# Contents

**1 Introduction** **4**

**2 Methodology** **4**

**3 VDex** **4**

3.1 Components . . . . . 4

3.1.1 Operating system . . . . . 4

3.1.2 Inter Contract Communication . . . . . 6

3.1.3 Side Chains . . . . . 6

3.1.4 Liquidity . . . . . 6

3.1.5 Hashed timelock contracts . . . . . 6

3.2 Database model . . . . . 6

3.3 Network Topology . . . . . 7

3.3.1 Nodes . . . . . 7

3.3.2 Hubs . . . . . 7

3.3.3 Latency . . . . . 8

3.4 OrderBook . . . . . 8

3.4.1 Data structures . . . . . 8

3.4.2 Operations . . . . . 8

3.4.3 Online order book . . . . . 8

3.4.4 Offline order book . . . . . 8

3.4.5 Order(Loopring) . . . . . 9

3.4.6 Order(EOSIO) . . . . . 9

3.4.7 Consensus model . . . . . 10

3.5 Ring mining . . . . . 11

3.6 VTX . . . . . 11

3.6.1 Token creation . . . . . 11

3.6.2 Token use . . . . . 12

3.6.3 Token distribution . . . . . 12

3.6.4 Fee model . . . . . 13

3.7 Inter blockchain communication . . . . . 14

3.8 Security model . . . . . 14

3.8.1 Introduction . . . . . 14

3.8.2 General actions . . . . . 15

3.8.3 Contract security . . . . . 15

3.8.4 Malware detection by auditing processes . . . . . 15

3.8.5 Random diversification . . . . . 15

3.8.6 Multiple factor identification . . . . . 16

3.8.7 Logs . . . . . 16

3.8.8 Transaction as Proof of Stake (TaPoS) . . . . . 16

3.8.9 Double spend . . . . . 16

3.8.10 Front running . . . . . 16

3.8.11 Forged identities . . . . . 17

3.8.12 Insufficient Balance . . . . . 17

3.8.13 Timing attack . . . . . . . . . . . . . . . . . . . . . . . . . . 17
3.8.14 EOSIO provisions . . . . . . . . . . . . . . . . . . . . . . . . 17
3.9 Inter blockchain communication . . . . . . . . . . . . . . . . . . . . 17
3.10 Multi blockchain . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
3.11 User experience . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
3.12 True decentralization . . . . . . . . . . . . . . . . . . . . . . . . . 18
3.13 System recovery . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
3.14 Scalable and modular architecture . . . . . . . . . . . . . . . . . . 19
3.14.1 Problem decomposition . . . . . . . . . . . . . . . . . . . . . 19
3.14.2 Minimize state space . . . . . . . . . . . . . . . . . . . . . . . 19
3.14.3 Determine and minimize state replication . . . . . . . . . . 19

**4 Risk** **19**

**5 Timeline** **20**

**6 Conclusion** **20**

# VDex White Paper v0.5

The Volentix Labs Team
info@volentixlabs.com

June 29, 2018

**Abstract**

The capacity of systems to recursively self regulate is one of the mechanisms used by evolution to enhance species. Distributed applications are doing this very well and their influence can be seen in the architecture and designs of emerging cryptocurrency exchanges. Because they handle such large amounts of capital, exchanges are powerful entities, in a global sense. Can the adoption by traders of decentralized exchanges act as a driving force in the acceptance and proliferation of decentralized ideology as a whole? With this momentum we introduce VDex, a decentralized exchange with the user and community in mind. Using some of the most recent paradigms and established protocols for security, ease of use and multi asset support, this low friction peer-to-peer exchange abides by open standards and ensures a harmonious and seamless flow among decentralized applications. Focused on functionality, this collection of smart EOSIO contracts are publicly accessible and contain easy to use options for security, anonymity, speed of payment, liquidity and profit margin. VDex is a DAO and its governance allows for non disruptive and collaborative action among VTX holders towards the growth and stability of the VTX token.

