm.

Reputation is implemented as a multi-level stacking system. Small penalties (`f = -0.5`) are given to data providers who join the market but never report the data. Medium penalties (`f = -1`) are given to data providers who voted, were not part of the consensus but are within three standard deviations of the mean from the `start_event` (if not implemented with the timestamp). Big penalties (`f = -3`) are given to data providers who voted, were not part of the consensus and were outside of the three standard deviations of the mean from the `start_event` (which probably means they were "gambling" and trying to predict the result of the market). A multiplier is added to the variable `f` for each violation. The reputation score can of course increase, which means that if you are a reputable participant in the system, you can "afford" more errors.

15.3. Disputes

Eventum is a self-converging system which in nearly all cases delivers the truth. Nevertheless, there can be cases where a large group of people form a consensus, but their vote was based on false information (intentionally or unintentionally). To mitigate this risk, developers can define a factor `Z` which defines what percentage of data providers, that did not form the majority, will receive a notification that their data was not accepted as correct. Percentage of data providers that get this notification is exponentially increased (e.g. if 96 people out of 100 reach consensus, all 4 will get the notification).

Validation nodes send this notification after `start_dispute_time` passes. Every data provider that receives this notification, can disagree with the market outcome and open a dispute. He needs to stake a large number of his tokens (exact amount depends on his reputation level) which gets returned to him if the dispute is successful. The dispute event gets broadcasted to other data providers who are subscribed to similar data feeds (same category, similar events, ...) who can now earn a reward for a positive dispute resolution. New data providers (`u`) are being recruited for the dispute resolution until they form a group (together with the people who formed a minority (`n`)) at least as large as the people that formed the majority (`m`) in the original data report: `n+u >= m`. If `n+u` decides with 100% that m was right (or u alone is 2/3 or more confident that m was right), the dispute is overthrown and staked tokens from the person who opened the dispute goes to everyone who helped to decide the outcome (along with additional rewards from the reward pool). The person who opened the dispute also loses reputation. If `n+u` agrees with 2/3 majority that `m` was wrong, tokens from m go to `n+u` (along with additional rewards from the reward pool and a bigger reward for the data provider who opened the dispute).

15.4. Selling Data

For the dispute mechanism to work, data providers need to send the result of the consensus to at least some of the data providers that reported the data.

They send the result to `Z%` of the people that were not included in the consensus after `start_dispute_time`. Z and start_dispute_time is set by the developer (e.g. `Z = 30%`, `start_dispute_time = 30min`). With those 2 parameters developer controls how strongly he believes in the consensus result (how fast he wants to receive dispute if there could be one, which could mean the original data he received was wrong) and how fast and how easy/costly data providers can receive information about consensus result (e.g. what is the value of consensus result to developer over time defined by `start_dispute_time`), which in return means, how easy is for data providers to "sell" consensus information to 3rd party clients even when they are voting randomly.
The developer also controls `K`, which defines the stake which in turn also decreases profitability for data providers to sell/share the consensus results. People who were wrong can dispute for up to `max_dispute_time`. After last event (in case of multiple-event type market) ends, `max_dispute_time` is added, and if there is no dispute, rewards/stakes are taken.

Let's take a look at someone trying to sell consensus data with the following variables set: `C/n = $1, K = 3, Z=20%`, binary decision market. Expected value or gain (`E`) when playing fair is `= $1, E` when playing random is `= - $1,` if selling consensus data, sell price needs to be more than `$15` to be profitable for the attacker and `$25` or more if attacker want it to be more lucrative than playing fair (if we assume fair player always know the answer, forms the majority and he is not fast enough to receive higher reward than the average reporter (so he does not receive speed_factor bonus), nor that he receives market signup bonus, which could all add to his winnings). This assumes that attacker can sell the data even though he knows that by playing random he only has `10%` chance to get the data at all (which developer can decrease even further by lowering `Z`), which is by itself hard (or most probably impossible) to sell. More people are in the market, more developer can decrease Z and still be sure there are enough people who will start the dispute if needed - at Z=`5%`, the attacker is profitable at `$60` and more lucrative than playing fair at `$100`.

Because the developer can also increase `start_dispute_time`, data needs to be valuable even after that time passes, which if the developer chooses wisely, greatly diminishes (or void) the value of the data altogether.
If the developer raises factor `K`, he can decrease `E` for attacker even further. Because attacker is flagged after `X` amount of times he plays against the market, and his identity is probably verified, he has no initiative to even start an attack like that even if all other factors are on his side.

