Contents

VDex White Paper v0.6

The Volentix Labs Team
info@volentixlabs.com

July 7, 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

EOS.IO contracts are publicly accessible and contain easy to use options for security, anonymity, speed of payment, liquidity and profit margin. Volentix 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 will provide easy to use settings for speed, cost, anonymity and security within the scope of scalability. Power users will feel enabled with the freedom to choose and thrive. Equally, new users will feel welcomed and secure. The sustainability of the product will reside on an architecture with 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 EOS.IO decentralized operating system. To provide a flexible modular product, developers 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 EOS.IO's ***cleos*** command line through functional python testing. This software resides at: https://gitaggregator.com/Volentix/ezeos

3 VDex

The Volentix system consists of five pillars of which VDex is the fulcrum:

3.0.1 Venue is an app for tracking signature campaigns.

1. Credibility metric
2. Live rewards platform
3. Real-time information
4. Data sources
5. User rankings
6. Payout information

3.0.2 Verto wallet

Verto is a desktop application acting as a multi-currency wallet for VDex. Traders use Verto to understand their holdings and positions, and understand the crypto assets they can trade into. It is the central component for all Volentix platforms and is necessary to grant the user full custody of their private keys for each token they own. Verto employs a system of smart contracts to maintain the state between two trading clients.

3.0.3 Vespucci analytical engine

Vespucci is an application that provides information and analytics about tradeable digital assets. It is also a tool to graph and compare tradeable digital assets. It provides a well rounded picture of each digital asset of interest and a live rating representing an amalgam of complex indicators .

1. An intuitive analytical agent.
2. A rating system for crypto.
3. Easy to understand overview of crypto.
4. A dashboard for crypto news.
5. A sentiment gage for cryptocurrencies.

Vespucci data sources consist pf data presently scattered among blockchain explorer sites and chat rooms, historic trading records, forum sentiment analysis, trading trends, developer activity and plan analysis. digital asset distribution and governance information, terms and conditions and current and historic address balances.

3.0.4 VDex

VDex will provide high-quality exchange services directly from the user's wallet. Both public and private keys are maintained locally by the wallet and liquidity is provided by the network itself, by way of users signalling their desire to make a particular trade. The swapping of crypto assets does not involve the temporary custody of the tokens by a central operator and therefore no market manipulation can occur. There are no risks of coins getting hacked and no risk regarding laws governing the exchange where the account is held. VDex allows users to trade crypto assets with other users without losing custody of the tokens they hold. Furthermore, VDex allows digital tokens to remain on the originator's wallet until all required transactions are completed. VDex supports trading in multiple cryptocurrencies, interoperable with any blockchain.

The Volentix Ecosystem

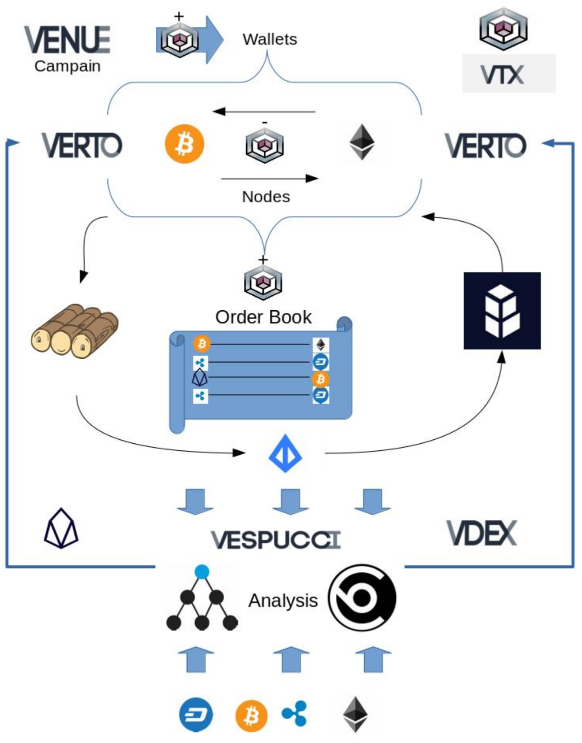

Figure 1:

3.1 Components

3.1.1 Operating system

EOS.IO 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 EOS.IO 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: [10]

1. **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

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 allows faster transaction times and increases bandwidth. EOS.IO blocks are produced every 500 ms.

