

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 10, 2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Connor Callahan
Bradley Lostak

ENTITLED

**VelaChain: A Decentralized Exchange Built for Cross-Chain
Communication**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING


Yuhong Liu (Jun 11, 2022 08:08 PDT)

Thesis Advisor


N. Ling (Jun 11, 2022 08:11 PDT)

Department Chair

VelaChain: A Decentralized Exchange Built for Cross-Chain Communication

by

Connor Callahan
Bradley Lostak

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 10, 2022

VelaChain: A Decentralized Exchange Built for Cross-Chain Communication

Connor Callahan
Bradley Lostak

Department of Computer Science and Engineering
Santa Clara University
June 10, 2022

ABSTRACT

The blockchain industry is one of the fastest growing industries in the world right now. Billions of dollars of venture capital money is being put to work to innovate in this new space. One of the problems that has plagued the blockchain space since its inception is the lack of interoperability between blockchains. A single blockchain is great at dictating its own state but they rarely contain functionality to communicate with other blockchains. Furthermore, users on these blockchains have limited ways to swap cryptocurrencies from one blockchain to another. For this project, we built a cryptocurrency decentralized exchange built on top of the Cosmos software development kit to include cross-chain communication. We successfully implemented the basic functionality needed for a decentralized exchange to serve both sides of the market: users and liquidity providers.

Acknowledgments

We would like to thank our advisor, Dr. Yuhong Liu, for her ever present support and encouragement.

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Existing Solutions	1
1.3	Our Solution	2
2	Background	4
2.1	Blockchain	4
2.2	Cross-Chain Communication	5
2.3	Polkadot	5
2.4	Near	6
2.5	Cosmos	6
3	Requirements	7
3.1	Functional	7
3.2	Nonfunctional	8
4	Technologies Used	9
5	Vela Architecture	10
5.1	Architecture Overview	11
5.2	Persistent Storage	12
5.3	Pools	12
5.4	Liquidity Providers	13
5.5	Messages	13
5.6	Transactions	13
5.7	Queries	14
6	Functionality and Analysis of Vela	15
6.1	Core Functionality	15
6.1.1	Create Pool	15
6.1.2	Join Pool	16
6.1.3	Exit Pool	16
6.1.4	Swap	17
6.1.5	Query Pools	17
6.1.6	Query Liquidity Providers	17
6.2	Elective Functionality	17
6.2.1	Add Liquidity	17
6.2.2	Remove Liquidity	18
6.2.3	Send Shares	18
6.3	Swap Fees	19

6.4	Exit Fees	19
6.5	Slippage	19
6.6	Limitations	20
6.7	Interoperability	20
7	Deployment Impacts	22
7.1	Ethical Impacts	22
7.1.1	Ethical Justification	22
7.1.2	What it means to be a good engineer?	23
7.1.3	Ethical Pitfalls	24
7.2	Societal Impacts	24
7.3	Civic Engagement	24
8	Conclusion	26
8.1	Obstacles Encountered	26
8.2	Summary and Next Steps	26

List of Figures

5.1	Relevant Chain Architecture	11
5.2	Available Transaction Commands for the Vela CLI	14
5.3	Available Querying Commands for the Vela CLI	14
6.1	Command Line Create Pool Transaction	16
6.2	Command Line Join Pool Transaction	16
6.3	Command Line Exit Pool Transaction	16
6.4	Command Line Swap Transaction	17
6.5	Command Line Add Liquidity Transaction	18
6.6	Command Line Remove Liquidity Transaction	18
6.7	Command Line Send Shares Transaction	18

Chapter 1

Introduction

1.1 Motivation

Throughout the last decade the popularity of blockchains has increased dramatically. What started as one blockchain called Bitcoin has quickly become a cacophony of blockchain networks, all of which claim to be the solution that the world needs. Each of these blockchains has its own set of rules which governs the activity which occurs on the blockchain. These blockchains all have their own set of features which might attract users to their network, however, there are very few trustworthy options for someone to operate between two separate blockchains. Because each blockchain has its own set of features it makes sense to be able to take one's assets on one blockchain and use them to accomplish some action on another blockchain. However, each chain will undoubtedly have its own set of currencies that are accepted or popular on that particular blockchain network. For example the bitcoin currency is the native currency on the Bitcoin network while ether native currency is the native currency on the Ethereum blockchain network. There are many more blockchains, each with their own set of tokens. If one wanted to interact between blockchains they would also need to be able to exchange the tokens of one chain to interact with the applications on another chain. This is the problem which we have tackled in our senior design project. We have built an exchange on top of a blockchain which has support for cross-chain communication such that users can bridge their tokens from one chain to ours and then swap them for whichever tokens they desire in a decentralized fashion.

1.2 Existing Solutions

When it comes to cross-chain communication solutions there are a few that have some popularity. The most notable are Cosmos, Polkadot, and Near. All three of these are projects which provide developers with the tools to build blockchains that have cross-chain communication solutions built within them. We will go into

greater depth about the differences between these projects in the background chapter.

