Mobius

A Universal Protocol Suite for the Blockchain Ecosystem and Real World Data

October 13, 2017 (v1.5)
info@mobius.network
Panchain, Inc.

Abstract

Mobius integrates the old internet with the new decentralized internet of value. Just as *Stripe* integrates payment processing into apps, Mobius (MOBI) integrates the blockchain ecosystem into apps. Our simple public APIs abstract away the underlying complexity of blockchain integration and development; so any developer can build modular constructs on the blockchain without specific domain expertise. As a result, Mobius makes it easy to connect any application, device, or data stream to the blockchain ecosystem. The live DApp Store allows every developer to securely distribute and scale cross-blockchain applications for mass adoption. The MVP use-case is our live DApp Store that makes it trivial for developers to accept in-app cryptocurrency micro-payments. The Mobius protocol encompasses standards for interoperable blockchain login, payment, smart contracts, governance, and oracles.

The blockchain at heart is a distributed transactions ledger where code is law: this envisages the possibility of fully autonomous decentralized markets. Our goal is to allow market participants to leverage Mobius to consume big data feeds and distill it into mission critical insights. Mobius makes this a reality with the conception of the Blockchain Smart Markets protocol: an incentive compatible two-sided periodically clearing combinatorial auction. The Mobius smart market brings the possibility of distributed hyper-efficient marketplaces, unprecedented in scale, that would coordinate trade in data and micro-services between an ever increasing number of rational programmatic agents. Developers can easily design market mechanisms and implement cryptographically secure distributed markets over the blockchain, which until recently would have been unimaginable except under pure monopoly conditions. Without the blockchain, there is no decentralized IoT marketplace that provides the data to fuel live AI applications. The simplest way to implement the smart market is through the Mobius blockchain protocol.

Table of Contents

1. Introduction

The Blockchain is comparable to the early internet, with 9,600 bps modems just entering the market. Ethereum running at a mere ~13 tps is unscalable for real world applications. Ethereum significantly encumbers future adoption with problematic high fees and major attack surface. The continual hard-forks portend future instability.

Blockchain developer tools and protocols are in their infancy. They are comparable to the primitive web developer tools of the early 1990s, an era without Stripe, who created a standardized simple protocol to accept credit card payments, or the Amazon AWS cloud, which made it easy for anyone to deploy a web application. There is a fundamental need to abstract the underlying implementations. Mobius does this now with universal APIs to convert the myriad of app developers into blockchain developers -- just as Stripe unleashed e-commerce through its suite of APIs to integrate payment solutions. Mobius APIs will ease the integration of the blockchain ecosystem within everyday applications.

2. The Token Economy

There are over 850 tokens and growing, with many created this year. These organizations have collectively raised almost $1.3 billion in the first half of 2017. The blockchain ecosystem has soared to a market capitalization greater than $150 billion. The number of blockchain users continues to grow. Despite Facebook-like growth, the number of blockchain users is minuscule compared to the 4 billion internet users or 8.4 billion IoT devices. This gap between the internet world and blockchain world represents a multi-trillion dollar opportunity. The internet App Economy alone is predicted to be $6.3 trillion by 2021, while smart devices will grow to over 50 billion by 2020.

Strong developer communities along with open APIs are the key to the success of the internet and its ecosystem of enterprises. This includes the likes of Facebook, Salesforce, Amazon, and Alibaba. Similarly, the success of the expanding the blockchain ecosystem requires both developers and robust API protocols. The open bidirectional Mobius protocol APIs turn the 18.2 million software developers in the world into blockchain developers through simple and standardized protocols. Any developer can easily connect any application, device, or data stream to the blockchain ecosystem. Mobius is closing the gap between the internet and the blockchain world through a multifaceted solution creating easy developer tools, a consumer DApp Store, and a Smart Market to allow hyper-efficient transactions of processed real world data between autonomous agents with our Universal Proof of Stake Oracle protocol.

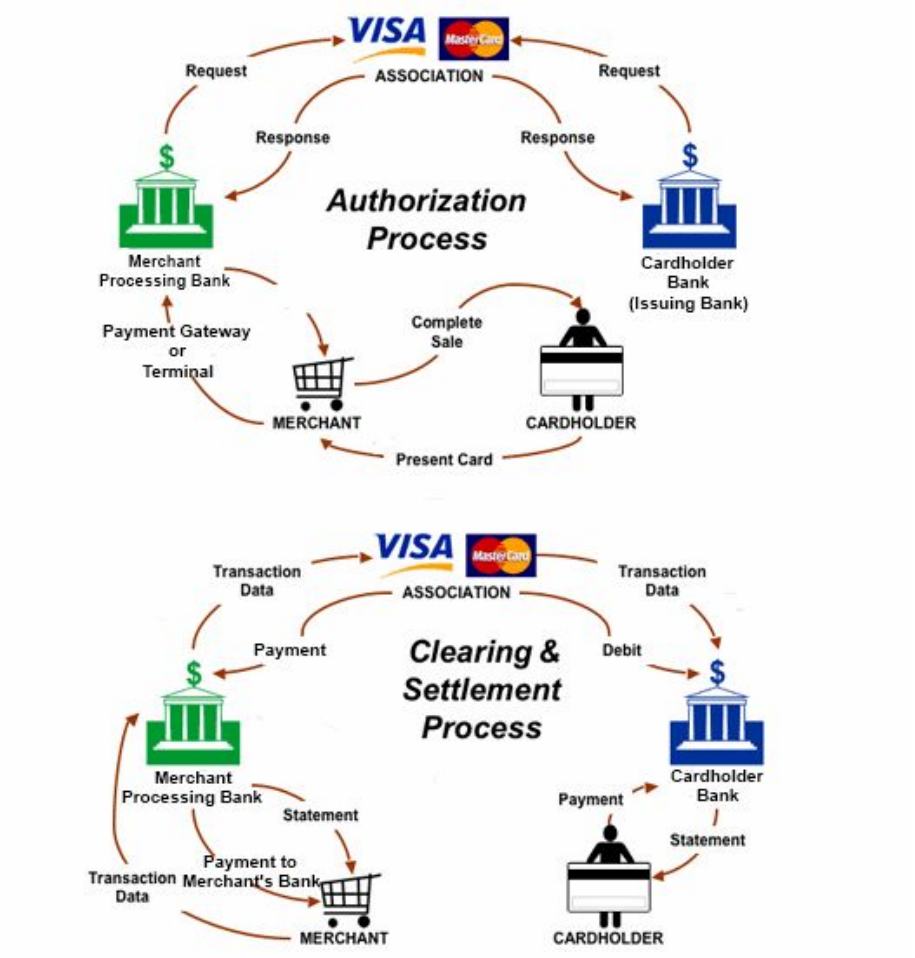
In addition, the Mobius protocols define universal cross-blockchain standards for payments, login, and oracles through simple APIs, developer frameworks, and webhook callbacks. In the payment space, Mobius is analogous to Stripe, which made it easy for developers to accept any credit card. Mobius similarly makes it easy for developers to transact with any blockchain token and benefit from significantly lower transaction fees by disintermediating legacy centralized institutions. The Mobius DApp Store works

in conjunction with the Mobius API to solve the consumer discovery problem and features fees that are over 70% lower than the centralized Apple and Google analogs. The short to intermediate aim is to progressively decentralize the entire global app economy.

3. Universal Blockchain Payment Protocol

The current global payment infrastructure is centralized, convoluted, slow, and expensive. These legacy models are long overdue for an overhaul to a more efficient, trustless, and dynamic digital alternative.

Accepting credit and debit cards involves a three key steps: authorization, clearing, and settlement process. The process depends on payment gateways, terminals, merchant banks, credit card associations, cardholder banks, and other stakeholders. As a result, this leaves participants with high fees and slow payment finalization. This is why legacy payments are expensive, often costing approximately $0.30 + 2.9% per transaction.



Source: https://www.pathwaypayments.com/processing-diagram.html

Blockchain technology will revolutionize and simplify global payments by disintermediating large financial institutions, significantly reducing transaction costs. However, current blockchain developer APIs and frameworks are in their infancy and are very hard to use. Developer technology for blockchain today is similar to payment developer frameworks before Stripe.

