Vega Technical Overview

vegaprotocol.io

July 29, 2019

1 Introduction

Vega is a technology protocol that allows decentralised public or private networks to facilitate the fully automated end-to-end trading and execution of financial products. Networks are secured with a byzantine fault tolerant consensus layer and implement pseudonymous margin trading using a novel market based liquidity incentivisation scheme to solve the problem of attracting and allocating market making resources in a decentralised system.

The technology choices and design of Vega are driven by criteria that align our engineering approach with the overall vision for Vega:

- **Security and correctness:** Vega must be designed with security and testability in mind.
- **Blockchain performance:** Vega will be and will remain at the cutting edge of public blockchain performance in terms of latency and throughput.
- **Application performance:** the application layer must perform on a par with professional (including non-blockchain based) trading systems.
- **Flexibility:** Vega cannot be tied to any specific blockchain or cryptocurrency for operation or trading and settlement.
- **Developer experience:** Vega must be easy to build against for all types of developer and use case.

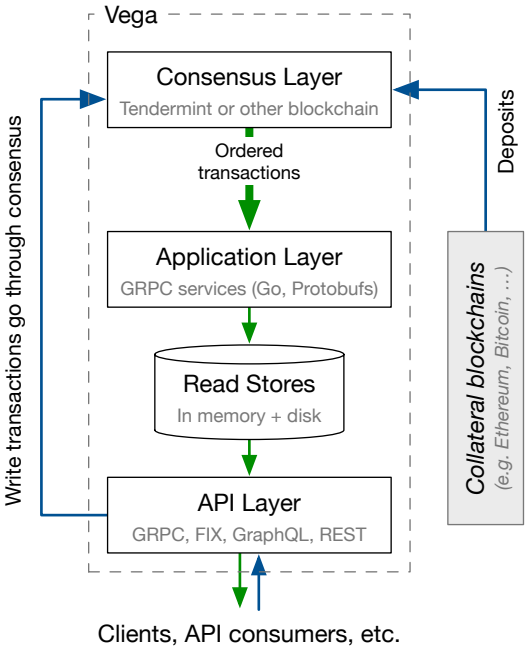
This short paper covers some of the more technical aspects of networks using the Vega protocol, and of the reference implementation developed by the Vega team. For a description of the protocol itself, see the Vega protocol whitepaper[1].

2 Architecture

We use *Command Query Responsibility Segregation*[2] (CQRS) and a modular design to enforce strict separation between the *consensus* (blockchain), *application*, and *API* layers in the reference implementation. Transactions are **Protocol Buffers** messages passed from the consensus layer to the application layer, in an order that is guaranteed by the consensus algorithm to be the same for all nodes[3].

[1] https://vega.xyz/papers/vega-protocol-whitepaper.pdf
[2] https://martinfowler.com/bliki/CQRS.html
[3] Vega therefore requires a blockchain with immediate finality.

Vega nodes also read from other blockchains that are used for collateral, and post transactions to the Vega network when they recognise a *deposit* or *withdrawal* on that blockchain. Vega therefore supports collateral from multiple blockchains, and cross-chain settlement.



Blockchain layer

Vega runs its own proof-of-stake blockchain network for performance, scalability, and flexibility. We currently use **Tendermint**[4] for consensus which provides a 1 second block time and can process 1000–4000 transactions per second (tps). Transactions undergo initial validation before being accepted, and are processed by the Vega application once each block has been finalised.

The separation between the blockchain and application means that Vega is *blockchain independent*, because the application layer can process valid, ordered transactions from any source. This allows Vega to migrate to a new consensus protocol if better technology becomes available. Blockchain independence also means that the Vega protocol and core implementation can be easily reused in other decentralised, distributed, or even traditional server-based environments to cater for a wider range of use cases.

[4] https://tendermint.com/

Application layer

The application (a.k.a. *trading core*) processes incoming transactions from the consensus layer — in the form of protocol buffers messages — sequentially and deterministically. This guarantees that all nodes will arrive at exactly the same state. The entire protocol currently requires just 11 transaction types:

governance: proposal to open a market, proposal to close a market, proposal to update a parameter, vote on a proposal.

trading: submit instruction (order), amend instruction, cancel instruction, notify observable (oracle data).

collateral: notification of deposit (on collateral chain), request for withdrawal, validation of withdrawal.

