Joystream: A Protocol for User Governed Content Platforms

Jsgenesis

June 8, 2020
DRAFT Version 0.1

Abstract

The Joystream protocol attempts to formalize a content platform that is governed and operated by the platform users. The centerpiece of the protocol is the shared platform state, implemented on top of a blockchain consensus system, which coordinates and provides incentives to all stakeholders. Almost every aspect of a content platform is endogenous to the protocol, including

(a) governance

(b) membership system, with screening and policing

(c) storage and distribution

(d) curated content directory

(e) content search, browsing, and recommendations

(f) software development finance and coordination

(g) content production financing

(h) advertising auctions and placement

(i) communication: messaging and message boards

and every subsystem is fully accountable to, and directed by, the users, as represented by the governance system. Capturing such a broad range of systems in the protocol is *the* distinguishing characteristic of this proposal. It is motivated by the thesis that a high level of platform accountability can be achieved by empowering two user capabilities.

First, the ability to voice discontent and subsequently implement changes based on such voices, which relies on an immutable history of actions, reliable information sharing, and binding execution of policy changes.

Second, the ability to exit the platform at a low cost and create an alternative when the platform decay has gone too far. This relies on having the entire platform state available for wholesale replication, which is not possible if critical parts of the platform state are exogenous to the protocol.

Contents

1 Preface

This document aims to give a high level overview of the design approach for of a new protocol for content platforms. This is an evolving document meant as a basis for an iterative review, technical specification and testing, and revised versions will be developed on the basis of that process. Multiple aspects of the protocol are still subject to active research, and any part of the current design may be amended or entirely abandoned as a result of subsequent considerations. Lastly, the descriptive resolution varies substantially across the proposal as a result of different parts at radically different levels of maturity.

2 Introduction

2.1 Motivation

Due to a mixture of conspiring factors, such as platform externalities, economies of scale and, jurisdictional arbitrage, dominant contemporary Internet platforms have become some of the least accountable organizations today. The traditionally constraining institutions of market competition, regulation and litigation simultaneously appear to be unable to push back against their market power. The social cost of this equilibrium is multifaceted, leading to lower innovation, platform rents, and more broadly to the fact that platform policy is only incidentally, not primarily, guided by the objectives of the largest platform stakeholder: *users*. Users should be broadly understood as anyone who is participating on a platform in any capacity.

2.2 Joystream

The Joystream protocol is an attempt at formalizing the structure and function of a content platform, where user accountability is a key objective and organizing principle. This accountability is generated by technically codifying two well-known complementary responses to organizational decline [1].

- **Voice**

  Users have effective means of sharing information, coordinating, and reaching decisions about key collective action problems, namely how to allocate and regulate the use of any shared platform asset.

- **Exit**

  Users have effective means of creating new platform instances that can preserve the entire platform history and state. This ability to fully replicate the technical, administrative, and economic state of a platform has two positive effects on accountability: first, it empowers the voice of users through the threat of a low cost exist; and second, it supports the emergence of a diversity of platforms when the underlying interests of stakeholders are irreconcilably incompatible or where the organizational decline as gone sufficiently far.

In order to build in the capacity for such responses, the Joystream protocol is built as follows. The shared platform state lives on an open blockchain consensus system. As a result, it has a public state and history that is fully auditable and immutable. The protocol also has secure direct and broadcast communication capabilities integrated within a platform identity system. This means that it is very cheap for any participant to securely inspect the system and make irrefutable positive claims as part of subsequent public deliberation, learning, debating, or agitating. The identity system along with an immutable history of public actions and communications generates the incentive to invest in community standing and act with a long time horizon. Further, there is a governance mechanism that allows low cost coordination around the use and regulation of shared assets and the amendment of the protocol. Importantly this mechanism self executes and thus provides reliable and a counterparty-free implementation of governance decisions. All these properties work in hand to support *voice* as response.

The protocol itself is open and implemented in open source reference software. It has an open shared state and data accessible for all. This makes the copying step of creating an alternative instantiation trivial and thus supports *exit* as response.

3 Notes

Occasionally, there may be reference to data types in various schemes or concept definitions, assume the C++ type system.

When identifier fields are used in definitions, assume that they are unique and created by auto-incrementation in the context of some set of existing instances.