For example, below is an example of the Authorize.Net Merchant Web Services Simple Object Access Protocol (SOAP) CreateCustomerProfile request data structure:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
<soap:Body>
<CreateCustomerProfile xmlns="https://api.authorize.net/soap/v1/">
<merchantAuthentication>
  <name>API Login ID here</name>
  <transactionKey>Transaction Key here</transactionKey>
</merchantAuthentication>
<profile>
  <merchantCustomerId>Merchant Customer ID here</merchantCustomerId>
  <description>Profile description here</description>
  <email>customer profile email address here</email>
<paymentProfiles>
<CustomerPaymentProfileType>
  <customerType>individual</customerType>
    <payment>
      <creditCard>
        <cardNumber>Credit card number here</cardNumber>
                    <expirationDate>Credit   card   expiration   date
here</expirationDate>
      </creditCard>
    </payment>
</CustomerPaymentProfileType>
</paymentProfiles>
</profile>
<validationMode>liveMode</validationMode>
</CreateCustomerProfile>
</soap:Body>
</soap:Envelope>
```

Source: https://www.authorize.net/content/dam/authorize/documents/CIM_SOAP_guide.pdf

The Authorize.net API is verbose, cumbersome, and hard to use. Below are the equivalent Stripe REST API calls:

```
$ curl https://api.stripe.com/v1/tokens   \
    -u sk_test_BQokikJ0vBiI2HlWgH4olfQ2: \
    -d card[number]=4242424242424242 \
    -d card[exp_month]=12 \
    -d card[exp_year]=2018 \
    -d card[cvc]=123
```

https://stripe.com/docs/api/curl#create_card_token

