Contents

VDex White Paper v0.3

Cormier, Sylvain
sylvain@volentixlabs.com

Hallak, Nemr
nemr@volentixlabs.com

Lauzon, Shawn
shawn@.volentixlabs.com

June 13, 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 crypto-currency exchanges. Because they handle such large amounts of capital, exchanges are powerful entities, in a global sense. Can the acceptation by traders of decentralized exchanges act as a driving force in the acceptance and proliferation of decentralized

ideology as a whole? With this organic 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 to ensure 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

This document will listing some advantages and shortcomings of present day distributed exchanges. Subsequently, the VDex exchange architecture and its components will be exposed.

2 Distributed exchanges

A market economy is a decentralized economic system that acts through the distributed, local interactions in the market of individual investments. The final tendency of a market economy results from these local interactions and is not the product of one body's instructions or regulations.

2.1 Mass adoption

More than 99 percent of cryptocurrency still goes through centralized exchanges.

2.2 Privacy, policy and security

Some statistics

1. 73% of exchanges take custody of user funds
2. 23% let users control keys
3. 53% of small custodial exchanges have a written policy outlining what happens to customer funds in the event of a security breach resulting in the loss of customer funds, compared to 78% of large custodial exchanges
4. 33% of custodial exchanges have a proof-of-reserve component as part of their formal security audit

Use of public ledger The use of a public ledger can reveal transactions. This can potentially represent a strategic disadvantage for advanced traders.

Types of security issues

1. Denial-of-service attack (DoS)
2. Sybil attack
3. Front running
4. Actual double-mining
5. Wealth Attacks
6. Reputation Gaming
7. Technical Attacks
8. Nothing at stake
9. Email Phishing
10. Social Media Giveaway Scam

2.3 Lack of true decentralization

1. Distributed:
   not all the processing of the transactions is done in the same place.
2. Decentralized: not one single entity has control over all the processing.

Some services can remain off-chain. (ie. onchain orderbooks are expensive and not efficient enough) Architectural (de)centralization how many physical computers is a system made up of? How many of those computers can it tolerate breaking down at any single time? Political (de)centralization how many individuals or organizations ultimately control the computers that the system is made up of? Logical (de)centralization does the interface and data structures that the system presents and maintains look more like a single monolithic object, or an amorphous swarm? One simple heuristic is: if you cut the system in half, including both providers and users, will both halves continue to fully operate as independent units?

2.4 Consensus

A consensus algorithm is a process in computer science used to achieve agreement on a single data value among distributed processes or systems. Consensus algorithms are designed to achieve reliability in a network involving multiple unreliable nodes. As a result, consensus algorithms must be fault-tolerant.

2.4.1 Essential ctriteria

1. Agreement
2. Validity
3. Termination

2.4.2 Goals:CAP

1. Consistency
2. Availability
3. Partition tolerance

2.4.3 Problems to solve

1. Fail-stop(crash)
2. Fail-recover
3. Network partition
4. Byzantine failure

An extensive and reliable list of projects along with their consensus algorithm can be found here: https://github.com/distribuyed/index

2.5 Cost

There is a potential high costs per trade. On ethereum, deploying and calling contract both cost fee.

2.6 Immunity

Decentralized cryptocurrency exchanges are much harder to regulate or even shut down because they are not restricted to one physical location.

2.7 Persistence

Data committed to the blockchain is permanent.

IPFS

1. Retreival based on content rather than location.

2. Name derived from the hash of content.

3. Network layer for finding files(P2P).

4. Users must provide their own servers and related infrastructure.

5. Links to IPFS committed to the blockchain.

2.8 Interoperability

- Need for cross-chain exchanges
- More blockchains/dapps interoperability

2.9 Scalability

- Possible blockchain bloat with ethereum and bitcoin network.

2.10 Speed

- Blockchain transactions just take time to be validated.
- It is always as fast as the slowest of the two blockchains being traded. - There are higher delays due to peaks of demand.

2.11 Anonymity

- Most decentralized exchanges do require the creation of an account before beginning trading.
- Most decentralized exchanges allow anyone to create an account under any name they choose with very little or no approval process.

2.12 A lack of liquidity

Because of a fragmented market, liquidity is divided in just a few market places and large orders struggle to be matched.