## 1 Introduction

VDex should provide easy to use settings for speed, cost, anonymity and security within the scope of scalability. Power users should feel enabled with the freedom to choose and thrive. Equally, new users should feel welcomed and secure. The sustainability of the product will reside on an architecture whith the ability to replace its own components, integrate new technologies while allowing for the migration of 2nd generation blockchain functionality. The task of building VDex mainly resides in successfully merging and integrating today's best protocols, paradigms and patterns to match the Volentix requirements on top of the EOSIO decentralized operating system. To provide a flexible modular product, developer will needed an environment that serves its purpose without burdening design with redundant complexity.

## 2 Methodology

All assumptions made in this paper will be verified by prototyping with EOSIO's ***cleos*** command line through functional python testing. This software resides at: https://github.com/Volentix/ezeos

## 3 VDex

### 3.1 Components

#### 3.1.1 Operating system

EOSIO is an operating system-like framework upon which decentralized applications can be built. The software provides accounts, authentication, databases, asynchronous communication and scheduling across clusters. Components and protocols are already built into the platform and just a subset can be used to satisfy VDex requirements. VDex will initially benefit from the most standard features offered by EOSIO such as account and wallet creation and the recovery of stolen keys but will subsequently have to implement the paradigms and patterns suggested by most protocols for the creation of decentralized exchanges through its contracts and provided tools:

1. **Schema defined messages and database**
   The *Standardized Service Contract* design principle advocates that the

service contracts be based on standardized data models. Analysis is done on the service inventory blueprint to find out the commonly occurring business documents that are exchanged between services. These business documents are then modeled in a standardized manner. The Canonical Schema pattern reduces the need for the application of the data model transformation design pattern.

2. **Binary/JSON conversion**
   The contracts provide the human readability of JSON and the efficiency of binary

3. **Parallelisation and optimisation**
   Separating authentification from application provides scalability by offloading the blockchain.

4. **Web Assembly**
   Web assembly has the highest performance for web application while also providing a sandbox to secure applications
   Sandboxing is a software management strategy that isolates applications from critical system resources and other programs. It provides an extra layer of security that prevents malware or harmful applications from negatively affecting system. It can be implemented by enforcing the following:

   (a) Network-access restrictions.
   (b) Restricted filesystem namespace.
   (c) Rule-based execution.
   (d) Control over what processes are spawned.
   (e) Read and write control.
   (f) Garbage collection.

5. **C/C++ contracts**
   - STL and Boost containers
   - C++ 14 The exchange will have a polling micro-service which will fetch the next unprocessed action. By processing the history the system will be informed of when the transaction was confirmed. For example, an exchange-specify memo may be embedded on the withdraw request which can be used to map to the private database state, which in turn tracks the withdraw process.

6. **Context Free Actions**
   Most of the scalability techniques proposed by Ethereum (Sharding, Raiden, Plasma, State Channels) become much more efficient, parallelizable, and practical while also ensuring efficient inter-blockchain communication and unlimited scalability. A Context Free Action involves computations that depend only on transaction data, but not upon the blockchain state. ex: Parallel processed signature verifications

### 3.1.2 Inter Contract Communication

Data is shared between contracts via an oracle. Most reading the of data is done via polling **nodeos** (the blockchain instance)

### 3.1.3 Side Chains

For inter-blockchain communication, the EOSIO protocol creates a TCP like communication channel between chains to evaluate proofs. For each shard (a unit of parallelizable execution in a cycle) a balanced merkle tree is constructed of these action commitments to generate a temporary shared merkle root. This is done for speed of parallel computation. The block header contains the root of a balanced merkle tree whose leaves are the roots of these individual shard merkle trees.

### 3.1.4 Liquidity

In order for a token to effectively partake in the global token economy, its trading volume must cross a critical barrier where the matches between buyers and sellers become frequent enough to ensure a stable pool of ”coincidences of wants”. This reliability within an exchange is known as liquidity. We say a token is liquid if it is easily possible to buy or sell it without considerably affecting its price.