Constant values are displayed with a capitalized snake case as follows: FOO_BAR. These are values that cannot be changed through governance and require a hard fork or consensus upgrade. In some occasions, the symbol C is used to denote some constant that is specific to the context.

4 Blockchain

The Joystream protocol is stateful, and the infrastructure for the secure distribution and updating of this state is a blockchain system. A given instantiation of the Joystream protocol runs on its own single-purpose blockchain infrastructure, which only processes transactions related to this instantiation. In the rest of the document, this blockchain will be treated as a silent transaction ordering mechanism; however, in the following section, it will briefly describe the assumptions on the blockchain infrastructure in the protocol design. Here, and in the rest of the document, the *platform or platform state* will refer to the state upon which transactions operate. In Bitcoin, for example, this would be the UTXO set and be distinctive from

the state of the underlying blockchain infrastructure itself, namely the actual chain of blocks designated by the chain selection rule.

4.1 Consensus

The blockchain has a consensus algorithm in the classical BFT algorithm family that is adapted to use Proof-of-Stake-based voting power for a dynamic validator[1] set. A designated platform token, described further in section 5, is used by the validators to stake and is also the unit in which they are rewarded. There is a growing set of such consensus protocols [2, 3, 4], and the following general properties of these systems are of importance to the protocol design.

- **High throughput and low latency:** In benchmarks, these algorithms have been shown to support combinations of confirmation latency, transaction throughput, validator count, and geographic distribution, which are substantially more attractive than that found in typical production Nakamoto consensus chains [5].

- **Light client friendly:** The overwhelming majority of end users will need to securely access the platform in computing environments with resource constraints, such as browsers and mobile devices. They should also be able to quickly start interacting with the platform, even if the last sign-on was a long time ago or may even be the first time. In addition, the Joystream protocol will have a large state and transaction history. These constraints make a light client protocol the only genuine interaction model for almost all users.

  In these algorithms, a light client only needs to track any potential changes in the validator set, which in practice changes quite infrequently, for example, due to the unbonding period induced delay and not in large increments. Once an up-to-date validator set is identified, the client can securely read any part of the platform state by authenticating merkle proofs from full nodes against state commitments found in the relevant block header.

  This is in contrast to Nakamoto consenus systems where all block headers starting at genesis[2] must initially be downloaded, and one must keep up with new blocks as the system moves forward[3]. Headers are validated, and the chain selection rule is applied on a continuous basis. Even if feasible, this requires a long initial synchronization period whenever a client comes on line for the first time or at some point after a hiatus.

- **Finality:** The finality of block commitment is the property that once two-thirds of the current validator set has signed off on a block, then that block will become, and remain, part of the chain permanently. This is in contrast to Nakamoto consenus systems where the heaviest chain selection rule can in principle fork off any block, albeit with exponentially declining probability in the block depth, from the chain. This property has a number of critical benefits

  – *Easy interoperability*

[1] We will refer to a block producing actor as a *validator*, and anyone simply fully validating the chain a *full validator*.
[2] For reference, Ethereum has a chain of more than 7.1M blocks as of Jan. 22 2019.
[3] For reference, Ethereum commits 5760 new blocks per day as of Jan. 22 2019.

In order for secure assertions about the state of the Joystream chain to be feasible on a different chain, this chain will effectively need to behave as a light client that can track the most up-to-date committed block on the Joystream chain. Finality ensures that tracking this becomes very easy, as there are no reorganization events that can occur. Such events open up the possibility of reverting the basis critical state changes executed on the remote chain prior to the reorganization, e.g. changes in asset ownership. Dealing with this is complex and will often involve introducing delays. Moreover, the light client friendliness discussed above also helps in reducing the information to be submitted to the remote chain light client.

– *Safe launch and coexistence*

Finality ensures that the incentive to attack a chain to perform a double spend through a reorganization goes away. This risk is particularly high in the early stage of the lifetime of a new blockchain, as the initial amount of work (or stake) on the system may be particularly low. Even beyond the launch, the amount of value securing the system will always fluctuate, in particular for nascent systems; thus, finality provides a valuable guarantee.

– *Better usability*

Finality makes it very easy to write applications that interact with the blockchain, as complex logic for gracefully dealing with reorganizations is entirely omitted. There is no need to introduce arbitrary security delay, which is also a benefit to end users.

