A Peer-to-Peer Encyclopedia Network

Sam Kazemian, Kedar Iyer, Travis Moore, Theodor Forselius, Larry Sanger

sam@everipedia.com kedar@everipedia.com travis@everipedia.com theodor@everipedia.com larry@everipedia.com

DISCLAIMER: This Everipedia Network Technical White Paper version 1.23 (August 19 2018) is a work in progress and for informational purposes only. It is meant to serve as a proposal of ideas for free, open-source software.

Abstract

Emerging blockchain technology has made it possible to create an incentivized peer-to-peer network for submitting, curating, and governing a database of encyclopedia articles. Participants in the network earn tokens for curating and submitting content to the database, then use these tokens to vote on protocol upgrades and further submissions or modifications to the database of articles. Websites, businesses, or individuals can build their own user interface to interact with the network or a subset of the network. This allows websites and applications to access and collaborate on a synchronized database of human knowledge, a “greater wiki,” that is constantly updated by all participants and applications on the network.

We propose a three module system consisting of a token module, a governance module, and an article submission module. These components interact to create a sustainable, decentralized, immutable, incentivized network of editors that create quality, well-cited encyclopedia articles. The usefulness of having credible information included in this highly ordered, historically recorded, and community maintained distributed database becomes valuable. This usefulness and a stake in the overall network provides the token its utility and value.

Introduction: The Everipedia Network

The non-blockchain version of the Everipedia (https://everipedia.org) attracts approximately 1-2 million unique visitors each month (as of August 2018). We anticipate that after the code is merged from the the blockchain version (https://iqnetwork.io), there will be a massive opportunity for the average person to encounter blockchain technology, much like Steemit has done (https://steemit.com/).

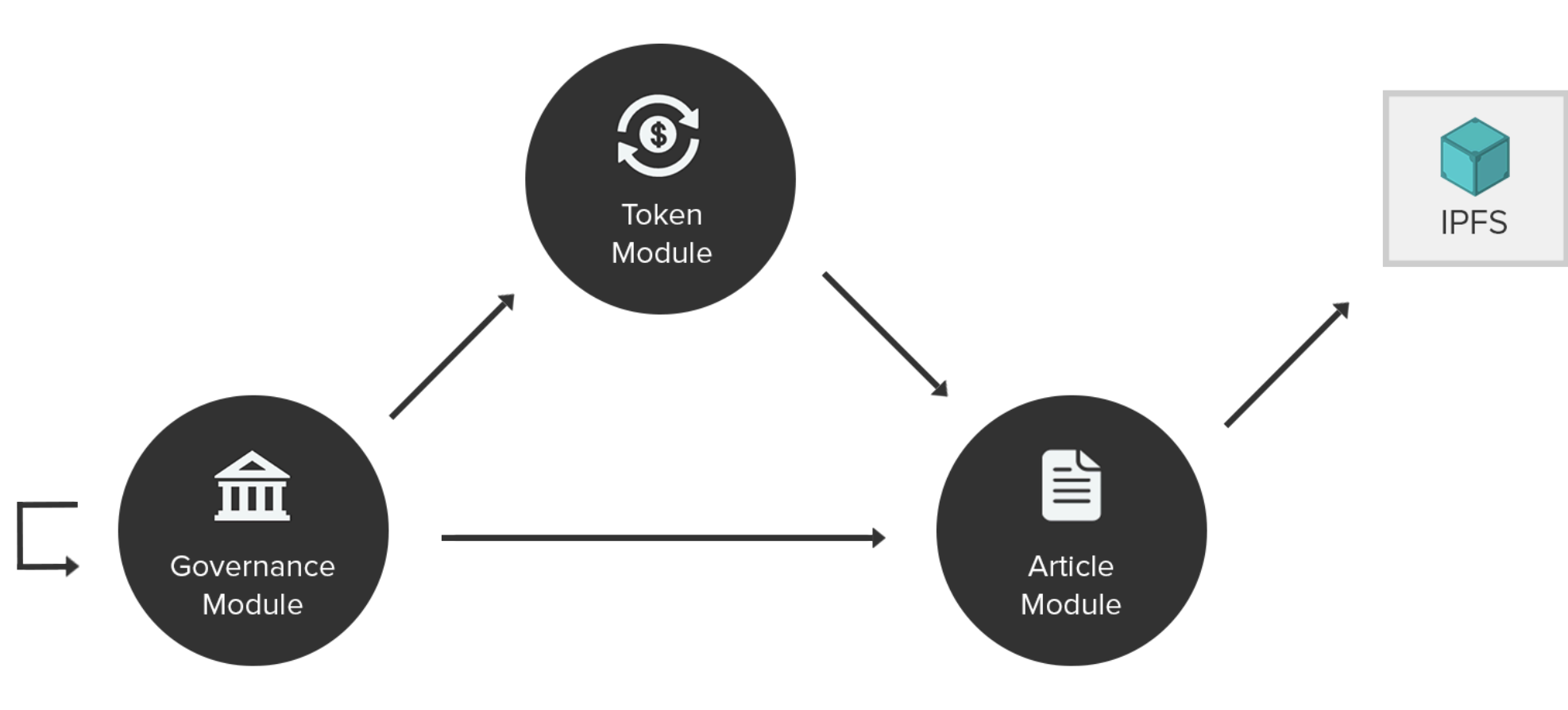
The Everipedia Network (EPN) is a decentralized encyclopedia database fully governed by token holders. Token holders can approve or reject edits, create network-wide rules that govern the encyclopedia, and buy and sell services for tokens on the network. This also means that the tokens play a central role in the consensus protocol of finalizing data entry into the network.