1. Implementation of the Loopring protocol with the use of EOSIO contracts acting as nodes
2. Implementation of the Bancor protocol used to bring stability to the token.

### 3.1.5 Hashed timelock contracts

A Hashed Timelock Contract (HTLC) is a type of smart contract enabling the implementation of time-bound transactions. Users will be offered a variable lock in period for their transations choosing or not to lock in their funds for a slightly greater time than the slowest blockchain in exchange for a discount on the transaction.

## 3.2 Database model

Each account has its own database which communicates through other accounts through message handlers.

**Database Iterators** Database API that is based upon iterators. Iterators give WebAssembly a handle by which it can quickly find and iterate over database objects. This new API gives a dramatic performance increase by changing the complexity of finding the next or previous item in a database from O(log(n)) to O(1).

**Persistence API** EOSIO provides a set of services and interfaces that enable contract developers to persist state across action and transaction boundaries. Without persistence, state that is generated during the processing of actions and transactions will be lost when processing goes out of scope.

The persistence components include:
- Services to persist state in a database
- Enhanced query capabilities to find and retrieve database content
- C++ APIs to these services
- C APIs for access to core services

**MongoDB bridge for other networks** Database state should be available for Raiden, Plasma, State Channels

## 3.3 Network Topology

### 3.3.1 Nodes

All wallets that have sufficient funds and well know to the network can be used as nodes

1. Merge order book information with others through inter contract communication sharing database state or with high speed dedicated channels for offline communication.
2. Nodes will mine order ring or chosen data structure.

### 3.3.2 Hubs

Hubs are geographically distibuted dedicated harware modules to speed up and ensure information flows across the network in an balanced and efficient manner.

1. Order cancelling for single and multi signatures
2. Scheduler (Timeouts)
3. Asset tokenization services
4. Order book browser populating
5. Load balancing
6. log analysis for security
7. receives logs from nodes
8. relays Vespucci information to Verto

### 3.3.3 Latency

EOSIO has low latency block confirmation (0.5 seconds). This latency can be provided if the currencies being traded are issued from blockchains that are equally fast, other wise, the transaction is as fast as the slowest block chain. On completion of the transaction, a transaction receipt is generated. Receiving a transaction hash does not mean that the transaction has been confirmed, it only means that the node accepted it without error, which also means that there is a high probability other producers will accept it. By means of confirmation, you should see the transaction in the transaction history with the block number of which it is included.

## 3.4 OrderBook

The flexibility of certain protocols allows for the malleability of the order book. According to various scenarios or concensus algorithms, VDex order books could morph to accomodate their cost, bandwidth, and security criteria:

### 3.4.1 Data structures

1. Loopring FIFO
2. EOSIO multi-index

### 3.4.2 Operations

1. Remove the existing orders if no match is found.
2. Add a new order that didn't fully match.
3. Remove a cancelled order.
4. Merge with remote book.
5. Morph to a new container.(Online only)

### 3.4.3 Online order book

Order books reside in a persistent container on each node belonging to the same account as all the other nodes.

### 3.4.4 Offline order book

A sparse distributed representation of hub log data and order book will act as an extra layer of magnification for anomalies. Also with concepts such as hierachical temporal memory and the BlockCollider protocol, offline order books will contribute to realtime analysis in **Vespucci** to create a VDex meta blockchain ledger.