4.2 Upgrading

It is possible to upgrade both the transaction validation rules and the current state of the chain through the transaction processing system itself. This can typically be enabled by storing the transaction validation rules in the state and then running them on top of some virtualization layer. This property is a requirement for the following:

- **Genuine accountability:** Without a formalized mechanism for both measuring the preferences of stakeholders on collective action decisions and exercising those decisions, there will be an inevitable development of off-chain social conventions and authorities who will operate as stuarts and coordinators in such scenarios. Such actors are not accountable to platform stakeholders in any well defined or transparent way, which is undesirable in itself.

- **Faster iteration:** For the protocol rules to quickly evolve, the process to coordinate around such changes must economize on critical ecosystem resources, such as attention, information processing, and legitimacy. Opaque off-chain upgrading processes risk becoming a perpetual source of conflict around questions of legitimacy. Moreover, the requirement for active upgrading of validator software and the non-committing signaling games that often surround such events are an additional practical friction. Instead, endogenous upgrading with an accompanying on-chain immutable record of deliberation and a history of such updates can offer an effective remedy against these difficulties.

- **Developer fungibility:** In general, a complex system where any failure is catastrophic will require constraining the number of developers who can securely contribute to the improvement of the system.

This constraint can in extreme cases support a market power in the hands of key developers, where they can end up becoming gate keepers for any change to be applied.

In the context of upgrading blockchain systems, the reliance on off-chain upgrades to the consensus rules, the state transition function must incorporate the full history of rule sets from genesis to the most recent changes to validate all blocks. In practice, this ends up with a monotonically increasing complexity confronting developers wanting to comprehend and modify the system, which is problematic, as explained.

4.3 Asynchronous Transaction Processing

Certain types of transactions that may take a long time to process are occasionally required. Specifically, they may even require more time to process more than what is feasible when consuming all the the block times across multiple blocks. In some cases, it may be possible to accommodate such processing by the following approach, referred to as Asynchronous Transaction Processing (ATP). When the transaction is processed, and is valid, an appropriate subset of the blockchain state is locked in the sense that all other transaction types that mutate it are considered invalid. In order to implement this in an orderly fashion, where it is practical to reason about what transactions are impacted by a given subset, one should confine the approach to suitable transaction types. Now, during this locking period, which typically will last for a predefined number of blocks, the validator nodes are free to conduct the desired processing for the initial transaction outside the normal sequential validation of blocks, for example, on a separate processing unit. The locking prevents any race condition. Finally, when the time has expired, the finalized processing result is committed to the blockchain state at the end of the corresponding block, and the substate locking is no longer in effect.

4.4 Fee Model

A standalone chain allows the freedom to implement a custom fee policy. Building on top of an existing chain would result in inheriting its existing fee model for on-chain capacity, which often is tied to generalized congestion control and financing security.

4.4.1 Models

There are two types of fee model modes available in the protocol.

- **Transactional:** The normal transactional pay-per-use model can be found in a majority of currency-focused systems such as Bitcoin. This mode applies to basic operations such as moving fund.
- **Block Range Action Quota (BRAQ):** An *action* refers to a combination of an actor, role, and transaction type. This model prevents a given action from occurring more than a given number of times over a given number of blocks. Successfully issuing such transactions within the given limits does not involve marginal outlay for the given actor.

Given that BRAQ will cover the majority of transactions, security will primarily be financed through minting tokens, which also more closely matches the public goods nature of chain security. This dual model has a number of critical benefits for platform.

First, it drastically reduces the transactions costs of onboarding new users who initially have no tokens and either face prohibitive costs in acquiring, storing, and using them or need to be persuaded about the value of the platform before being willing to incur such costs. Such users will instead face the option of being onboarded via a screening mechanism (see section 10). After this, they will be able to immediately interact with the system, within constraints. Users may later earn or purchase tokens to escape quota limitations. Alternatively, the platform itself may simply eventually abandon the free policy once these costs have declined sufficiently as a result of general increase in maturity of the blockchain ecosystem.

Second, this approach explicitly subjects the fee model to a dynamic governance process where all default limits and individual member quotas can be adjusted. This is an easy and possibly the only feasible long term mechanism for correcting the externalities associated with the long-term social cost of transaction processing and state utilization in public chains [6].