The Everipedia Network turns the non-profit knowledge industry into a knowledge economy where economic incentives can guide the creation of knowledge content. With the EPN one can connect directly to the world’s largest wiki database from any site. We imagine a future in which even non-programmers can create sites that utilize the Everipedia Network through something as simple as a Wordpress template.

Network effects have had a heavy centralization effect on the knowledge industry preventing serious competition and market forces from springing up. Wikipedia is one of the largest websites in the world boasting over 19 billion pageviews a month across all languages.[1][2] However, Wikipedia unintentionally traps the knowledge capital it generates within its own platform when its content could be used to create a thriving knowledge economy.

Additionally, one clear shortcoming Wikipedia has demonstrated is its inability to capture any of the monetary and intrinsic value of content that its platform and community has created, as evidenced by bi-annual donation banner campaigns.[3][4] In this regard, there is room to dramatically upend the status quo by creating an open, distributed knowledge base with technology that properly tracks the value creation of the community and returns this value back to the creators, curators, and developers of the platform. Additionally, a distributed platform which draws consensus, contributions, incentives, and value from the participants of the network also has a unique opportunity for the participants to take part in the actual hosting, storage, and distribution of the content on such a network.

Achieving such a goal would create a peer to peer distributed content system that is properly incentivized and coordinated. Such a system would be immune to single points of failure which makes it nearly impossible to censor or block access by various bad actors.



The three modules of the EPN and their interactions. The token module is responsible for holding token balances, minting schedule, and sending/receiving/fee functions. It can change the state of the article module. The article module is responsible for edit proposals and changing the state of the database of articles which are then sent to IPFS nodes. The governance module is responsible for proposals on changing network-wide rules. It can change the state of any module (including itself) by deploying new code that is voted through by token holders.

On-chain Knowledge & Facts

Current smart contracts are fairly simplistic and do not have the capability to replicate more nuanced financial arrangements in the real world. Additionally, the “oracle problem” is still unresolved. There is no standard, trustless, and sybil resistant method for agreeing on real-world knowledge so that financial transactions can reference them. One of the direct uses of the Everipedia Network should be to create the world’s first repository of on-chain knowledge so that financial transactions and decentralized applications can reference them without having to re-invent their own oracle systems. In this sense, the IQ token’s intrinsic value is the stake it provides in the largest distributed repository of facts used by a growing industry.

Governance Module

The governance module is an object which has scope to make changes to every module, including itself. Governance actions can modify the software for any of the three modules, but not the databases containing token balances and articles.

The governance module allows for submitting changes to the community of token holders for approval. If approved, the governance module then deploys those changes to the corresponding module(s). This allows for the community to come to social consensus on the rules that govern the network as a whole. Discussion about this consensus can take place off-chain on social media and other communication hubs, but deployment of new code needs to be done on-chain in a trustless manner.

The EPN will be a system of smart contracts on the EOS platform, so users will not be running full nodes of the Everipedia Network. This means users cannot vote on software updates by updating their client software as they do in Bitcoin or Ethereum. Instead, a trustless on-chain consensus process must be designed for deployment of new updates.

Without an internal consensus process, the only viable alternative for updating the module software would be for a trusted party (such as core developers or a foundation) to process off-chain consensus and deploy new updates with their elevated permissions. This is undesirable given the spirit of the industry and the clear trustless alternative that is possible.

Research and development, solutions to scaling, and improvements to the codebase are just as critical (if not more critical) as any service and feature built on top of the network. For this reason, we have laid the foundation for proposing edits and additions to the source code of the network through the governance module. This mechanism will be used for meta-governance of the network protocol itself in a fully trustless, on-chain process.

Funding the Network and Self-Sustainability

The Everipedia Network is a series of modules (smart contracts) which runs on the EOS Virtual Machine (EVM). The virtual machine’s storage, random access memory, and bandwidth/compute resources are allocated to accounts proportional to their EOS token balances. The Everipedia Network will require EVM resources to run and function. As the network increases in size, it would require a higher percentage of the EOS network’s compute power and as a result need more EOS tokens.

There are many ways for the EPN to fund itself ranging from older generation methods such as simple donations (such as Wikipedia.org), UX layer revenue as well as new generation methods based on token inflation and auctioning. One such funding method would be a governance action which would propose minting new IQ tokens to be auctioned for EOS tokens. If the action is approved by voters, the newly acquired EOS tokens would then be used to fund the EPN's new bandwidth, storage, and compute needs. This method is similar to the EOS network's own inflation based funding model where block producers are paid EOS tokens for their services.

Another possibility is to auction off the IQ generated from transaction fees for EOS, which would then be bound by an EOS smart contract to automatically purchase RAM or other resources as needed to sustain operations. An auction would only occur if it were detected that RAM was near capacity, say 90%.

Token Module

The token module is responsible for making changes to the token balances of addresses by transferring tokens, applying transaction fees, minting tokens, and locking tokens for the article consensus process. The fundamental unit of account in the Everipedia Network is the IQ token which is tracked within the token module.

Initial Supply & Minting Schedule

Tokens will be minted every 30 minutes ("IQ reward period") through the edit process and edit curation process. There will be a fixed amount of tokens minted each day. The amount will be reset every day at 00:00 UTC.

Proposed edits pass thresholds and receive IQ rewards based on votes of token holders.

At the end of the 30-minute IQ reward period, editors will receive IQ rewards in proportion to the value of their contributions. Assuming .08% yearly inflation of the supply, the following equation shows how rewards are calculated:

T_{Period} = 500 IQ

**NOTE (as of June 15 2020): The IQ rewards per 30 minute period is 500 IQ for a yearly inflation of about .08% annually (8,760,000 IQ minted yearly). The inflation percentage per year decreases as the total number of IQ minted is fixed but the total supply increases over time. The total IQ token supply will also be capped at 10.5 billion. No other IQ will be minted after the hard cap is reached unless there is a governance vote. Creator and curation rewards will then need to come from fees accrued in other products and services in the IQ ecosystem, either within the Everipedia dApp or other developed applications.