```
$ curl https://api.stripe.com/v1/tokens   \
    -u sk_test_BQokikJ0vBiI2HlWgH4olfQ2: \
    -d description = "Customer for
anthony.white@example.com" \
    -d source=tok_189gK92eZvKYlo2CLeB0l8KA
```

https://stripe.com/docs/api/curl#create_customer

The Stripe API is simple, elegant, and easy to use. Stripe leveraged the complicated payment developer infrastructure to turn its developer-first, easy to use API into a company that has raised more than $440 million and is valued at over $9 billion today. Mobius is solving the same problem, except for the emerging blockchain economy. Apple and Google App Store fees and credit card fees will take a meaningful percent of this revenue creating significant deadweight loss. Mobius will make it easy for developers to transition to the decentralized lower cost blockchain economy. Mobius fees are over 70% cheaper than existing centralized payment solutions.

The Mobius API provides a universal interface to accept any blockchain token. This is analogous to how Stripe provides a universal interface to accept different credit cards such as Visa, Mastercard, Discover, and American Express. Currently, it is very complicated to accept, manage, and secure tokens. For example, if a video game developer creates an Ethereum ERC20 token that acts as their in-game credits, it is complicated to move tokens between the Ethereum network and their backend accounting system.

Current Token Developer Process:

1. Run a full Ethereum node
2. Create a unique Ethereum address for each user
3. Monitor every user Ethereum address for incoming token transactions
4. Secure the private keys
5. Manage token accounting in their backend

The Mobius API simplifies all this by providing a modern easy to use REST API that returns JSON and uses secure webhooks to alert developers of incoming token transfers.The Mobius API already works with any ERC20 token such as Golem, Augur, Iconomi, Bancor, Storj, Status, and Credo. As we grow, we will keep expanding the API to support other blockchains and tokens besides Ethereum ERC20.

4. Universal Token Login Protocol

Tokens can be used to represent membership to a service and can be used as a login mechanism. Any service that uses tokens as a login mechanism will have similar technical infrastructure needs analogous to a token payment infrastructure. Mobius provides a simple REST API that abstracts low-level blockchain development required to verify token ownership at time of login and facilitate verified login.

Using a token to represent a subscription provides several benefits including increased anonymity, lower transaction fees, and the potential for an independent secondary market to develop providing greater value for subscribers and incentive for early subscription or membership purchase. The secondary market will allow people to anonymously buy and sell their membership without ever transacting over centralized financial institutions such as credit cards, debit cards, or bank accounts allowing for complete anonymity, assuming usage of an anonymous protocol tokens like Zcash, Monero, or PIVX.

Multiple tiers of membership can also be represented through tokens by requiring a different number of tokens to access different levels. For example, a basic membership might only require one token to login while the plus membership might require five tokens to login.

5. Universal Governance Protocol

Many organizations that create a token and plan to use it for platform governance will have similar use cases. For example, if Wikipedia and Reddit each create a token they are likely to have similar governance and proof of stake platform integration needs. On the governance side, both organizations may allow token holders to vote on proposals that dictate how the communities develop. On the platform integration side, both may require advanced users such as editors or moderators to have some vested

tokens at stake to reduce spam and harmful community behavior. Both organizations may also want to reward community members that improve the community and add value. These similar use cases can be abstracted much as web frameworks and databases, the building blocks of web applications, have been abstracted. The vast majority of developers do not create their own web framework such as Ruby on Rails or their own database server such as PostgreSQL. Instead they use one of the existing web frameworks or database servers to save time. Blockchain based governance will follow the same model and be offered to developers by Mobius via standard protocols and APIs that they can use to facilitate token based voting and vested at-stake micro-actions that result in token rewards or penalties based on community feedback.

6. Decentralized App Economy

One of the biggest problems hindering mass adoption of DApps and cryptocurrencies as fiat replacement is consumer discovery and adoption. Currently, there is no widely used DApp Store. Mobius is creating a universal DApp Store that will be similar to the Apple App Store and Google Play Store. Any app that accepts the Mobius Token will be listed in the DApp Store. The DApp Store will also integrate the Mobius Token for a universal decentralized credit system that has significantly lower fees than the Apple App Store or Google Play Store. The DApp Store fee will only be 8.8%. The 8.8% fee is only charged if tokens are used through the DApp Store credit system - if the developer uses the API to receive tokens there is only a 1% fee. These numbers are subject to change overtime and the goal is to gradually decrease fees even more as network adoption continues to expand.

DApps will be incentivized to accept the Mobius Token in the DApp Store because of the network effects of the DApp Store and Mobius Token. The Mobius Token will experience similar network effects to the Internet in which the token becomes more valuable as more people use it. This is in parallel to a developer having its own token primarily for governance reasons. A developer that has its own token will still want to accept the easiest to use and cheapest form of payment from transient users so they are not turned away. Simultaneously, a developer with its own token will want to limit governance and at-stake based micro-actions to users with a vested stake in their community and platform. The DApp Store and Mobius Network provide a platform to easily support both use cases.

6.1 MOBI Commerce Network Effects

In the past two decades, the developed world's economic vitality became progressively dependent on widespread internet accessibility. Likewise, blockchain adoption will prove to be equally indispensable to a nation's GDP: transforming centralized systems and inefficient legacy processes into more trustless, decentralized, vending-machine like transactions. Well defined smart contracts on public blockchains have the potential to turn noisy big data into value-added insights via prediction markets, create immutable public records, power decisions of autonomous AI agents, and much more in the years ahead. The DApps of tomorrow will differ substantially from today's apps, reflecting the coming wave of decentralization of the app economy. Existing apps will need to find an easy way to connect to the blockchain ecosystem. Our thesis is that blockchain "content" in the form of apps will only come to fruition if the best developers are given an economic incentive in advance of mass market adoption by end

users. Such an option is only accessible by major enterprises with significant funding. A notable example is PayPal's Referral program, which resulted in 7-10% daily growth in the early 2000's. The program had an initial CAC of $20.00, which triggered network effects since it offered free money to new users and their referrals. In a winner-takes-all marketplace, the DApp Store requires a similar strategy to seed content and increase adoption; more DApps stimulates adoption, increasing the economic incentive to create and deploy new DApp content and closing the feedback loop.

There is a classic chicken and egg problem for marketplaces: for the DApp Store, developers are more inclined to add content only when users are present, and users will adopt such content depending on its availability, unique value, and stickiness. As a result, if developers are incentivised to add content to the platform, then early adopting end users will gradually adopt the platform as well.

To persuade one group of early adopting developers to enlist in advance of strong market penetration, we will offer an economic incentive by giving them Mobius Tokens to seed initial growth in the short to intermediate term. This is akin to PayPal's Referral Program strategy of user acquisition and growth, which will drive a positive feedback loop of engagement and bolster network effects.

6.2 Growth With Two Sided Network Effects

Based on the Lotka Volterra population equations we can model the dynamics underlying the Mobius token and the DApp Store as follows:

$$dx/dt = \alpha x + \beta xy$$