2.13 Lack of transparency

- Actual costs and processes of trading are opaque and involve high trading costs, often higher than announced fees.
- Graphical interfaces are inaccurate and misleading to the point of suspicion of wrong doing.
Front-running orders is possible and illegal.

2.14 User experience

- Since centralized exchanges tend to be better funded, they can deliver a better user experience than decentralized exchanges.
- Certain governments could hold responsible an exchange for a poor customer experience.
- Better customer support and sense of professionalism is needed.

2.15 A lack of educated users / apathy

Markets are full of speculators unaware of how to safely deal with cryptocurrencies.

2.16 Conclusion

- Getting liquidity through a large adoption by the ecosystem is a long process.
- Traders do not join because they creatures of habit and are not easily moved from a platform that already matches their orders.

- There are potential performance issues, market manipulation, hardware failures, latency problems, and many other inherent problems when it comes to dealing with large volume.

The third blockchain generation consists of many organizations developing new solutions to help build decentralized applications. More recently solutions are trying to promote inter-blockchain communication and compatibility. They are sprouting new data structures built from the input of many blockchains, generating new meta chain protocols. This seems to be the beginning of the fourth generation blockchain but because of much hype, many solutions out there are imposing protocols with fully integrated solutions and no collaboration incentive. To distill the best concepts out of this ecosystem, a more collaborative integration effort is needed to gather all available useful technology and cull redundant features.

3 VDex

3.1 Introduction

VDex should supply the user with many flexible configuration options. The users should be able to choose custom settings for speed, cost, anonymity and security. In tandem, optimal performance required by a power user should not be compromised. Most of all, scalability should be possible to provide for the increasing demand for decentralized exchanges.

Certain current available framework options allow developers to focus on their product's functionality and usability instead of managing and optimizing operating procedures. To provide a flexible modular product, developers needed an environment that served its purpose without impeding design with unnecessary complexity.

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.

3.2 General architecture and components

3.2.1 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. This OS paradigm allows to easily implement complex distibuted systems with ease of development. Components and protocols already built into the platform and just a subset can be used to satisfy 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 have to implement the paradigms and patterns suggested by most protocols for the creation of decentralized exchanges through its contracts and the use of the tools provided by the system.

3.2.2 EOSIO Components

1. 1. Schema defined messages and database
   The Standardized Service Contract design principle advocates that the service contracts be based on standardized data models. and the analysis of 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
   - Human readability of JSON

- Efficiency of binary

3. Separate authentification from application

4. Use of web Assembly Provides sandboxing
   - Secure applications
   - Highest performance for web application

5. C/C++ contracts

Separation of message and (n...) transaction layers The separation of the message and transaction layers will augment the speed of transaction and keep unnecessary data off the blockchain.

Polling micro service The exchange is running a polling micro-service which will fetch the "next unprocessed deposit". By processing the history the system is informed of when the transaction was confirmed. 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.

Inter contract communication Smart contracts can only read data that is part of the transaction or stored in blockchain state. To pass external data into a contract it will need to be sent via an oracle. Reading the data is done via polling **nodeos** (the blockchain instance) When declaring a multi_index to use as a table contracta you would typically create a struct and typedef similar to this:

Listing 1: C++ code using listings

```
1     // @abi table myobjects
2     struct myobject {
3     name sender;
4     // ...
5
6     EOSLIB_SERIALIZE(myobject, (sender)(...))
7     }
8
9     typedef multi_index<N(myobjects), myobject> myobjects_t;
10
11    }
```

and then create an instance to use within the **contracta** code:

Listing 2: C++ code using listings

```
1     auto db = myobjects_t(_self, _self);
```

The first self refers to the code that defined the table and the second refers to the scope for the table in this case both are referring to **contracta**. In order for **contractb** to have access to that data it would need to also define the *struct* and *typedef* as above but when creating an instance for access the self would be replaced by N(contracta). The *struct* would need to have the same property names, types and order of definition and also be serialized in the same order. The **ABI** name and index type would also need to match. Then the data would be accessible for reading as if it's in the same contract.

Sandboxing 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.