### 3.4.5 Order(Loopring)

Listing 1: C++ code using listings

```
message Order {
address protocol;
address owner;
address tokenS;
address tokenB;
uint256 amountS;
uint256 amountB;
unit256 lrcFee
unit256 validSince; // Seconds since epoch
unit256 validUntil; // Seconds since epoch
marginSplitPercentage; // [1-100]
bool buyNoMoreThanAmountB;
uint256
walletId;
address authAddr;
// v, r, s are parts of the signature
uint8 v;
bytes32 r;
bytes32 s;
// Dual-Authoring private-key,
of our own order book follows an OTC model, where limit
// not used for calculating order's hash,
orders are positioned based on price alone. Timestamps of
string authKey;
uint256 nonce;
}
```

### 3.4.6 Order(EOSIO)

Listing 2: C++ code using listings

```

struct limit_order {
    uint64_t     id;
    uint128_t    price;
    uint64_t     expiration;
    account_name owner;

    auto primary_key() const { return id; }
    uint64_t get_expiration() const { return expiration; }
    uint128_t get_price() const { return price; }

    EOSLIB_SERIALIZE( limit_order , ( id )( price )( expiration )( owner ) )
};
```

### 3.4.7 Consensus model

VTX is required to participate in the consensus process and earn both block rewards and transaction fees. Time is an essential component to control distributed systems. No one clock can reside on one system to ensure a decentralized system. Since the task of matching orders is fairly easy, the simplest solutions will be chosen for what will be considered "Mining" of the VTX token. Providing a way to easily replace protocols as new ones emerge can ensure an even more democratic approach.

**Raft** Raft is a protocol using RPCs to ensure consensus among a node cluster. It is remarquable in its simplicity and effectiveness. It consists of log propagation mechanism punctuated by leader election

1. Leader election
   - Randomized timeouts are stamped on nodes
   - Timeout triggers candidacy
   - Follower becomes leader and starts election when timeout is reached
   - If leader discovers follower with larger term, it updates its term, cancels election and reverts to follower. - elections have to happen within the shortest timeout of all the
   followers within the cluster - If election is stuck because of two nodes timing out at the same time,
   next timeout is used
   - Cluster not accepting data from client in election phase.
   - Time to send messages to followers must be less than the time of the shortest timeout within the followers.

2. Log replication
   - Leader takes commands from clients, appends to log
   - Leader replicates its log to others
   - Commit state
   - Consistency checks for missing or extranuous entries

3. Safety
   - Only server with up to date log can become leader
   - No votes because no logs.

This prevents forking and allows for a strong consistency model.

As enunciated above, other protocols such as PARSEC and the introduction of Directed acyclic graphs are being considered but on the other hand The following protocols have been discarded:

**2PC** Commit-request phase (voting phase), and the commit phase, where the coordinator decides whether to commit or not, based on the information gathered in the voting phase. Problems in the presence of failures. Assumes there is storage that can be trusted at each node, that no node crashes, and that nodes can communicate with each other. Blocking protocol.

**MongoDB's consensus protocol** MongoDB uses asynchronous replication by default so there is a risk of losing data when the primary goes down. Async replication is fast since it needs not wait for the slaves to acknowledge a write before telling a client that everything is saved. But, this scheme will lose data when the master goes down because the client will think everything is safe when in fact its recent writes are gone.

**Augmented polling** As an excercise for democratizing the platform, Pre-sale VTX holders, founders and contributors participate in a proof of non-repudiation process(1 time mining) for token allocations.

## 3.5 Ring mining

**Mining Criteria**

1. Check for subrings
2. Load Balancing (Fill rate/stock vailability)
3. Rate is equal or less than the original buy rate set by user
4. Process cancellations
5. Order are scaled according to the history of filled and cancelled amounts

## 3.6 VTX

### 3.6.1 Token creation

The **eosio.token** contract from the EOSIO framework can be used to issue EOSIO compliant tokens. VTX should be generated on demand with each request for a transaction.
The purpose of the token is to create liquidity in the VDex exchange. All orders values are momentarily converted to VTX to ensure maximum demand and growth of the token.