Furthermore, there are also a significant amount of decentralized exchanges that are available within the cryptocurrency industry. The most notable of these being Uniswap. Uniswap is a project which first popularized the idea of creating a decentralized exchange on a blockchain. The difference between a decentralized exchange and a centralized exchange is who is providing the liquidity by which users trade with. In a centralized exchange users will usually trade directly with another user. For example, if you want to buy something for a given price and another user wants to sell something at that same price then the two users will be matched and the order will be fulfilled. This describes the order book model. Another form of centralized exchange is where there is some central authority providing the capital in which to "make the market" at a given price. When a user wants to exchange, they can immediately do so by trading with this centralized authority. In this way they are the market maker. Both of these approaches work fairly well in many cases, like the New York Stock Exchange, but are much less practical for exchanges that are built on top of a blockchain network. This is due to a variety of reasons. For one, computation is usually quite expensive on a blockchain network. This is because a blockchain network is essentially a single global state. Which means that whenever this state changes it must be updated on every computer that is running the blockchain across the world. Thus the order book model of an exchange is not practical for this reason. Another reason why these are not great for a blockchain network is that one of the primary purposes of a blockchain network is to create a decentralized form of value and computation. This means that users will likely be opposed to an exchange which relies on a single point of value such as a market maker model. We will detail further about how a decentralized exchange works in the background section.

There are many decentralized exchanges available to be used right now. Other than Uniswap, common alternatives are Sushiswap, Balancer, Bancor, and Pancakeswap. All of these have minor and major changes in how they operate.

1.3 Our Solution

Our solution is to build a decentralized exchange on top of cross-chain communication solutions such as Cosmos, Polkadot, and Near. The purpose of this is to create an application which both supports cross-chain communication and allows users to swap their tokens from one blockchain for that of another blockchain.

One might reasonably ask why someone would want to swap tokens from one blockchain for tokens of another. The primary reason is that not all blockchains are alike. For example, the most prominent blockchain, Bitcoin, is quite limited in its functionality compared to newer blockchains such as Ethereum or Solana. That

does not mean that Bitcoin is worse than Ethereum, Solana, or any other of the new "smarter" blockchains. It is just how Bitcoin was designed. However, if someone owned some bitcoin on the Bitcoin blockchain network and wanted to access the functionality of newer blockchains such as Ethereum they would need a way to exchange their bitcoin with the tokens that are used on Ethereum, namely ether. Once they have bridged their tokens onto the Ethereum network they are free to explore all of the functionality that is offered by Ethereum. One common example is to loan your tokens out and earn interest on the loan.

To accomplish this is where something like our solution would come into play. Using our decentralized chain built on top of a cross-chain communication solution users would be able to exchange their assets from a source blockchain network onto the destination blockchain network.

Chapter 2

Background

2.1 Blockchain

You have undoubtedly heard about blockchain at some point in the last few years. Most likely in relation to the Bitcoin blockchain network. The simplest way to explain what a blockchain is global state machine. In the case of the Bitcoin blockchain network that state refers to a series of accounts and how much bitcoin that account has. A blockchain is global in that this state is stored in computers running around the world. The way in which this state is changed is through a transaction. For example, if Alice wants to send 1 bitcoin to Bob, then she needs to submit this transaction to one of these computers, known as a miner. Once a miner accepts this transaction it is propagated to the rest of the computers running the Bitcoin blockchain software, also known as nodes.

Of course, there are many other blockchains than Bitcoin. Most of the new blockchains share a fundamental difference. Instead of using miners to accept new transactions, they use what are called validators. The difference between these goes beyond the scope of this paper. One will often refer to this as a difference in consensus mechanism. Bitcoin uses a proof-of-work consensus mechanism while newer blockchains use a proof-of-stake consensus mechanism. One of the most important differences between proof-of-work and proof-of-stake is that proof-of-stake does not require miners to accept new transactions. Meaning that the energy needed to mine is no longer expended in blockchain networks which use a proof-of-stake consensus mechanism. One other important difference between the proof-of-work consensus mechanism and the proof-of-stake consensus mechanism is that proof-of-stake achieves instant finality while proof-of-work experiences probabilistic finality. This means that in blockchain networks that use a proof-of-work consensus mechanism there is always a chance that the state that you believe to be the current state is overtaken by a fork of the blockchain network. The reason that this is is because of proof-of-work consensus mechanism is

governed by computational power. Meaning that if someone can gather 51 percent of the total computational power on the Bitcoin blockchain network they can outpace the rest of the network and create a fork of the Bitcoin blockchain network where there are different transactions included in the state in a sequence which is possibly different than before. Because of this chance the state in a proof-of-work consensus mechanism is said to be probabilistic. This is not the case for proof-of-stake consensus mechanism which does not rely on computational power like proof-of-stack consensus mechanism.

2.2 Cross-Chain Communication

Because of the difference in which some blockchains achieve consensus, it can be difficult to create a system in which these blockchains can communicate between each other. Imagine a situation where a proof-of-work blockchain is trying to show to proof-of-stake blockchain that a certain transaction occurred on its blockchain so that the proof-of-stake blockchain can change its state to account for that transaction. There is no way to know for certain whether that transaction will definitively exist in perpetuity. Thus, the proof-of-stake blockchain network can not be 100 percent certain whether it should accept this transaction or not. Because of this issue cross-chain communication between blockchain networks which utilize different consensus mechanisms is still a problem with no good solution.

Cross-chain communication between blockchain networks that both use proof-of-stack is much more reasonable. Because both blockchains achieve instant finality one blockchain network can be certain that the transactions of another blockchain network are final and can be accepted as such.

For our project we began by exploring different forms of cross-chain communication solutions. Most of them focused on creating a platform by which developers can create blockchains that use proof-of-stack consensus mechanism which utilize a protocol which allows it to communicate with other similarly made blockchains.