4.4.2 BRAQ

In the BRAQ model, two parameters are required for each action: the *range length*, denoted by R, and the *action quota*, denoted by Q; both are positive integers. A combined range length and transaction limit is known as a *BRAQ quota*. This model prevents an action in a given block if there are already L such actions in the current R blocks prior. The system must maintain a sequence of positive integers $H = (h_1, \ldots, h_N)$, known as the *event list*, for each action, initially set to an empty list. A combination of BRAQ quota and such a list is referred to as *BRAQ instance*. An action should not be allowed in block height h when

$$L \leq |\{h_i \mid h_i \geq h - R \text{ for some } i = 1, \ldots, N\}|$$

otherwise it will proceed to normal validation. If the corresponding transaction is subsequently found to be accepted, then the new value for H should be

$$(h, h_1, \ldots, h_M)$$

where $M = N$ if $N < L$, otherwise $M = N - 1$. Hence, H is a list of block heights for, up to, the last N successful actions. The benefit of the event list is that it makes it easy for a light client to track the availability of a given action at any given time, as the entire BRAQ instance is securely available in the state. A full node could in principle track and enforce an instance without having the H in the state.

Lastly, when using this model, it is in some cases desirable to share the same quota across a range of instances. This is typically if you are dealing with a very large number of instances all of which have very similar actors. At the same time, it is also ideal to retain the flexibility to have explicitly custom quotas for specific actors based on policy or discretion. These two goals are accommodated by the concept of a *BRAQ quota proxy*. This is either a normal quota or an identifier resolving a normal quota in some context-specific pool of quotas or simply a sentinel to use some other context specific quota.

4.5 Interoperability

Some of the assets that are of value to the platform will inevitably not live in the state of the same blockchain. This can, for example, be the state of a DNS mapping that lives on some other system, such as ENS [7] or Handshake [8]. It may even be desirable to move certain assets from the Joystream blockchain, such as

tokens, onto other blockchain systems. While proposals are being developed to support very general inter-blockchain transaction routinely, such as Cosmos [5] for tokens and Polkadot [9], it is not clear when or if they will be fully deployed, and to what extent the Joystream blockchain will be able to benefit, or more simply whether the Joystream governance process will converge on such an integration.

A direct and simple case-by-case integration will be possible by using the same standard technique in almost all secure blockchain integration proposals. For each blockchain that needs to be integrated with Joystream, deploy a light client instance on each side, for the opposite side, where there is a requirement to write to the state on the first. This obviously requires that the given side is expressive and economical enough to support such an on-chain light client. Take the example of wanting to subject an ENS mapping, on Ethereum, to the governance processes on the Joystream blockchain. This will only require deploying a Joystream light client contract on Ethereum, which also is set as the owner to the given mapping. Block headers committed to the Joystream blockchain will then have to be submitted to this light client, at the very least when there are changes in validator sets. A designated party from the Joystream side can be incentivised to regularly conduct this. This header history in the state of the contract will allow it to be securely convinced of the ENS-related signaling actions on the Joystream side through Merkle proofs, since these will be committed to in the corresponding headers[4]. The contract can then in turn execute the corresponding on-chain contract call to put this signal into effect in ENS.

5 Accounts and Tokens

The platform has a standard for account-based (fungible) token ownership. There will be a primary token, called the *native token*, which will be part of the initial state of the platform. This native token will, as will be described later, be used for value transfer, bonding, and governance activities.

New tokens are minted continuously for a very wide range of purposes, all primarily to reward some actor for some behavior. This minting is perpetual, and how much is minted for what purpose is controlled by the platform governance process. New tokens are also burned by actors in scenarios where they need to contribute to the platform through the token; e.g. purchasing ad placements. There is no platform treasury that will otherwise absorb these tokens. As a result of these three separate exogenous dynamics - and the platform upgradeability functionality, there is no ex-ante certainty about the total supply of the token at different points in time in the future.

The other tokens are related to the content finance market, described further in section 15. They are issued by publishing a *token profile* into a registry called the *token registry*. A token profile includes key information, such as a standard token symbol, issuer identifier, description, and also the ownership state itself known as the *token balances*. The token balances simply maps a public key to a positive integer, and a single mapping represents the quantity of tokens under the control of whoever holds the private key corresponding to the public key. The registration of a key in the balances of a token is referred to an *account*. Reuse of the same key pair across accounts for the same actor is an individual policy choice. Normal spending operations can be conducted from an account to any new or existing account on the same token by signing a message with a private key that matches the public key corresponding to the original account. The token registry is a mapping from a *token identifier*, which is a unique positive integer, to the corresponding token profile.

