EOS.IO and TRON

History and Background

Blockchain technology was introduced in 2008 with the launch of the Bitcoin currency, and since then entrepreneurs and developers have attempted to generalize the technology to support a wider range of applications on a single blockchain platform.

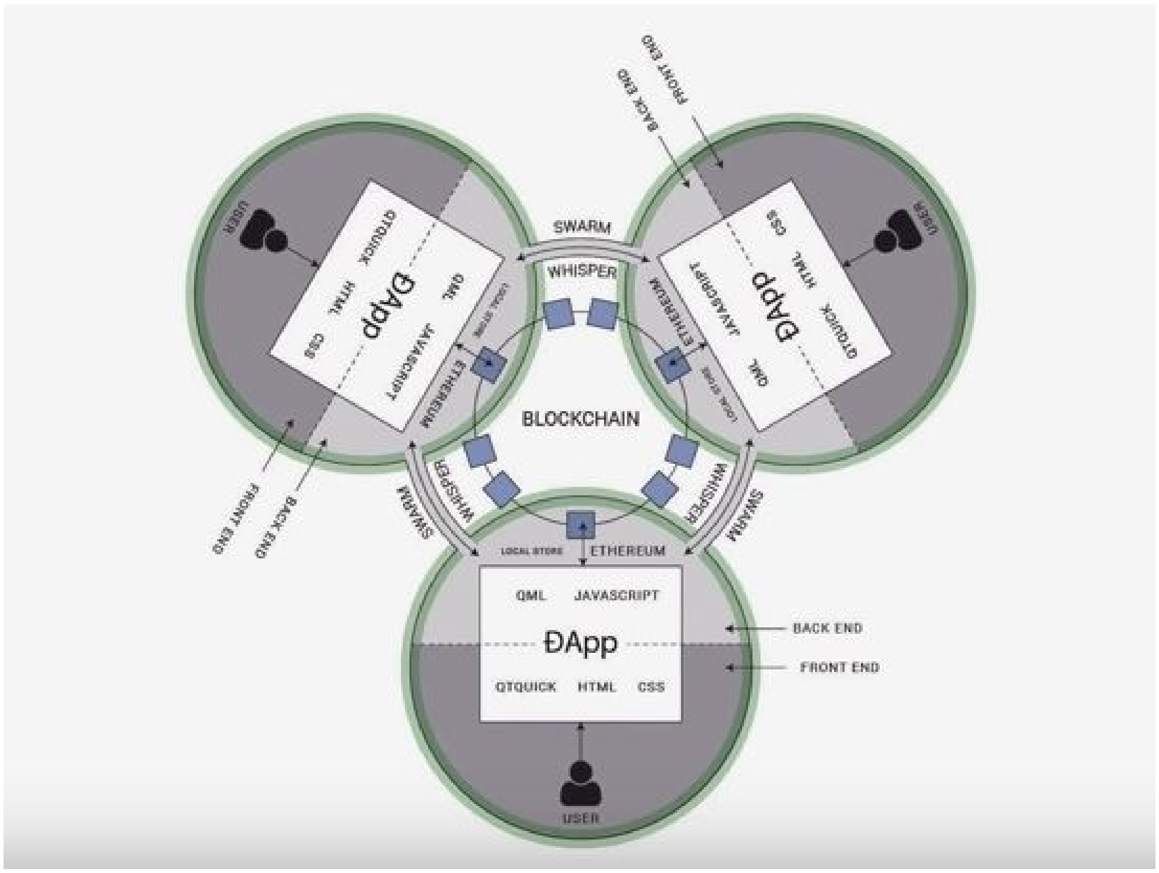
While a number of blockchain platforms have struggled to support functional decentralized applications, application specific blockchains such as the BitShares decentralized exchange (2014) and Steem social media platform (2016) have become heavily used blockchains with tens of thousands of daily active users. They have achieved this by increasing performance to thousands of transactions per second, reducing latency to 1.5 seconds, eliminating per-transaction fees, and providing a user experience similar to those currently provided by existing centralized services.

Existing blockchain platforms are burdened by large fees and limited computational capacity that prevent widespread blockchain adoption.

EOS and EOS.IO

EOS, by block.one, is a blockchain-based, decentralized operating system, designed to support commercial-scale decentralized applications by supplying each of the essential core performance (including databases, accounts with permissions, scheduling, authentication, and handling communication between the program and the internet), thus allowing developers to focus on their own business process.

EOS.IO is a software that allows companies to construct blockchain programs that resemble present online software, using an architecture like website frameworks. It emulates most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk storage) with the computing resources distributed equally among EOS cryptocurrency holders. EOS.IO operates as a smart contract platform and decentralized operating system intended for the deployment of industrial-scale decentralized applications through a decentralized autonomous corporation model. The smart contract platform claims to eliminate transaction fees and also conduct millions of transactions per second.



[1] DAPP Arhtecture

EOS.IO Features and comparison to other Blockchains

What do DAPPs require?