Article Module

The article module is used to propose edits to be included in the database. Proposed edits are tuple objects which contain an IPFS hash pointing to the immediate parent version and an IPFS hash pointing to the new version. An example of an edit object could be: [QmXvHQCbvxp3vQm96VmZDBaTX8Aae6vVcoTvVB6QQsMXnM, QmeAv3LJo4Kre6dR7GQBqjJFztY9YWZD131W5tYGStUcbM]

Staking IQ Tokens

Staking IQ tokens is required to propose edits, vote on edits, and propose/vote on network governance actions. IQ is staked by locking up tokens in a 21-day vesting period. This is similar to other blockchains such as Steem which requires users to “power up” (essentially lock and vest) their tokens in order to vote/stake them on content published on the platform.[9] The process of staking IQ is also called powering up (as a token of appreciation to Steem’s pioneering design). IQ that is locked up for 21 days gives the holder “Brainpower” (BP) at a 1:1 ratio. Brainpower is not a fungible or transferable token and only spent during usage of the EPN by the staking account. Once an account’s BP is entirely spent, they must wait for the IQ staking period of 21 days to end before re-staking their tokens for BP to use the EPN. Otherwise, they can acquire more IQ tokens and stake the newly acquired IQ for BP.

Example: A user has 150 IQ tokens. They can call the staking function to lock up their 150 IQ tokens for 21 days and get their account allocated 150 Brainpower to use for proposing edits, governance actions, and voting.

The second feature of the article module is using BP for token holder voting of inclusion or exclusion of proposed edits in queue. The validation of articles goes through a validation algorithm (below) with parameters that can be changed through a governance vote.

Validation Algorithm (the content consensus method)

One of the most important processes in the network is the validation of state changes to the database - that is, approval of changes to articles or creation of new ones. The validation algorithm is a function which takes in as arguments the proposed edit object of the article and the BP votes for that proposal. It returns token rewards, accepted status, and slashing conditions.

The validation period for each edit lasts a maximum of 30 minutes. Front-ends are free to use their own criteria to display pending edits since even pending edit objects are located inside the database. For example, some front-ends could choose to display all pending edits with time-sensitive content.

Curation Rewards

$PeriodReward_{Curation}$ = Curation mint rate; number of IQ tokens minted for curation rewards per period P (initial network P = 30 minutes)

$Votes_{Curator}$ = Sum of one user's Brainpower who voted on a majority side during period P

$Votes_{Majority}$ = Sum of all users' Brainpower who voted on a majority side during period P

$Reward_{Curator}$ = Voter/Curator's curation reward per period P

Curation rewards will not be given for proposals that are ties (50% / 50%).

Edit Rewards (Content Rewards)

$PeriodReward_{Editor}$ = Edit mint rate; number of IQ tokens minted for editor rewards per period P (initial network P = 30 minutes)

$ApprovalVotes$ = Sum of all Brainpower votes for an approval (including curators) in a period.

$Reward_{Editor}$ = Editor's curation reward per period P

Editor rewards will not be given for proposals that are ties (50% / 50%), but they will still be approved.

Examples

Assuming period reward of 10 IQ:

Scenario 1

User	Vote	Reward
Sam (Editor)	+10	8.235 IQ
Travis	+25	0.588 IQ
Kedar	+50	1.176 IQ

Sam = ((10 + 25 + 50) / (10 + 25 + 50)) * 8 IQ + (10 / (10 + 25 + 50)) * 2 IQ = 8.235 IQ

Travis = (25 / (10 + 25 + 50)) * 2 IQ = 0.588 IQ

Kedar = (50 / (10 + 25 + 50)) * 2 IQ = 1.176 IQ

Scenario 2

User	Vote	Reward
Sam (Editor)	+10	8.333 IQ
Travis	-25	25 IQ slashed for an extra 747106 seconds
Kedar	+50	1.666 IQ

Sam = ((10 + 50) / (10 + 50)) * 8 IQ + (10 / (10 + 50)) * 2 IQ = 8.333 IQ

Travis = ((10 - 25 + 50) / (10 + 25 + 50)) * 1814400 seconds = Current 25 IQ stake extended for an extra 747106 seconds (slashing)

Kedar = (50 / (10 + 50)) * 2 IQ = 1.666 IQ

Slashing Conditions

Editing articles can be a fairly contentious activity. However, there must be sybil-resistant economic incentives to have token holders arrive at the common, salient answer of the edit proposal. That is, there must be incentives for each individual agent to arrive at what they believe to be the common response - the correct one. Likewise, there should be disincentives proportional to arriving at the minority response - the incorrect one. Unlike proof of stake based consensus methods which slash/burn the staker’s tokens for voting on incorrect blocks, the EPN slashing condition does not permanently burn tokens for voting incorrectly (voting on the minority side).

Slashing increases the staking lock-up period inversely proportional to the minority voting ratio. If a user’s account is slashed, their lock-up period for withdrawing their IQ tokens increases by some amount. This solves two issues: 1. Repeatedly voting on the minority side effectively burns IQ tokens since the lock-up period consistently increases so that repeated attacks on the network de facto remove the attacker’s IQ tokens from circulation (since they will not be able to withdraw and re-stake their tokens for a very long duration). 2. Participation in genuine contentious discussions (“edit wars”) is not discouraged since losing sides do not get their tokens burned permanently. Additionally, the more contentious an edit is, the less penalty there is for voting on the minority side (since the minority is almost as large as the majority).