2.3 Polkadot

Polkadot is a cross-chain communication which is centered around what they call the relay chain. In Polkadot every blockchain is connected to this relay chain. Whenever there is a transaction that takes place on one of the blockchains there are a set of validators that confirm this transaction back to the relay chain [2]. Furthermore any of these blockchains that are connected to the relay chain can query the relay chain for transactions. In this way cross-chain communications are enabled. Because transactions are visible to all blockchains within

the Polkadot network these blockchains can effectively interact with each other.

One unique thing about Polkadot is that in order to become one of these blockchains you have to pay and there is a limited amount of spots. This is because the validation is provided by the relay chain and that comes at a cost. Because of this Polkadot is not the most open when it comes to allowing anyone to create a blockchain network which uses cross-chain communication.

2.4 Near

Near is a much simpler model than Polkadot. Instead of creating a blockchain network from the ground up, it simply builds on top of existing blockchains to enable them to communicate between blockchains. For example, to create a bridge between the Ethereum blockchain network and another blockchain network Near would have a computer monitoring each network and relaying them to the other blockchain.

2.5 Cosmos

The last cross-chain communication solution that we research is Cosmos. Cosmos is the most flexible of the solutions. With Cosmos anyone can create a blockchain and communicate to other blockchains using their Inter Blockchain Communication protocol. Any blockchain that has incorporated this Inter Blockchain Communication protocol will have the means to interoperate with each other [1]. Unlike Polkadot each blockchain does need to provide their own source of validation. For our project we chose to build on top of the Cosmos blockchain network because we felt that it would provide us with the most realistic and effective solution for the amount of time that we have to work on our project. Because of its open design and strong community we decided that Cosmos was the best solution for us.

Chapter 3

Requirements

3.1 Functional

In the order that we approached them:

1. Create Pool

see §8.1.1

2. Query Pools

see §8.1.5

3. Query Liquidity Providers

see §8.1.6

4. Join Pool

see §8.1.2

5. Exit Pool

see §8.1.3

6. Swap

see §8.1.4

7. Add Liquidity

see §8.2.1

8. Remove Liquidity

see §8.2.2

9. Send Shares

see §8.2.3

10. Local Relayer

see §8.7

3.2 Nonfunctional

- Swap Fees

see §8.3

- Exit Fees

see §8.4

- Fee Slippage

see §8.5

Chapter 4

Technologies Used

Because we decided to build our project on top of the Cosmos blockchain project it was natural to use the Cosmos software development kit. Using the Cosmos software development kit, one can easily get up and running with a simple blockchain and not have to worry about the details of consensus, cross-chain communication, and the many other things that make up a blockchain.

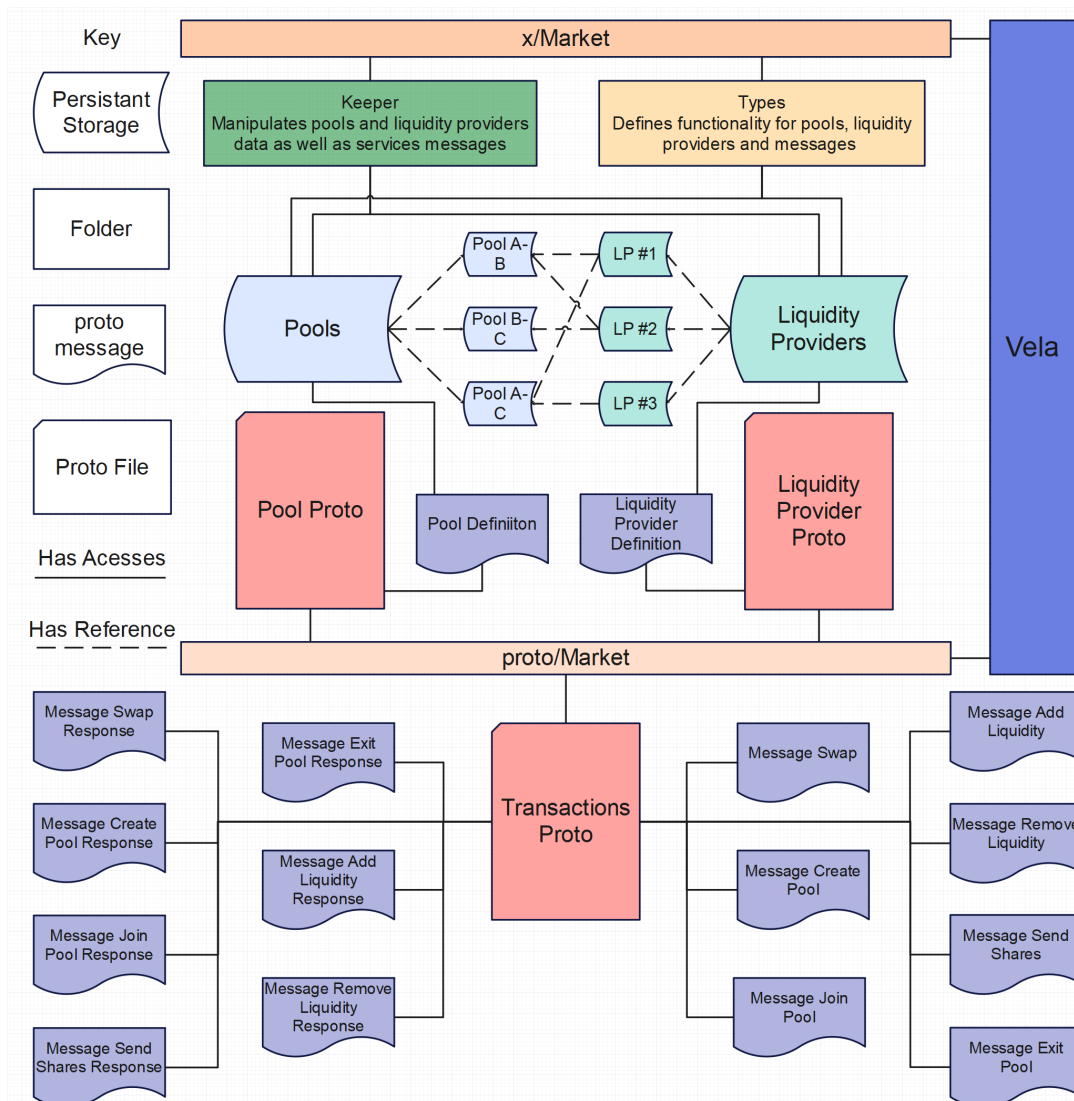
The Cosmos software development kit uses the Go programming language to interact with the Tendermint consensus mechanism using their ABCI socket interface. Tendermint not only handles network consensus but also the networking requirements of the chain, including gossiping new messages, broadcasting transactions and routing packets. This is all abstractly taken care of, enabling developers to focus solely on the implementation of the blockchain's application and treat network communication as a black box.