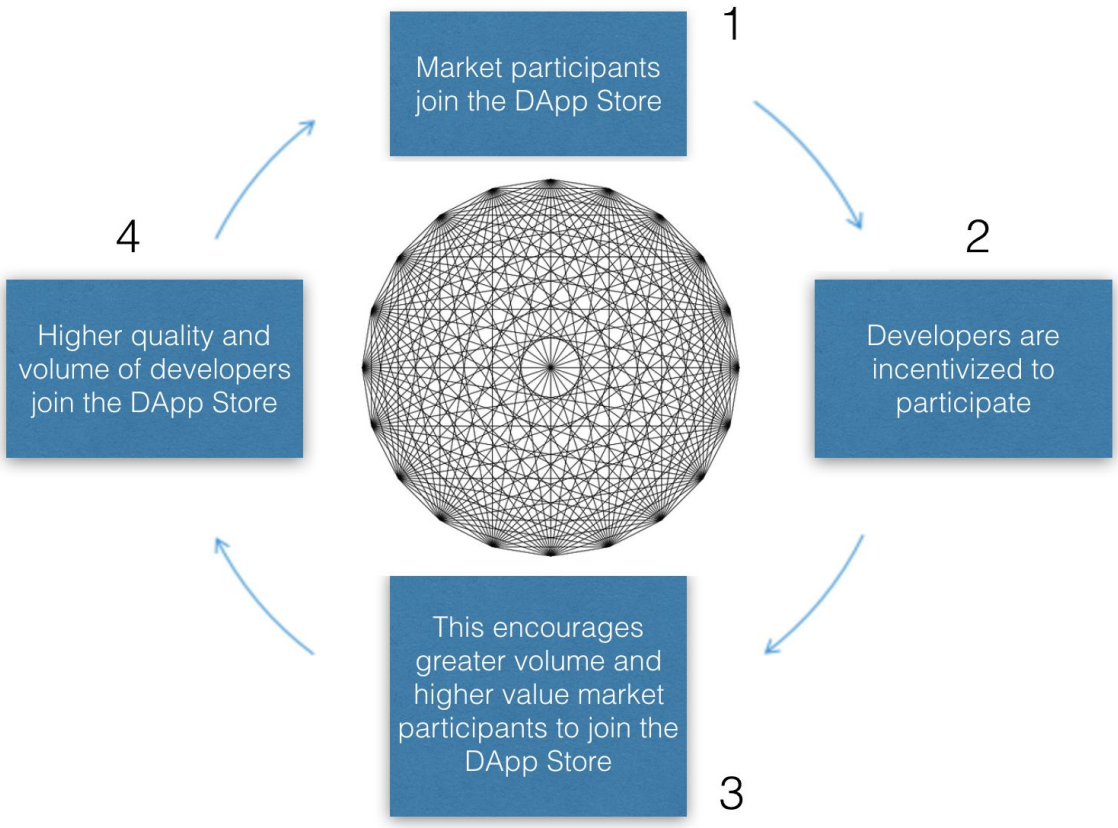
$$dy/dt = \delta xy + \gamma y$$

where:

- x is the number of market participants in the DApp Store, which represents the demand or consumer side
- y is the number of developers in the DApp Store
- t represents time
- dx/dt is the growth rate of market participants with respect to time
- dy/dt is the growth rate of developers with respect to time
- $\alpha, \beta, \gamma, \delta$ are positive real parameters that describe the interactions of the developers and market participants.

Essentially, the rate of change of market participants is dependant on number of current market participants and number of developers, similarly this is true for developers.

This leads to a virtuous cycle. As more market participants start using the DApp, more developers are encouraged to participate. This further incentivizes new market participants to start using the DApp store.



1) Token buyers join the platform, resulting in more demand side users. 2) This encourages sellers (developers with DApps) to participate (increases supply side of users). 3) As a result, this event encourages more buyers to join (more demand side users), 4) which brings more and better sellers to the table (more supply side users). 5) Repeat loop.

6.3 Network Effects of the Market Participants

Let us assume that a fraction ζ of the potential population is already using the DApp Store. Any individual pays a price, measured in units of MOBI, to become a market participant of the DApp Store. Each individual will have a maximum limit on the price they are willing to pay to participate. We call this the reserve price, which is measured in MOBI. The reserve price may be represented as follows:

$$reserve\ price\ =\ g(x)f(\zeta)\,,$$

where:

- $g(x)$ as before is the desire of some person x in becoming a market participant
- $f(\zeta)$ is the measure of the benefit that each new market participant derives from having a ζ fraction of the total population already being market participants
- $f(\zeta)$ is an increasing function in terms of ζ: it tells us how much more valuable the DApp store is when more people are using it.

- $g(x)f(\zeta)$ comes from the Lotka Volterra population equations.

For the reserve price, the term $g(x)f(\zeta)$ represents the fact that market participants who derive a larger benefit, benefit more from an increase in the fraction of the population becoming market participants than those market participants that have smaller values.

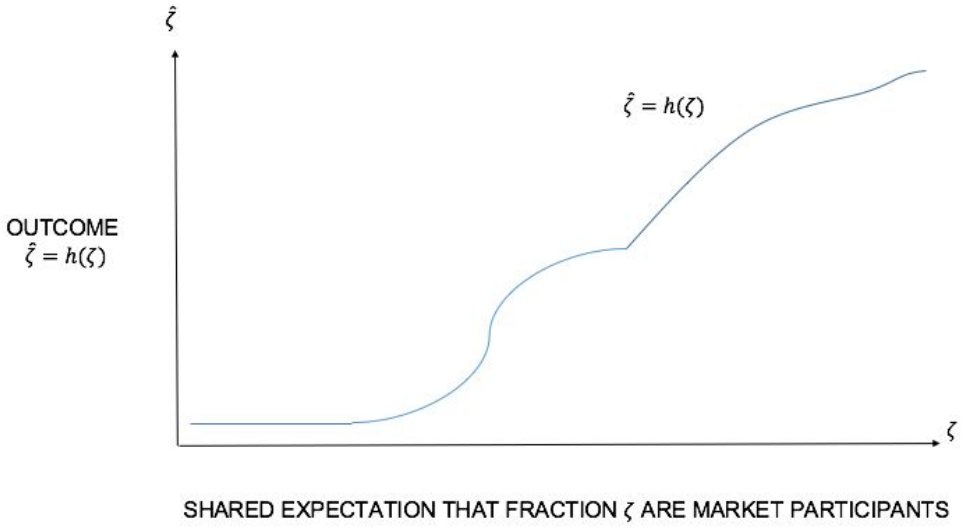
As elucidated by the virtuous cycle, not only does the number of market participants and developers increase, but the quality increases, too. That is, the market participants will spend more MOBI and developers who list better quality apps.

6.4 Equilibria with Network Effects in B2B/C & P2P Adoption of DApp Store

Individuals derive a shared expectation that the fraction of population that are market participants is ζ, and if each of these individuals then makes a decision to become a market participants by purchasing MOBI based on the above expectation, then the fraction of people who actually end up purchasing MOBI is ζ.

This is the self-fulfilling expectations equilibrium for the quantity of purchasers ζ: if the community expects that a ζ percentage of the population of potential consumers will become market participants of the DApp Store and spend to buy Mobius, then this expectation is in turn fulfilled by their behavior.

6.5 Network Effects of Consumer Adoption in the DApp Store Ecosystem



We can define a function $\widehat{\zeta} = h(\zeta)$ as follows. If potential consumers expect a ζ fraction of the population to become market participants of the DApp Store then a $h(\zeta)$ fraction will become market participants of the DApp Store. The function $h(\zeta)$ is the real world adoption rate based on networking effects.

Furthermore, if the potential consumer base expects a ζ fraction of the population will commit and become market participants, then there is potential market participant x who will want to become a market participant provided the reserve price for participation is higher than the amount the potential customer has to pay.

$$\text{If } g(x)f(\zeta) \geq m^*$$

where m^* is the cost in Mobius that the individual must pay. Hence, if anyone wants to participate, the set of people who will participate will be between 0 and $\widehat{\zeta}$,

where: $\widehat{\zeta}$ is the solution of the following equations

$$g(\widehat{\zeta})f(\zeta) = m^*$$

$$\Rightarrow \quad g(\widehat{\zeta}) = m^*/f(\zeta)$$

Taking the inverse we can obtain $\widehat{\zeta}$:

$$\widehat{\zeta} = g^{-1}(m^*/f(\zeta))$$

This gives us a relationship between the mobius price to be paid to become a market participant and the adoption rate based on network effects.

Thus we can drive adoption of the DApp Store based on a greater understanding of the dynamics of the relationship between users and developers.

6.6 Hackathons for the Best Developers: Enhancing Network Effects

One possibility is to hold Regular Hackathons for DApp Store Developers.

Developers get rewarded for performance according to the following criteria[1]:
i) New unique users that join and engage the platform.
ii) For retention, where returning users consume new content.

[1] These criteria can be optimized according to our DApp development roadmap.

A fixed amount of tokens are allocated for the hackathons, from the 32.5% community pool, on a monthly basis and may be adjusted to a weekly cadence, if desired by decentralized governance made by community of MOBI. For instance, a hackathon can occur for a specified time and the community can vote in proportion to the MOBI they have for the apps they desire to win the developer prize. These tokens are redistributed based on performance of the developers.

Once a critical inflection point is reached in terms of app popularity, developers will have to pay a small token contribution to list their app on the DApp Store. Hackathons will continue on a regular monthly cycle. The tokens paid to list new apps in the DApp Store will be redistributed back to top performers in the Hackathons. The distribution of tokens is expected to follow a Zipfian distribution.

Formally, let:

- N be the number of developers;
- k be their rank based on performance;
- s is the value of the exponent characterizing the actual distribution observed.

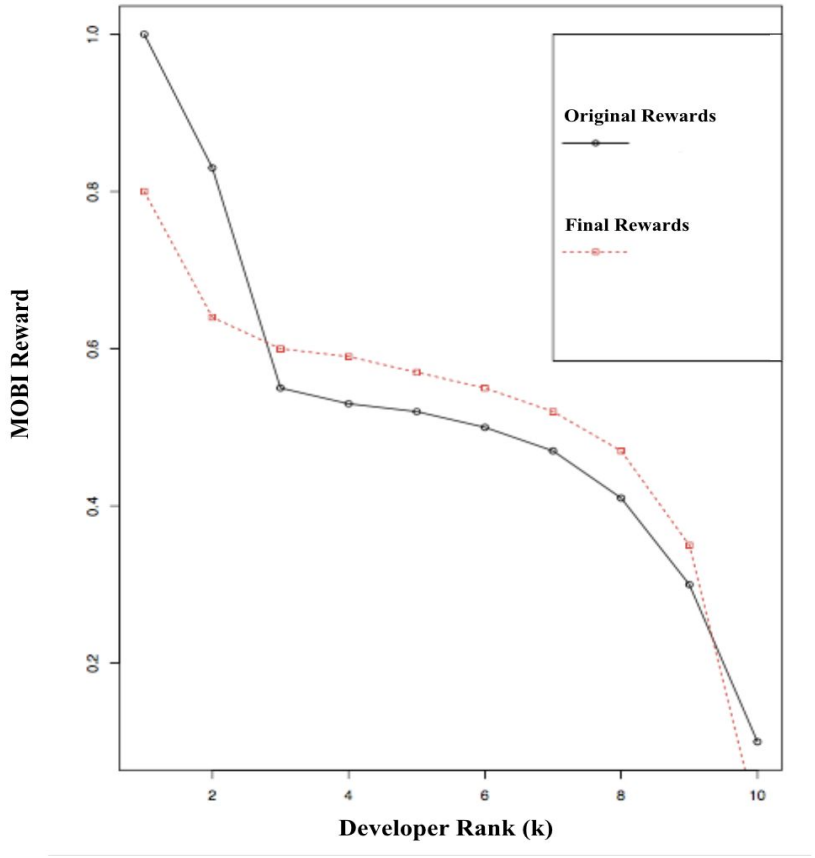
Zipf's law then predicts that out of a population of N elements, the frequency of elements of rank k, $f(k;s,N)$, is:

$$f(k;s,N) = (1/k^s)/ \sum_{n=1}^{N} (1/n^s)$$

s is calculated at the end of each month based on transactions observed. This distribution is then scaled to modify the rewards given out.

We hope to reduce the reward earned by the highest performers marginally to better reward middle performers, while lower end performers below a cutoff dont get any reward. This can be accomplished as follows: 1) Cutting off developers below the 80% rank 2) if $s > 1$, then replacing s with its logarithmic value and distributing rewards proportionally.

Developer Rank vs. Reward Distribution in Mobius DApp Store using Zipf's Law



The goal is to optimize the incentivization structure for as many developers to make as many quality apps as possible; middle-ranks should get a higher reward in comparison to what the Zipfian power law distribution would recommend.

Essentially, if the black curve describes original distribution of reward, then the final distribution is represented by the red curve. The area under the curve represents total reward amount.

7. Universal Proof of Stake Oracle Protocol

Blockchains exist in the digital world and only have access to data that is fed into them. In the blockchain ecosystem, oracles are specialized applications that provide access to external data, essentially feeding the blockchain with information, which is consumed by well-defined smart contracts. Oracles can set up tunnels for data streams from any viable source, ranging from the bloomberg price feeds to internet of things (IoT) devices.

Smart contracts are built to consume information from oracles. Smart contracts are designed to be triggered by events, which are reported to them by oracles, performing pre-programmed tasks based on data inputs. Thus, one of the primary tasks of an oracle is to provide information to smart contracts. Oracle data can be consumed by end users apart from smart contracts.

Most importantly, data provided by any oracle must be trustworthy, meaning that smart contracts must be able to trust the reliability of the data source before executing. In the blockchain world transaction rollbacks are not possible, and yet, the central issue of data reliability is not dealt with adequately at the moment. Furthermore, currently there is no standard secure oracle protocol that allows oracles to operate cross-blockchain and feed their valuable data into multiple blockchain systems. Mobius solves these fundamental architectural weaknesses by creating a standardized proof of stake based protocol that enables developers to easily create oracles that securely input data into the blockchain ecosystem, particularly into smart contracts.

