EOSIO / **Documentation**

Branch: master ▾ **Documentation** / TechnicalWhitePaper.md Find file Copy path

gleehokie Moved images to being hosted on GitHub a95c323 12 days ago

8 contributors

`467 lines (263 sloc)   50.3 KB`

EOS.IO Technical White Paper v2

March 16, 2018

Abstract: The EOS.IO software introduces a new blockchain architecture designed to enable vertical and horizontal scaling of decentralized applications. This is achieved by creating an operating system-like construct upon which applications can be built. The software provides accounts, authentication, databases, asynchronous communication, and the scheduling of applications across many of CPU cores or clusters. The resulting technology is a blockchain architecture that may ultimately scale to millions of transactions per second, eliminates user fees, and allows for quick and easy deployment and maintenance of decentralized applications, in the context of a governed blockchain.

PLEASE NOTE: CRYPTOGRAPHIC TOKENS REFERRED TO IN THIS WHITE PAPER REFER TO CRYPTOGRAPHIC TOKENS ON A LAUNCHED BLOCKCHAIN THAT ADOPTS THE EOS.IO SOFTWARE. THEY DO NOT REFER TO THE ERC-20 COMPATIBLE TOKENS BEING DISTRIBUTED ON THE ETHEREUM BLOCKCHAIN IN CONNECTION WITH THE EOS TOKEN DISTRIBUTION.

DISCLAIMER: This EOS.IO Technical White Paper v2 is for information purposes only. block.one does not guarantee the accuracy of or the conclusions reached in this white paper, and this white paper is provided “as is”. block.one does not make and expressly disclaims all representations and warranties, express, implied, statutory or otherwise, whatsoever, including, but not limited to: (i) warranties of merchantability, fitness for a particular purpose, suitability, usage, title or noninfringement; (ii) that the contents of this white paper are free from error; and (iii) that such contents will not infringe third-party rights. block.one and its affiliates shall have no liability for damages of any kind arising out of the use, reference to, or reliance on this white paper or any of the content contained herein, even if advised of the possibility of such damages. In no event will block.one or its affiliates be liable to any person or entity for any damages, losses, liabilities, costs or expenses of any kind, whether direct or indirect, consequential, compensatory, incidental, actual, exemplary, punitive or special for the use of, reference to, or reliance on this white paper or any of the content contained herein, including, without limitation, any loss of business, revenues, profits, data, use, goodwill or other intangible losses.

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.

Requirements for Blockchain Applications

In order to gain widespread use, applications on the blockchain require a platform that is flexible enough to meet the following requirements:

Support Millions of Users

Competing with businesses such as eBay, Uber, AirBnB, and Facebook, require blockchain technology capable of handling tens of millions of active daily users. In certain cases, an application may not work unless a critical mass of users is reached and therefore a platform that can handle very large numbers of users is paramount.

Free Usage

Application developers need the flexibility to offer users free services; users should not have to pay in order to use the platform or benefit from its services. A blockchain platform that is free to use for users will likely gain more widespread adoption. Developers and businesses can then create effective monetization strategies.

Easy Upgrades and Bug Recovery

Businesses building blockchain based applications need the flexibility to enhance their applications with new features. The platform must support software and smart contract upgrades.

All non-trivial software is subject to bugs, even with the most rigorous of formal verification. The platform must be robust enough to fix bugs when they inevitably occur.

Low Latency

A good user experience demands reliable feedback with a delay of no more than a few seconds. Longer delays frustrate users and make applications built on a blockchain less competitive with existing non-blockchain alternatives. The platform should support low latency of transactions.

Sequential Performance

There are some applications that just cannot be implemented with parallel algorithms due to sequentially dependent steps. Applications such as exchanges need enough sequential performance to handle high volumes. Therefore, the platform should support fast sequential performance.

Parallel Performance

Large scale applications need to divide the workload across multiple CPUs and computers.

Consensus Algorithm (BFT-DPOS)

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. Anyone may choose to participate in block production and will be given an opportunity to produce blocks, provided they can persuade token holders to vote for them.

The EOS.IO software enables blocks to be produced exactly every 0.5 second and exactly one producer is authorized to produce a block at any given point in time. If the block is not produced at the scheduled time, then the block for that time slot is skipped. When one or more blocks are skipped, there is a 0.5 or more second gap in the blockchain.

Using the EOS.IO software, blocks are produced in rounds of 126 (6 blocks each, times 21 producers). At the start of each round 21 unique block producers are chosen by preference of votes cast by token holders. The selected producers are scheduled in an order agreed upon by 15 or more producers.

If a producer misses a block and has not produced any block within the last 24 hours they are removed from consideration until they notify the blockchain of their intention to start producing blocks again. This ensures the network operates smoothly by minimizing the number of blocks missed by not scheduling producers who are proven to be unreliable.