Example for valuable data, big n but small min_consensus, `C/n = $10, K = 6, Z = 3%`, `start_dispute_time = 1h`, flagged `= 3`: attacker needs to be able to sell 1 hour old consensus data while having `1.5%` chance to receive the consensus result at all and he needs to sell it for at least `$2000` to be profitable. That is of course statistically good for him, but he will on average need to vote randomly more than 66 times before receiving consensus data and selling it for `$2000`, but because he gets flagged after 3 times of playing the market wrong, there is no way for him to pull off an attack like this.

15.5. Review System

To further reduce the chance for fraudulent providers to collectively enter the market and reach consensus on some random data, a review system is built into the service layer of Eventum, where a set of AI algorithms tries to recognize dishonest behaviour and asks random users in the system (preferably in similar markets or with same requirements) to confirm the outcome of the suspicious market before the dispute deadline. This reviewers also get a special reward from the reward pool to review and (optionally) open dispute. Review system uses many different parameters to recognize what is suspicious behaviour, including network statistics (latencies, node origins, …) and behaviour analysis (similar voting patterns, same markets, ...).

16. Token Model

Validation nodes earn fees for correctly capturing, processing and delivering data in Eventum tokens (**`EVT`**). Tokens are also used in staking mechanism, reputation system, disputes and governance. `EVT` is an essential part of the platform, and it fuels the most critical parts of the system. Because every single piece of data that goes through the system is connected to a token which is being paid as a fee to validation nodes, the increased usage of the system increases the demand for the tokens.

All `EVT` tokens will be minted and distributed in the token generation event (TGE), and their total supply can never change.

Users of the Eventum platform that create markets (developers) are charged a fee based on the amount of data that is being validated. Validation nodes are then paid for validating this data and acting as an intermediary between data reporters and developers providing security in the network.

The validation fee is approximated when the market is created. Since the approximations can be off, the platform requires the developer to “lock-in” X-times the approximated amount. The fees used to pay validation nodes are set by the validation node owners themselves. Every validator can decide what is the minimum fee they are willing to validate for. This market dynamic forces the validation market to strive towards and reach supply-demand equilibrium. If the minimum fee that is required to validate data is changed during the events, the developer can increase/decrease the fee. This fee increase/decrease can occur automatically or manually by the developer (he can opt-in to be notified of these changes).

While the fees are primarily paid in EVT, it is possible to use other ERC-20 tokens. To incentivise the use of our token and ensure that validation nodes are still consistently paid in EVT, all other ERC-20 tokens will be automatically exchanged to EVT on decentralized exchanges. Additionally, paying fees with other `ERC-20` tokens will increase the fee for `N%` (N never being under `1%`). N is calculated based on how many developers decide to use other `ERC-20` tokens to pay the fees, rising when the ratio goes up and falling when it goes down. This mechanic keeps the token viable and usable while at the same time letting developers use other tokens if they choose to. The difference between other `ERC-20` token and EVT payments is sent to the "reward pool".

Literature

[1] Connie Crosby (2011). The Importance of Real Time News and Information. link

[2] James Cotton (2014). Data: The importance of accuracy, integrity and real-time integration link

[3] Watson, Hugh J., et al. "Real-time business intelligence: Best practices at Continental Airlines." Information Systems Management 23.1 (2006): 7.

[4] Watson, Hugh J., and Barbara H. Wixom. "The current state of business intelligence." Computer 40.9 (2007).

[5] Sportradar AG, link

[6] Gartner, Inc. (2017). "Gartner Says Worldwide Business Intelligence and Analytics Market to Reach $18.3 Billion in 2017" link

[7] PennState News (2011). "Real-time search market worth more than $30 million a day" link

[8] George Collins, principal, Deloitte Consulting LLP, and chief technology officer, Deloitte Digital US, The Wall Street Journal (2015) "Companies Assessing Business Value of APIs" link

[9] Statista Inc. (2017). Number of smartphone users worldwide from 2014 to 2020

[10] Vitalik Buterin (2014). SchellingCoin: A Minimal-Trust Universal Data Feed link

[11] James Surowiecki (2005). The Wisdom of Crowds, 978-0-385-50386-0

[12] Ethereum Foundation, link

[13] Swarm, link

[14] Whisper, link

[15] Decentralizing Everything with Ethereum's Vitalik Buterin | Disrupt SF 2017, link

[16] NEO, link

[17] Lisk, link

[18] Sidechains, link

[19] Sharding in Ethereum, link

[20] Shah, Devavrat. "Gossip algorithms." Foundations and Trends® in Networking 3.1 (2009): 1-125.

[21] Civic, link