The trading core in our implementation is written entirely in **Go**, selected as it is a mature language that is ideal for writing dependable, maintainable, and high performance server applications. The Vega application is divided into functional components, which are described later in this document.

Read stores

The *read stores*, also implemented in Go, ingest and interpret a stream of events from the trading core and store the resulting data in memory and disk backed data structures designed to service API queries. Events include state changes for orders, trade executions, price and risk numbers, settlement cashflows, deposits and withdrawals, and governance actions.

API layer

Clients connect to various APIs, which perform *queries* on the read stores and post *commands* to the consensus layer. The APIs are designed to provide a great developer experience for different types of client system.

The GRPC and FIX APIs provide for high performance trading and data systems integration; whereas the REST and GraphQL APIs, which include support for streaming market data, are designed for quickly and easily building high performance front end applications and scripting.

3 Trading core components

Vega's trading core is a modular application with functional separation between components that allows for maximum configurability, including selective use of a subset of components in permissioned deployments that do not need the protocol's full functionality.

- **Matching engine:** a limit order book that operates either in continuous trading or auction mode for open markets. It will also support *request for quote* (RFQ) and *matched trades* for trading on *over the counter* (OTC) markets.
- **Risk engine:** evaluates the *risk model* for each market to calculate margin requirements for each participant's net open position. The risk engine then ensures that sufficient margin is allocated to each net position with allocation requests to the collateral manager, and initiates *closeout* trades if not.
- **Collateral engine:** maintains each participant's balance for every crypto-asset they have deposited by processing deposit notifications from collateral blockchains and settlement instructions from the settlement engine. It also handles the allocation of collateral to markets for margin.
- **Settlement engine:** generates settlement instructions for the collateral engine when markets mature, products create interim cashflows, and any time a position is fully or partially closed. Employs the *position resolution algorithm* if there is a shortfall at settlement.
- **Governance engine:** manages the creation and closure of markets on the network, and the modification of parameters. Actions are taken in response to voting that occurs following acceptance of a proposal transaction from one or more participants.

4 Performance

Blockchain

- The use of proof-of-stake allows for significant (multiple orders of magnitude) performance improvement over existing proof-of-work chains.
- Tendermint performance figures:

  Block time: 1 second

  Latency: 0.5–1.5 seconds

  Throughput: 1000–4000 tps[5]
- The architectural separation of the blockchain layer from the application layer allows for further improvements in future if more performant solutions become available.

Application

- Vega's matching engine has been tested with order books containing millions of orders and consistently processes submit, amend, and delete instructions in 5–15μs on a standard laptop.
- Risk models run on the 'bare metal' and can therefore take advantage of the full capabilities of the underlying hardware, including GPU acceleration and parallelisation.

API layer

- The GRPC API uses binary Protocol Buffers messages, and data is retrieved from optimised in-memory stores for high performance queries.

[5] Estimated production performance based on early testing.

5 Scaling

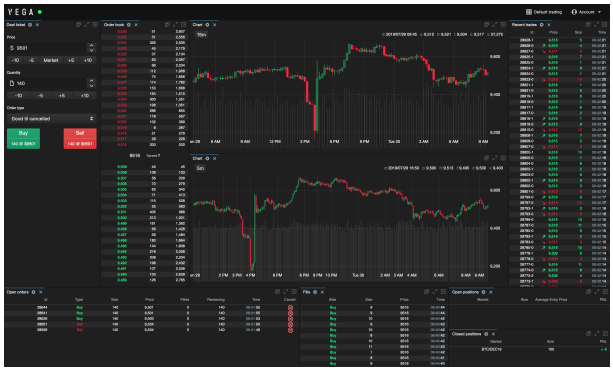
Vega has been designed from the outset to scale beyond the early adopter phase, to the level required for serious real-world use as part of the world's financial infrastructure.