A proof of stake protocol is used to incentivise oracles to provide correct data and penalize oracles that provide incorrect data. The Mobius Universal Proof of Stake Oracle Protocol (PSOP) will require oracles to vest and stake MOBI before inputting data into the blockchain ecosystem. Oracles that provide incorrect data will be penalized, losing their vested MOBI while oracles that provide correct data will be compensated with MOBI by data consumers. Any oracles found to be providing incorrect data will lose some portion of their vested and staked MOBI. Market designs that incentivize authentication, validation, and ultimately successful aggregation of information will be implemented as part of the Proof of Stake Protocol.

The broad adoption and deployment of oracles will create an additional blockchain monetization method through oracle devices. Oracle pools analogous to mining pools will also develop. In these pools, people all around the world will combine their oracles to provide more data to the blockchain ecosystem and better monetize it.

For example, farmers in Yuanyang County, Yunnan, China could purchase and install temperature sensor oracles to provide distributed and verified temperature data to the blockchain ecosystem. The temperature readings will receive a score based on the number of oracles that agree on the reading. When the temperature data is purchased through an open marketplace the providers and verifiers of the data will receive MOBI as compensation. Any farmer providing a single point of temperature data may rarely be compensated for their data if they provide it alone because much like successfully mining a bitcoin, it is rare someone will want a specific temperature data point. However, a temperature oracle pool with world wide temperature coverage will become the standard temperature provider, thus even those that provide rarely used data can be compensated based on how they help complete the dataset instead of just providing a single datapoint.

7.1 The Market for Lemons: Quality Uncertainty of Oracle Data

If quality is difficult to ascertain due to information asymmetry, then lower quality oracles will seek to masquerade as higher quality oracles. Smart contracts will take this adverse incentive into consideration and assume that the quality of oracles as a whole is uncertain. Smart contracts would then only pay MOBI considering the overall average quality of oracles. Thus in case the market fails to distinguish between higher and lower quality oracles, then this leads to the higher quality oracles being underpaid and driven out of the market. To overcome this issue will have segmented markets where only oracles above a predefined quality threshold would be allowed to participate.

If smart contracts had the ability to tell wheat from chaff, then fair payment in terms of mobius could be achieved. A segmented markets for higher and lower quality data would naturally come into existence. If smart contracts can not tell the quality differences, then a single market would form where the smart contracts would be willing to pay only average prices.

One way to overcome quality issues is to devise a rating mechanism like a quality score. In the forward auction, the smart contracts would be provided with a quality rating based on the oracles past performance and mobius staked. Quality scores are aggregated based on past performance, staking MOBI, and **Proof of Verification**. We will design a Proof of Verification Protocol to allow audits of oracles. **Prequalifying** oracle's entry into the market helps set up a **segmented** market for oracle data. High fidelity oracles would participate in a separate market from low fidelity & newly minted oracles. One format would run a high fidelity auction where entry would require a minimum quality score which could be achieved from a combination of good past history and staking mobius.

7.2 Oracle and Smart Contract Ecosystem

Software oracles provide access to online data like an oracle for trending twitter handles while **Hardware oracles** provide access to real world data like IoT or RFID data. A further classification is the direction of the data flow with **Inbound oracles** providing data to the blockchain and **Outbound oracles** providing data to the real world from the blockchain. An important subclass are the **Consensus oracles** which as the name suggests work by consensus. With multiple oracles providing similar data, which may have reliability issues, consensus based rules are used to parse and aggregate the data provided. For instance, multiple weather IoT devices provide various humidity readings and a consensus humidity reading is calculated.

Consider a game theoretic viewpoint: Here we are interested in the private information that oracles hold and their actions as self-interested rational agents while participating in a market to sell their data.

Let $N = \{0, 1, ...n\}$ denote a list of oracles in a mechanism for the sale of data on the blockchain. Let $\bar{t}$ be a vector of the type profiles of all the oracles. So $\bar{t} = (t^1, t^2, ..., t^N)$ includes a type for each oracle. Type t^i represents all the value, information, beliefs, utility and preferences of the oracle i.

Oracles with **independent private values**: In game theory parlance oracles whose valuations of the data they supply only depends on their own type t and the types are statistically independent. Further the valuations are privately held by the oracles and not publicly known. They are not affected by other competing oracles during participation in the data smart market. This independence in valuation leads to a strong dependence between the bids placed in the reverse auction and the private valuations of the oracle.

Oracles with **interdependent values** do not know their own valuations themselves, they are prone to mimic valuations of competing oracles, more strongly the distribution of types t of the oracles are statistically dependent. This leads to a weaker relationship between the bids placed and the actual valuations. Oracles who want to supply data may be required to sign a contract with an **ancillary staker** that verifies, validates, and authenticates the oracle's data and then stakes Mobius on behalf of the oracle, to participate in the auction. In case of issues in the data, the ancillary staker is held responsible.

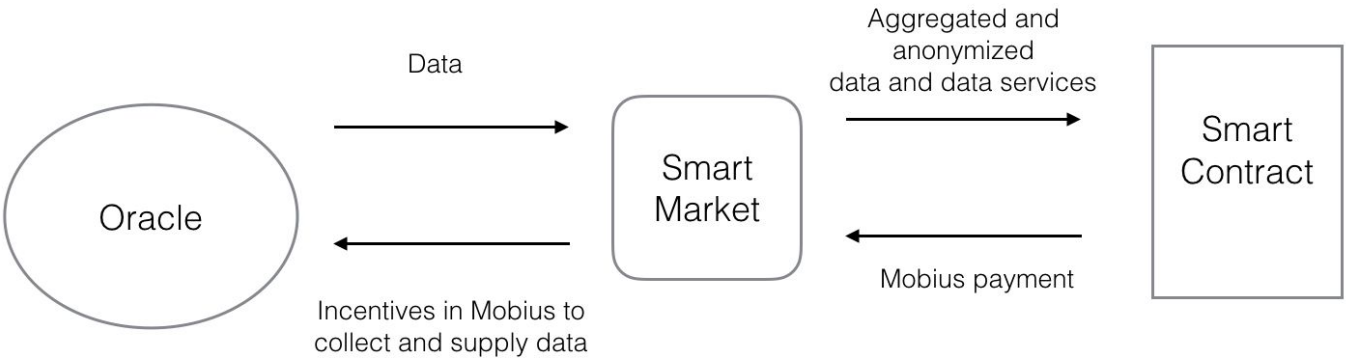
Concept of **Ancillary Stakers**: Ancillary stakers are essentially a kind of smart contract with a specialized role in the Proof of Stake Oracle Protocol. Stakers are third parties: independent from the oracles, whose primary job is to validate oracle data and stake their own Mobius and reputation on the data, for which they can charge oracles a percentage cut of the proceeds from the sale of the data. Stakers are like reverse insurance agents. This is the scenario where the oracle may not be willing to stake on its own.

Some other situations and kinds of smart contracts arise and are also part of the Proof of Stake Oracle Protocol. **Intermediary smart contracts** who buy data from an oracle(s), process it and resell to other smart contracts or outbound oracles. This may take place under the intermediary's brand and represents a kind of white labeling. **Insurance smart contracts** provide insurance to other smart contracts with the underlying being the quality of data they are buying.

8. Blockchain Smart Markets

Blockchain Smart Markets are auctions harnessing blockchain technology that clear periodically. Transactions take place between distinct pools of smart contracts or other end users acting as buyer and sellers rather than bilaterally. Pools of oracles may also participate but are restricted to being sellers. Decentralized smart contracts or end users submit bids to buy and offers to sell data or services in a commoditized manner. The whole process is managed by an auction manager or 'auctioneer'. Clearing the market usually involves the auctioneer solving complex mathematical optimization problems with arbitrary constraints periodically to maximize the gains from trade. Smart markets are designed to reduce transaction costs significantly and eliminate externalities while allowing for competition not possible in more traditional settings.

Smart markets allow for coordination between diverse smart contracts or other end users, which is usually only possible under monopoly conditions. This coordination ability is a natural complement to the blockchain world. The use of cryptocurrencies and distributed ledgers allows for the implementation and operation of complex smart markets with ever larger number of participants.



The time period of smart markets is the time between successive instances of the market clearing. It may range from a few milliseconds to a few days. Markets include one sided forward auctions (demand only), one sided reverse auctions (supply only) and two sided auctions with supply and demand components.