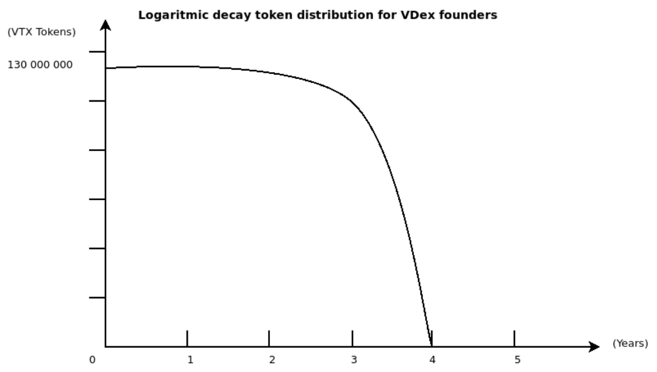

Figure 1:

### 3.6.2 Token use

The token can be used:

1. to pay for fees to use the exchange.
2. to stake and accumulate value.
3. to ensure mining incentive.

### 3.6.3 Token distribution

Table 1: Token distribution

| **amount** | **details** |
|---|---|
| 2.1 Billion Tokens | Supply 1.3B |
| 5% Pre-public crowdsale | 0.000016-0.000020 BTC |
| 28% Public crowdsale | 0.000021 BTC+ |
| 10% Founders | Time Locked* |
| 10% (130M VTX) Prior Work, Team and Advisors | Time Locked* |
| 35% Future Decentralized Treasury | |
| 12% Core Development) | |

*Logarithmic decay token distribution over 4 years.

Pre-sale VTX holders, founders and contributors participate in a proof of non-repudiation process(1 time mining) for token allocations. The algorithm used implements a augmented vote for each of the modified percentages among VTX members.

### 3.6.4 Fee model

#### Transaction fees

1. Fees can be collected in any currency.
2. Fees can be assigned to specific accounts
3. A small fee collected for general maintenance by developers

#### Lock-in period

1. User can lock in funds for 24hrs and have a free transaction
2. User can lock in funds for 24hrs+ and generate VTX

#### Mining fees

1. A pre-determined amount of VTX is required to participate in the consensus process
2. A wallet can become a node and earn VTX through mining
3. From the loopring protocol: 'When a user creates an order, they specify an amount of VTX to be paid to the ring-miner as a fee, in conjunction with a percentage of the margin (marginSplitPercentage) made on the order that the ring-miner can claim. This is called the margin split. The decision of which one to choose (fee or margin split) is left to the ring-miner. This allows ring-miners to receive a constant income on low margin order-rings for the tradeoff of receiving less income on higher margin order-rings.'
    (a) If the margin split is 0, ring-miners will choose the flat VTX fee and are still incentivized.
    (b) If the VTX fee is 0, the income is based on a general linear model.
    (c) When the margin split income is greater than 2x(VTX fee), ring-miners choose the margin split and pay VTX to the user.

**EOSIO** If the main EOSIO network scales according to plan and is used for building VDex, the following considerations will be applicable for deploying the exchange:

1. Deploying contract has cost but free to use.
2. Developers have to stake eos tokens to deploy contract. After the contract is destroyed, the locked tokens are returned.
3. Dapps must allocate resources to their contracts, memory, cpu, bandwidth.
4. The payer of resources is up to the dapp.
5. Multiple messages in one transaction and multiple accounts can be asigned to the same thread.

**Budget forecast** Since the acceptation of decentralized exchanges is not just yet arrived, but is deemed to happen, it is safe to say that the value of VTX should increase with demand over a longer period of time, say 5 years.

## 3.7 Inter blockchain communication

EOSIO is designed to make Inter-Blockchain-Communication (IBC) proofs lightweight. For chains with an insufficient capacity for processing the IBC proofs and establishing validity, there is the option to degrade to trusted oracles/escrows. To directly control other currency transactions with an EOSIO based smart contract a trusted mutisig wallet holding the currency in escrow is used to persuade the signing/publishing of the currency transaction based on IBC proofs from the originating chain.