Under normal conditions a DPOS blockchain does not experience any forks because, rather than compete, the block producers cooperate to produce blocks. In the event there is a fork, consensus will automatically switch to the longest chain. This method works because the rate at which blocks are added to a blockchain fork is directly correlated to the percentage of block producers that share the same consensus. In other words, a blockchain fork with more producers on it will grow in length faster than one with fewer producers, because the fork with more producers will experience fewer missed blocks.

Furthermore, no block producer should be producing blocks on two forks at the same time. A block producer caught doing this will likely be voted out. Cryptographic evidence of such double-production may also be used to automatically remove abusers.

Byzantine Fault Tolerance is added to traditional DPOS by allowing all producers to sign all blocks so long as no producer signs two blocks with the same timestamp or the same block height. Once 15 producers have signed a block the block is deemed irreversible. Any byzantine producer would have to generate cryptographic evidence of their treason by signing two blocks with the same timestamp or blockheight. Under this model a irreversible consensus should be reachable within 1 second.

Transaction Confirmation

Typical DPOS blockchains have 100% block producer participation. A transaction can be considered confirmed with 99.9% certainty after an average of 0.25 seconds from time of broadcast.

In addition to DPOS, EOS.IO adds asynchronous Byzantine Fault Tolerance (aBFT) for faster achievement of irreversibility. The aBFT algorithm provides 100% confirmation of irreversibility within 1 second.

Transaction as Proof of Stake (TaPoS)

The EOS.IO software requires every transaction to include part of the hash of a recent block header. This hash serves two purposes:

1. prevents a replay of a transaction on forks that do not include the referenced block; and
2. signals the network that a particular user and their stake are on a specific fork.

Over time all users end up directly confirming the blockchain which makes it difficult to forge counterfeit chains as the counterfeit would not be able to migrate transactions from the legitimate chain.

Accounts

The EOS.IO software permits all accounts to be referenced by a unique human readable name of up to 12 characters in length. The name is chosen by the creator of the account. The account creator must reserve the RAM required to store the new account until the new account stakes tokens to reserve its own RAM.

In a decentralized context, application developers will pay the nominal cost of account creation to sign up a new user. Traditional businesses already spend significant sums of money per customer they acquire in the form of advertising, free services, etc. The cost of funding a new blockchain account should be insignificant in comparison. Fortunately, there is no need to create accounts for users already signed up by another application.

Actions & Handlers

Each account can send structured Actions to other accounts and may define scripts to handle Actions when they are received. The EOS.IO software gives each account its own private database which can only be accessed by its own action handlers. Action handling scripts can also send Actions to other accounts. The combination of Actions and automated action handlers is how EOS.IO defines smart contracts.

To support parallel execution, each account can also define any number of scopes within their database. The block producers will schedule transaction in such a way that there is no conflict over memory access to scopes and therefore they can be executed in parallel.

Role Based Permission Management

Permission management involves determining whether or not an Action is properly authorized. The simplest form of permission management is checking that a transaction has the required signatures, but this implies that required signatures are already known. Generally, authority is bound to individuals or groups of individuals and is often compartmentalized. The EOS.IO software provides a declarative permission management system that gives accounts fine grained and high-level control over who can do what and when.

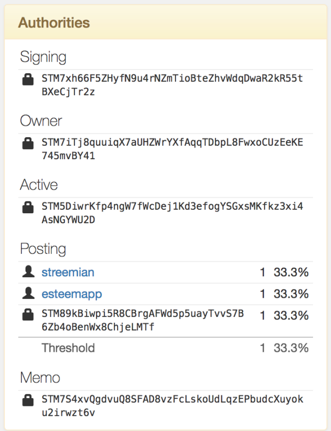
It is critical that authentication and permission management be standardized and separated from the business logic of the application. This enables tools to be developed to manage permissions in a general-purpose manner and also provide significant opportunities for performance optimization.

Every account may be controlled by any weighted combination of other accounts and private keys. This creates a hierarchical authority structure that reflects how permissions are organized in reality and makes multi-user control over accounts easier than ever. Multi-user control is the single biggest contributor to security, and, when used properly, it can greatly reduce the risk of theft due to hacking.

EOS.IO software allows accounts to define what combination of keys and/or accounts can send a particular Action type to another account. For example, it is possible to have one key for a user's social media account and another for access to the exchange. It is even possible to give other accounts permission to act on behalf of a user's account without assigning them keys.

Named Permission Levels

Using the EOS.IO software, accounts can define named permission levels each of which can be derived from higher level named permissions. Each named permission level defines an authority; an authority is a threshold multi-signature check consisting of keys and/or named permission levels of other accounts. For example, an account's "Friend" permission level can be set for an Action on the account to be controlled equally by any of the account's friends.