- The protocol allows for the network to be sharded by *risk universe*[6], effectively meaning that each market can have it's own network. This means that Vega has essentially unlimited horizontal scalability and can therefore cater for any number of instruments and markets.
- Each blockchain, and therefore each market, is limited in transaction throughput by the blockchain technology in use as well as the physical network and computing infrastructure in place. This limit will increase over time with both hardware and software upgrades, which may include migrating to a different consensus protocol and implementation. For markets that are at the limit, transaction aggregation will allow participation to continue to increase, at the cost of latency for some transactions.
- Risk models can be run asynchronously, with results going through consensus, allowing for slower and more complex Monte Carlo type risk models as well as fast closed form calculations.

6 Reference trading UI

Whilst we expect many users to connect to the various APIs directly, Vega will provide a fully functional reference implementation of a trading UI for Vega.

The screenshot below is of an early prototype of this application, which will be developed into a professional quality trading application with a variety of trading and risk management tools. The trading front end will also be extensible with third party extensions.



The reference UI is a JavaScript + HTML *dApp* built with **TypeScript**, **React**, and **GraphQL** (Apollo) and can be run from a hosted server or directly from a user's local machine. Like the rest of Vega, it is fully decentralised, requiring only a connection to a full or passive Vega node to run.

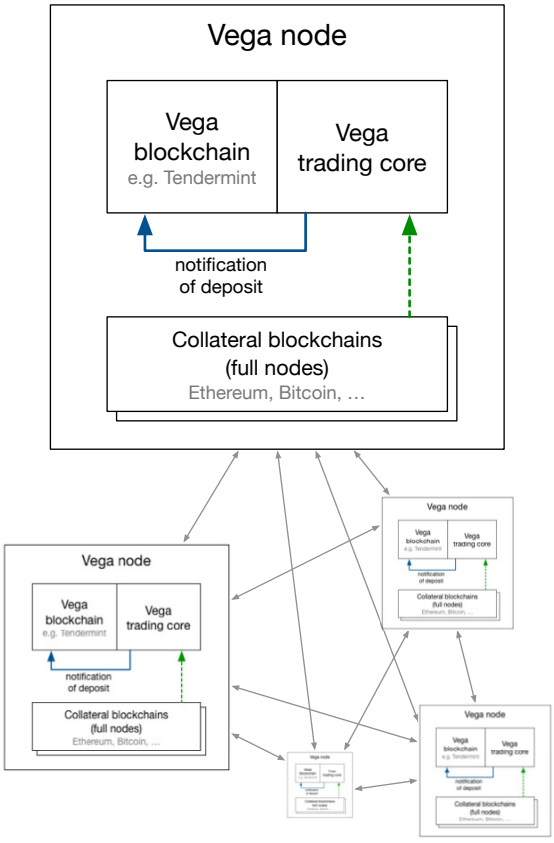
[6] A market or set of related markets that allow for perfect netting, for instance various maturities of a futures contract, or options with different strikes on the same underlying.

7 Multi-chain collateral

Vega is designed for high performance trading and settlement, but does not host the assets being traded. Instead, Vega is designed to work with the wide range of digital coins, tokens, and other assets held on existing blockchains.

However, although it manages the balances of any assets deposited with the network, the Vega blockchain itself doesn't hold these crypto-assets. In this sense, Vega is a *'second layer'* solution or *side-chain*, providing functionality on top of that provided by other blockchains such as Bitcoin and Ethereum, which hold the underlying assets being traded.

To achieve this, every Vega *full node* operator will also need to run full nodes of each supported collateral blockchain.



For example, a smart contract on the Ethereum blockchain is used to hold funds deposited to Vega and regulate withdrawals. Payments into the smart contract are recognised and ready for use on Vega once they have been observed by two thirds of the nodes. To withdraw funds, a request is made via a Vega transaction. If it is valid, the Vega nodes sign a multi-sig withdrawal transaction that can be posted to the Ethereum blockchain once it receives enough (two thirds of nodes, again) signatures to complete the transaction.

A similar approach can be taken for every collateral blockchain, however in practice we will at some point look for multi-chain / cross-chain solutions to support collateral from a wider range of source chains with a single integration.