See-Thru Paper

A readable Specification for Ethereum

Micah Dameron

November 12, 2017

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.

The Zen of Python

Executive Summary

The goal of this paper is threefold. First, we aim to clarify concepts from the Yellowpaper that are not given in the Yellowpaper in sufficient detail. Second, we propose a formally correct pseudocode to replace the Yellowpaper's formal math. Last, we aim to give an open-source specification for Ethereum that the developer community can learn, use, and grow from, without needing a Ph.D. and a lot of spare time to understand.

Acknowledgements

Thanks to Kirk Dameron for bringing to bear the principles of project management on this endeavor. Thanks to John Mardlin for giving the paper new direction from its start. Thanks to Mike Goldin for reviewing and explaining pseudocode. Thanks to John Packel for his review, and thanks to Chris Lundkvist and Daniel Ellison for joining the next round of reviews.

Contents

1 Introduction

Over the past decade, blockchain technology has proven its longevity and veracity through a number of systems, most notably through Bitcoin, the first electronic currency of its kind to succeed. While the **Bitcoin Protocol** has been successful in its endeavor to create currency using Blockchain technology, we propose the **Ethereum Protocol** which expands on this simple *currency application* application by creating a *globally-distributed virtual machine* with shared memory using Blockchain technology. Relying on concepts from parallel computing, such as:

1. `message-passing`
2. `shared-state concurrence` and
3. `transaction processing`

the **Ethereum Protocol** fosters an environment where software developers are able to precisely execute machine instructions that have the same level of veridicity and certainty as do the financial transactions on other secure blockchains like Bitcoin.

2 FAQ

Q: Who Is This Paper For?

This paper is for anyone who wants a readable specification for Ethereum.

Q: Why is The Ethereum Virtual Machine (EVM) transaction based?

A: Since resources on The EVM are limited, they must have a computational cost associated with them. If computational costs are transactional, then they will not go through even partially unless the transaction goes through fully.

Q: Bitcoin doesn't have a formal specification, why does Ethereum need one?

A: The goal of Bitcoin was to create a decentralized payment network; at that it has succeeded marvelously. The goal of Ethereum is to create a decentralized computer. A formal specification allows developers to build new clients that aren't part of the core wallet. This keeps there from being an unofficial monopoly on the protocol by those who are able to manipulate it.

Q: Is there a good way to define Ethereum in a few words?

A: Yes:

"Ethereum is a transaction-based virtual machine, following a singleton design pattern and adhering to a shared state."

Don't worry about understanding all of that yet. It will be unpacked and precisely explained throughout the paper.

Q: How is this paper organized?

A: The main body aims to reproduce Yellowpaper content in a more readable form, while the appendices critique Yellowpaper content in its original form.

3 Key Terms and Concepts

3

This section Key Terms section aims to cover the highest-level and most important concepts you need to know to understand Ethereum. Each of the following sections breaks down the EVM one of these abstractions at a time.

A few key terms: state, state-transition, state, transaction, state machine

Ethereum is a transaction-based state machine. A state-machine can only be in one of a set number of configurations depending on its inputs. Picture a gearshift, and picture each

change in gears as a change of the machine's (car/truck's) state. Picture the hand on the gearshift as the input. Since all of the possible states in Ethereum are determined beforehand, impossible states are excluded necessarily.

A state is a particular static configuration that Ethereum is in. In order to affect the state, valid transaction(s) must occur. Only committed transactions change the state, and committed transactions must be mined.[a] Transactions which throw an error, whether an out of gas error or some other error, are transactions that have failed in their execution and thus fail to initiate a valid **state transition**. A valid state transition is when the hand applies enough energy to the gearshift to push it into another gear and make it switch. An invalid state transition, like say, out of gas errors, would be if the hand didn't have enough energy in its push to get it into the next gear. Alternatively, the hand might try pushing it toward a gear its already in, or towards a gear that doesn't exist (bad jump dest) and fail to change the state. Ethereum is a turing-complete abstract model, that excludes these things by definition. It would be as unlikely for someone to hack Ethereum and make counterfeit Ether as it would that your car would pop out another gear and behave autonomously like something out of a Steven King novel. It's just not in Ethereum's instructions to do that. State machines are limited for this exact reason.