One technology that we used on top of the Cosmos software development kit is ignite. Ignite is a tool which uses the cosmos software development kit to scaffold basic blockchains from a command line interface. With it one can add functionality to support the inter-blockchain communication protocol module and many more. Using ignite was the starting point for our project. However, after a certain point we were unable to use ignite for the specific details of our application because ignite does not provide us with that level of detail. Thus, at those points we went into the actual Go code to accomplish what we need.

Chapter 5

Vela Architecture

5.1 Architecture Overview



11
Figure 5.1: Relevant Chain Architecture

Our code's architecture was scaffolded by the Ignite CLI. The Ignite CLI incorporates the Cosmos SDK's structure and functionality into a ready-made boilerplate chain, set up to be built upon. Network message passing and network consensus is handled by the Tendermint consensus mechanism. The Cosmos SDK interacts with Tendermint through the ABCI interface. These interactions occur over a socket, enabling Tendermint to work with any socket-enabled language. The Cosmos SDK is built primarily in Go, as is the project. Proto3 protocol buffers and JSON encoding are used to serialize data for persistent storage.

Most go modules in the Cosmos SDK have a types package and a keeper package. Type packages contain relevant module constants, methods for messages and structures, and keeper coordination between modules through selective exposure of only specified functions. Only a module's keeper has access to the module store. Every query and transaction uses the keeper methods to view and manipulate the state of the module's store. Keeper packages typically contain gRPC query files, servers and services for messages, and CRUD for the store. The types and keepers packages within each module and between modules work in concert with one another following the object capabilities (OCAP) design of the Cosmos SDK. These packages are where most of our development process took place while building the chain we call Vela.

5.2 Persistent Storage

Persistent data held on the chain includes a map of liquidity pools, indexed by their associated token denominations, and a map of liquidity providers, indexed by pool name and account address. Liquidity providers are associated with pools however the reverse is not true. A pool without any liquidity providers will have residual liquidity from the last exit fee applied. Therefore, liquidity pools are only ever created and never destroyed. Joining/exiting a pool adds/removes a liquidity provider to/from the map of liquidity providers. Add/Remove liquidity exchanges an amount of shares for an amount of tokens in. Send shares enables liquidity providers to send their pool shares to another user without exiting the pool. The data being changed by these transactions is stored on the chain, Transactions potentially change the data, resulting in a new state and therefore a new block.

5.3 Pools

Pools store a pair of assets sent to it. The ratio between the assets determines the swap price from one to another. The ratio in practice should generally reflect the market price's ratio between the tokens. Any deviance from the ratio creates an arbitration opportunity for a swap followed by an immediate sell on a

centralized exchange.

5.4 Liquidity Providers

Liquidity providers link an account address to shares owned in a pool. By limiting shares to only being valid on chain, without IBC, shares can be sent from one user to another, regardless of if the receiving account is already a liquidity provider for the share's pool or any other pool. Joining a pool will create a new liquidity provider, while exiting a pool deletes the associated liquidity provider from the persistent storage. Shares owned for a pool, by a liquidity provider, are held in the liquidity provider structure.

5.5 Messages

Messages are submitted by users to validating nodes. These nodes will gossip the message to neighboring nodes if the message is valid. A validated message can then go on to manipulate the state of the blockchain or query the state for relevant information.

5.6 Transactions

Messages which result in the state changing are bundled into transactions. When a validating node submits a new block, it does not submit a message or list of messages, but rather a list of transactions. Each transaction bundles relevant messages with the transaction fee and gas price for the transaction. These transactions are then grouped and inserted into a new block which is proposed for the rest of the network's nodes to validate. Once a new block has been validated, the transactions and therefore messages contained are executed. This is when the persistent storage of the blockchain is actually manipulated. If valid, the fees and amounts are deducted and the transaction is finalized along with the block. If rejected, the transaction does not modify the persistent store.

```

bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad tx market
market transactions subcommands

Usage:
  velad tx market [flags]
  velad tx market [command]

Available Commands:
  add-liquidity    Broadcast message add-liquidity
  create-pool      Broadcast message create-pool
  exit-pool        Broadcast message exit-pool
  join-pool        Broadcast message join-pool
  remove-liquidity Broadcast message remove-liquidity
  send-shares      Broadcast message send-shares
  swap             Broadcast message swap