Authorities

Signing

🔒 STM7xh66F5ZHyfN9u4rNZmTioBteZhvWdqDwaR2kR55tBXeCjTr2z

Owner

🔒 STM7iTj8quuiqX7aUHZWrYXfAqqTDbpL8FwxoCUzEeKE745mvBY41

Active

🔒 STM5DiwrKfp4ngW7fWcDej1Kd3efogYSGxsMKfkz3xi4AsNGYWU2D

Posting

streemian	1	33.3%
esteemapp	1	33.3%
🔒 STM89kBiwpi5R8CBrgAFWd5p5uayTvvS7B6Zb4oBenWx8ChjeLMTf	1	33.3%
Threshold	1	33.3%

Memo

🔒 STM7S4xvQgdvuQ8SFAD8vzFcLskoUdLqzEPbudcXuyoku2irwzt6v

Another example is the Steem blockchain which has three hard-coded named permission levels: owner, active, and posting. The posting permission can only perform social actions such as voting and posting, while the active permission can do everything except change the owner. The owner permission is meant for cold storage and is able to do everything. The EOS.IO software generalizes this concept by allowing each account holder to define their own hierarchy as well as the grouping of actions.

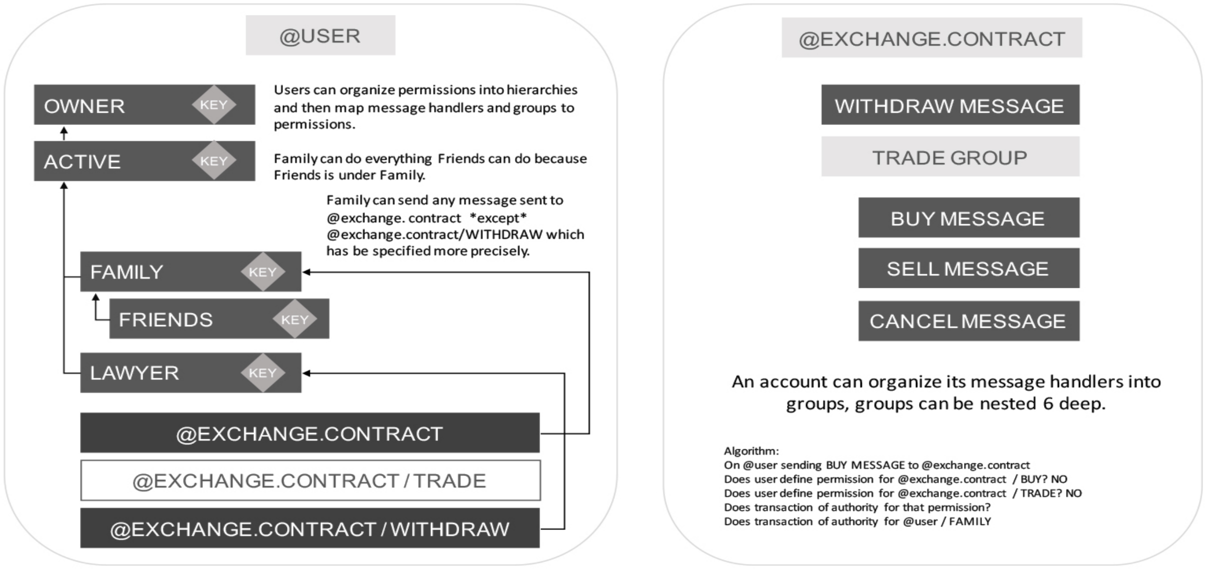
Permission Mapping

EOS.IO software allows each account to define a mapping between a contract/action or contract of any other account and their own Named Permission Level. For example, an account holder could map the account holder's social media application to the account holder's "Friend" permission group. With this mapping, any friend could post as the account holder on the account holder's social media. Even though they would post as the account holder, they would still use their own keys to sign the Action. This means it is always possible to identify which friends used the account and in what way.

Evaluating Permissions

When delivering an Action of type "**Action**", from **@alice** to **@bob** the EOS.IO software will first check to see if **@alice** has defined a permission mapping for **@bob.groupa.subgroup.Action**. If nothing is found then a mapping for **@bob.groupa.subgroup** then **@bob.groupa**, and lastly **@bob** will be checked. If no further match is found, then the assumed mapping will be to the named permission group **@alice.active**.

Once a mapping is identified then signing authority is validated using the threshold multi-signature process and the authority associated with the named permission. If that fails, then it traverses up to the parent permission and ultimately to the owner permission, **@alice.owner**.



Users can organize permissions into hierarchies and then map message handlers and groups to permissions.

Family can do everything Friends can do because Friends is under Family.

Family can send any message sent to @exchange. contract *except* @exchange.contract/WITHDRAW which has be specified more precisely.

An account can organize its message handlers into groups, groups can be nested 6 deep.

Algorithm:
On @user sending BUY MESSAGE to @exchange.contract
Does user define permission for @exchange.contract / BUY? NO
Does user define permission for @exchange.contract / TRADE? NO
Does transaction of authority for that permission?
Does transaction of authority for @user / FAMILY

@exchange.contract/ WITHDRAW may be done by either the Lawyer permission, Active or Owner.

Default Permission Groups

The EOS.IO technology also allows all accounts to have an "owner" group which can do everything, and an "active" group which can do everything except change the owner group. All other permission groups are derived from "active".