Voters in the majority are given a proportional amount of the IQ inflationary token reward allotted for content. Voters that vote in the minority have their token lock-up period increased for an extended amount of time, inversely-proportional to the minority stake. The slashing ratio can be modeled with the following:

(Total Majority BP Votes - Total Minority BP Votes) / (Total BP Votes) = Slashing Ratio

Ex: User has 200 Brainpower and votes “No” using all their BP for an edit proposal. Total votes equal 2000 Brainpower with 1500 BP votes “Yes” and 500 total BP votes “No.” Because the user voted on the minority side, the slashing ratio is calculated as: ((1500) - (500)) / (2000) = (.5) then multiplied by (21 days) to get the increased lock-up time.

The 200 IQ that generated the 200 BP is locked up for an additional 10.5 days. Note: The more contentious the voting, the less the penalty is for voting in the minority.

Identity, Reputation, and Account Histories

It is possible to leverage upcoming on-chain identification and reputation systems to incorporate into the validation algorithm and edit approval process such that previous edit histories and identities of editors can be measured in the consensus process. Such identity systems could include uPort or native EOS.IO user ID/reputation systems. Since edit proposals (and their approval/disapproval) are already stored on-chain, this user history can be incorporated into new updates to the validation algorithm coupled with the identification/credentials of the user.

Delegating Votes

Users who do not wish to personally vote using their BP can delegate their BP to another entity or “pool” for consensus voting. Pools will be operated by the community and vote on behalf of their users according to transparent principles published in a constitution, wiki, or similar document. This could form a secondary market for the price of BP if there is sufficient demand similar to how secondary markets are likely to form over EOS.IO bandwidth, RAM, and storage. Additionally, this could allow for passive earning of IQ tokens if delegates pass back some amount of the IQ earned by curating content to the original delegator.

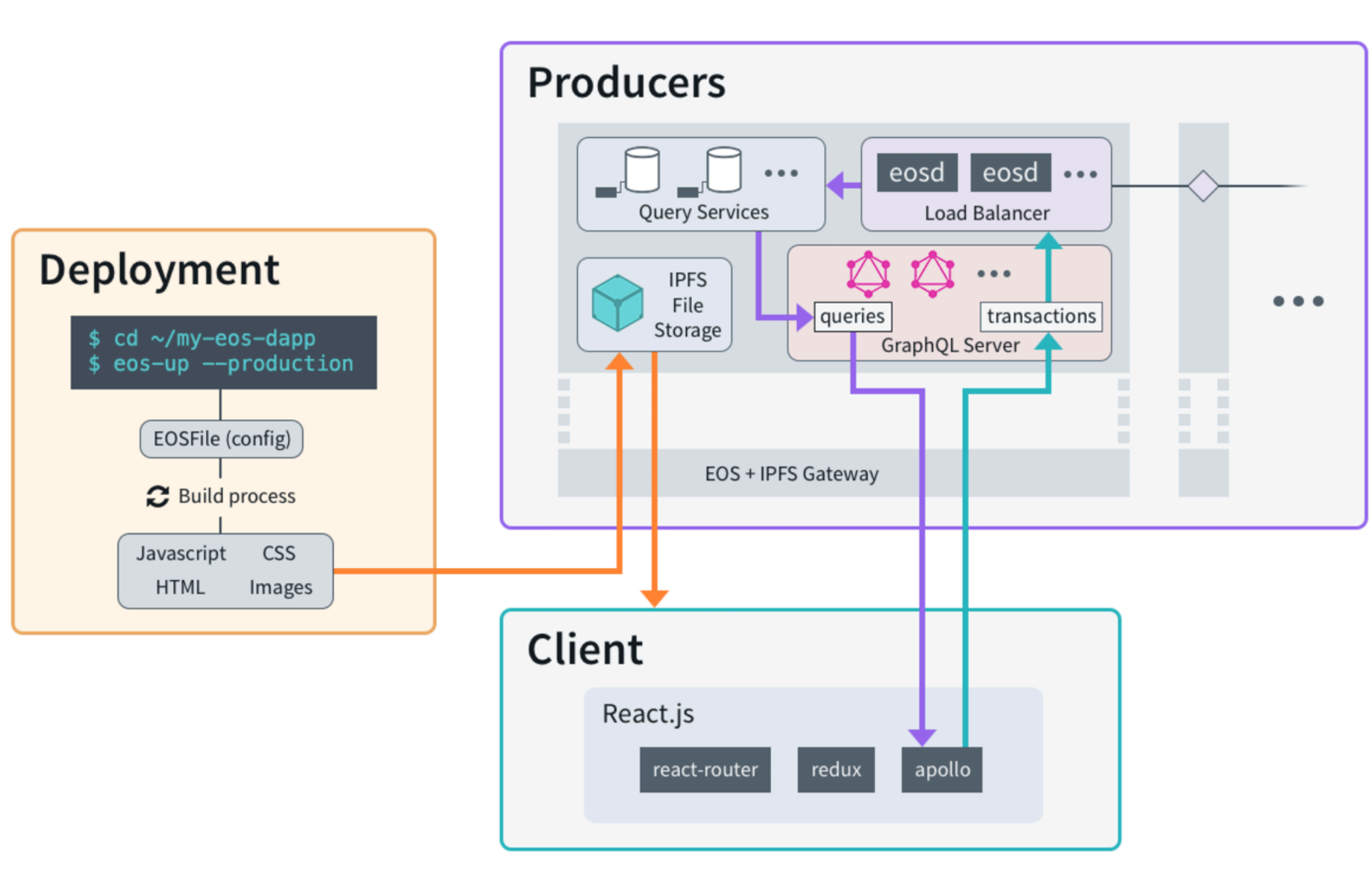
Building a delegate layer on top of the base voting protocol would allow a market to form over article narratives as “thought leaders” pledge curation ideals they would follow should they be delegated votes. For example, an anti-censorship staking pool could state that they will vote in favor of any edit that adequately cites its sources regardless of the specifics of the content. Users who believe in this vision for the encyclopedia network could delegate their votes to the staking pool.