```

Figure 5.2: Available Transaction Commands for the Vela CLI

5.7 Queries

Queries are exactly as the name suggests; messages which do not change the state of the blockchain, but instead simply observe it. From being aware of the current state of the pools on the chain, to checking the shares of a liquidity provider, queries enhance the UX and enable a more straightforward integration for UI. Custom queries in our project include listing all, or just one, pool(s) or liquidity provider(s). Default queries include the ability to check account balances with an account address, viewing chain parameters, and viewing valid commands on a per module basis.

```

bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad q market
Querying commands for the market module

Usage:
  velad query market [flags]
  velad query market [command]

Available Commands:
  list-liq-prov list all liqProv
  list-pool     list all pool
  params        shows the parameters of the module
  show-liq-prov shows a liqProv
  show-pool     shows a pool

```

Figure 5.3: Available Querying Commands for the Vela CLI

Chapter 6

Functionality and Analysis of Vela

By the end of our development phase of the project, we'd successfully built a running version of our application specific blockchain, Vela. We met our core functionality goals for the chain but did not meet all of our elective functional requirements. The blockchain was only ever built and tested using a single node to run a simulated network, and more steps would be needed to be taken before deployment on a test-net. In this single node environment, our test results were as expected although if the project were to continue, a more refined approach should be developed further before launching on a test-net to address the issues raised below.

6.1 Core Functionality

The core functional requirements we set for ourselves were aimed at creating a working automated market maker model. Those messages and queries include the following:

6.1.1 Create Pool

- `~$ velad tx market create-pool [amount-a] [denom-a] [amount-b] [denom-b] [min-shares]`
- The create-pool command creates a new pool for the given denomination pair, if no such pool already exists, and creates a new liquidity provider with at least min-shares of the pool.

```
bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad tx market create-pool \
> 10000 alpha \
> 10000 beta \
> 1000 \
> --from alice
{"body":{"messages":[{"@type":"/VelaChain.vela.market.MsgCreatePool","creator":"cosmos1s6rtmvvy8vk9epytkhx66m2uz759r637l1nnqrg","amountA":"10000","denomA":"alpha","amountB":"10000","denomB":"beta","minShares":"1000"},"memo":"","timeout_height":"0","extension_options":[],"non_critical_extension_options":[],"auth_info":{"signer_infos":[],"fee":{"amount":[],"gas_limit":"200000","payer":"","granter":""},"signatures":[]}}]}
confirm transaction before signing and broadcasting [y/N]:
```

Figure 6.1: Command Line Create Pool Transaction

6.1.2 Join Pool

- `~$ velad tx market join-pool [amount-a] [denom-a] [amount-b] [denom-b] [min-shares]`
- The join-pool command creates a new liquidity provider with at least min-shares for the pool associated with the given denomination pair.

```
bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad tx market join-pool \
> 1000 alpha \
> 1000 beta \
> 100 \
> --from bob
{"body":{"messages":[{"@type":"/VelaChain.vela.market.MsgJoinPool","creator":"cosmos1ts46vq19nwmn99av5gpn2v2ctd43k5nk0qputh","amountA":"1000","denomA":"alpha","amountB":"1000","denomB":"beta","minShares":"100"},"memo":"","timeout_height":"0","extension_options":[],"non_critical_extension_options":[],"auth_info":{"signer_infos":[],"fee":{"amount":[],"gas_limit":"200000","payer":"","granter":""},"signatures":[]}}]}
confirm transaction before signing and broadcasting [y/N]:
```

Figure 6.2: Command Line Join Pool Transaction

6.1.3 Exit Pool

- `~$ velad tx market exit-pool [denom-a] [denom-b]`
- The exit-pool command converts all of the message creator's pool shares into the tokens for the pool and deletes the liquidity provider from the persistent storage.

```
bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad tx market exit-pool \
> alpha \
> beta \
> --from alice
{"body":{"messages":[{"@type":"/VelaChain.vela.market.MsgExitPool","creator":"cosmos1s6rtmvvy8vk9epytkhx66m2uz759r637l1nnqrg","denomA":"alpha","denomB":"beta"},"memo":"","timeout_height":"0","extension_options":[],"non_critical_extension_options":[],"auth_info":{"signer_infos":[],"fee":{"amount":[],"gas_limit":"200000","payer":"","granter":""},"signatures":[]}}]}
confirm transaction before signing and broadcasting [y/N]:
```

Figure 6.3: Command Line Exit Pool Transaction

6.1.4 Swap

- `~$ velad tx market swap [amount-in] [denom-in] [min-amount-out] [denom-out]`
- The swap command adds amount-in to the pool for at least min-amount-out using amount-out = amount-in * (pool-amount-out / pool-amount-in) to convert the value of the denom-in coin to the value of the denom-out coin according to the pool's ratio.

```
bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad tx market swap \  
> 1000 alpha \  
> 990 beta \  
> --from carol  
{ "body": { "messages": [ { "@type": "/VelaChain.vela.market.MsgSwap", "creator": "cosmos1wf7yyz28xfx456qu2zfz7e8f15he80rk4aw4nt", "amountIn": "1000", "denomIn": "alpha", "minAmountOut": "990", "denomOut": "beta" }, { "memo": "", "timeout_height": "0", "extension_options": [], "non_critical_extension_options": [] }, { "auth_info": { "signer_infos": [], "fee": { "amount": [], "gas_limit": "200000", "payer": "", "granter": "" }, "signatures": [] } } ] } }  
confirm transaction before signing and broadcasting [y/N]:
```

Figure 6.4: Command Line Swap Transaction

6.1.5 Query Pools

- `~$ velad q market list-pool`
- Lists the data structure for each pool.