State Machine

priorstate The state immediately preceding the current state.

presentstate The present state of the Blockchain i.e. now.

poststate The state immediately following the current state.

transaction An atomic operation; a message, data modification, or other procedure that is guaranteed to perform completely or not at all.

state-transition The process occurring between two inert states that constitutes the creation of a new state.

state-transition[ing] function One of several possible functions that took Ethereum from a priorstate to a presentstate or takes Ethereum from a presentstate to a poststate.

state-machine A state machine rests in a universal, stable, singular condition, called a state. State machines can transition to new states given certain compatible inputs.

state As a whole, the state is the sum total of database relationships in the **state database**. The state is an inert position on the chain, a position between prior state and post state; a block's frame of reference, and a defined set of relationships to that frame of reference.

More key content to come...Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is

[a] The Yellowpaper denotes the concept of Ethereum's state with a Greek symbol; a lowercase Sigma:σ

no need for special content, but the length of words should match the language.

4 Transactions

2

4.1 Message Calls

2

4.2 Contract Creation

5 Gas

2

5.1 Gasprice

5.2 Gaslimit

Any unused gas is refunded to the user.

5.3 Gasused

5.4 Execution

5.5 Execution Model

2

5.6 Execution Environment

5.7 State Machines

2

5.8 Miner Choice

Miners choose which gas prices they want to accept.

6 States

6.1 Substate

7 State Transitions

8 Currency

The smallest unit of currency in Ethereum is the Wei, which is $1 * 10^18$ Ether. All units at the machine level are counted in Wei.

9 Machine State

10 Account Creation

11 Ethash

11.1 GHOST Protocol

12 A message call

13 RLP

13.1 Well-Formed RLP

14 Blocks

14.1 Transaction Receipts

14.2 Nonce

14.3 Ommers

15 OPCODES

15.1 Lower-Level Lisp

15.2 Solidity

15.3 EVM Code

15.4 Formal Signs

16 Contract Creation

17 Bit Sequences

18 ECDSA

18.1 Public Keys

18.2 Private Keys

19 Byte Arrays

20 Apply Rewards

21 Verification

Verifies Ommer headers

21.1 Ommershash

21.2 Beneficiary

21.3 Bloom Filter

21.4 isSibling Property

21.5 Mining

22 Halting

23 World State

The MAPPING of **addresses** and **account states** (RLP data structures), this is also known as *state*, or σ. This mapping is not stored on the blockchain, rather it is stored as a Merkle-Patricia trie in a DATABASE BACKEND[a] that maintains a mapping of bytearrays to bytearrays.[b] The cryptographic internal data going back to the root node represents the *State* of the Blockchain at any given root, i.e. at any given *time*.[c]

[a] A database backend is accessed by users indirectly through an external application, most likely an Ethereum client; see also: state database

[b] A bytearray is specific set of bytes [data] that can be loaded into memory. It is a structure for storing binary data, e.g. the contents of a file.

[c] This permanent data structure makes it possible to easily recall any previous state with its root hash keeping the resources off-chain and minimizing on-chain storage needs.

[a] Formally **world state** $= \sigma$.