- **Support For Millions of Users**

It should be scalable enough for millions of users to use it. This is especially true for D[1]APPs that are looking for mainstream acceptance.

[1] [1]https://i.stack.imgur.com/jzm8y.png

- **Free Usage**

The platform should enable the devs to create dapps which are free to use for their users. No user should have to pay the platform to gain the benefits of a dapp.

- **Easily Upgradable**

The platform should allow the developers the freedom to upgrade the dapp as and when they want. Also, if some bug does affect the DAPP, the devs should be able to fix the DAPP without affecting the platform.

- **Parallel Performance**

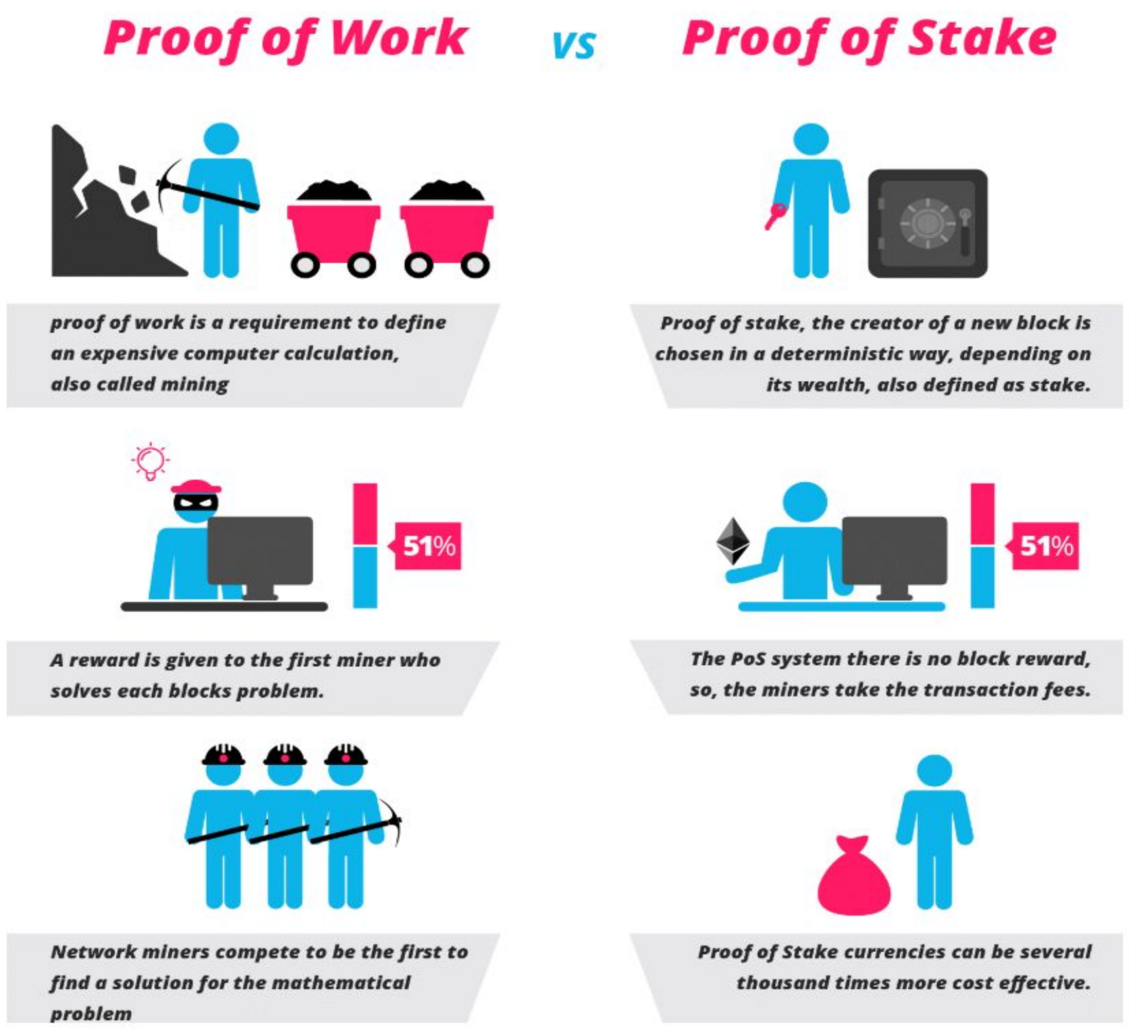
A platform should allow their DAPPS to be processed parallelly in order to distribute the workload and save up time.

- **Sequential Performance**

However, not all the functions on a blockchain should be done that way. Think of transaction execution itself. Multiple transactions can't be executed in parallel; it needs to be done one at a time to avoid errors like double spends.

- **Low Latency**

A DAPP should run as smoothly as possible and with the lowest possible latency.