6.1.6 Query Liquidity Providers

- `~$ velad q market list-liq-prov`
- Lists the data structure for each liquidity provider.

6.2 Elective Functionality

These transactions were not needed for the initial bare-bones proof of concept.

6.2.1 Add Liquidity

- `~$ velad tx market add-liquidity [amount-a] [denom-a] [amount-b] [denom-b] [min-shares]`
- The add-liquidity command adds at least min-shares of the pool to the message creator's liquidity provider for the pool.

```
bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad tx market add-liquidity \
> 1000 alpha \
> 1000 beta \
> 100 \
> --from bob
{"body":{"messages":[{"@type":"/VelaChain.vela.market.MsgAddLiquidity","creator":"cosmos1ts46vql9nwmn99av5gpn2v2ctd43ksnk0qputh","amountA":"1000","denomA":"alpha","amountB":"1000","denomB":"beta","minShares":"100"},"memo":"","timeout_height":"0","extension_options":[],"non_critical_extension_options":[],"auth_info":{"signer_infos":[],"fee":{"amount":[],"gas_limit":"200000","payer":"","granter":""},"signatures":[]}}]}
confirm transaction before signing and broadcasting [y/N]:
```

Figure 6.5: Command Line Add Liquidity Transaction

6.2.2 Remove Liquidity

- `~$ velad tx market remove-liquidity [shares] [denom-a] [denom-b]`
- The `remove-liquidity` command removes some, but not all, of a liquidity provider's shares and converts the amount to the pool's assets, using the current pool ratio as a spot price.

```
bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad tx market remove-liquidity \
> 100 \
> alpha \
> beta \
> --from bob
{"body":{"messages":[{"@type":"/VelaChain.vela.market.MsgRemoveLiquidity","creator":"cosmos1ts46vql9nwmn99av5gpn2v2ctd43ksnk0qputh","shares":"100","denomA":"alpha","denomB":"beta"},"memo":"","timeout_height":"0","extension_options":[],"non_critical_extension_options":[],"auth_info":{"signer_infos":[],"fee":{"amount":[],"gas_limit":"200000","payer":"","granter":""},"signatures":[]}}]}
confirm transaction before signing and broadcasting [y/N]:
```

Figure 6.6: Command Line Remove Liquidity Transaction

6.2.3 Send Shares

- `~$ velad tx market send-shares [shares] [denom-a] [denom-b] [address]`
- The `send-shares` command sends shares amount of shares for the denom-a denom-b pool from the liquidity provider associated with the message creator and denomination pair, to the address provided, creating and deleting new liquidity providers appropriately.

```
bradley@DESKTOP-R6DRIIT:~/VelaChain/vela$ velad tx market send-shares \
> 100 \
> alpha \
> beta \
> "cosmos15tw0lkn8ku72c4mlkaqyapfpwj4k74g0asrqc8" \
> --from bob
{"body":{"messages":[{"@type":"/VelaChain.vela.market.MsgSendShares","creator":"cosmos1ts46vql9nwmn99av5gpn2v2ctd43ksnk0qputh","shares":"100","denomA":"alpha","denomB":"beta","address":"cosmos15tw0lkn8ku72c4mlkaqyapfpwj4k74g0asrqc8"},"memo":"","timeout_height":"0","extension_options":[],"non_critical_extension_options":[],"auth_info":{"signer_infos":[],"fee":{"amount":[],"gas_limit":"200000","payer":"","granter":""},"signatures":[]}}]}
confirm transaction before signing and broadcasting [y/N]:
```

Figure 6.7: Command Line Send Shares Transaction

6.3 Swap Fees

In our pool model, a fee is charged for using a pool's liquidity for a swap. This swap fee is applied to the amount being added to the pool. The amount to remove and send to the message creator is calculated using the swap fee reduced amount in, $(1 - \text{SwapFee}) * (\text{Amount In})$. An implication of this fee structure is that swap fees are not actively paid but applied under the hood and implicitly distributed to liquidity providers by increasing the value of the pool's shares. These fees, in conjunction with exit fees, help to offset the almost unavoidable situation of realized impermanent loss and acts as an incentive for liquidity providers to provide liquidity for longer periods of time.

6.4 Exit Fees

Exit fees are applied similarly to the approach used for swap fees. When exiting a pool, shares are converted into their respective amounts of each of the tokens in the pool and sent to the user after the shares are burned. In the same manner that swap fees have the fee applied before the math for conversions is applied, the number of shares to convert is reduced by the exit fee and then converted to the tokens according to the number of shares and ratio in the pool.

6.5 Slippage

A phenomenon known as slippage can occur when the pool's spot price for a swap changes in the time between a swap being broadcast and the transaction being processed. Slippage can occur favorably but it can also have very drastically negative effects in inopportune sequences of swaps. It is therefore important to prevent the price from slipping too much between the broadcasting and processing of a message. Our pool model handles this for swap transactions by using the entire amount from the transaction, minus the swap fee portion, and comparing the result to a minimum amount out which is set on the command line when submitting a swap transaction. What this means for users making swaps is that the slippage can be at most as bad as the minimum amount requested out, but the potential positive benefits of slippage (getting more tokens out than anticipated) are not limited.