4. **Web Assembly(WASM)**
   Web assembly has the highest performance for web application while also providing sandboxing to secure applications. This will provide VDex with network access restrictions, filesystem namespace restrictions, enforced rule-based execution and better control over what processes are spawned.

5. **C/C++ contracts**
   At the time of this writing, C++ has best tooling for and execution speeds for WASM. C++ is a much more mature language than for instance Solidity used for Ethereum contracts. It also has better debugging support as well as libraries that have been tested over the years and provide reliable functionality. The EOS codebase has also very heavy usage of templates. For example, C++ allows us to use templates and operator overloading to define a runtime cost-free validation of units. Managing memory is much easier building smart contracts because the program reinitializes to clean

state at the start of every message. Furthermore, there is rarely a need to implement dynamic memory allocation. The WebAssembly framework will automatically reject any transaction addressing memory wrong. Since EOS.IO contracts use C++14 one can resort to smart pointers if dynamic memory allocation is needed.

6. **Schema defined messages and database**
   Service contracts are standardized to guarantee a baseline measure of interoperability associated with the harmonization of data models. 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. [9]

3.1.2 Inter Contract Communication

Data is shared between contracts via an oracle which creates a transaction embedding the data in the chain. ”An oracle, in the context of blockchains and smart contracts, is an agent that finds and verifies real-world occurrences and submits this information to a blockchain to be used by smart contracts.” [3] Every node will have an identical copy of this data, so it can be safely used in a smart contract computation. Oracles pushes the data onto the blockchain rather than the smart contract pulling the information. Most reading the of data is done via polling **nodeos** (the blockchain instance) to monitors the blockchain's state and performs certain actions in response.

3.1.3 Side Chains

In EOS.IO, issuing tokens is a process a which creates sidechain.Sidechains are emerging mechanisms that allow tokens and other digital assets from one blockchain to be securely used in a separate blockchain and then be moved back to the original blockchain if needed. There can be multiple side chains where different tasks are distributed accordingly for improving the efficiency of processing. For inter-blockchain communication, the EOS.IO 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. [10]

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” [11]. 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 EOS.IO contracts acting as nodes (relays)[24]
2. Implementation of the Bancor protocol used to bring stability to the tokenwhile promoting liquidity.[11]

3.1.5 Hashed timelock contracts

A Hashed Timelock Contract (HTLC)[15] 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 Network Topology

3.2.1 Nodes

Nodes are the endpoints with which users will propagate their orders to VDex. If a user chooses, the node can become the one who can, through an election process, settle the order book for that iteration. Hence, any wallet with sufficient funds and posessing a good track record with VDex can be used as node. Nodes are the endpoints of the VDex network. Their function is to:

1. Act as a portal to VDex trough the Verto wallet.
2. Merge order book information with others.
3. Settle order book.
4. Order cancellation.
5. Assign timeouts for the **Raft** Protocol. (seen later in the document)

3.2.2 Aggregators

The VDex aggregators are dedicated Volentix servers for simulator and security purposes. One of their functions is to pull logs and order book data from nodes into sparse distributed representations for hierarchical temporal memory as intrusions [16] analysis for detecting anomalies in the system such. The aggregators will also be host to other components such as a metachain ledger[4]and blockchain scrapers from which Vespucci will elaborate scenarios to build user advice.

3.2.3 Latency

EOS.IO has low latency block confirmation (0.5 seconds).[10] 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. For example a bitcoin block takes 9 minutes to mine at time of this writing. 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.

3.3 OrderBook

An order book is the list of orders that VDex will use to record the interest of buyers and sellers. A matching engine uses the book to determine which orders can be fulfilled. The Loopring protocol allows for customizing the order book data structure. [24] This will be to the advantage of VDex to offer flexibility in the choice of data structure to be used according to the corresponding currencies.

3.3.1 Data structures

1. Loopring FIFO First in first out circular buffer, as suggested by the protocol. Nodes can design their order books in any number of ways to display and match a user's order. Follows an OTC model, where limit orders are positioned based on price alone. [24]

2. EOS.IO persitence API The order book will take advantage of the powerful multi index container shared among nodes through the same EOS.IO account.

3.3.2 Online order book

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

3.3.3 Offline order book

Residing on the aggregator, offline order books will serve for simulator and security purposes.