Proof of Work
vs
Proof of Stake
proof of work is a requirement to define an expensive computer calculation, also called mining
Proof of stake, the creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as stake.
51%
51%
A reward is given to the first miner who solves each blocks problem.
The PoS system there is no block reward, so, the miners take the transaction fees.
Network miners compete to be the first to find a solution for the mathematical problem
Proof of Stake currencies can be several thousand times more cost effective.

[2]PoW and PoF

EOS.IO software utilizes the only known decentralized consensus algorithm proven capable of meeting the performance requirements of applications on the blockchain, Delegated Proof of Stake (DPOS). Under this algorithm, those who hold tokens on a blockchain adopting the EOS.IO software may select block producers through a continuous approval voting system. [2]Anyone may choose to participate in block production and will be

[2] [2]https://blockgeeks.com/guides/eos-blockchain/

given an opportunity to produce blocks, provided they can persuade token holders to vote for them.

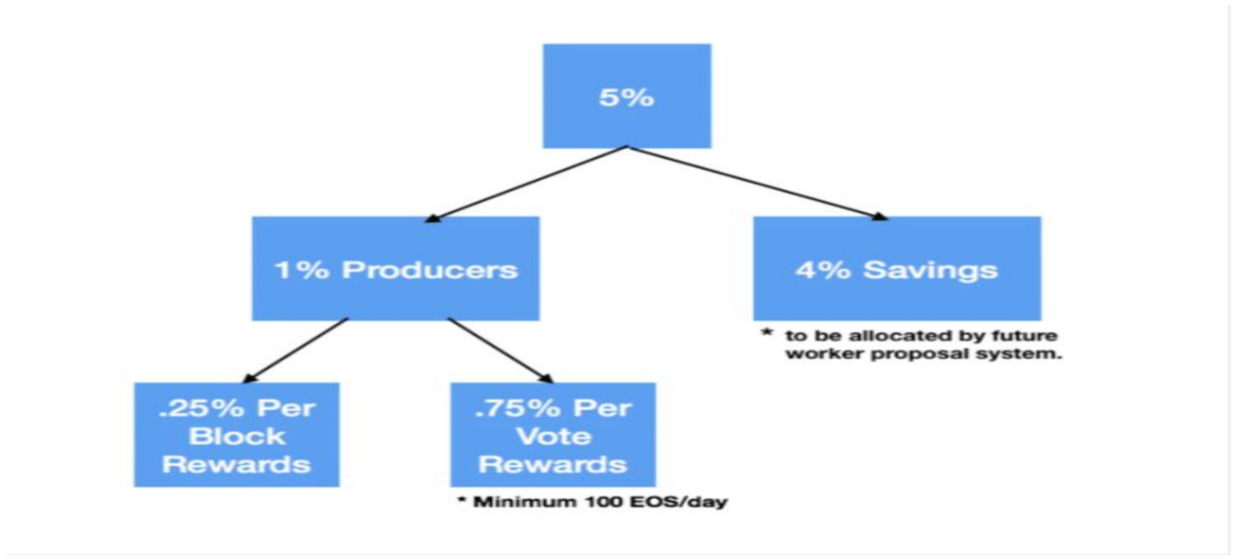
- Blocks are produced in the rounds of 21.
- At the start of every round 21 block producers are chosen. Top 20 are automatically chosen while the 21st one is chosen proportional to the number of their votes relative to the other producers.
- The producers are then shuffled around using a pseudorandom number derived from the block time. This is done to ensure that a balance connectivity to all other producers is maintained.
- To ensure that regular block production is maintained and that block time is kept to 3 seconds, producers are punished for not participating by being removed from consideration. A producer has to produce at least one block every 24 hours to be in consideration.

Eliminating Transaction Fees

EOS works on an ownership model whereby users own and are entitled to use resources proportional to their stake, rather than having to pay for every transaction. So, in essence, if you hold N tokens of EOS then you are entitled to N*k transactions. This, in essence, eliminates transaction fees.

The costs of running and hosting applications on Ethereum can be high for a developer who wants to test their application on the blockchain. The gas price involved in the early stages of development can be enough to turn off new developers.

The fundamental difference between the way Ethereum and EOS operate is that while Ethereum rents out their computational power to the developers, EOS gives ownership of their resources. So, in essence, if you own 1/1000th of the stake in EOS then you will have ownership of 1/1000th of the total computational power and resources in EOS.