A similar approach is used when requesting shares out of a pool. To account for slippage when joining or adding liquidity to a pool, a minimum number of shares is requested in the transaction. If by the time the transaction is processed, this minimum number of shares is unobtainable, the transaction is rejected and slippage is avoided.

6.6 Limitations

During testing, one issue with our model's implementation that stood out was the relative ease with which one can effectively drain a pool's liquidity. If a pool's liquidity is small enough, or an account's balance is large enough, the price impact that a swap can make on a pool can be drastic. For example, consider a pool with 100 alpha and 100 beta and disregard fees. If Frank uses this pool to swap 50 alpha for 50 beta, the pool's new liquidity will be 150 alpha and only 50 beta. Frank will have gone from having 50 alpha to having 50 beta, but now the pool's price for 150 alpha is 50 beta. Frank can therefore swap back that 50 beta into the pool and receive 150 alpha. Because Frank's transactions had such a significant price impact on the pool, he created a scenario in which he could effectively drain the pool with negligible costs to him. If this project were to continue, addressing this issue would be extremely important.

Existing DEXs have addressed this problem by incorporating the price impact a swap has on the pool in the processing of a message. A simple implementation of this involves breaking up a large impact swap into many smaller impact swaps and processing them in a sequence. This mitigates the impact of a single large swap because the spot price, or ratio, for the pool is changing with each swap.

6.7 Interoperability

Our blockchain was built to be an IBC enabled chain, and it is. When creating the Market module, a dependency on IBC was established. This means that a relayer can be set up to connect our blockchain to any other IBC enabled blockchain. No custom packets were implemented specifically for using a common module between chains. Instead, our chain can send and receive IBC tokens using the bank module to any other connected chain. What happens once it gets there is up to the recipient of the IBC transaction. So if Alice wants to use Vela's liquidity pools for a swap, but does not have any vela, she can simply send her tokens directly to her account on Vela using an IBC transfer. From there, she can make swaps as she pleases within the chain. When Alice is done swapping and wants to remove her tokens, she can use an IBC transfer to move her tokens off of Vela and onto any IBC enabled chain with a relayer connecting to Vela.

While all of the IBC interactions should work, we did not complete our non-functional requirement of actually setting up and properly testing a relayer between Vela and another IBC enabled chain. However, the IBC-Transfer module used is a standard package with widespread use. While more testing is always a good thing, assuming that IBC-Transfer packets would be correctly handled by the Cosmos SDK is a fair assumption, with any errors more likely originating from an incorrectly set up relayer. To clarify, if an

IBC enabled chain were to successfully send an IBC-transfer packet to Vela, all of the previously described functionality can be used with the IBC transferred token. However, this has not been fully tested.

Chapter 7

Deployment Impacts

7.1 Ethical Impacts

7.1.1 Ethical Justification

The ethical justification for our project is to enable freedom for people to transact freely between separate assets. We believe that financial freedom should be available to everyone in the world. Unfortunately, this is very much not the case at the moment. While one may not realize this if they live in a country like the United States of America, many countries restrict the financial freedom of their citizens. Some of the countries with the least economic freedom include Iran, Democratic Republic of the Congo, Sudan, and many more. These countries have extension regulation that restricts economic freedom, prevent international trade, and often have high inflation in their native currency. When you consider the position that citizens from these countries are in, it becomes much clearer why someone would seek alternative forms of money such as cryptocurrencies. Our project is building on top of the economic freedom created from cryptocurrencies. With our decentralized exchange built on top of cross-chain communication solutions we aim to further increase the economic freedom available throughout the world. Not only does our project allow users to swap between cryptocurrencies it also allows them to earn yield from their cryptocurrencies by providing liquidity. Such financial actions are not commonly seen for many people living in non-economically free countries.

One important aspect of any good form of money is the rate at which it inflates. In other words how much the value of the currency decreases each year. Usually, the United States aims at a two percent inflation rate each year. However, during the COVID-19 pandemic the United States dollar has inflated at much higher rates. If you are a United States citizen and are uncomfortable with the rate of inflation that the United States dollar is experiencing then we believe that you have the right to opt out of that currency and find a currency that better aligns with your financial goals. Fortunately most cryptocurrencies outline a clear and immutable

rate of inflation so anyone can decide whether to buy it long-term or not. Bitcoin for example incrementally slows its rate of inflation until it reaches 21 million total bitcoin in circulation. This point will occur in the year 2140. At that point all the bitcoin that will ever be created will have been put into circulation and the rate of inflation will be zero percent. It is for this reason that some people believe that bitcoin is a stronger store of value than currencies controlled by governments such as the United States dollar because the government is keen to print billions of their currency if they deem it necessary to stimulate the economy. It is for this reason that we believe that our project is justified in that it provides the ability for people to choose their form of money.

7.1.2 What it means to be a good engineer?

The first way in which we believe that this project has made us into better engineers is in the commitment to the public good. While one might not initially think that an exchange is a public good, since our exchange is built on a blockchain it is open to anyone and there is no central authority who owns this exchange. Even though we created it once we publish it, anyone can look at the code and can make changes to it as they see fit. In this way our project is very much a public good in that nobody owns it and provides economic freedom to everyone. This project has made it clear to us that creating public goods comes with a unique set of challenges. On one hand you do not own the thing you are creating, but on the other hand you have the opportunity to create a much larger impact in the world because it is public.