Staking by Article or Topic

It is possible to explore per article or per topic staking such that staking IQ tokens to a specified subsection of the network’s content gives the staker more BP authority over such article/topic but only allows them to partake in voting for that specified article/topic. The validation algorithm which computes BP votes would need to check for the type of stake (general vs. per article/topic) and weight the vote accordingly. This feature would allow users to “own” a certain jurisdictional authority over certain topics proportional to their stake in that area of the network’s content.

EOS.IO Implementation

EOS.IO software is a Turing-complete distributed computing platform and smart contract system in which block producers allocate bandwidth & storage for large scale decentralized applications proportional to the user/developer EOS token balance.[7]



EOS.IO schematic: Block producers of the network provide bandwidth, IPFS storage, query services, servers, and gateway interfaces. They are incentivized to do this through newly minted EOS tokens, similar to how Bitcoin miners are incentivized to provide SHA256 hashing services to the network through Bitcoin block rewards. Users (clients) then connect to services made available by block producers similar to using classical internet services. Since there are multiple block producers, there is a distributed entry point to all software on the EOS.io network unlike classical centralized web applications. img-source: EOS.IO retrieved from: steemit.com

This allows for feeless transactions since accounts are only rate-limited based on their EOS token balance. The more EOS tokens an account holds, the more of their transactions an account can expect to be included in the ledger. This gasless protocol (compared to Ethereum) allows for the building of rich decentralized consumer applications.

Everipedia Network articles (as well as their histories) will be stored using IPFS protocol nodes by community members and front end service providers. In order to block access to Everipedia Network content, actors would have to prevent any TCP/IP interaction and packet exchange between end users and the EOS mainnet entirely as even a single IPFS peer can provide access to Everipedia content. This task, while theoretically feasible, is substantially more difficult than blocking a single domain and is more akin to attempting to shut down torrent networks by going after all individual torrent seeders - a task that has consistently proven impossible for many well-funded organizations and state actors. Additionally, it is possible for community members to host any set of articles from their own private IPFS daemon which listens for edits of articles by connecting to the EOS network. This means that anyone who wishes to host articles (or some subset of articles) can do so from any location or server by running their own IPFS node and light EOS client.

Under the EOS.IO implementation, an Everipedia “reader” is any individual with an internet connection that can access the EOS main network and query an IPFS protocol for an article hash as recorded in the EPN article module smart contract. Reading and requesting Everipedia content is planned to be free to all end users and does not require any IQ token balance. Conversely, an Everipedia “user” is any individual who possesses a functional wallet (an EOS.IO account) with a valid balance of IQ tokens to propose edits. Users will sign all their transactions with the associated private key to demonstrate ownership of a valid balance. Using the EOS.IO account system allows for various web standards and benefits that are not currently available on other blockchains such as account recovery and human readable usernames. This would essentially create a seamless user experience similar to using a web application with classical user and account authentication. Additionally, it is possible to leverage upcoming EOS.IO identification and reputation systems to incorporate into the validation algorithm and edit approval process such that previous edit histories and identities of editors can be measured in the consensus process.

Database Schema

Collective collaboration requires agreed norms and standards, especially norms in handling the data that is stored, updated, and consumed. Bitcoin’s unspent transaction output (UTXO) database model is only effective because all network participants agree to keep account values stored in such a structure. Otherwise, the same security guarantees would not be possible.

The “database schema” refers the structure of on-chain storage in the EOS.IO blockchain inside of the article module (the associated smart contract). The database for the Everipedia Network will be a 2 column element that pertains to the current hash of the article state. The second element is the immediate preceding hash of the previous state of the article which functions as a pointer. This allows for a tree-like structure of the history of IPFS hashes of the article and allows quick querying of the Everipedia Network database to find the current state of an article in relation to some historical snapshot (as well as any forks or merges). This is similar to a git protocol tree of all previous commits to a codebase except the work done to be committed is an encyclopedia article or edit. This schema has two technical guarantees that make it ideal for distributed ledger storage: 1. Querying for a complete state tree of an article (therefore showing all previous historical edits). 2. Easy branching and merging of articles while keeping a unified history of previous states.

An example database of 10 rows is shown below:
{
[QmPgLhaXAPYxx9isxGuxnBZA28WoUxhYsiRaPPbMKdvPAm, null],
[QmSRvE5W6WV4KGRwhttumnamJgGGyiaAGYLEaubRgRreP2,

QmPgLhaXAPYxx9isxGuxnBZA28WoUxhYsiRaPPbMKdvPAm]

[QmcmBixvVnzrswZ24BbzjbrkTvi4ZMbxZcZpfqjzkxmLYA, null]

[QmeAv3LJo4Kre6dR7GQBqjJFztY9YWZD131W5tYGStUcbM, null],

[QmXvHQCbvxp3vQm96VmZDBaTX8Aae6vVcoTvVB6QQsMXnM,

QmeAv3LJo4Kre6dR7GQBqjJFztY9YWZD131W5tYGStUcbM],

[QmYP7vvmy1MX7x6QDazd8opK4KB2NuhiLGbwBQWgxR552q,

QmXvHQCbvxp3vQm96VmZDBaTX8Aae6vVcoTvVB6QQsMXnM]

[QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco, null],

[QmcHn96sJBZY4QaGeDh6vVBDznygqbVCgh7bwHbskxcoaY,

QmeAv3LJo4Kre6dR7GQBqjJFztY9YWZD131W5tYGStUcbM]