[3]

Architecture

EOS is a collection of applications that interact within a distributed database structure. The software system is comprised of three primary components: Nodeos, Cleos, and Keosd. Users interact with the network [3]

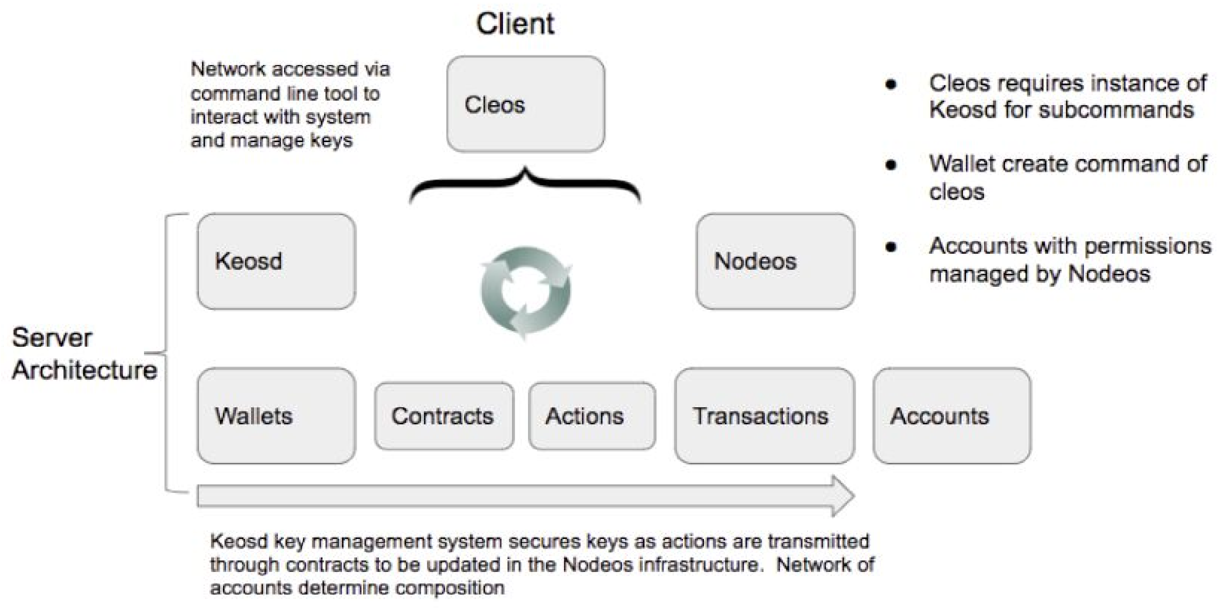
[3] [3]https://www.pinterest.com/pin/382031980886313013/

through the command line tool, Cleos, which subsequently connects through Nodeos to the rest of the database management system (DBMS).

Nodeos: The core component of the node setup is configured as a daemon using various plugins that allow the node to run. Nodes can either be producing nodes (block producers) or non-producing nodes that are only affiliated with an account. Non-producing nodes can be implemented as a public HTTP-RPC API for developers. These nodes can also act as private endpoints for apps that wish to utilize the computational infrastructure for local development.

Cleos: Clients can access the EOS network via the Cleos command line interface (CLI) tool. This tool allows interaction with the overall network and configuration of interactions within the network through Nodeos. The block producers in the network host,deliver, and manage most of the resources consumed by the clients, allowing them to directly access information about transactions.

Keosd: This component of the system is designed to store private keys that are used to sign transactions within the network. Keosd is run locally and stores keys locally. Users are able to use and query the keys through its RPC API.


Client
Network accessed via command line tool to interact with system and manage keys
Cleos
Cleos requires instance of Keosd for subcommands
Wallet create command of cleos
Accounts with permissions managed by Nodeos
Keosd
Nodeos
Server Architecture
Wallets
Contracts
Actions
Transactions
Accounts
Keosd key management system secures keys as actions are transmitted through contracts to be updated in the Nodeos infrastructure. Network of accounts determine composition

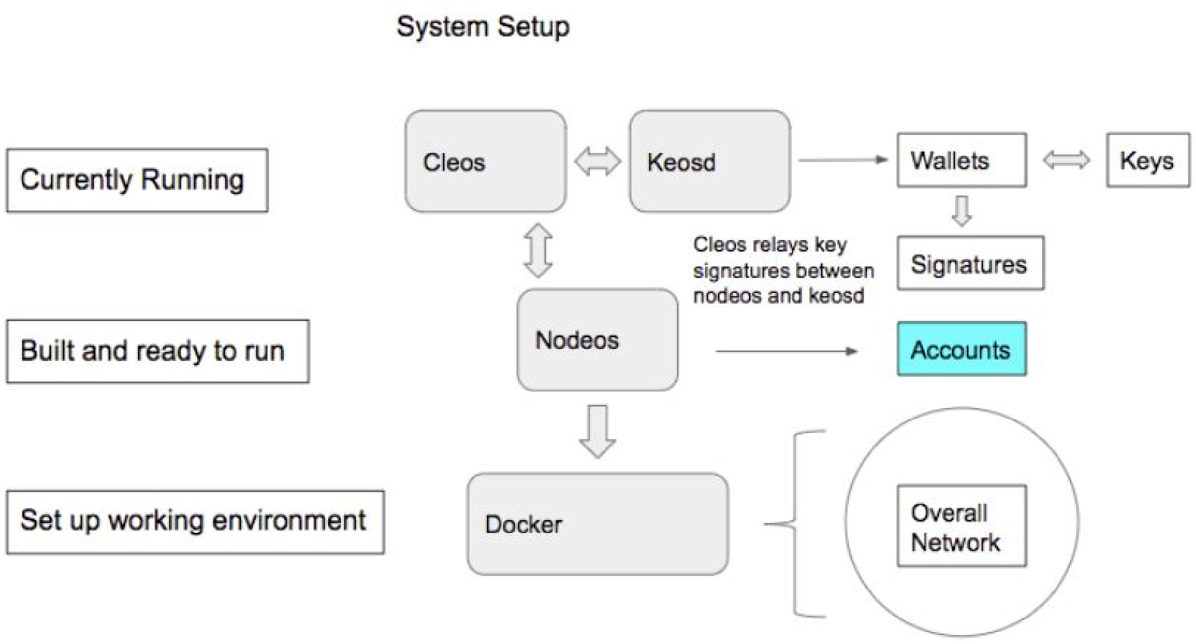
[4]