Another key characteristic that this project has taught us is in the form of effective teamwork. While there were only two of us in our team, this project informed our views about the importance of working together in a close-knit manner. Initially we were focused on splitting up the responsibilities cleanly and neatly such that we could be most efficient. While there are certainly times when it is necessary to divide and conquer, we learned that often times when we are both focusing on the same problem we will come up with ideas that were much better than those that we would have thought of individually. At the very least it makes us think differently and stay focused on the problem. Furthermore, we learned that while it is important to work together closely, if someone does not prepare for the meetings by not doing the requisite research, working together can become an impedance. This is because one person is ahead of the other and will not be as effective while the other person catches up.

One other way in which this project has informed us is related to courage. Whenever one is embarking on a large project where you do not really know what you are doing it takes a large degree of courage to even get started. This project has taught me this and much more. Another thing this project taught me is not to be

afraid to fail and to look forward to the opportunities where you have a chance to learn. In these ways my view on the importance of courage in being a good engineer has been sculpted.

7.1.3 Ethical Pitfalls

The primary ethical consideration when it comes to pitfalls is whether total economic freedom is actually for the better of the individual. There is an argument to be made that not everyone has enough knowledge in order to make the best decisions for themselves. For example, there are many instances of people losing all of their money in cryptocurrencies because they invested in a scam or simply made poor decisions. In this sense the primary risk and ethical pitfall is that there is an argument to be made that the government has a better argument to have control over the economic activities of their citizens.

7.2 Societal Impacts

In terms of the impact on society, this project, and others like it can have a profound impact. A world in which currencies that are not governed directly by a central bank like the United States dollar is drastically different than the one that we live in currently and it is difficult to imagine all the ways in which our world would change. One of the most important changes is that individuals will have a much greater say in which currencies they choose to store their money in. Most people currently will save in whatever the local currency is, but if the government and money are fully decoupled and everyone has access to exchanges there is much more freedom in the ways in which one stores value.

7.3 Civic Engagement

From a political point of view, many politicians oppose cryptocurrencies. Even in freedom oriented countries like the United States of America there are a number of politicians that oppose cryptocurrencies. There are a number of reasons, many say it is because of the environmental impact that cryptocurrencies have. While there is some validity to these statements, most new blockchains do not employ a proof-of-work consensus mechanism so this argument falls flat in those cases. Another less visible reason is that politicians want to be able to control the economic activity of their currency. They want their currency to be a global reserve currency like the United States dollar is. One obvious example of a country that wants complete control over their citizens economic activity is China. They have attempted to ban Bitcoin countless times and are attempting to create their own digital currency to completely monitor their citizens' economic activity.

While most countries are not nearly as restrictive as China, most politicians lean towards retaining control in the form of money used by its citizens. There is a decent argument to be made in their favor. Perhaps politicians and governments are better at regulating currency. However, many people do not believe this due to the misaligned incentives of being in control of a country's primary form of money.

Chapter 8

Conclusion

8.1 Obstacles Encountered

Throughout this senior design project the most challenging obstacle was embarking on a project that we knew very little about how to accomplish. Over time we learned how to take it one step at a time and focus on small milestones that eventually add up to a larger project. We begin by researching cross-chain communication solutions. Once we chose Cosmos we learned the basics of how to use the Cosmos software development kit. We also spent time learning the basics of the Go programming language. Once we broke the project into smaller more reasonable chunks, our confidence and progress increased.

8.2 Summary and Next Steps

Overall, we have created a rudimentary decentralized exchange which supports inter blockchain communication. The next steps to making this project function in a production environment are listed below.

1. Address and reduce swap price impact on a pool for significantly large transactions. With the current system, and ignoring fees, an account with about half of the total value locked in a pool can use that amount to effectively drain the pool entirely (see §8.6)
2. Run a local relayer to connect Vela and a generic inter blockchain communication chain.
3. Spin up a local test net.
4. Join Cosmos Network test-net.
5. Test functionality with multiple accounts.
6. Launch on Cosmos Network main-net.

Bibliography

- [1] Ethan Buckman Jae Kwan. Cosmos whitepaper, 2014.
- [2] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework, 2016.









Thesis_VelaChain

Final Audit Report

2022-06-11

Created:	2022-06-10
By:	Darcy Yaley (dyaley@scu.edu)
Status:	Signed
Transaction ID:	CBJCHBCAABAAEEri9RicdVXkpJcLa5wteurNarGkw3CU

"Thesis_VelaChain" History

-  Document created by Darcy Yaley (dyaley@scu.edu)
2022-06-10 - 11:55:36 PM GMT
-  Document emailed to Yuhong Liu (yhliu@scu.edu) for signature
2022-06-10 - 11:56:42 PM GMT
-  Email viewed by Yuhong Liu (yhliu@scu.edu)
2022-06-11 - 1:21:27 AM GMT
-  Document e-signed by Yuhong Liu (yhliu@scu.edu)
Signature Date: 2022-06-11 - 3:08:42 PM GMT - Time Source: server
-  Document emailed to N. Ling (nling@scu.edu) for signature
2022-06-11 - 3:08:44 PM GMT
-  Email viewed by N. Ling (nling@scu.edu)
2022-06-11 - 3:10:47 PM GMT
-  Document e-signed by N. Ling (nling@scu.edu)
Signature Date: 2022-06-11 - 3:11:11 PM GMT - Time Source: server
-  Agreement completed.
2022-06-11 - 3:11:11 PM GMT