Currently, smart contracts are completely public and any data they consume becomes public. This precludes **public blockchain** smart contracts from participating in blockchain data auctions requiring data to be kept private. **Private blockchain** smart contracts with proprietary systems to keep data private are the primary use case. On the public blockchain, end users may consume data privately off chain. The actual transfer of data ideally will take place through apps on the DApp store. For example, an autonomous car may buy a contract for navigation data at auction and consequent off-chain navigation alerts are obtained through an app on the DApp store.

8.1 Blockchain Smart Markets Protocol

Mobius will implement the Blockchain Smart Markets Protocol, which will allow auctioneers, market designers and game theorist to work in-sync with developers to easily design and quickly deploy smart markets to enable the trade of commoditized data and services in a secure and transparent manner.

A smart market for the sale of data will rely on both the Proof of Stake Oracle Protocol and the Blockchain Smart Markets Protocol. Apart from the earlier discussed smart market for data, additional use cases for the Blockchain Smart Markets Protocol include a smart market for decentralized cloud services including file storage and cloud computing. Note that actual data may be consumed in real time, may be stored off-blockchain, or encrypted versions may be stored on the blockchain. This will depend on the characteristics of the data and the kind of market functioning. Since there are currently significant

blockchain scaling issues, it may not be possible to keep encrypted data on the blockchain, and off chain solutions, or hybrid solutions, will be important.

9. Market Design for Proof of Stake Oracle Protocol

9.1 Background

Underlying Theoretical Construct: Game Theory
Game theory is concerned with the mathematical modeling strategic behavior of market participants under specified rules. It is the part of economics concerned with the detailed rules and procedures of economic institutions (like markets and auctions). Auctions are games designed by the auctioneer and played by the bidders.

Given fixed Rules ➡ Study Outcomes

Reverse Game Theory: Market Design
Economists not only analyze markets, but design or engineer them. Mechanism design, often called reverse game theory takes an engineering viewpoint towards designing a market that achieves desired objectives given strategic settings with rational players. An engineering approach to design markets towards desired objects in strategic settings with rational players includes efficiency of auction design, optimal and equilibrium bidding strategies, and revenue comparison. Fairly accommodating all participants' needs is critical.

To design a market ➡ Specify set of rules to govern the market
Given target objective ➡ Design market to achieve the predefined objective

The **standard** is to decide on goals we need to achieve and then design the market around it.

9.2 Stated Market Design Goals

We seek to design a mechanism (i.e. market) for the trade or exchange of data using blockchain infrastructure, among oracles and end users. The exchange can be facilitated by smart contracts as bidding agents for end users that consume the data. This market is to be implemented using the Proof of Stake Oracle Protocol and the Blockchain Smart Markets Protocol. As noted elsewhere smart contracts may be the end users on private blockchains with proprietary methods that allow smart contracts to consume data privately. On public blockchains smart contracts are completely public and any data they consume becomes public, losing its value. On public blockchains, end users would ideally consume the data privately off chain. This could be implemented by providing the data to the winning user on a private blockchain or off-chain. The purpose of this section is to examine the market design that would support a well functioning IoT data marketplace using the blockchain ecosystem.

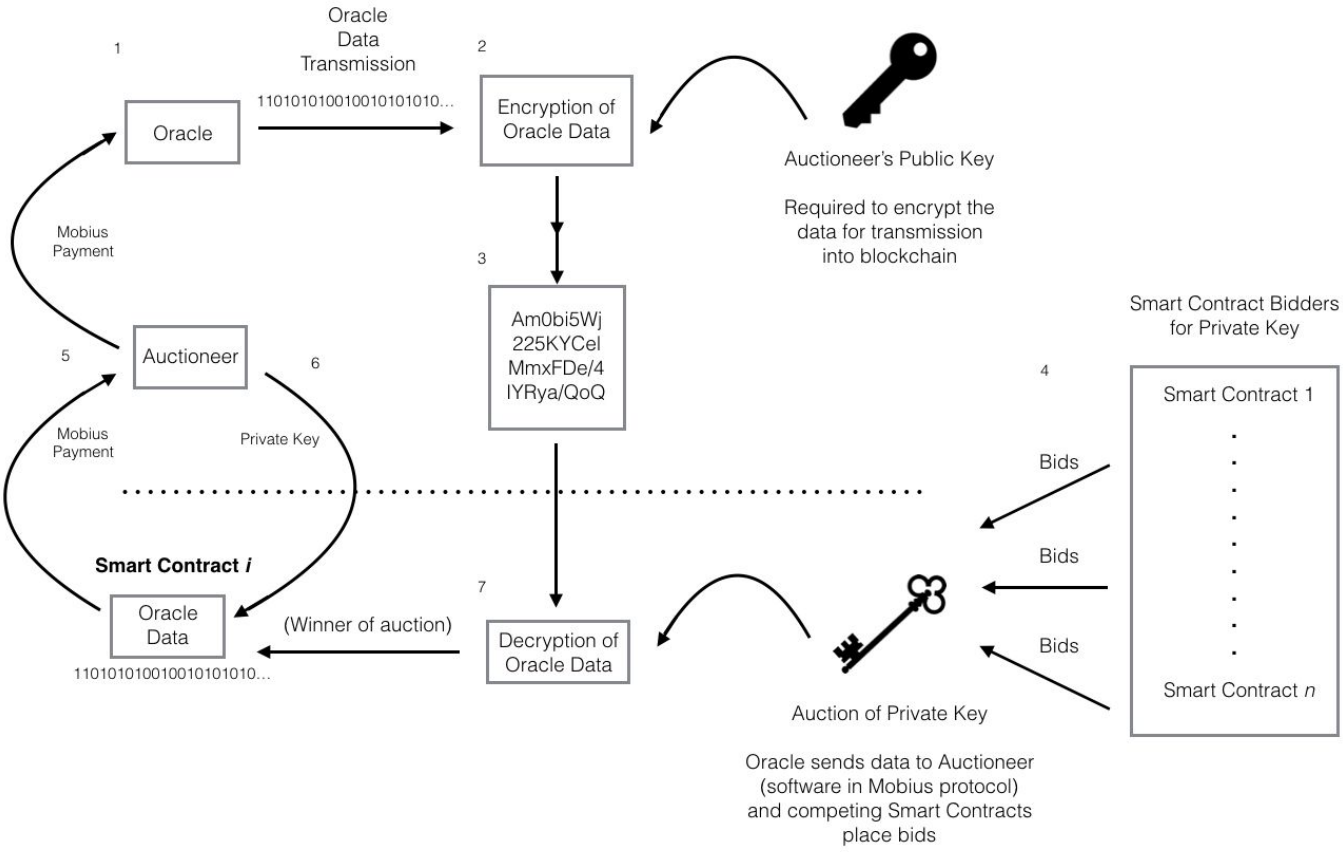
As per our market design methodology, we first define the goals:

- Facilitate an efficient exchange of information between oracles and end users through smart contracts
- Minimize false, invalid data; only authentic and validated data is exchanged
- Simple to understand and participate; transparent (or easy to replicate outcomes) and with similar outcomes in repeated trials
- All monetary transfers are settled in Mobius
- All data is encrypted, added to the blockchain or kept off-blockchain and information exchange is accomplished via the transfer of encryption keys between oracles and end users
- Incentivizes oracles to get data; auctions are voluntary or individually rational; bidders incentivized to participate (participants are at least better off taking part in the auction)
- Incentive-Compatible: Minimize strategic behavior, honesty is the best or dominant strategy for all participants; avoid windfalls for bidders and winner's curse problem
- Simple to understand and participate in auction; transparent and easy to replicate
- Shapley-Shubik Core prices: Prices are theoretically guaranteed to be close to the actual optimum value of the information (Shapley & Shubik 1969).

9.3 Design & Implementation

We propose a two sided auction to determine prices with some contractual language that guarantees data authenticity. Assuming that there are no scaling issues, data supplied by oracles would be encrypted and added to the blockchain. This would be implemented using an auction as follows:

Smart Market: Forward Auction on Blockchain



Note: This auction mechanism is implemented on a public blockchain with data submitted to a private blockchain or otherwise kept off-chain.

Conceptualization: Data As A Commodity

From a production view point, information or data (like IoT device data) is 'produced' according to predefined terms in the auction contract. Data integrity, authenticity and validity requirements are part of the terms that govern the auction. This is made available by 'oracles', encrypted and added onto the blockchain. Encryption keys are sold at auction. *RSA* or another form of encryption is used to encrypt the data before it is added to the blockchain.

An auction **Private** and **Public key** pair is created. The Public key is declared and added to the blockchain. Oracles wanting to take part in the auction add data to the blockchain and encrypt it using the auctioneer's public key (in this case, the auctioneer is the oracle). The winning bidders are then provided

with private keys to decrypt the data after the auction ends. This decryption takes place off chain to keep the data private.

Any and all transfers as part of the auction are settled in MOBI. In case the oracle sells multiple copies, then the data is encrypted with the public key and the private key is sold multiple times. Encryption keys are transferred to winning end users after payment in MOBI. The end users consume the data and have a fixed time to report any violations of terms. Oracles are paid after the expiry of this time period.

Auction Overall Structure

- Step 1: Forward Auction

Incentive Compatible Direct Mechanism for the sale of Information is required and we propose a modified Combinatorial VCG Auction. The auction allows multiple keys for the same data set to be sold. Bidders may bid additional amounts to restrict sale of additional keys

- Step 2: Reverse Auction

To run a reverse auction (procurement auction) to buy data from oracles. Simultaneous Descending (Dutch) auction is used to select oracles which would be selling the data to end users. Reserve price (opening price in case of the dutch auction) is set according to the closing price in the forward auction plus a markup. The initial price (or the effective reserve price) is the clearing price in the forward auction plus a markup.

- Step 3: Clearing Price

We combine bids in the reverse and forward auction to determine how many and which combination of keys to sell.

9.4 The Forward Auction

Encryption keys would be sold at a *forward* auction using a modified Combinatorial auction version of the weighted Vickrey-Clarke-Groves mechanism. We discuss a game theoretical set-up for VCG Auction of oracle data to bidders including end users or smart contracts (on private blockchains) below.

Given the following:

Consider a list of bidders participating in an auction of data. Let $N = \{0, 1, ...n\}$ denote the bidders taking part in the auction, with 0 denoting the auction operator. The auction operator is a type of a smart contract or a combination of a distributed demand and supply side platform. Let $\bar{t}$ be a vector of the type profiles of all the bidders. So $\bar{t} = (t^1, t^2, ..., t^N)$ includes a type for each bidder. Type t^i represents all the value, information, beliefs, utility and preferences of the bidder i.

