EOS.IO Software Roadmap

This document outlines the development plan from a high level and will be updated as progress is made toward version 1.0. It should be noted that this roadmap applies only to the blockchain software and not to the other tools and utilities such as wallets and block explorers which will have their own teams and dedicated roadmaps once Phase 1 is complete.

Everything contained in this document is in draft form and subject to change at any time and provided for information purposes only. block.one does not guarantee the accuracy of the information contained in this roadmap and the information is provided "as is" with no representations or warranties, express or implied.

Phase 1 - Minimal Viable Testing Environment - Summer 2017

The goal of this phase is to establish the APIs that developers will require to start building and testing applications on EOS.IO. In order for developers to start testing their applications they will require the following to be implemented:

Standalone Node (Dan & Nathan)

A standalone node operates a test blockchain and produces blocks while exposing an API. This node does not need to concern itself with any P2P networking code.

Native Contracts (Nathan)

The EOS.IO software has a number of native contracts. These are contracts that manage the core operations of the blockchain and exist outside the Web Assembly interface. These contracts include:

1. @eos - manages EOS token transfers
2. @stake - manages locked EOS, voting, and Producer Election
3. @system - manages permissions, messages, and contact code updates

Virtual Machine API (Dan)

Contracts are compiled to WebAssembly (WASM) and WASM must interface with the blockchain via a defined API. This API is what developers depend upon to build applications and be relatively stable before developers can really start to build on EOS.

RPC Interface (Arhag, Nathan)

A simple JSON RPC over HTTP interface will be provided that enables developers to broadcast transactions and query application state. This is critical for both publishing and interacting with test applications.

Command line Tools (Arhag)

Command line tools facilitate integrating the RPC interface with developer build environments.

Basic Developer Documentation (Josh)

Documents that teach developers how to get started with building on EOS.IO blockchains. This includes documentations of the WASM API, RPC Interface, and Command Line Tools.

Phase 2 - Minimal Viable Test Network - Fall 2017

Everything in Phase 1 assumes a trusted environment that only runs the developer's own code. Before a test network can be deployed several additional features need to be implemented and tested.

P2P Network Code (Phil)

This is a plugin that is responsible for synchronizing the blockchain state between two standalone nodes.

WASM Sanitation & CPU Sandboxing (Brian)

The WASM code needs to be sanitized to check for non-deterministic behavior such as floating point operations and infinite loops.

Resource Usage Tracking & Rate Limiting (Arhag)

To prevent abuse the resource monitoring and usage tracking rate limits users accoding to staked EOS.

Genesis Import Testing (DappHub)

Tools need to be developed to export data from the EOS Token Distribution state and create a genesis configuration file. This will enable anyone participating in the Token Distribution to acquire some initial test EOS (TEOS).

Interblockchain Communication (Nathan)

This feature involves verifying the Merkle hashing of transactions is proper.

Phase 3 - Testing & Security Audits - Winter 2017, Spring 2018

During this phase the platform will undergo heavy testing with a focus on finding security issues and bug. At the end of Phase 3 version 1.0 will be tagged.

Develop Example Applications

Example applications are critical to proving the platform provides the features required by real developers.

Bounties for Succesfully Attacking Network

Attacking the network with spam, virtual machine exploits, and bug crashes, and non-deterministic behavior will be a heavily involved process but necessary to ensure that version 1.0 is stable.

Language Support

Adding support for additional langauges to be compiled to WASM: C++, Rust, etc.

Documentation & Tutorials

Phase 4 - Parallel Optimization Summer / Fall 2018

After getting a stable 1.0 product released, we will move toward optimizing the code for parallel execution.

Phase 5 - Cluster Implementation The Future