Parallel Evaluation of Permissions

The permission evaluation process is "read-only" and changes to permissions made by transactions do not take effect until the end of a block. This means that all keys and permission evaluation for all transactions can be executed in parallel. Furthermore, this means that a rapid validation of permission is possible without starting costly application logic that would have to be rolled back. Lastly, it means that transaction permissions can be evaluated as pending transactions are received and do not need to be re-evaluated as they are applied.

All things considered, permission verification represents a significant percentage of the computation required to validate transactions. Making this a read-only and trivially parallelizable process enables a dramatic increase in performance.

When replaying the blockchain to regenerate the deterministic state from the log of Actions there is no need to evaluate the permissions again. The fact that a transaction is included in a known good block is sufficient to skip this step. This dramatically reduces the computational load associated with replaying an ever growing blockchain.

Actions with Mandatory Delay

Time is a critical component of security. In most cases, it is not possible to know if a private key has been stolen until it has been used. Time based security is even more critical when people have applications that require keys be kept on computers connected to the internet for daily use. The EOS.IO software enables application developers to indicate that certain Actions must wait a minimum period of time after being included in a block before they can be applied. During this time, they can be cancelled.

Users can then receive notice via email or text message when one of these Actions is broadcast. If they did not authorize it, then they can use the account recovery process to recover their account and retract the Action.

The required delay depends upon how sensitive an operation is. Paying for a coffee might have no delay and be irreversible in seconds, while buying a house may require a 72 hour clearing period. Transferring an entire account to new control may take up to 30 days. The exact delays are chosen by application developers and users.

Recovery from Stolen Keys

The EOS.IO software provides users a way to restore control of their account when keys are stolen. An account owner can use any owner key that was active in the last 30 days along with approval from their designated account recovery partner to reset the owner key on their account. The account recovery partner cannot reset control of the account without the help of the owner.

There is nothing for the hacker to gain by attempting to go through the recovery process because they already "control" the account. Furthermore, if they did go through the process, the recovery partner would likely demand identification and multi-factor authentication (phone and email). This would likely compromise the hacker or gain the hacker nothing in the process.

This process is also very different from a simple multi-signature arrangement. With a multi-signature transaction, another entity is made a party to every transaction that is executed. By contrast, with the recovery process the recovery partner is only a party to the recovery process and has no power over the day-to-day transactions. This dramatically reduces costs and legal liabilities for everyone involved.

Deterministic Parallel Execution of Applications

Blockchain consensus depends upon deterministic (reproducible) behavior. This means all parallel execution must be free from the use of mutexes or other locking primitives. Without locks there must be some way to guarantee that transactions that may be executed in parallel do not create non-deterministic results.

The June 2018 release of EOS.IO software will run single threaded, yet it contains the data structures necessary for future multithreaded, parallel execution.

In an EOS.IO software-based blockchain, once parallel operation is enabled, it will be the job of the block producer to organize Action delivery into independent shards so that they can be evaluated in parallel. The schedule is the output of a block producer and will be deterministically executed, but the process for generating the schedule need not be deterministic. This means that block producers can utilize parallel algorithms to schedule transactions.

Part of parallel execution means that when a script generates a new Action it does not get delivered immediately, instead it is scheduled to be delivered in the next cycle. The reason it cannot be delivered immediately is because the receiver may be actively modifying its own state in another shard.

Minimizing Communication Latency

Latency is the time it takes for one account to send an Action to another account and then receive a response. The goal is to enable two accounts to exchange Actions back and forth within a single block without having to wait 0.5 seconds between each Action. To enable this, the EOS.IO software divides each block into cycles. Each cycle is divided into shards and each shard contains a list of transactions. Each transaction contains a set of Actions to be delivered. This structure can be visualized as a tree where alternating layers are processed sequentially and in parallel.

```
Block

  Region

    Cycles (sequential)

      Shards (parallel)

        Transactions (sequential)

          Actions (sequential)

            Receiver and Notified Accounts (parallel)
```

Transactions generated in one cycle can be delivered in any subsequent cycle or block. Block producers will keep adding cycles to a block until the maximum wall clock time has passed or there are no new generated transactions to deliver.

It is possible to use static analysis of a block to verify that within a given cycle no two shards contain transactions that modify the same account. So long as that invariant is maintained a block can be processed by running all shards in parallel.

Read-Only Action Handlers

Some accounts may be able to process an Action on a pass/fail basis without modifying their internal state. If this is the case, then these handlers can be executed in parallel so long as only read-only Action handlers for a particular account are included in one or more shards within a particular cycle.

Atomic Transactions with Multiple Accounts

Sometimes it is desirable to ensure that Actions are delivered to and accepted by multiple accounts atomically. In this case both Actions are placed in one transaction and both accounts will be assigned the same shard and the Actions applied sequentially.

Partial Evaluation of Blockchain State

Scaling blockchain technology necessitates that components are modular. Everyone should not have to run everything, especially if they only need to use a small subset of the applications.