## 3.8 Security model

### 3.8.1 Introduction

The assumptions made in this section are made from the analysis on gathered information. Full security prototyping, testing and securing will occur according to the chart in the **Timeline** section. The security concerns with the VDex environment can generally be categorized as variants of the following scenarios:

1. The attacker executes malicious code within a transaction
2. The order of transactions is manipulated
3. The timestamp of a block is manipulated

### 3.8.2 General actions

1. Filter incoming data
   (a) Data from blockchains
   (b) Data received by nodes
   (c) Data received by wallet
2. Filter outgoing data
   (a) Scheduler
   (b) Cancel order
   (c) Commit

### 3.8.3 Contract security

1. Keep vast majority of funds in a time-delayed, multi-sig controlled account
2. Use multi-sig on the hot wallet with several independent processes/servers double-checking all withdrawals
3. Deploy a custom contract that only allows withdrawals to KYC'd accounts and require multi-sig to white-list accounts
4. Deploy a custom contract that only accepts deposits of known tokens from KYC'd accounts
5. Deploy a custom contract that enforces a mandatory 24-hour waiting period for all withdrawals
6. Utilize contracts with hardware wallets for all signing, even automated withdrawals
7. Seamless sytem to upgrade broken contracts
8. Ability to pause the functionality of a contract
9. Ability to delay an action of a contract

### 3.8.4 Malware detection by auditing processes

The system will provide insights on rogue processes during the transaction period.

### 3.8.5 Random diversification

By using the raft protocol in the election process, a certain level of randomization is aquired.

### 3.8.6 Multiple factor identification

As in many existing applications, this measure is efficient and already known to the public at large.

### 3.8.7 Logs

Ensure inspection of logs as control.

1. Raft.
2. Anomaly detection with AI(Numenta).
3. Script investigations of certain non token purchases related addresses.

### 3.8.8 Transaction as Proof of Stake (TaPoS)

Prevents a replay of a transaction on forks that do not include the referenced block signals the network that a particular user and their stake are on a specific fork.

### 3.8.9 Double spend

A double spend is an attack where the given set of coins is spent in more than one transaction.

1. Send two conflicting transactions in rapid succession into the network. This is called a race attack.
2. Pre-mine one transaction into a block and spend the same tokens before releasing the block to invalidate that transaction. This is called a *Finney* attack.
3. Own 51+% of the total computing power of the network to reverse any transaction, as well as have total control of which transactions appear in blocks. This is called a 51% attack. This is impossible according to EOSIO, Loopring or Raft. If a block producer takes an unreasonable amount of runtime, is not profitable enough, the process is blacklisted.

### 3.8.10 Front running

To prevent someone from copying another node's trade solution, and have it mined before the next supposed transaction in the pool, a higher fee per transaction is charged.
The major scheme of front-running in any protocol for order-matching is order-filch: when a front-runner steals one or more orders from a pending order book settlement transaction. EOSIO and loopring both have remedies to this. In both cases keys are not part of the on-chain transaction and thus remain unknown to parties other than the ring-miner itself. Nodes dictate how they manage orders. Each node could potentially use a diffent solution for resolving the order books, inducing yet another level of randomness to promote security.

#### 3.8.11 Forged identities

Malicious users acting as themselves or forged identities could send a large amount of small orders to attack Loopring nodes. However, most of these orders will be rejected for not yielding satisfying profit when matched. Again, nodes should dictate how they manage orders.

#### 3.8.12 Insufficient Balance

Malicious users could sign and spread orders whose order value is non-zero but whose address actually has zero balance. Nodes could monitor and notice that some orders actual balance is zero, update these order states accordingly and then discard them. Nodes must spend time to update the status of an order, but can also choose to minimize the effort by, for example, blacklisting addresses and dropping related orders.

#### 3.8.13 Timing attack

Timing attacks are a class of cryptographic attacks through which a third-party observer can deduce the content of encrypted data by recording and analyzing the time taken to execute cryptographic algorithms.The randomness of the timeouts in the raft algorithm prevents this.