Accounts

In the EOS architecture, accounts are presented as authorization structures that can define senders and receivers of actions. Hierarchical authorization can be granted through permissions management and contracts. Accounts can be granted permissions and configured to provide individual or group access for validating transactions [9]. They act as the base-unit data modules that allow sending and receiving of transactions in the system. [4]

[5]Accounts are used to execute contracts by sending structured actions to block producers. These actions are routed through the rest of the network in order to reach the destination accounts

[4] [4]https://www.whiteblock.io/library/eos-test-report.pdf
[5][5] https://www.whiteblock.io/library/eos-test-report.pdf



[5]

Transactions and Contracts

The way that transactions and contracts interact with a blockchain system defines certain levels of robustness that exist within the infrastructure.

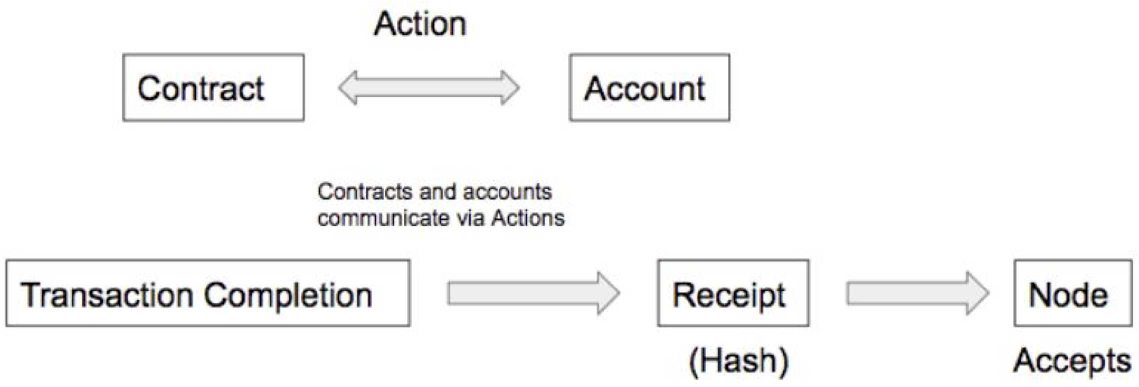
Contracts: Smart contracts in EOS are defined by a combination of actions and action handlers. Action handlers are designed to access different accounts through sending actions that can interact with the private databases affiliated with each account. Block producers schedule transactions in order to optimize conflicts over memory and resource intensive actions within the network.

WASM is the instruction format that the EOS virtual machine understands, meaning that the interaction between the EOS library and WASM binary code happens through Web Assembly modules.

Actions: These are the single atomic units of operation. Actions represent the communication interface between contracts and accounts. Most of the EOS network is built using actions that interact with each other and form the the underlying architecture.