24 Account State

The account state is the state of any particular account during some specified world state σ.[a]

24.1 Nonce

The **nonce** aspect of an ACCOUNT'S STATE is the number of transactions sent from, or the number of contract-creations by, the address of that account.[b]

24.2 Storage Root

The **storage root** aspect of an ACCOUNT'S STATE is the hash of the trie[c]

24.3 Code Hash

The **code hash** aspect of an ACCOUNT'S STATE is the HASH OF THE EVM CODE of this account. Code hashes are STORED in the **state database**. Code hashes are permanent and they are executed when the address belonging to that account RECEIVES a message call.[def]

24.4 Balance

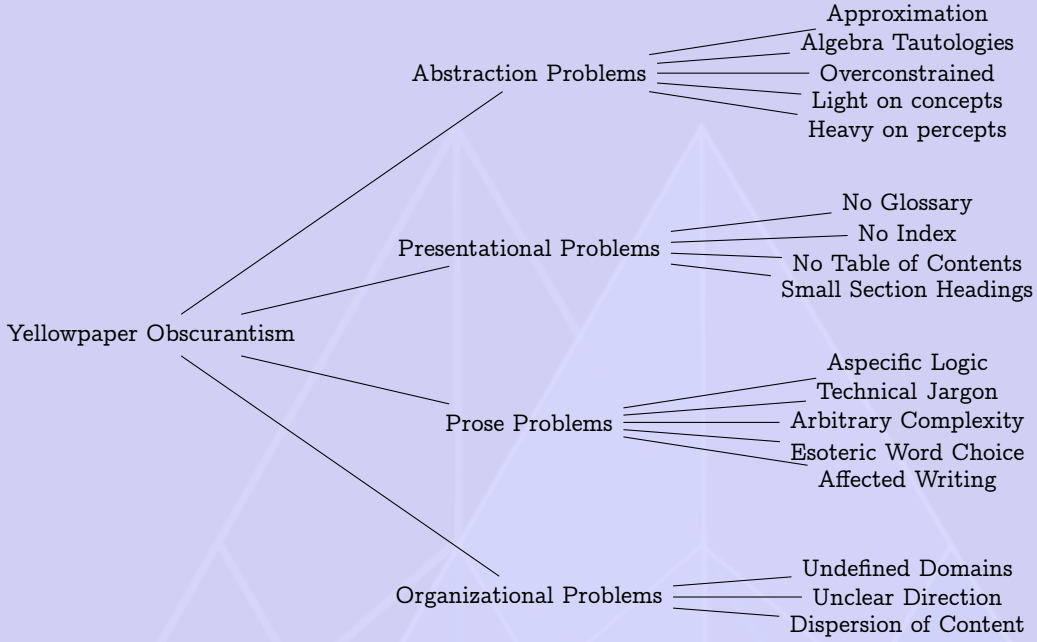
The amount of **Wei** OWNED by this account. [g]

- Key/value pair stored inside the root hash. [h]
- L_I^*, is defined as the element-wise transformation of the base function[i]
- The *element-wise transformation of the base-function* refers to all of the key/value pairs in L_I
- L_I refers to a particular trie.

[b] σ is the world state at a certain given time, and *n* is the number of transactions or contract creations by that account.
[c] A particular path from root to leaf in the **state database** that encodes the STORAGE CONTENTS of the account.
[d] A message call is any interaction with the account on-chain.
[e] Formal notation is $\sigma[a]_c$
[f] More formal notation is $\texttt{KEC}(\mathbf{b}) = \boldsymbol{\sigma}[a]_c$
[g] This formal notation is $\sigma[a]_b$
[h] $\texttt{TRIE}\big(L_I^*(\boldsymbol{\sigma}[a]_\mathbf{s})\big) \equiv \boldsymbol{\sigma}[a]_s$
[i] L_I, given as: $L_I\big((k,v)\big) \equiv \big(\texttt{KEC}(k), \texttt{RLP}(v)\big)$

.1 What problems did the original Yellowpaper have?