An exchange application developer runs full nodes for the purpose of displaying the exchange state to its users. This exchange application has no need for the state associated with social media applications. EOS.IO software allows any full node to pick any subset of applications to run. Actions delivered to other applications are safely ignored if your application never depends upon the state of another contract.

Subjective Best Effort Scheduling

The EOS.IO software cannot obligate block producers to deliver any Action to any other account. Each block producer makes their own subjective measurement of the computational complexity and time required to process a transaction. This applies whether a transaction is generated by a user or automatically by a smart contract.

On a launched blockchain adopting the EOS.IO software, at a network level all transactions are billed a computational bandwidth cost based on the number of WASM instructions executed. However, each individual block producer using the software may calculate resource usage using their own algorithm and measurements. When a block producer concludes that a transaction or account has consumed a disproportionate amount of the computational capacity they simply reject the transaction when producing their own block; however, they will still process the transaction if other block producers consider it valid.

In general, so long as even 1 block producer considers a transaction as valid and under the resource usage limits then all other block producers will also accept it, but it may take up to 1 minute for the transaction to find that producer.

In some cases, a producer may create a block that includes transactions that are an order of magnitude outside of acceptable ranges. In this case the next block producer may opt to reject the block and the tie will be broken by the third producer. This is no different than what would happen if a large block caused network propagation delays. The community would notice a pattern of abuse and eventually remove votes from the rogue producer.

This subjective evaluation of computational cost frees the blockchain from having to precisely and deterministically measure how long something takes to run. With this design there is no need to precisely count instructions which dramatically increases opportunities for optimization without breaking consensus.

Deferred Transactions

EOS.IO Software supports deferred transactions that are scheduled to execute in the future. This enables computation to move to different shards and/or the creation of long-running processes that continuously schedule a continuance transaction.

Context Free Actions

A Context Free Action involves computations that depend only on transaction data, but not upon the blockchain state. Signature verification, for example, is a computation that requires only the transaction data and a signature to determine the public key that signed the transaction. This is one of the most expensive individual computations a blockchain must perform, but because this computation is context free it can be performed in parallel.

Context Free Actions are like other user Actions, except they lack access to the blockchain state to perform validation. Not only does this enable EOS.IO to process all Context Free Actions such as signature verification in parallel, but more importantly, this enables generalized signature verification.

With support for Context Free Actions, scalability techniques such as Sharding, Raiden, Plasma, State Channels, and others become much more parallelizable and practical. This development enables efficient inter-blockchain communication and potentially unlimited scalability.

Token Model and Resource Usage

PLEASE NOTE: CRYPTOGRAPHIC TOKENS REFERRED TO IN THIS WHITE PAPER REFER TO CRYPTOGRAPHIC TOKENS ON A LAUNCHED BLOCKCHAIN THAT ADOPTS THE EOS.IO SOFTWARE. THEY DO NOT REFER TO THE ERC-20 COMPATIBLE TOKENS BEING DISTRIBUTED ON THE ETHEREUM BLOCKCHAIN IN CONNECTION WITH THE EOS TOKEN DISTRIBUTION.

All blockchains are resource constrained and require a system to prevent abuse. With a blockchain that uses EOS.IO software, there are three broad classes of resources that are consumed by applications:

1. Bandwidth and Log Storage (Disk);
2. Computation and Computational Backlog (CPU); and
3. State Storage (RAM).

Bandwidth and computation have two components, instantaneous usage and long-term usage. A blockchain maintains a log of all Actions and this log is ultimately stored and downloaded by all full nodes. With the log of Actions, it is possible to reconstruct the state of all applications.

The computational debt is calculations that must be performed to regenerate state from the Action log. If the computational debt grows too large then, it becomes necessary to take snapshots of the blockchain's state and discard the blockchain's history. If computational debt grows too quickly then it may take 6 months to replay 1 year worth of transactions. It is critical, therefore, that the computational debt be carefully managed.

Blockchain state storage is information that is accessible from application logic. It includes information such as order books and account balances. If the state is never read by the application, then it should not be stored. For example, blog post content and comments are not read by application logic, so they should not be stored in the blockchain's state. Meanwhile the existence of a post/comment, the number of votes, and other properties do get stored as part of the blockchain's state.

Block producers publish their available capacity for bandwidth, computation, and state. The EOS.IO software allows each account to consume a percentage of the available capacity proportional to the amount of tokens held in a 3-day staking contract. For example, if a blockchain based on the EOS.IO software is launched and if an account holds 1% of the total tokens distributable pursuant to that blockchain, then that account has the potential to utilize 1% of the state storage capacity.

Adopting the EOS.IO software on a launched blockchain means bandwidth and computational capacity are allocated on a fractional reserve basis because they are transient (unused capacity cannot be saved for future use). The algorithm used by EOS.IO software is similar to the algorithm used by Steem to rate-limit bandwidth usage.

Objective and Subjective Measurements

As discussed earlier, instrumenting computational usage has a significant impact on performance and optimization; therefore, all resource usage constraints are ultimately subjective, and enforcement is done by block producers according to their individual algorithms and estimates. These would typically be implemented by a block producer via the writing of a custom plugin.