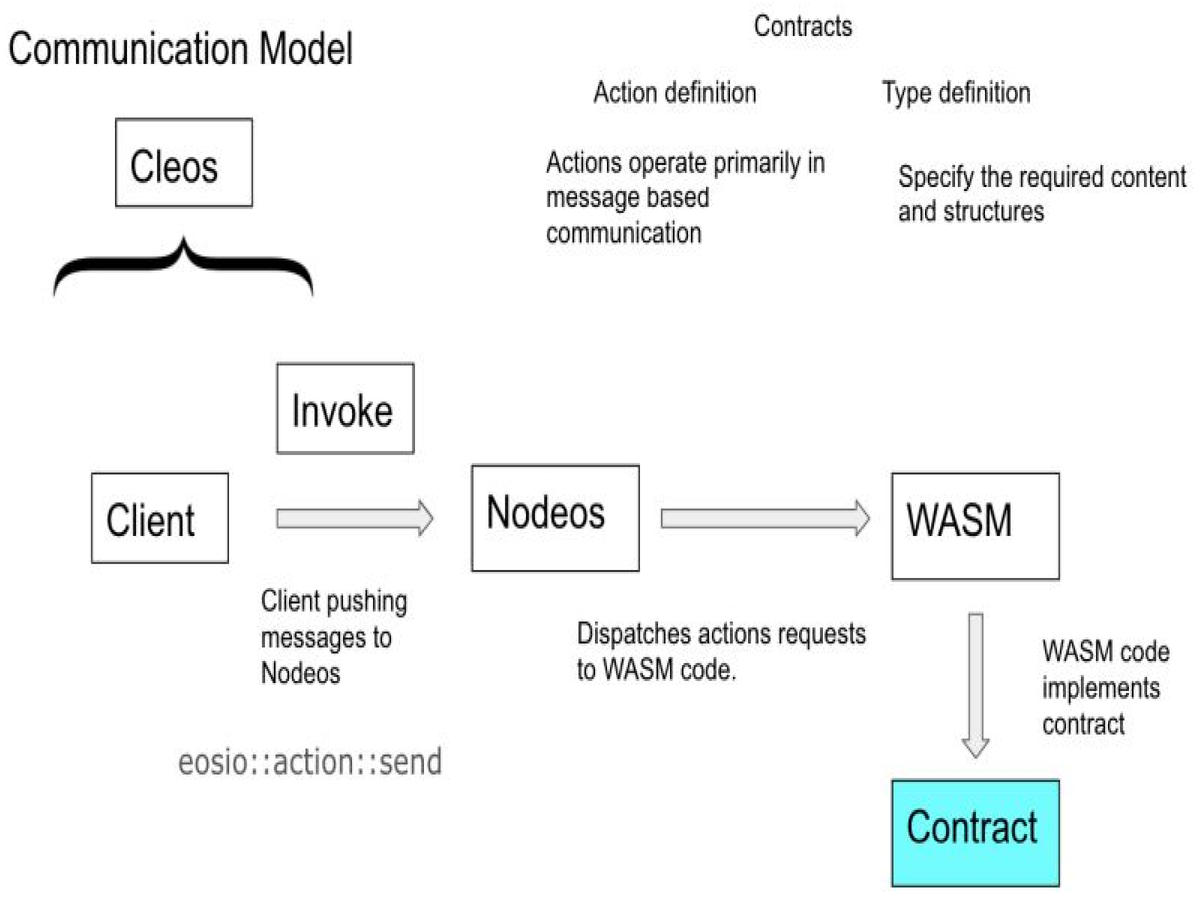
Transactions: Transactions represent the execution of actions. Transactions can consist of individual actions or multiple actions. The successful execution of transactions dictate the change of state in the EOS blockchain.

[6]



[6]

[6] [6]https://www.whiteblock.io/library/eos-test-report.pdf

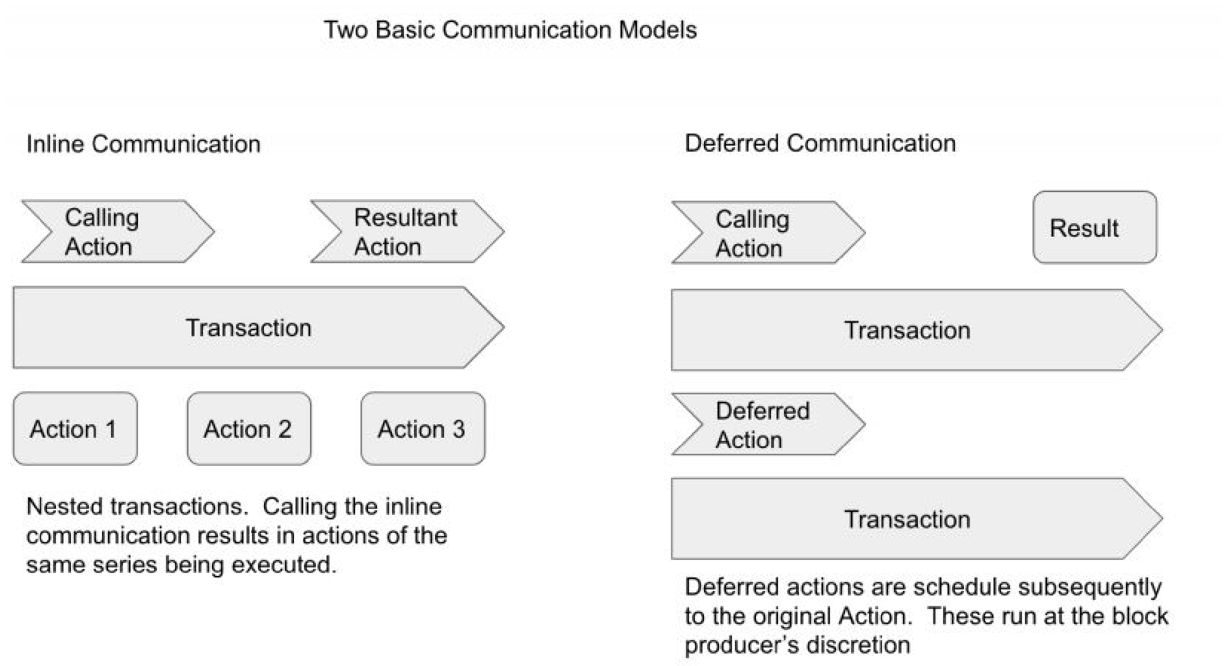


Communication Model
Cleos
Invoke
Client
Nodeos
WASM
Contract
Client pushing messages to Nodeos
eosio::action::send
Dispatches actions requests to WASM code.
WASM code implements contract
Contracts
Action definition
Type definition
Actions operate primarily in message based communication
Specify the required content and structures