There are too many to list them all individually, but this map shows the general ideas:



.2 Symbol Key

Sign	Meaning
σ	world state
Υ	state transition function
Π	block-level state transition function
Ω	block-finalization state transition function
T	substitutes any given transaction
x_t	the transaction $_t$ corresponding to x
$\equiv$	is equivalent to
B	substitutes any given block
( )	contain symbols that are logically related
...	a defined pattern in a known function

Glossary

abstract . 5, 11

abstract machine An abstract machine is a conceptual model of a computer that describes its own operations with perfect accuracy. Since abstract machines are theoretical, all possible outputs can be determined beforehand. 11

Address A 160-bit (20-byte) code used for identifying Accounts. 11

addresses 160-bit (20-byte) identifiers–the last 20 characters of an Ethereum address. 11

Autonomous Object A notional object existent only within the hypothetical state of Ethereum. Has an intrinsic address and thus an associated account; the account will have non-empty associated EVM Code. Incorporated only as the Storage State of that account. 11

Balance A value which is intrinsic to accounts; the quantity of Wei in the account. All EVM operations are associated with changes in account balance. 11

beneficiary The 160-bit address to which all fees collected from the successful mining of this block be transferred. 11

Bit The smallest unit of electronic data storage: there are eight bits in one byte. The Yellowpaper gives certain values in bits (e.g. 160 bits instead of 20 bytes). 11

block header The information in a block besides transaction information. It consists of several elements: the, the, , , , , and , , the, and ,. 11

blockchain A consensus-based record of agreement where chunks of data[j] (called blocks) are stored with cryptographic hashes linking each Block to the next, ensuring veracity. 11

Contract A piece of EVM Code that may be associated with an Account or an Autonomous Object. 11

Dapp An end-user-visible application hosted in an Ethereum browser.. 11

design pattern a pattern of design in OOP. 11

difficulty A scalar value corresponding to the difficulty level of a current block. This can be calculated from the previous block's difficulty level and the timestamp; formally H_d. 11

Ethereum Runtime Environment The environment which is provided to an Autonomous Object executing in the EVM. Includes the EVM but also the structure of the world state on which the relies for certain I/O instructions including CALL & CREATE. 11

Ethereum Browser [k] A cross-platform GUI of an interface similar to a simplified browser (a la Chrome) that is able to host applications, the backend of which is purely on the Ethereum protocol.. 11

Ethereum Foundation The non-profit organization in charge of executing the development processes of Ethereum in line with the Whitepaper. 11

[j] As of November 12, 2017, roughly between 1 and 20 kilobytes in size[Etherscan2017]

[k] a.k.a. Ethereum Reference Client

Ethereum Virtual Machine A secure, consensus-based global computer, and the subject of this paper. 11

EVM Assembly The human readable version of EVM code. 11

EVM Code The bytecode that the EVM can natively execute. Used to formally specify the meaning and ramifications of a message to an Account. 11

External Actor A person or other entity able to interface to an Ethereum node, but external to the world of Ethereum. It can interact with Ethereum through depositing signed Transactions and inspecting the blockchain and associated state. Has one (or more) intrinsic Accounts. 11

Gas The fundamental network cost unit. Paid for exclusively by Ether (as of PoC-4), which is converted freely to and from Gas as required. Gas does not exist outside of the internal Ethereum computation engine; its price is set by the Transaction and miners are free to ignore Transactions whose Gas price is too low. 11

gas used A scalar value equal to the total gas used in transactions in this block; formally H_g. 11

gas limit A scalar value equal to the current limit of gas expenditure per block; formally H_l. 11

hacker ethic A maxim purporting that knowledge about systems should be free and unhindered[Levy2010]. 11

leaf node the bottom-most node in a particular tree, of blocks, one half of the "key" the other half being the root node, which creates the path between. 11

logs bloom The Bloom filter composed from indexable information (logger address and log topics) contained in each log entry from the receipt of each transaction in the transactions list; formally H_b.. 11

Lower-Level Lisp The Lisp-like Low-level Language, a human-writable language used for authoring simple contracts and general low-level language toolkit for trans-compiling to. 11

Message Data (as a set of bytes) and Value (specified in Wei) that is passed between two Accounts, either through the deterministic operation of an Autonomous Object or the cryptographically secure signature of the Transaction. 11