[QmbtygxuXiQDYn1BWDjsJtHUVYuFzUUV6fdJZg8AL4axVB,

QmSRvE5W6WV4KGRwhttumnamJgGGyiaAGYLEaubRgRreP2]

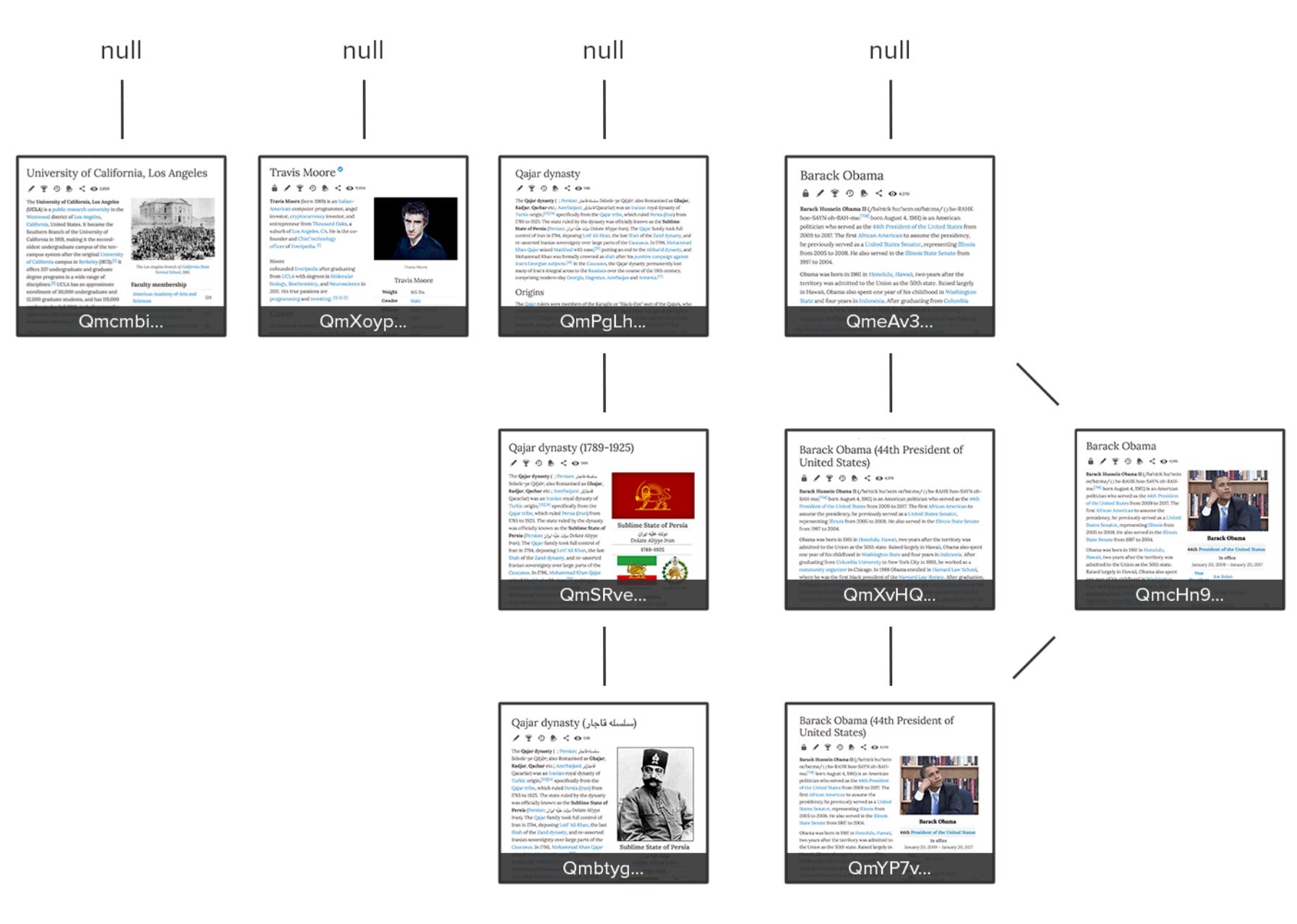
[QmYP7vvmy1MX7x6QDazd8opK4KB2NuhiLGbwBQWgxR552q,

QmcHn96sJBZY4QaGeDh6vVBDznygqbVCgh7bwHbskxcoaY]
}

The first element in each tuple marks some current state, while the second element refers to the previous article state. If the second element is null, this means that the article state has no previous state (meaning the article was just created). In the example above, rows 1, 3, 4, and 7 depict the creation of a new article with no previous state. Row 2 simply shows the article in line one being edited. Row 9 shows the article state in row 2 being edited (now with 3 total edits in its history)

Line 5 and 8 demonstrate an article of the same state forking into two different states (perhaps edited by two different communities or telling different narratives of the same topic). Both line 5 and 8 share the previous state of an article created on row 4. Row 6 shows the article in 5 being edited further but the forked off article in row 8 clearly ignores the change because row 9 depicts an edit made to the forked article. Row 10 shows the original forked article in row 8 being merged back into the article that it forked from. The two forked articles are now merged into one again (perhaps due to an edit proposal and community agreement).

While the above structure has many advantages, it has several issues that should be directly addressed. For example, concurrent edit proposals can potentially cause issues. If one individual proposes an edit to an article and submits an edit object, then before that object is finalized through consensus, another edit object referencing the same parent hash essentially creates an unintended fork of the article. A potential way to resolve this issue would be to only allow one pending edit proposal referencing the same parent hash at a time in the queue of pending edits.