[7]

7

[7] [7]https://www.whiteblock.io/library/eos-test-report.pdf


Two Basic Communication Models
Inline Communication
Calling Action
Resultant Action
Transaction
Action 1
Action 2
Action 3
Nested transactions. Calling the inline communication results in actions of the same series being executed.
Deferred Communication
Calling Action
Result
Transaction
Deferred Action
Transaction
Deferred actions are schedule subsequently to the original Action. These run at the block producer's discretion

[8]

TRON[8]

General Overview

The TRON Protocol, one of the largest blockchain based operating systems in the world, offers base public blockchain support of high throughput, high scalability, and high availability for all decentralized applications in the TRON ecosystem.

TRX is the name of the currency used in the TRON network. TRX can be used to vote for super representatives and obtain bandwidth. Freezing the

[8] [8]https://www.whiteblock.io/library/eos-test-report.pdf

TRX balance in a wallet gives the user TRON Power (TP), which is used to vote for Super Representatives (SRs). To keep the network operating smoothly, TRON network only allows every account to initiate a transaction for free once every 10 seconds.

Architecture

TRON adopts a 3-layer architecture comprised of storage layer, core layer, and application layer.

- **Storage Layer**

The tech team of TRON designed a unique distributed storage protocol consisting of block storage and state storage.
The notion of a graph database was introduced into the design of the storage layer to better meet the need for diversified data storage in the real world.
Blockchain Storage TRON blockchain storage chooses to use LevelDB, which is developed by Google and proven successful with many companies and projects. It has high performance and supports arbitrary byte arrays as both keys and values, singular get, put and delete, batched put and delete, bi-directional iterators, and simple compression using the very fast Snappy algorithm.
State Storage TRON has a KhaosDB in the full-node memory that can store all the newly forked chains generated within a certain period of time and supports witnesses to switch from their own active chain swiftly into a new main chain. It can also protect

blockchain storage by making it more stable from being terminating abnormally in an intermediate state.

- **Core Layer**

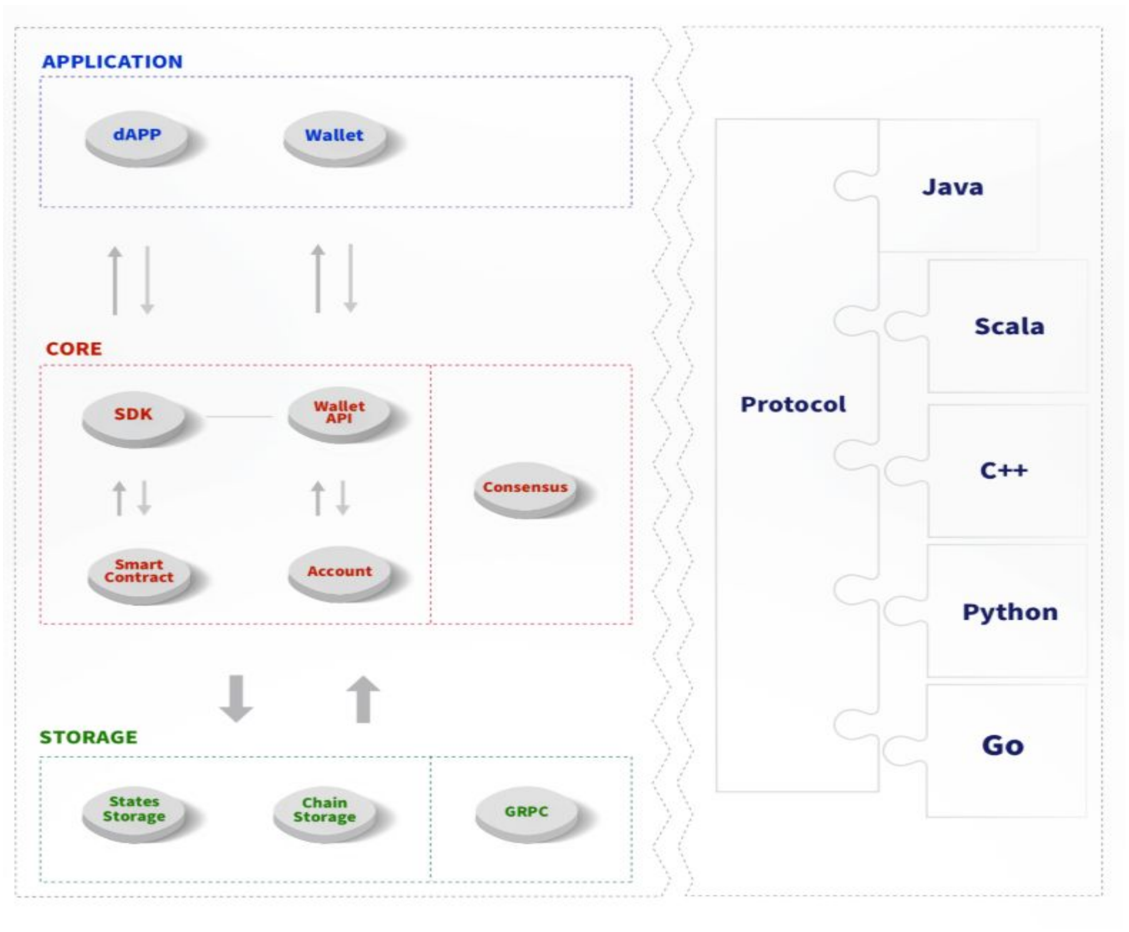
Smart contract module, account management module, and consensus module are core layer's three modules. It is TRON’s vision to base its functions on a stacked virtual machine and optimized instruction set. In order to better serve the development of DApps, Java is designated as the language for smart contracts, which is to be further supplemented by other high-level programming languages. In addition, innovations are made to TRON’s consensus on the basis of DPOS to fulfill its special needs.