#### 3.8.14 Other EOSIO security provisions

1. No uses of mutex or locks for on chain parallellisation
2. All accounts must only read and write in their own private database

## 3.9 Inter blockchain communication

Transactions sent to a foreign chain will require some facilities on the foreign chain to be trustless. In the case of two EOSIO based chains, the foreign blockchain will run a smart contract which accepts block headers and incoming transactions from untrusted sources and is able to establish trust in the incoming transactions if they are provably from the originating chain. For chains with an insufficient capacity for processing the IBC proofs and establishing validity, the options degrade to trusted oracles/escrows. For instance, if you wanted to directly control bitcoin transactions with an EOSIO based smart contract you would need something like a trusted mutisig wallet that holds the bitcoin in escrow and can be Is data recoverable by any participant at any tpersuaded to sign/publish the bitcoin transaction based on IBC proofs from the originating chain.

## 3.10 Multi blockchain

Multi blockchain information can be obtained by aggregating blockchain timelines in parallel order with variance in the frequency where the state is changed, into a comprehensible data structure. This will enable a system to trigger multichain load balancers, transfer states, draw data outputs from smart contracts and trigger execution of transactions on foreign blockchains. Relative block distance, relative global state and timestamped events are recorded on a global ledger to optimize and confirm transaction before they actually happen on the native chain. This can be used for instance to determine block production coincidence between chains to determine optimal liquidity.

## 3.11 User experience

**Simulator** In an effort to provide a better and safer user experience, a VDex trading simulator will be provided with convenient scenarios.

**Templates** Easy to use templates for standard transactions will be provided.

**Comprehensive and detailed interface**

1. Shows the entire market and fluctuations
2. Shows Shows wallet: balance and previous transactions.
3. Shows detailed history with built in tax calculator.
4. Contains toggles for advanced features.

## 3.12 True decentralization

Decentralization is the process by which planning and decision-making are distributed or delegated away from a central, authoritative location or group. EOSIO is a free, open source, scalable and bug free infrastructure for decentralized applications. EOSIO software tries to ensure a fair and transparent block producer (BP) election process utilizing a democratic delegated proof of stake (DPoS) consensus.s. A pragmatic understanding of the complexity of the task at hand combined with the unpredictability of human interactions might contribute in accepting the fact that a minimal amount of centralization is required in the initial stages. After all, the ideas that emanated from Vitalik Buterin and Dan Larimer were once centralized in their head. The criteria of decreasing the *incidences of centralization* with time should be mandatory. VDex will always vote and have direct influence in this direction In any case, the exchange itself will possess several mechanisms to promote decentralization. For instance the system for electing nodes when solving order books will not use a shared central clock or (DPoS) but should for example be based on random timeouts for the determination of leaders in an election (RAFT) or using graph theory (DAG) in the case of the PARSEC protocol.

## 3.13 System recovery

By ensuring that the latest version of the ledger is always put forward, the RAFT and PARSEC protocols provides robust system for recovery in case of node failure by design. Security measures are also provided in case of IBC for trading with native blockchains. In the case of non identification of the chain, a fall back occurs to the native rules to then wait until the next block or default to a short time lock.

## 3.14 Scalable and modular architecture

To secure the potential for innovation, the principles, concepts and paradigms proposed by components of the system must favour decoupling of technologies. Since creating and maintaining distibuted and decentralized systems is very complex we must use different strategies:

### 3.14.1 Problem decomposition

Problem solving strategy of breaking a problem up into a set of subproblems, solving each of the subproblems, and then composing a solution to the original problem from the solutions to the subproblems.

### 3.14.2 Minimize state space

Dynamic programming and templating are hard because of complexity and debugging challenges. Nesting conditions can also seem unimportant for normal program execution. Special attention to these patterns and details in the initial design will allow to maximize efficiency while allowing for easy replacement or addition of components. In the context where CPU, bandwidth and RAM are monetized, this is even more important.