3.3.4 Decentalization process of order book settlement

To ensure decentralization, nodes will take turns to settle the Order Book. The settling node must be designated and all order book entries from all nodes must be available to the settling node.The RAFT[7] and PARSEC[19] protocols offer elegant and simple solutions. The concepts of RAFT are simple to implement and have been around since the the time of PAXOS[16] while PARSEC is fairly recent but more efficient, using directed acyclic graphs, eliminating the need for copying logs.

Raft

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.

3.4 Order Settlement

In the case of a FIFO container for the order book, settling occurs as such:

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

3.5 VTX

VTX is both the cryptocurrency used on the vDex exchange. VTX can be used in the five pillars of the Volentix system.

1. To pay transaction fees on the vDex.
2. To invest.
3. To vote on proposals submitted to the network, using the voting rights allocated to VTX holders.
4. To submit proposals to the network.
5. To stake support for project building, liquidity, or proposal reviews.
6. As a fee redistribution token on the VDex exchange.
7. To incite users to partake in order book settlement.
8. To reward participants in the consensus process and in Venue campains.

3.5.1 Token creation

The **eosio.token** contract from the EOS.IO framework will be used to issue 2.1 Billion Tokens EOS.IO compliant tokens with a supply of 1.3 Billion.

3.5.2 Initial Token offering

Distribution

1. The founders The initial promoters of the project. Conceptualization, plan and initialization.
2. Prior work core team Consists software engineers, blockchain analysts, legal experts and business development specialists. In addition to the core team, Volentix has a Board of Advisors consisting of top experts from a variety of fields relevant to the project.
3. Decentralized treasury Publicly controlled funds for community projects and salaries for ongoing development. Small percentage or every transaction goes back into the treasury.
4. Ongoing core development Incentives for core team.

Justification of token crowdsale vote Pre-sale VTX holders, founders and the core team will be given the opportunity to participate in an *augmented vote*[6] to know if the Volentix community wants a crowdsale or not. Augmented votes will require voters to take an easy test which proves they acknowledge facts concerning crowdsales. Participants to the elaboration of this test grid will compile facts linked to publications regarding this issue and will earn VTX in the process. The resulting vote will therefore be 'augmented' by ensuring all voters vote with the same established knowledge regarding this issue.

Table 1: Token distribution

Percentage of issued tokens	**price**	**Timelock***
35% Future Decentralized Treasury		
5% Initial funding	0.000016-0.000020 BTC	
28% Distribution	0.000021 BTC+	
12% Founders		yes
10% (130M VTX) Prior Work, Team and Advisors		yes
10% Ongoing Core Development enticement		

*Logarithmic decay token distribution over 4 years.

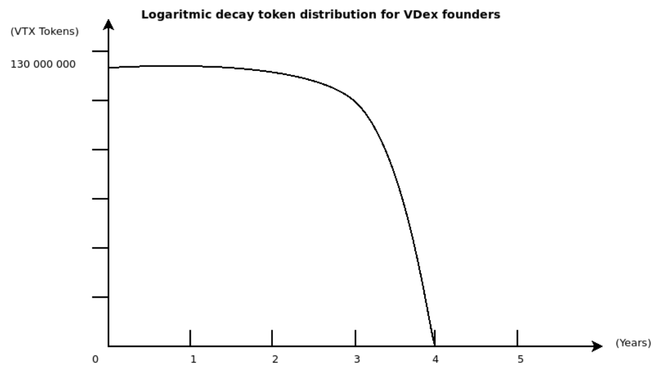

Figure 2:

Multi signature account The raised funds will be deposited to a multi sig account owned by the founders that will only be only be accessed with the consent of at least two of three signees.

3.5.3 Fee model

Transaction fees

1. Fees will 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 fee
A Hashed Timelock Contract (HTLC)[15] is a type of smart contract enabling the implementation of time-bound transactions.

1. User can lock in funds for 24hrs and have a free transaction

2. User can lock in funds for 24hrs+ and generate VTX

Order book settlement fees A wallet can become a node and earn VTX through order book settlement. 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.'[24] A pre-determined amount of VTX is required and certain conditions will need to be met in order to participate in the consensus process.