- **Application Layer**

Developers can utilize interfaces for the creation of diverse DApps and customized wallets. The TRON protocol adheres in entirety to Google Protobuf, which intrinsically supports multi-language extension.

Node Types

The three types of nodes on Tron’s network are Super Representative Witness (SR Full Node), Full Node, and Solidity Node. SR Full Nodes are responsible for block production; Full Nodes provide APIs, and broadcast transactions and blocks; Solidity Nodes synchronize irrevocable blocks and provide inquiry APIs. Exchanges need to deploy a Full Node and a Solidity Node. The Solidity Node connects to the local Full Node, which connects to the Mainnet.



[9]

TRON Power (TP) is needed to vote and the amount of TP depends on the voter's frozen assets(TRX).

TP is calculated in the following way: 1 TP for 1 frozen TRX.

Every account in the TRON network has the right to vote for your own SRs.

After the release(unfreeze), you don't have any frozen assets and lose all TP [9] accordingly. As a result, all votes become invalid for the ongoing and future voting round unless to freeze again to vote.

[9] [9]https://developers.tron.network/docs/architecture

Note that the TRON network only records your most recent vote, which means that every new vote will negate all previous votes.

Token Holder Categories

1. **Super Representatives:** top 27 individuals among the 127 candidates, voted once every 6 hours. Super Representatives play a key role in governing the TRON community by ensuring basic functions, e.g. block generation and bookkeeping, and obtain corresponding earnings.
2. **Super Representative Candidates:** 127 individuals elected through voting by the entire token holder community. Votes are updated once every 6 hours.
3. **Token Holder:** Individual holding any amount of TRX.

Bandwidth Points

Scaling up any blockchain's network may lead to delays on transaction confirmation, as seen in the Ethereum and Bitcoin networks. To ensure smooth network operation, the TRON network grants every account a free pool of Bandwidth Points for free transactions every 24 hours. To engage in transactions more frequently requires freezing TRX for additional bandwidth points, or paying the fee in TRX. Transactions are transmitted and stored in the network in byte arrays. Bandwidth points consumed in a transaction equals the size of its byte array. If the length of a byte array is 200 then the transaction consumes 200 bandwidth points.

Energy

The creation and operation of a smart contract consume CPU resources. It takes time for smart contracts to operate in virtual machines (VMs), and the time consumed in the system is calculated in microseconds. CPU resources are consumed in energy, which means 1 Energy = 1 Microsecond (μs). If a contract takes 100 μs to execute in a VM, it needs to consume 100 Energy. The total CPU resources provided by the TRON network are 100,000,000,000 Energy within 24 hours.Energy can only be obtained by freezing the TRX. Energy obtained = the TRX frozen for gaining Energy / the total TRX frozen for gaining Energy in the entire network * 100,000,000,000, which is the equally-divided fixed Energy for all users based on the frozen TRX.

For example, suppose the total amount of TRX frozen for gaining Energy is 1,000,000,000 TRX in the current network, and one account freezes 1000 TRX, which is one-millionth of the total and equals 100,000 microseconds. If executing a contract takes 1000 microseconds, then the user can trigger the contract 100 times.

Bibliography

https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

https://blocksdecoded.com/what-is-eosio/

https://medium.com/quillhash/beginners-guide-to-eos-and-basic-smart-contract-7d61c41037bb

https://developers.tron.network/docs/getting-started

https://coinswitch.co/info/tron/what-is-tron