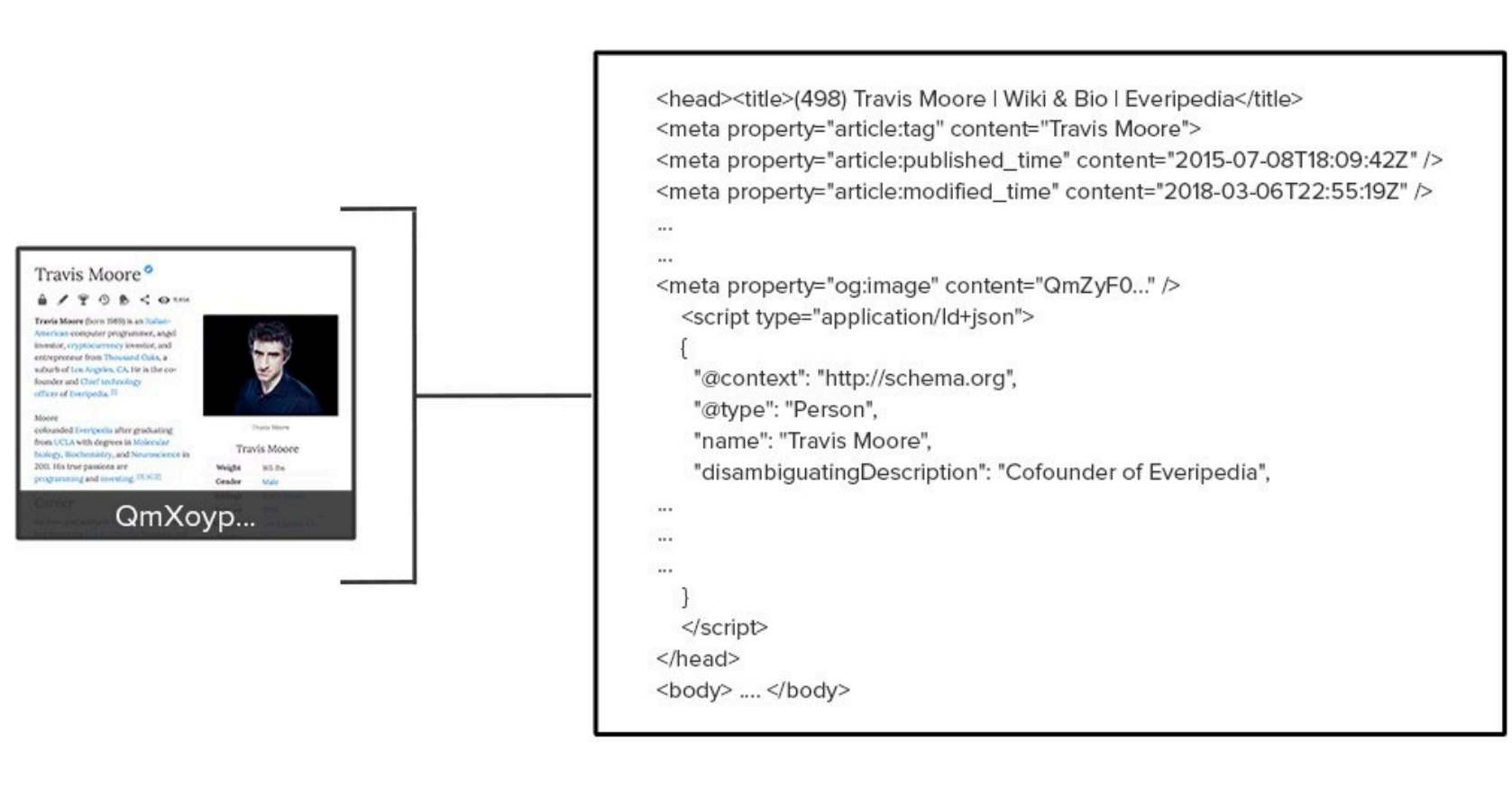


Visual schematic of the above database schema showing 4 distinct articles: “University of California, Los Angeles,” “Travis Moore,” “Qajar Dynasty,” and “Barack Obama.” Each article is hashed. The first edit of an article (article creation) points to a null parent hash. Subsequent edits create new content and hence a new hash and point to the article’s immediate previous state. This forms a tree which makes the article history.

Article Data Structure

Articles will be stored as gzipped HTML files. This is because using the HTML document object model allows for high compatibility of possible front end combinations for the data which is retrieved from the network. By using HTML, any developer can create a front end to display articles in customized manners, submit edits, and analyze data inside the document. Designated HTML class attributes will be used to mark specific portions of articles so that all front end software can easily detect standard sections of an article. Such classes could include “infobox,” “citation,” or “title.” A data structure standard should be decided on by community members and early adopters of the technology which should include linking schemes and hyperlink rules (such as whether only internal links to the database can be embedded in articles etc). Although the standards can, in theory, be changed with a governance vote, practical application of such a data-structure change is highly cumbersome as some changes are not backwards compatible and would need to be retroactively applied to all older articles within the network.

Finally, within the HTML file, there will be one designated JSON-LD class for linked data to the object topic of the article. This allows easy retrievability of key-value descriptors from any article using API endpoints and allows bots to add information easily to the JSON-LD class. JSON-LD is used to build rich graph relationships between objects and is currently used in web pages to describe semantic relationships between the webpage’s topic and content. Having a robust data class for building rich graph relationships between topics within the article helps create a meta-network layer which can be queried for rich data by machines, services, and artificial intelligence training.



Each article is an HTML file that can be served from IPFS nodes or EOS.IO storage across many jurisdictions and participants. CSS and styling is applied on the front end UX layer and can be customized by website owners that serve the content and compete with other front end services. The article content, metadata, and class objects are edited by network participants and require token holder consensus.