Possible outcomes are (X, B):

Let X denote the set of all possible decisions in the auction. An outcome in the auction is denoted as $(x, \overline{m})$:

where

$$x \in X$$
$$\overline{m} = (m^0, m^1, ..., m^N)$$

$\overline{m}$ denotes a vector of positive or negative payments between bidders including the auctioneer and is measured in mobius.

Every smart contract values the outcome of the auction and we denote bidder i's value as $v^i(x, t^i)$. The value is dependant on the outcome of the auction and the type t^i of the smart contract. Furthermore, the utility that the bidder i derives from auction is:

$$u^i((x, m), \bar{t}) = v^i(x, t^i) - m^i$$

Given the assumptions and denotations above, and based on our stated market design goals, we must achieve allocative efficiency. Allocative efficiency denoted by the decision x^* is achieved if the auction decision maximizes the total value of the allocation.

$$\Rightarrow \quad x^* \in argmax_{x \in X} \sum_{i=1}^{N} v^i(x, t^i)$$

Furthermore, for budget balance, the auction should not produce a net loss.
The limiting condition in this case is:

$$m^0 \equiv 0$$
$$\Rightarrow \quad \sum_{i \in N} m^i = 0$$

Therefore, the auctioneer denoted as bidder 0 solves the following optimization problem to determine the outcome of the auction along with the constraints above:

$$V(X, N, \bar{t}) = max_{x \in X} \sum_{i=1}^{N} v^i(x, t^i)$$

$$\hat{x}(X, N, \bar{t}) \in argmax_{x \in X} \sum_{i=1}^{N} v^i(x, t^i)$$

This represents an incentive compatible direct mechanism that leads to the highest value smart contract winning the auction and paying its negative externality in the form of a payment measured in terms of mobius.

In the VCG auction, if any bidding agent i reports a type t^i, the VCG mechanism charges this smart contract a penalty if his report changes the optimal allocation x^* thereby hurting other bidders. Such a report is termed as **pivotal**.

$$\widehat{m^i}(X, N, \bar{t}) = \sum_{j \in N-i} v^j(\hat{x}(X, N-i, t^{-i}), t^j) - \sum_{j \in N-i} v^j(\hat{x}(X, N, \bar{t}), t^j)$$

This extra payment is specified to compensate the remaining bidders $N-i$ for their loss on account of smart contract i. This is exactly the negative externality of the smart contract.

The auction mechanism at equilibrium is incentive compatible. The weakly dominant strategy for each smart contract is to truthfully bid its true value. This is functionally equal to a second price auction.

9.5 How Much To Sell

A mechanism needs to be in place to maximize the revenue from the auction. Same data can be sold multiple times or equivalently multiple private encryption keys to the same data can be sold. The analog of this in the non-blockchain world is the sale of mp3 songs at a digital music store. This leads to some fundamental questions. How should bids be combined to determine the quantity transacted? Should the effect of bids on quantity transacted be recognized in the reverse auction incentive analysis?

This analysis depends on the type of data and the values of the agents bidding for the data. Agents will have the ability to be sole consumers of the data available at auction by outbidding the sum of all the competing bids. The number of mobius required to buy oracle data at equilibrium, essentially the clearing price, is set by the oracles as a response to smart contract demand. This **endogeneity** of the quantity of data traded holds true even in a two sided auction.

The Second Price Auction

We conduct a second price auction, the winning smart contract pays the second highest bid. The examples discussed below in this section take place on a private blockchain with proprietary methods to keep data consumption private.

For Example: Consider 1 oracle selling a single instance of a data set with three smart contracts bidding. Only one copy of the data is to be sold at auction. All transfers are in Mobius. The lowest bid increment in 0.01 mobius.

Smart Contract	Bid in Mobius (MOBI)	Outcome	Payment
S.C. #1	7	Won	5.01 MOBI
S.C. #2	5	Lost	0
S.C. #3	4	Lost	0

The highest bidder S.C. #1 wins the auction and gets the private encryption key of the oracle data set, but only pays 5.01 Mobius, the second highest bid incremented once. In this scenario, the highest bidding smart contract is charged the minimum bid it needed to rank above its closest competitor and win the contract.

This ensures that smart contracts bid their true actual private values and that there are no incentives for smart contracts to place bids much lower than their actual values.

Selling Multiple Keys and Incremental Bidding

Multiple keys to the same data can be sold. The oracle may sell the same data multiple times, this can be easily accomplished by selling multiple private encryption keys.

Incremental Bidding
For Example: Forward auction for the sale of IoT data supplied by 5 distinct oracles, only two encryption keys are to be sold per oracle. Three smart contracts participate. As before auction is on a private blockchain.

Smart contracts place binary demand tuples of the keys that they want, where a 1 indicates wanted & 0 indicates not wanted.

Smart Contract	Demand Set	Total demand quantity	Bid per oracle	Set Won	Payment required Per oracle	Total payment
S.C. #1	{1,1,1,1,1}	5	9	{1,1,1,1,1}	{5.01,2.01,5.01,5.01,9}	26.04 MOBI
S.C. #2	{1,0,1,1,0}	3	5	{1,0,1,1,0}	{5,0,5,5,0}	15 MOBI
S.C. #3	{1,1,0,1,0}	3	2	{0,1,0,0,0}	{0,2,0,0,0}	2 MOBI

The winning smart contract pays its bid in Mobius unless there is a lower winning bidder. If there is a lower winning bidder, then the winning smart contract pays the lower bidders price.

Blocking Additional Sales

The smart contracts will have the option to block further sales by paying an increment called the blocking increment. The blocking increment is the sum of all the blocked bids. All blocking increments must be payed in Mobius.

Only highest value bidders are allowed to block, if the blocking smart contract is not the highest bidder then the blocking bid is treated as a normal bid.

STEP 1: The auction is run as if no blocking takes place, and second price payments in Mobius are calculated.

STEP 2: Blocking increment is calculated as the sum of all blocked winning bids:

$$\text{Blocking increment} = \sum_{i \in \textit{blocked bids}} m^i \quad .$$

STEP 3: All blocked winning bids are then added to the blocking winners payments calculated in STEP 1. The blocked winner's bids are decremented appropriately from their payments.

Blocking Winner Payment = Second price bid + Blocking increment
Blocked Winner Payment = Second price bid - Blocked increment

Additionally, in the combinatorial auction, oracle data is priced incrementally.

Thus, any smart contract buying oracle data will automatically be charged the blocking increment for the section of data that no other smart contract ends up buying.

Smart contracts place demand tuples of the keys that they want, where 1 indicates wanted, 0 indicates not wanted, X indicates blocking.

Smart Contract	Set Demanded	Quantity Demanded	Bid per oracle in Mobius	Second Price Set Won	Second Price Payment	Total payment
S.C. #1	{X,X,X,1,1}	5	9	{1,1,1,1,1}	{5.01,2.01,5.01,5.01,9}	26.04 MOBI
S.C. #2	{1,0,1,1,0}	3	5	{1,0,1,1,0}	{5,0,5,5,0}	15 MOBI
S.C. #3	{1,1,0,1,0}	3	2	{0,1,0,0,0}	{0,2,0,0,0}	2 MOBI

Smart Contract	Blocked Set	Blocked Bids	Calculation	Final Payments	Set Won
S.C. #1	{0,0,0,0,0}	{0,0,0,0,0} = 0	26.04 + 10	36.04 MOBI	{1,1,1,1,1}
S.C. #2	{X,0,X,0,0}	{5,0,5,0,0} = 10	15-10	5 MOBI	{0,0,0,1,0}
S.C. #3	{0,X,0,0,0}	{0,2,0,0,0,} = 2	2 - 2	0 MOBI	{0,0,0,0,0}

Revenue Equivalence: Both auctions with blocking and without blocking yield the same revenue. This is important to avoid any adverse incentives to manipulate bids by the blocking winner to pay lower amounts.

Unlimited Supply & Singleton Demand

If blocking is not an option and only complete sets of oracle data are to be traded. This represents the scenario where k private encryption keys of oracle data are available for sale, with N smart contracts wanting to buy exactly one key. Under VCG auction rules:

$$\text{if } k < N$$
$$\textit{clearing price} = b^{k+1}$$

$$\text{if } k \geq N$$
$$\textit{clearing price} = 0$$

In the case of unlimited supply, the keys sell for zero under VCG conditions. Furthermore, the VCG mechanism implies that every key is simply sold for the amount of the $k+1st$ highest bid if we can restrict the number of keys being auctioned to less than the number of bidders.

The first key may actually be very expensive to produce but oracles can produce additional keys at zero marginal cost, hence there is effectively an unlimited supply of encryption keys. The oracles do not have any supply restrictions other than those they voluntarily impose themselves or restrictions imposed by the auctioneer. The oracles and the auctioneer actually will have to artificially reduce supply in order to increase revenues.