1. Network-access restrictions.
2. Restricted filesystem namespace.
3. Rule-based execution.
4. Control over what processes are spawned
5. Read and write control
6. Garbage collection

lock-in period A variable lock in period will also be offered where users can choose to lock in their funds for a slightly greater time than the slowest blockchain in exchange for a discount on the transaction.

Liquidity

1. Implementation of the Loopring protocol with the use of EOS.IO contracts acting as relays
2. Bancor protocol formula

generalized signature verification - Inter-blockchain communication. - Inter contract communication

3.3 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 EOS.IO 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

3.4 Network Topology

3.4.1 Nodes Relays

All wallets that have sufficient funds can be used as relays
Functions

1. Merge orderbook information with others
2. Mine ring

3.4.2 Hubs

Function

1. Order cancelling for single and multi signatures
2. Scheduler (Timeouts)
3. Asset tokenization services
4. Order book browser populating

3.4.3 Zones

Zones are defined geographically with hubs.

3.4.4 Latency

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

Speedy confirmation 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.

3.5 OrderBook

3.5.1 Data structure

1. Loopring:FIFO
2. EOSIO:multi-index

3.5.2 Online order book

OrderBooks reside in a persistent container on each relay belonging to the same account as the other relays.

3.5.3 Offline order book

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.

3.5.4 Order(Loopring)

Listing 3: 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
```

```
11 marginSplitPercentage; // [1-100]
12 bool buyNoMoreThanAmountB;
13 uint256
14 walletId;
15 address authAddr;
16 // v, r, s are parts of the signature
17 uint8 v;
18 bytes32 r;
19 bytes32 s;
20 // Dual-Authoring private-key,
21 of our own order book follows an OTC model, where limit
22 // not used for calculating order's hash,
23 orders are positioned based on price alone. Timestamps of
24 string authKey;
25 uint256 nonce;
26 }
```

3.5.5 Order(EOSIO)

Listing 4: C++ code using listings