Blockchain Agnosticism

This whitepaper discusses the implementation of the Everipedia Network using EOS.IO software. It is entirely possible for the three modules discussed above to be built and implemented using different distributed ledger technology while preserving the fundamental substance and utility proposed. Additionally, it is entirely possible to build the three modules using different distributed ledger technology which interact through cross-blockchain transfers of value and data. One example of such an implementation would be building an entire blockchain in a proof of stake system using the Ethereum Plasma Framework. IQ would act as the staking token for production and proposition of blocks.[5][6] Another example of cross-chain implementation could be building the governance module using the upcoming Tezos network since Tezos software is bundled with sophisticated self-governance features to be used for its own network rule-making. Additionally, the storage of article blobs through IPFS nodes can be incentivized using different implementations, such as Filecoin, in addition to EOS.IO Storage (like discussed above).

Wikipedia started with 1990s server hardware and architecture but upgraded to caching infrastructure and state of the art data centers. The Everipedia Network is a long term self-sustainable project that has the goal of creating the first global decentralized knowledge base. Given that goal, the network should specifically use technology that has long term potential and developers should keep open minds towards future scaling solutions. This allows network developers and maintainers to make bets on what will be valuable blockchain technology today and in the foreseeable future, but allows building an expandable and platform agnostic network which can use multiple breakthrough technologies to function efficiently.

Bounties, Services, Bots, Private Edits, Marketplaces, and Network Development

The Everipedia Network is a light-protocol, meaning that it is agnostic to features and services built on its underlying infrastructure. For this reason, the network does not come directly bundled with services hard-coded such as classical advertising or subscriptions. Instead, the network imposes a nominal transaction fee on all token interactions which is burnt (governance based transactions and messages do not accrue a fee). This creates a base layer for an ecosystem of services, bounty/freelance work, and marketplaces for citation verification services and other novel uses. In this way, the Everipedia Network could support established media organizations such as the Wikimedia Foundation (Wikipedia), CNN, and NYTimes taking part in the editing and verification process by creating their own services and organization accounts on the network. This in effect creates an ecosystem where participants are not only encouraged to create the most reliable content but also encouraged to create the best tools and services for the creation of content. This also creates a large volume of transactions which supplies sizable transaction fees which are burnt.

Additionally, in future updates, it would be possible to use zero knowledge succinct non-interactive argument of knowledge proofs (zk-SNARKs) to publish state changes to articles with complete privacy. This has a twofold effect: 1.) Individuals can propose content without being publicly identified in the ledger. 2.) Curators can vote for or against the proposal without being publicly identified in the ledger. Implementation of zk-SNARKs (or other zero knowledge proofs) would be done in the same way that private transaction proofs are performed in z.cash and likely to be implemented in Ethereum.[8] To properly use such technology, the base blockchain used for the network would likely need to support zero knowledge proofs in some capacity so that entire SNARK implementations do not have to be “hand-rolled” by the Everipedia Network’s contract code itself.

Conclusion

The Everipedia network uses IQ tokens, which track the stakes of users and act as incentive rewards for content creators and curators. The network uses a three module system (token, governance, and article) to create a self-sustaining database of encyclopedia articles. The structure of content is uniform and standardized so that it is easily analyzed, consumable, and improved upon by services, front-end websites, AI, and bots. The database schema allows all the technical guarantees of prior wiki software (such as historical snapshot of all article states) as well as new functions not possible before (such as distributed, real-time hosting and scalable forking & merging of articles)

Innovation in the encyclopedia industry has stagnated in the past 10 years. Creating a synchronized, collaborative encyclopedia database accessible by all applications on the internet will generate the network effects required to re-ignite growth and innovation in the documenting of knowledge.

References

[1] Page Views for Wikipedia, Both sites, Normalized. (2005). Retrieved Oct. & nov., 2017, from https://stats.wikimedia.org/EN/TablesPageViewsMonthlyCombined.htm

[2] Matei, S. A., & Britt, B. C. (2017). Analytic Investigation of a Structural Differentiation Model for Social Media Production Groups. Lecture Notes in Social Networks Structural Differentiation in Social Media, 69-84. doi:10.1007/978-3-319-64425-7_5

[3] Denning, P., Horning, J., Parnas, D., & Weinstein, L. (2005). Wikipedia risks. Communications of the ACM, 48(12), 152. doi:10.1145/1101779.1101804

[4] Blockchain Investing - Olaf Carlson-Wee and Aaron Harris (25:01). (2017, July 19). Retrieved August 10, 2017, from https://www.youtube.com/watch?v=9SYVX2wcMVM&feature=youtu.be&t=25m1s

[5] Buterin, V., & Poon, J. (2017, August 11). Plasma: Scalable Autonomous Smart Contracts. Retrieved August 12, 2017, from http://plasma.io/plasma.pdf

[6] Poon, J. (2017, June 17). OmiseGO: Decentralized Exchange and Payments Platform. Retrieved July 3, 2017, from https://cdn.omise.co/omg/whitepaper.pdf

[7] Larimer, D., (Bytemaster), & Lavin, J., (hkshwa). (2017, June 3). EOS.IO Technical White Paper. Retrieved June 11, 2017, from https://github.com/EOSIO/Documentation/wiki/Whitepaper-Test

[8] Buterin, V. (2017, November 9). STARKs, Part I: Proofs with Polynomials. Retrieved November 10, 2017, from http://vitalik.ca/general/2017/11/09/starks_part_1.html

[9] Larimer D., Scott N., Zavgorodnev V., Johnson B., Calfee J., Vandeberg M. Steem: An incentivized, blockchain-based social media platform. March 2016. Retrieved January 7th, 2018 from https://github.com/steemit/whitepaper/commit/da16f36bf23bc53d30b57787d7b9044d9c07399c.