Message The act of passing a message from one Account to another. If the destination account is associated with non-empty EVM Code, then the VM will be started with the state of said Object and the Message acted upon. If the message sender is an Autonomous Object, then the Call passes any data returned from the VM operation. 11

mixhash a 256-bit hash which proves, combined with the nonce, that a sufficient amount of computation has been carried out on this block; formally H_m.. 11

nonce A value equal to the number of transactions sent from this address or the number of contract creations made by this account. a 64-bit hash which proves, combined with the mix-hash, that a sufficient amount of computation has been carried out on this block; formally H_m. 11

number A scalar value equal to the number of ancestor blocks. The genesis block has a number of zero; formallyH_i.. 11

Object Synonym for Autonomous Object. 11

ommers hash The Keccak 256-bit hash of the ommers list portion of this block; formally H_o. 11

parent hash The Keccak 256-bit hash of the parent block's header, in its entirety; formally H_p. 11

public key A term originating from *cryptography* and corresponding to **private key**, this is the 42-byte (i.e. 42-character) string of ASCII digits which transacts on the public network. 11

receipts root The Keccak 256-bit hash of the root node of the true structure populated with the receipts of each transaction in the transactions list portion of the block; formally H_e. 11

root node the uppermost node in a particular tree, of blocks, representing a single world state$^{\sigma}$ at a particular time. 8, 11

serialization Serialization is the process of converting an object into a stream of bytes in order to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called deserialization.[billwagner]. 11

singleton A design pattern in Object-Oriented Programming which specifies a class with one instance but with a global point of access to it[sourcemaking]. 11

specification Technical descriptions, instructions, and definitions from which other people can create working prototypes. 4, 11

state A permanent, static, state H_m.. 4, 11

state root The Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalizations applied; formally H_r. 11

State Database A database backend that maintains a mapping of bytearrays to bytearrays. 11

state database A database stored off-chain, [i.e. on the computer of some user running an Ethereum client] which contains a trie structure mapping bytearrays [i.e. organized chunks of binary data] to other bytearrays [other organized chunks of binary data]. The RELATIONSHIPS between each node on this trie constitute a MAP, a.k.a. a MAPPING of all PREVIOUS STATES on the **EVM** which a client might need to reference. 5, 8, 9, 11

state machine A state machine rests in a universal, stable, singular condition, called a state. State machines transition to new states given certain compatible inputs.. 4, 11

state-transition . 4, 11

Storage State The information particular to a given Account that is maintained between the times that the Account's associated EVM Code runs. 11

timestamp A scalar value equal to the reasonable output of Unix's time() at this block's inception; formally H_s. 11

transaction An input message to a system that, because of the nature of the real-world event or activity it reflects, requires to be regarded as a single unit of work and must either be processed completely or rejected[Ngondi2016]. 4, 5, 11

Transaction A piece of data, signed by an External Actor. It represents either a Message

or a new Autonomous Object. Transactions are recorded into each block of the blockchain. 11

transactions root The Keccak 256-bit hash of the root node of the true structure populated with each transaction in the transactions list portion of the block; formally H_t. 11

trie A tree-structure for organizing data, the position of data in the tree contains the particular path from root to leaf node that represents the key (the path from root to leaf is "one" key) you are searching the trie structure for. The data of the key is contained in the trie relationships that emerge from related nodes in the trie structure. 8, 9, 11

turing-complete . 5, 11

Whitepaper A conceptual map, distinct from the Yellowpaper, which highlights the development goals for Ethereum as a whole[EF2017]. 11

Yellowpaper Ethereum's first formal specification,[Wood2017] written by Dr. Gavin Wood, one of the founders of Ethereum. Notorious for its difficulty to read,[reddit] it provides both a boon and an obstacle to potential adopters of Ethereum. 11

Acronyms

ERE Ethereum Runtime Environment. 11

EVM Ethereum Virtual Machine. 4, 11

LLL Lower Level Lisp. 11

OOP Object-Oriented Programming. 11

YP Yellowpaper. 11