```
1
2  struct limit_order {
3      uint64_t      id;
4      uint128_t     price;
5      uint64_t      expiration;
6      account_name owner;
7
8      auto primary_key() const { return id; }
9      uint64_t get_expiration() const { return expiration; }
10     uint128_t get_price() const { return price; }
11
12     EOSLIB_SERIALIZE( limit_order, ( id )( price )( expiration )( owner ) )
13 };
```

3.5.6 Consensus model

VTX is required to participate in the consensus process and earn both block rewards and transaction fees. Time is anj essential comonent to control distributed systems. No one clock can reside on one system to ensure a decentralized system. Raft is a protocol using RPCs to ensure consensus among a node cluster.

Raft Log propagation punctuated by leader election

1. Leader election

   - Randomized timeouts are stamped on relays
   - 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 relays 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.

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 Pre-sale VTX holders, founders and contributors participate in a proof of non-repudiation process(1 time mining) for token allocations.

3.6 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

3.7 VTX

3.7.1 Token creation

The **eosio.token** contract from the EOS.IO framework can be used to issue EOS.IO compliant tokens.
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.

3.7.2 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 encentive.

3.7.3 Token distribution

Table 1: Token distribution

amount	**details**
2.1 Billion Tokens	Supply 1.3B
5% Pre-public crowdsale	$0.14-0.18
28% Public crowdsale	$0.19+
10% Founders	Time Locked
10% (130M VTX) Prior Work, Team and Advisors	Time Locked
35% Future Decentralized Treasury	
12% Core Development)	

3.7.4 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 relay 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.'

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.

EOSIO

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.
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. EOS.IO can process signature verification in parallel.
7. No uses of mutex or locks for on chain parallelisation
8. All accounts must only read and write in their own private database
9. Accounts processes messages sequentially and parrallellism is done at the account level.
10. Schedule is deterministic

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.

3.8 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.

3.9 Security model

3.9.1 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 timeline. The security concerns with the VDex environment can generally be categorized as variants of the following scenarios:

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

3.9.2 General actions

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

3.9.3 Contract security

1. Upgrade broken contracts
2. Pause the functionality of a contract
3. Delay an action of a contract

3.9.4 Malware detection by auditing processes

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

3.9.5 Random diversification

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

3.9.6 Multiple factor identification

Provided by EOSIO

3.9.7 Logs

Inspection of logs as control.

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

3.9.8 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.

3.9.9 Double spend

3.9.10 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. When a front-runner steals the entire order book from a pending transaction. When a submit transaction is not confirmed and is still in the pending transaction pool, anyone can easily spot such a transaction and replace **minerAddress** with their own address (the **filcherAddress**), then they can re- sign the payload with **filcherAddress** to replace the order- ring's signature. The filcher can set a higher price and submit a new transaction hoping block-miners will pick his new transaction into the next block instead of the original **submitRing** transaction. Dual Authoring, involves the mechanism of setting up two levels of authoriza- tion for orders - one for settlement, and one for ring-mining

1. For each order, the wallet software will generate a random public-key/private-key pair, and put the key pair into the order's JSON snippet. (An alternative is to use the address derived from the public-key instead of the public-key itself to reduce byte size. We use **authAddr** to represent such an address, and **authKey** to represent **authAddr**'s matching private-key).
2. Compute the order's hash with all fields in the order except r, v, s, and **authKey**), and sign the hash using the owner's private-key (not **authKey**).

3. The wallet will send the order together with the **authKey** to relays for ring-

4. When an order-ring is identified, the ring-miner will use each order's **authKey** to sign the ring's hash, **minerAddress**, and all the mining parameters. If an order-ring contains n orders, there will be n signatures by the n **authKeys**. We call these signatures the **authSignatures**. The ring-miner may also need to sign the ring's hash together with all mining parame- ters using **minerAddress**'s private-key.

   Relays dictate how they manage orders. Notice that **authKeys** are NOT part of the on-chain transaction and thus remain unknown to parties other than the ring-miner itself.

5. The Loopring Protocol will now verify each against the corresponding **authAddr** and reject the order-ring if any is missing or invalid. The result is that now: The order's signature (by the private-key of the owner address) guarantees the order cannot be modified, including the **authAddr**. The ring-miner's signature (by the private-key of the **minerAddress**), if supplied, guarantees nobody can use his identity to mine an order-ring. The **authSignatures** guarantees the entire order-ring cannot be modified, including **minerAddress**, and no orders can be stolen. Dual Authoring prevents ring-filch and order-filch while still ensuring the settlement of order-rings can be done in one single transaction. In addition, Dual Authoring opens doors for relays to share orders in two ways: non- matchable sharing and matchable sharing. By default, and then discard them. Nodes must spend time to update Supports of limit-price orders, meaning that orders' timestamps are ignored. This implies that front-running a trade has no impact on the actual price of that trade, but does impact whether it gets executed or not.

3.9.11 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. Relays dictate how they manage orders.

3.9.12 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.

3.9.13 Timing attack

The randomness of the raft algorithm prevents this.

3.9.14 Eclipse attack

3.9.15 Knapsack

3.9.16 Trojan malware

3.9.17 Sybil attack

3.9.18 Eclipse attack

3.9.19 Timing attack

3.9.20 Knapsack

3.9.21 Trojan malware

3.10 IBC

Transactions sent to a foreign chain will require some facilities on said 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.

3.11 Multi blockchain

Timelines in parallel order with variance in the frequency where state is changed. Multichain load balancers transfer state, draw data outputs from smart contracts and trigger execution of transactions on other blockchains. Relative block distance, relative global state timestamped events. Global ledger.

3.12 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.

3.13 True decentralization

The system does not use a shared central clock.

3.14 System recovery

The raft protocol provides for system recovery

3.15 Potential for innovation

3.16 Scalable and modular design

Since creating and maintaining distibuted and decentralized systems is very complex we must use different strategies:

3.16.1 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.

3.16.2 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. But 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.

3.16.3 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.

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	15/03/2018
White paper	v.0.1	08/06/2018
White paper	v.0.5	15/06/2018
Atomic swap prototype	n/a	07/15/2018
Liquidity pool prototype	n/a	09/01/2018

6 Conclusion

Although certain assumptions made in this paper still remain to be verified, a very distinctive direction for VDex architecture can be distilled in its choice for the simplest solutions. A highly flexible and modular MVP based on concepts and protocols that stand out by their simplicity will allow for the implementation of a modularity that will benefit the system with the capacity of easily add or replace components in the prospect of advancing functionality.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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