1. If the margin split is 0, ring-miners will choose the flat VTX fee and are still incentivized.
2. If the VTX fee is 0, the income is based on a general linear model.
3. When the margin split income is greater than 2x(VTX fee), ring-miners choose the margin split and pay VTX to the user.

EOS.IO The following considerations will be applicable for deploying the exchange on the EOS.IO platform

1. Deploying contract has cost but is free to use.
2. Developers have to stake EOS.IO 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 will increase with demand over a longer period of time, around 5 years.

3.6 Inter blockchain communication

EOS.IO 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 EOS.IO 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.

3.7 Security model

3.7.1 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

3.7.2 General actions to be taken

1. Filter incoming data
   (a) Data from blockchains and order book resolution.
   (b) Data received by nodes
   (c) Data received by wallet
   (d) Data received by aggregators
2. Filter outgoing data
   (a) Cancel order
   (b) Commit

3.7.3 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

3.7.4 Malware detection by auditing processes

The system will provide insights on rogue processes during the transaction period with AI analysis residing on the aggregators.

3.7.5 Random diversification

By using the RAFT protocol in the election process, a certain level of randomization is aquired with varying length of timeouts.

3.7.6 Multiple factor identification

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

3.7.7 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.

3.7.8 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.

3.7.9 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 EOS.IO, Loopring or Raft. If a block producer takes an unreasonable amount of runtime or is not profitable enough, the process is blacklisted.[24]

3.7.10 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. EOS.IO 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 order book settling node. Nodes dictate how they manage orders. Each node will use a different solution for resolving the order books, inducing another level of randomness to promote security.

3.7.11 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.

3.7.12 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.

3.7.13 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.

3.7.14 Other EOS.IO security attributes

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

3.8 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 EOS.IO 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, wanting to directly control bitcoin transactions with an EOS.IO based smart contract 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.[10]

3.9 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.[4]

3.10 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 also be provided.

Comprehensive and detailed interface

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

3.11 True decentralization

Decentralization is the process by which planning and decision-making are distributed or delegated away from a central, authoritative location or group. EOS.IO is a free, open source, scalable and bug free infrastructure for decentralized applications. EOS.IO software tries to ensure a fair and transparent block producer (BP) election process utilizing a democratic delegated proof of stake (DPoS) consensus. 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. The criteria of decreasing the *incidences of centralization* with time should be mandatory. VDex will always vote and have direct influence in this direction and can someday hope to make a difference in numbers. 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.

3.12 System recovery

By ensuring that the latest version of the ledger is always put forward, the RAFT and PARSEC protocols provide 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.

3.13 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:

3.13.1 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.

3.13.2 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.

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

6 Conclusion

Constructs, concepts and protocols that stand out by their simplicity while retaining their effectiveness will help implement a modular design which will

Table 2: VDex R&D Timeline

component	**version**	**date**
ICO		15/07/2018
Wallet, account and token creation prototype		10/07/2018
White paper	v.1.0	15/07/2018
White paper	v.0.5	15/06/2018
White paper	v.0.1	08/06/2018

enhance the system with the capacity to easily add or replace components with the prospects of carefully advancing functionality as micro services are created. Although certain assumptions made in this paper still remain to be verified, a very distinctive direction for VDex architecture will be distilled as a highly flexible and modular MVP capable of adaptation and reaction to the changing technological ecosystem. For this, provisions of the EOS.IO operating system, Loopring, Bancor, RAFT and PARSEC protocols have been retained.

Acknowledgements

Bibliography

References

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

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

[3] blockchainhub.net, *blockchain-oracles*, (2017).

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

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

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

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

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

[9] T. Earl, *Soa principles of service design*, (2016).

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

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

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

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

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

[15] K. Kurokawa, *Atomic cross chain transfer, an overview*, (2015).

[16] L. Lamport, *The part time parliament*, (1998).

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

[18] S. R. M.P.M-S, Aniket Kate Matteo Maffei, *Concurrency and privacy with payment-channel networks*, (2017).

[19] F. H. Q. M. S. S. Pierre Chevalier, Bart lomiej KamiÂťnski, *Protocol for asynchronous, reliable, secure and efficient consensus (parsec)*, (2018).

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

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

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

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

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

[9] [3] [10] [5] [23] [13] [24] [22] [18] [11] [12] [14] [2] [8] [17] [20] [21] [7] [1] [4] [6]