That said, there are certain things that are trivial to measure objectively. The number of Actions delivered, and the size of the data stored in the internal database are cheap to measure objectively. The EOS.IO software enables block producers to apply the same algorithm over these objective measures but may choose to apply stricter subjective algorithms over subjective measurements.

Receiver Pays

Traditionally, it is the business that pays for office space, computational power, and other costs required to run the business. The customer buys specific products from the business and the revenue from those product sales is used to cover the business costs of operation. Similarly, no website obligates its visitors to make micropayments for visiting its website to cover hosting costs. Therefore, decentralized applications should not force its customers to pay the blockchain directly for the use of the blockchain.

A launched blockchain that uses the EOS.IO software does not require its users to pay the blockchain directly for its use and therefore does not constrain or prevent a business from determining its own monetization strategy for its products.

While it is true that the receiver can pay, EOS.IO enables the sender to pay for bandwidth, computation, and storage. This empowers application developers to pick the method that is best for their application. In many cases sender-pays significantly reduces complexity for application developers who do not want to implement their own rationing system. Application developers can delegate bandwidth and computation to their users and then let the “sender pays” model enforce the usage. From the perspective of the end user it is free, but from the perspective of the blockchain it is sender-pays.

Delegating Capacity

A holder of tokens on a blockchain launched adopting the EOS.IO software who may not have an immediate need to consume all or part of the available bandwidth, can delegate or rent such unconsumed bandwidth to others; the block producers running EOS.IO software on such blockchain will recognize this delegation of capacity and allocate bandwidth accordingly.

Separating Transaction costs from Token Value

One of the major benefits of the EOS.IO software is that the amount of bandwidth available to an application is entirely independent of any token price. If an application owner holds a relevant number of tokens on a blockchain adopting EOS.IO software, then the application can run indefinitely within a fixed state and bandwidth usage. In such case, developers and users are unaffected from any price volatility in the token market and therefore not reliant on a price feed. In other words, a blockchain that adopts the EOS.IO software enables block producers to naturally increase bandwidth, computation, and storage available per token independent of the token's value.

A blockchain using EOS.IO software also awards block producers tokens every time they produce a block. The value of the tokens will impact the amount of bandwidth, storage, and computation a producer can afford to purchase; this model naturally leverages rising token values to increase network performance.

State Storage Costs

While bandwidth and computation can be delegated, storage of application state will require an application developer to hold tokens until that state is deleted. If state is never deleted, then the tokens are effectively removed from circulation.

Block Rewards

A blockchain that adopts the EOS.IO software will award new tokens to a block producer every time a block is produced. In these circumstances, the number of tokens created is determined by the median of the desired pay published by all block producers. The EOS.IO software may be configured to enforce a cap on producer awards such that the total annual increase in token supply does not exceed 5%.

Worker Proposal System

In addition to electing block producers, pursuant to a blockchain based on the EOS.IO software, token holders can elect a number of Worker Proposals designed to benefit the community. The winning proposals will receive tokens of up to a configured percent of the token inflation minus those tokens that have been paid to block producers. These proposals will receive tokens proportional to the votes each application has received from token holders, up to the amount they request for performing their work. The elected proposals can be replaced by newly elected proposals by token holders.

The system contracts that implement Worker Proposals may not be in place at initial launch in June 2018, but the funding mechanism will. It will begin to accumulate funds at the same time block producer awards start. Since the Worker Proposal System will be implemented in WASM it can be added at a later date without a fork.

Governance

Governance is the process by which people in a community:

1. Reach consensus on subjective matters of collective action that cannot be captured entirely by software algorithms;
2. Carry out the decisions they reach; and
3. Alter the governance rules themselves via Constitutional amendments.

An EOS.IO software-based blockchain implements a governance process that efficiently directs the existing influence of block producers. Absent a defined governance process, prior blockchains relied ad hoc, informal, and often controversial governance processes that result in unpredictable outcomes.

A blockchain based on the EOS.IO software recognizes that power originates with the token holders who delegate that power to the block producers. The block producers are given limited and checked authority to freeze accounts, update defective applications, and propose hard forking changes to the underlying protocol.

Embedded into the EOS.IO software is the election of block producers. Before any change can be made to the blockchain these block producers must approve it. If the block producers refuse to make changes desired by the token holders then they can be voted out. If the block producers make changes without permission of the token holders then all other non-producing full-node validators (exchanges, etc) will reject the change.

Freezing Accounts

Sometimes a smart contact behaves in an aberrant or unpredictable manner and fails to perform as intended; other times an application or account may discover an exploit that enables it to consume an unreasonable amount of resources. When such issues inevitably occur, the block producers have the power to rectify such situations.

The block producers on all blockchains have the power to select which transactions are included in blocks which gives them the ability to freeze accounts. A blockchain using EOS.IO software formalizes this authority by subjecting the process of freezing an account to a 15/21 vote of active producers. If the producers abuse the power they can be voted out and an account will be unfrozen.