### 3.14.3 Determine and minimize state replication

State machine replication is a general method for implementing a fault-tolerant service by replicating servers and coordinating client interactions with server replicas. Replication has a cost and PARSEC has an elegant solution for consensus without replication.

# 4 Risk

The sheer amount of transactions VDex hopes to one day process is hard to visualize but in the context of the growing interest for decentralized exchanges, more transactions equate with more risk. Managing the risks of handling currency also proved initially to be a challenge to previous centralized providers.It was a long and arduous process.
- The evolution of the efficiency of decentralized exchanges should not be expected to be any different but unlike centralized exchanges, dexes have the

support of the open source developer community which quickly contribute forward solutions to resolving problems and enhancing the product.
- The Volentix DAO will ensure VTX counsel nomination within assembly. This structure poses a risk by its novel and sparsely tested approach but ensures solidity by analysis of its theoretical precepts.

## 5 Timeline

Table 2: VDex R&D Timeline

| **component** | **version** | **date** |
|---|---|---|
| Wallet, account and token creation prototype | n/a | 15/07/2018 |
| White paper | v.0.1 | 08/06/2018 |
| White paper | v.0.5 | 15/06/2018 |
| White paper | v.0.8 | 15/07/2018 |

## 6 Conclusion

Constructs, concepts and protocols that stand out by their simplicity while retaining their effectiveness will help implement a modular design which will enhance the system with the capacity to easily add or replace components with the prospects of carefully advancing functionality while implementing micro services. Although certain assumptions made in this paper still remain to be verified, a very distinctive direction for VDex architecture could be distilled as a highly flexible and modular MVP capable of adaptation and reaction to the changing technological ecosystem. For this provisions of the EOSIO operating system and the Loopring and Bancor protocols have been retained.

## References

[1] Aelf, *A multi-chain parallel computing blockchain framework*, (2018).

[2] ARK, *A platform for consumer adoption*, (2018).

[3] BlockColliderTeam, *Block collider white paper*, (2018).

[4] V. Buterin, *Ethereum: a next generation smart contract and decentralized application platform*, (2013).

[5] S. Cormier, *A machine based societal model for curbing citizen cynicism*, (2017).

[6] J. O. Diego Ongaro, *In search of an understandable consensus algorithm*, (2018).

[7] M. Duncan, Quale, *Halo platform*, (2018).

[8] EOS.IO, *Eos.io technical white paper v2*, (2018).

[9] G. B. Eyal Hertzog, Guy Benartzi, *Bancor protocol: Continuous liquidity for cryptographic tokens through their smart contracts*, (2018).

[10] S. D. K. M. T. S. H. Garcia-Molina, *The eigentrust algorithm for reputation management in p2p networks*, (2018).

[11] M. R. Garrick Hileman, *Global cryptocurrency benchmarking study*, (2017).

[12] Komodo, *An advanced blockchain technology, focused on freedom*, (2018).

[13] Q. Liquid, *Providing liquidity to the non-liquid crypto economy*, (2018).

[14] A. K. M. M.-S. R. Malavolta, Pedro Moreno-Sanchez, *Concurrency and privacy with payment-channel networks*, (2017).

[15] SingularityNET, *A decentralized, open market and inter-network for ais*, (2018).

[16] M. M. Timo Hanke and D. Williams, *Dfinity technology overview series consensus system*, (2018).

[17] W. F. Wang, Zhou, *Loopring: A decentralized token exchange protocol*, (2018).

[18] A. B. Will Warren, *0x: An open protocol for decentralized exchange on the ethereum blockchain*, (2017).

[19] G. Wood, *Ethereum: A secure decentralised generalised transaction ledger.ethereum project yellow paper*, (2014).

[4] [19] [11] [17] [18] [14] [9] [10] [12] [2] [7] [13] [8] [15] [16] [6] [1] [3] [5]