Given a truthful valuation reporting mechanism an optimal single price m^* and optimal quantity of encryption keys k^* for clearing the auction with maximum revenue is solved as follows:

Let there be N smart contracts with values v^i arranged as an ordered set such that:

$$v^i > v^{i+1}$$

$$\Rightarrow \quad k^* \in argmax_{i \in N} i * v^i$$
$$\Rightarrow \quad m^* = v^{k^*+1}$$

The optimal singleton price is the $k^* + 1th$ highest bid which in the VCG mechanism is the value of the $k^* + 1th$ highest bidding smart contract.

An approximate auction in the absence of a truthful ordering of values or a very large number of smart contracts can be done by randomly sampling valuations.

The random sampling optimal price auction game may be played as follows:

Step 1: Randomly partition the smart contracts N into two sets $N1$ and $N2$.
Such that each smart contract has probability = $\frac{1}{2}$ of being randomly assigned to one of the sets and

$$N = N1 \cup N2$$
$$N1 \cap N2 = \oslash$$

Step 2: Using the above procedure find payments m_1^* & m_2^*; the optimal singleton prices for each set.

Step 3: Clearing prices and allocations are calculated as follows:

$\forall i \in N1$ if $v^i > m_2^*$
Smart contract i wins a single key and pays $m = m_2^*$

Similarly,

$\forall j \in N1$ if $v^j > m_1^*$

Smart contract j wins a single key and pays $m = m_1^*$

Random sampling optimal price auctions are **dominant-strategy truthful**, **weakly budget balanced** and **ex post individually rational**.

If a minimum two keys are sold at auction then it can be shown that in this random sampling optimal price auction game

$$Expected\ Total\ Revenue \geq \frac{1}{4.68}\ m^*$$

10. Smart Markets Use Cases

10.1 Block Button Auction

Button auctions are those where the only options for the bidder is to decide when to withdraw from the auction. The block button auction is a demand side forward auction when static oracle data supply is available. Smart contracts or end users are the participants and are assisted by demand side distributed platforms. All transactions are settled in Mobius and are settled in the next block created at the end of the auction.

In the block button auction, any participating smart contract has one choice to make, whether to participate in the next block or drop out. Once a smart contract drops out there is no re-entry. The last smart contract remaining wins the auction. Each round the bids and demand tuples change.

At the start, all interested bidders are participants and the oracle data price is its reserve price in Mobius. After each new block is created the price increments by a fixed amount of Mobius. If a bidder is willing to pay the current block price it must keep participating, otherwise the smart contract drops out of the auction. Re-entry is strictly prohibited. If in any block there is only one smart contract remaining, then that smart contract is the winner. The accounts are settled in the next block created with the exchange of private encryption keys for final display price in mobius.

Suppose a bidder has a private value v for the data. Then its dominant strategy is to participate till the block price exceeds v.

This leads to an incentive compatible truthful mechanism: bidders will act on their private true value, regardless of other participating bidders.

At equilibrium all bidders play their dominant strategies, the outcome is:

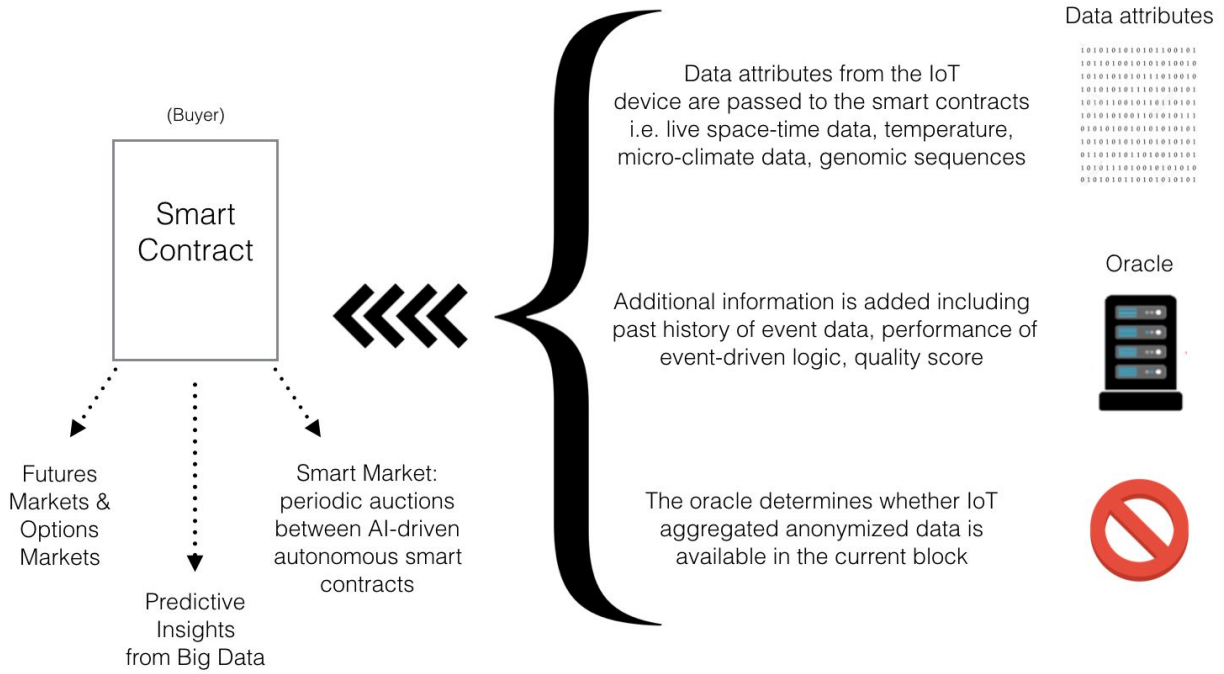
- The winning bidder is the buyer with the highest valuation.
- The final block price is the second highest bidder's valuation.

In a simultaneous descending auction situations may arise where different bidders have differing preferences on the packaging of oracle data sets or where auction requirements lead to intractable constraints on the auction optimization problem. In such scenarios a **package auction** may lead to better results like more participants, higher revenues and more efficient allocations.

10.2 Real Time Intra-block Bidding

Applications in:

1) IoT Data Marketplace
2) Demand & supply side distributed platforms



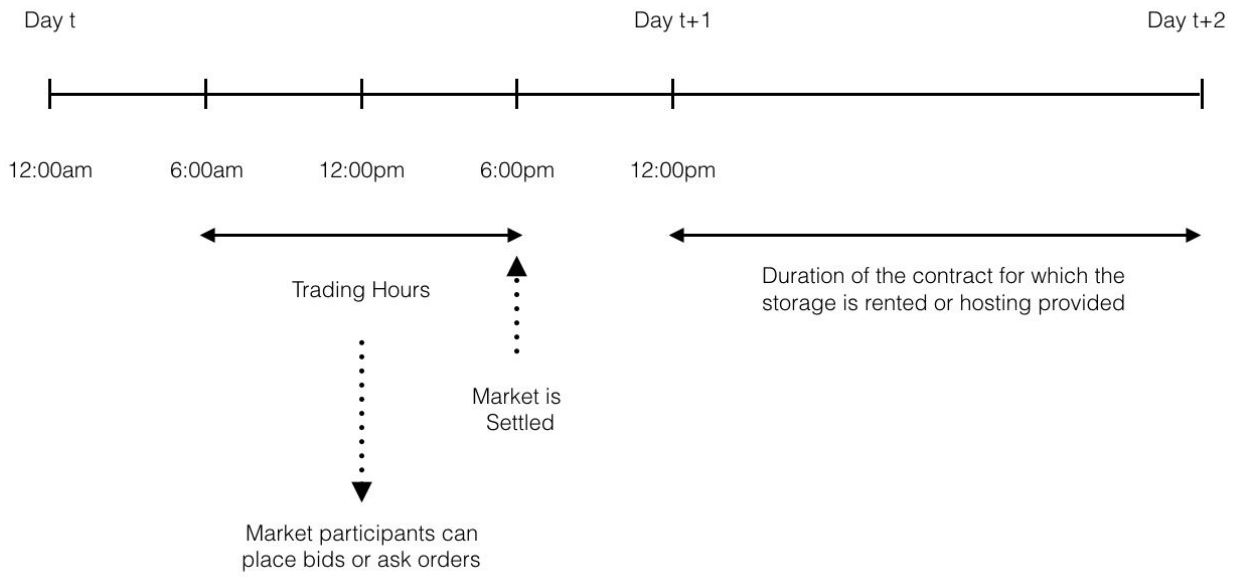
Real-time block bidding is the buying and selling of oracle data through real-time auctions that occur in the time it takes a new block to be added to the blockchain. All transactions are settled in MOBI and must complete before the next block is created. The system may actually run in real time and be independent of the blockchain adding new blocks.

Those auctions are to be facilitated by supply-side distributed platforms, which are designed to assist oracles to sell their data. Here, the entire trade must take place within the time it takes to create a new block.

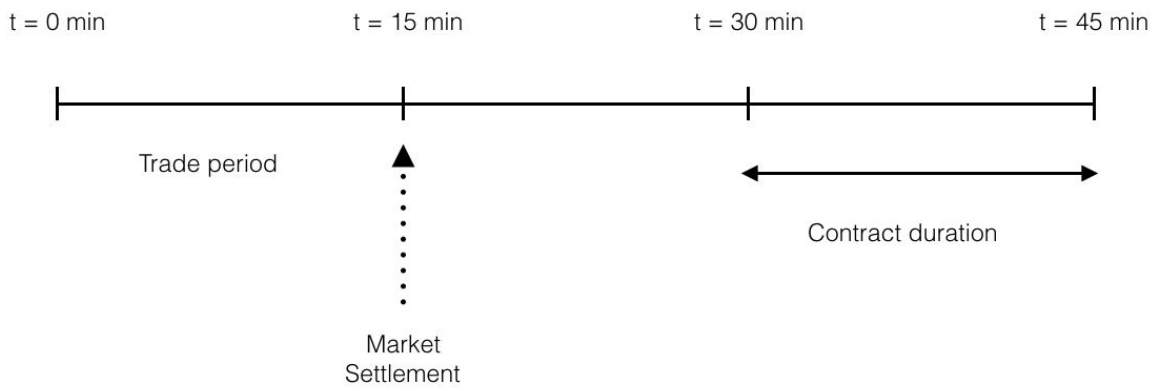
10.3 Cloud Services Market

Apart from long term contracts, a simple order matching market suffices for long term contracts. These may be termed as **Look Ahead** markets for web services on the cloud including cloud based file storage. Consider a **Day Ahead** market. The following setup is for a generalized system in trade of cloud based web services over the blockchain. As an example, let us consider a market to rent and host storage space for the next calendar day in 15 minute blocks. It is a closed double-sided anonymous auction for each 15 minute blocks for the following day.

Timeline



A **Term Ahead** Market provides for trade in storage 15 minutes after settlement time. This is carried out for each 15 minute block in the term ahead or the day ahead market.



- Buy Side

Interested buyers specify requested durations, level of data redundancy and maximum buying price.

- Sell Side

Sellers specify the time blocks, capacity in TB, minimum price in mobius and Bandwidth charges. All sellers are required to provide stakes as collateral against defaulting on their potential contracts. The stake sizes are fixed by the auctioneer and pro-rated according to TB.

- Market Clearing

At the end of the current trade/call period, the bid and ask orders are accumulated to determine market clearing price and market clearing volume. Orders are matched after each call period. An equilibrium or clearing price is calculated and all orders are settled at this clearing price.

11. Development Progress

Mobius was entirely self-funded prior to the Pre-Sale.

The first version of the Mobius Universal Blockchain Payment Protocol is in beta. If you are interested in using it please email beta@mobius.network.

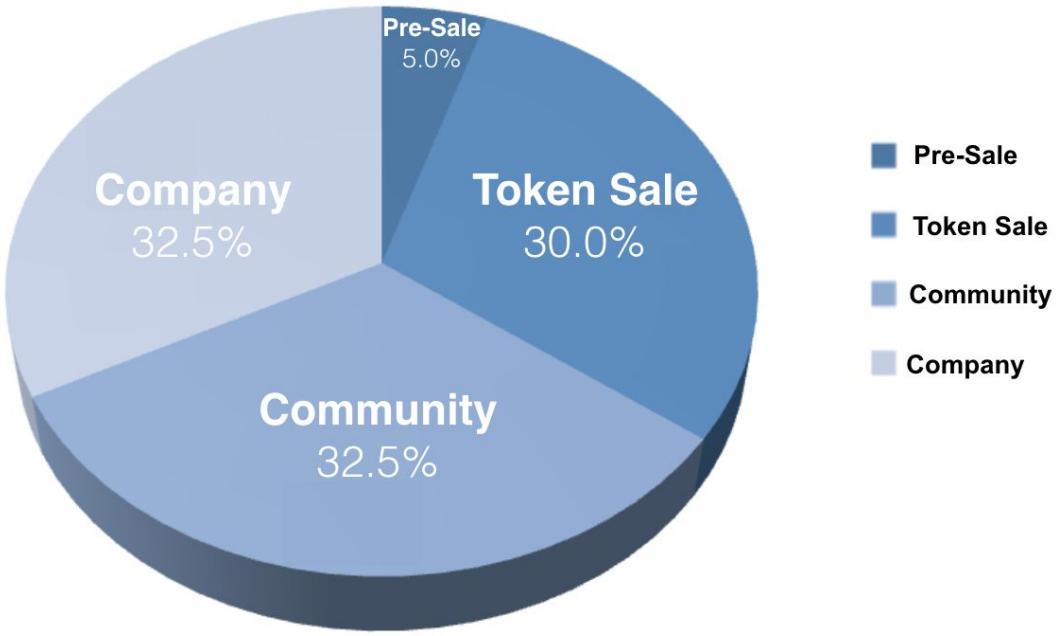
12.Roadmap

13. MOBI Allocation Summary

In the MOBI Smart Contract Token Generation 888,000,000 MOBI will be created. The majority of MOBI will be sold to the public or reserved for the community to incentivize network growth and reward contributors.The remaining tokens will be held by the company.

Prior to the token sale, up to 5% of MOBI out of the community portion will be distributed to DApp Store Apps. The distribution process will take place by giving users that sign-up for the DApp Store prior to the token sale credits in the DApp Store than can be deposited into Apps. After the token sale, any credits that a developer has earned through their Apps will be converted into MOBI on a 1 to 1 basis. This distribution will bootstrap the DApp Store and drive submission of apps into the DApp Store prior to the token sale, demonstrating the store's real-world value and use.

MOBIUS TOKEN DISTRIBUTION



- 5.0% of MOBI will be sold in a pre-sale at a discounted rate to the token sale.
- 30.0% of MOBI will be sold in the token sale.
- 32.5% of MOBI will be reserved for the community to grow the network and reward contributors.
- 32.5% of MOBI will be reserved by the company for growth and R&D.

14. MOBI Token Sale Details

The token sale is scheduled for November 8th, 2017. If interested in purchasing ahead of time please emails tokens@mobius.network.

15. Team

Founders

David S. Gobaud

Stanford University
B.S. Computer Science
Harvard Law School, J.D.
Past: Co-Founder, Yoshi
White House
Y Combinator
Resume

Cyrus S. Khajvandi

Stanford University
B.S. Biological Sciences, with Honors
Past: Co-Founder & CEO, Incentru
Advisor to Credo & BitBounce.io
Early Adopter of BTC, ETH/C
NSF & HHMI Researcher

Monis Rahman

Stanford, Ph.D. Computational & Mathematical Engineering (on leave)
Auction & Market Design, Deep Learning, Big Data
Past: CTO, Next 2 Percent
Stanford University School of Medicine, Researcher in Immunology
Purdue, B.S. Computer Engineering

Advisors

Jed McCaleb

Co-Founder of Stellar.org
Co-Founder of Ripple
Creator of eDonkey2000

Jackson Palmer

Creator of Dogecoin

Daniel Cawrey

Co-Founder & CEO Pactum Capital

16. Supporting Documents & Links

Important Information

- Website: https://mobius.network
- DApp Store: https://mobius.network/store
- Dev API Docs: https://mobius.network/docs/
- Node SDK: https://www.npmjs.com/package/@mobius-network/mobius-node
- Email: http://eepurl.com/cWcydr
- Blog: https://medium.com/mobius-network
- Github: https://github.com/mobius-network/

Community

- Twitter: https://twitter.com/mobius_network
- Rocket Chat: https://mobius.rocket.chat