Changing Account Code

When all else fails and an "unstoppable application" acts in an unpredictable manner, a blockchain using EOS.IO software allows the block producers to replace the account's code without hard forking the entire blockchain. Similar to the process of freezing an account, this replacement of the code requires a 15/21 vote of elected block producers.

Constitution

The EOS.IO software enables blockchains to establish a peer-to-peer terms of service agreement or a binding contract among those users who sign it, referred to as a "constitution". The content of this constitution defines obligations among the users which cannot be entirely enforced by code and facilitates dispute resolution by establishing jurisdiction and choice of law along with other mutually accepted rules. Every transaction broadcast on the network must incorporate the hash of the constitution as part of the signature and thereby explicitly binds the signer to the contract.

The constitution also defines the human-readable intent of the source code protocol. This intent is used to identify the difference between a bug and a feature when errors occur and guides the community on what fixes are proper or improper.

Upgrading the Protocol & Constitution

The EOS.IO software defines the following process by which the protocol, as defined by the canonical source code and its constitution, can be updated:

1. Block producers propose a change to the constitution and obtains 15/21 approval.
2. Block producers maintain 15/21 approval of the new **constitution** for 30 consecutive days.
3. All users are required to indicate acceptance of the new constitution as a condition of future transactions being processed.
4. Block producers adopt changes to the source code to reflect the change in the constitution and propose it to the blockchain using the hash of the new constitution.
5. Block producers maintain 15/21 approval of the new **code** for 30 consecutive days.
6. Changes to the code take effect 7 days later, giving all non-producing full nodes 1 week to upgrade after ratification of the source code.
7. All nodes that do not upgrade to the new code shut down automatically.

By default, configuration of the EOS.IO software, the process of updating the blockchain to add new features takes 2 to 3 months, while updates to fix non-critical bugs that do not require changes to the constitution can take 1 to 2 months.

Emergency Changes

The block producers may accelerate the process if a software change is required to fix a harmful bug or security exploit that is actively harming users. Generally speaking it could be against the constitution for accelerated updates to introduce new features or fix harmless bugs.

Scripts & Virtual Machines

The EOS.IO software will be first and foremost a platform for coordinating the delivery of authenticated messages (called Actions) to accounts. The details of scripting language and virtual machine are implementation specific details that are mostly independent from the design of the EOS.IO technology. Any language or virtual machine that is deterministic and properly sandboxed with sufficient performance can be integrated with the EOS.IO software API.

Schema Defined Actions

All Actions sent between accounts are defined by a schema which is part of the blockchain consensus state. This schema allows seamless conversion between binary and JSON representation of the Actions.

Schema Defined Database

Database state is also defined using a similar schema. This ensures that all data stored by all applications is in a format that can be interpreted as human readable JSON but stored and manipulated with the efficiency of binary.

Generic Multi Index Database API

Developing smart contracts requires a defined database schema to track, store, and find data. Developers commonly need the same data sorted or indexed by multiple fields and to maintain consistency among all the indices.

Separating Authentication from Application

To maximize parallelization opportunities and minimize the computational debt associated with regenerating application state from the transaction log, EOS.IO software separates validation logic into three sections:

1. Validating that an Action is internally consistent;
2. Validating that all preconditions are valid; and
3. Modifying the application state.

Validating the internal consistency of a Action is read-only and requires no access to blockchain state. This means that it can be performed with maximum parallelism. Validating preconditions, such as required balance, is read-only and therefore can also benefit from parallelism. Only modification of application state requires write access and must be processed sequentially for each application.

Authentication is the read-only process of verifying that an Action can be applied. Application is actually doing the work. In real time both calculations are required to be performed, however once a transaction is included in the blockchain it is no longer necessary to perform the authentication operations.

Inter Blockchain Communication

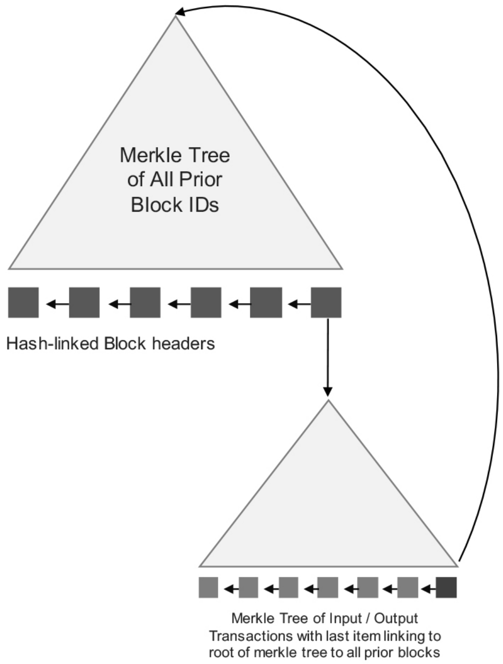
EOS.IO software is designed to facilitate inter-blockchain communication. This is achieved by making it easy to generate proof of Action existence and proof of Action sequence. These proofs combined with an application architecture designed around Action passing enables the details of inter-blockchain communication and proof validation to be hidden from application developers, enabling high level abstractions to be presented to developers.

Merkle Proofs for Light Client Validation (LCV)

Integrating with other blockchains is much easier if clients do not need to process all transactions. After all, an exchange only cares about transfers in and out of the exchange and nothing more. It would also be ideal if the exchange chain could utilize lightweight merkle proofs of deposit rather than having to trust its own block producers entirely. At the very least a chain's block producers would like to maintain the smallest possible overhead when synchronizing with another blockchain.

The goal of LCV is to enable the generation of relatively light-weight proof of existence that can be validated by anyone tracking a relatively light-weight data set. In this case the objective is to prove that a particular transaction was included in a particular block and that the block is included in the verified history of a particular blockchain.

Bitcoin supports validation of transactions assuming all nodes have access to the full history of block headers which amounts to 4MB of block headers per year. At 10 transactions per second, a valid proof requires about 512 bytes. This works well for a blockchain with a 10 minute block interval, but is no longer "light" for blockchains with a 0.5 second block interval.


Merkle Tree of All Prior Block IDs
Hash-linked Block headers
Merkle Tree of Input / Output Transactions with last item linking to root of merkle tree to all prior blocks

The EOS.IO software enables lightweight proofs for anyone who has any irreversible block header after the point in which the transaction was included. Using the hash-linked structure shown it is possible to prove the existence of any transaction with a proof less than 1024 bytes in size.

Given any block id for a block in the blockchain, and the headers a trusted irreversible block. It is possible to prove that the block is included in the blockchain. This proof takes ceil(log2(N)) digests for its path, where N is the number of blocks in the chain. Given a digest method of SHA256, you can prove the existence of any block in a chain which contains 100 million blocks in 864 bytes.

There is little incremental overhead associated with producing blocks with the proper hash-linking to enable these proofs which means there is no reason not to generate blocks this way.

When it comes time to validate proofs on other chains there are a wide variety of time/ space/ bandwidth optimizations that can be made. Tracking all block headers (420 MB/year) will keep proof sizes small. Tracking only recent headers can offer a trade off between minimal long-term storage and proof size. Alternatively, a blockchain can use a lazy evaluation approach where it remembers intermediate hashes of past proofs. New proofs only have to include links to the known sparse tree. The exact approach used will necessarily depend upon the percentage of foreign blocks that include transactions referenced by merkle proof.

After a certain density of interconnectedness, it becomes more efficient to simply have one chain contain the entire block history of another chain and eliminate the need for proofs all together. For performance reasons, it is ideal to minimize the frequency of inter-chain proofs.

Latency of Interchain Communication

When communicating with another outside blockchain, block producers must wait until there is 100% certainty that a transaction has been irreversibly confirmed by the other blockchain before accepting it as a valid input. Using an EOS.IO software-based blockchain and DPOS with 0.5 second blocks and the addition of Byzantine Fault Tolerant irreversibility, this takes approximately 0.5 second. If any chain's block producers do not wait for irreversibility it would be like an exchange crediting a deposit that was later reversed and could impact the validity of the blockchain's consensus. The EOS.IO Software uses both DPOS and aBFT to provide rapid irreversibility.

Proof of Completeness

When using merkle proofs from outside blockchains, there is a significant difference between knowing that all transactions processed are valid and knowing that no transactions have been skipped or omitted. While it is impossible to prove that all of the most recent transactions are known, it is possible to prove that there have been no gaps in the transaction history. The EOS.IO software facilitates this by assigning a sequence number to every Action delivered to every account. A user can use these sequence numbers to prove that all Actions intended for a particular account have been processed and that they were processed in order.

Segregated Witness

The concept of Segregated Witness (SegWit) is that transaction signatures are not relevant after a transaction is immutably included in the blockchain. Once it is immutable the signature data can be pruned and everyone else can still derive the current state. Since signatures represent a large percentage of most transactions, SegWit represents a significant savings in disk usage and syncing time.

This same concept can apply to merkle proofs used for inter-blockchain communication. Once a proof is accepted and irreversibly logged into the blockchain, the 2KB of sha256 hashes used by the proof are no longer necessary to derive the proper blockchain state. In the case of inter-blockchain communication, this savings is 32x greater than the savings on normal signatures.

Another example of SegWit would be for Steem blog posts. Under this model a post would contain only the sha256(blog content) and the blog content would be in the segregated witness data. The block producer would verify that the content exists and has the given hash, but the blog content would not need to be stored in order to recover the current state from the blockchain log. This enables proof that the content was once known without having to store said content forever.

Conclusion

The EOS.IO software is designed from experience with proven concepts and best practices, and represents fundamental advancements in blockchain technology. The software is part of a holistic blueprint for a globally scalable blockchain society in which decentralized applications can be easily deployed and governed.