[4]This could be in the event logging system, or individual transaction types being included in blocks.

The native token has identifier 0.

Note that the following account model is distinct from the smart contract account model in general, where all transactions are tied to a given account, although such platforms also have an account-based token ownership model. Transactions are instead on Joystream, at least most of the time, tied to membership, which has been explained in further detail in section 6.

6 Member

6.1 Overview

The membership concept is meant to unify all platform-level participation for the same actor in a way that is independent of token ownership in the account system. This means that all platform level actions are with reference to a particular membership. A given member may of course occupy a range of different roles through the same membership. Membership is conceptually a base role in itself. Having an integrated representation of the participation of a single actor is very valuable in supporting efficient communication and collaboration and supports pro-social investment in the actor's reputation. Separating this from token holdings is valuable, as it allows for some type of participation to be possible without tokens, for example, as a means for actors to earn their first tokens

6.2 Member, Profile and Registry

A *member* is an actor who is registered in the *membership registry* and is defined as follows:

Member

- **ID:** Unique integer identifier.
- **Key:** Public key. This is the key used to authenticate transactions as a member.
- **Handle:** String used as human readable identifier (UTF-8).
- **Avatar:** Storage identifier for avatar image (see section 11).
- **Description:** String of capped length (UTF-8).
- **Added:** Date and time when the membership was first established .
- **Entry:** If membership was established through screening (see section 10), the this is set to ID of screening authority which created membership. If it was established through payment (see section 6.3), then this is the paid term ID of the terms. Otherwise, its blank.
- **BRAQs Instances:** Set of BRAQ instances for all base member actions (see section 4.4.2)
- **Suspended:** Whether member is suspended.
- **Subscription:** If at least one subscription has been entered, then this is the date and time of that event and the corresponding subscription term ID (see section 6.4). Otherwise its blank.

The membership registry is simply a mapping that associates the identifier (ID) of a member with the corresponding profile. Lastly, there is also a set of BRAQ quotas, called the *default membership quotas*, used for all base membership BRAQ instances with indirect proxy quotas.

The suspension field only impacts a member's capacity to act through their base membership capacities; any action derived from other roles is unaffected. Also, a member may be suspended even if this field is not set (see section 10.4).

A membership may be established for free, as explained in section 10, or it may be paid for a one time cost of burning a given amount of tokens. As a result of free entry, a given key may be associated with a membership but no account. The converse is also possible by definition. Once a membership has been established, it is permanent. For a given key, there may both a corresponding account and membership, or either one exclusively. While an actor may find it practical to identify with the same key in both capacities, the system cannot, and does not, enforce this.

6.3 Paid Membership

Paid membership terms represent a set of conditions for a prospective membership, through payment, on the platform, and is defined as

Paid Membership Terms

- **ID:** Unique integer identifier known as *term ID*.
- **Fee:** Quantity of native token that must be provably burned.
- **Proxy Quota:** Initial quota for membership.
- **Text:** String of capped length (UTF-8) describing the human readable conditions to be agreed upon.

The platform has a set of terms, called the *active paid membership terms*, which are currently in place for anyone seeking paid membership. It is updated through a council proposal. Any new terms introduced by the council must have an ID greater than the prior active terms; the initially active terms have an ID of 0. The full history of such terms that were once active is kept in a list known as the *paid membership terms record*, which maps the term ID to the corresponding terms.

A new actor may join as a member at any time through a request, which will burn the required fee from their account, so long as the platform is accepting members. This is gated by a platform variable referred to as the *platform membership gate* and can be changed through a proposal.

6.4 Subscription

Subscription terms represent a set of conditions for a prospective subscriber on the platform and are defined by the following:

Subscription

- **ID:** Unique integer identifier.
- **Fee:** Quantity of native token that must be provably burned.
- **Duration:** Number of blocks for which the subscription is valid.
- **Proxy Quota Delta:** Set of proxy quotas added to the base level quotas to expand quotas.
- **Text:** String of capped length (UTF-8) describing the human readable conditions to be agreed upon.

A member with a subscription that is *active*, that is, the current block height is lesser than the sum of the subscription entry time and duration is referred to as *a subscriber* in this period.

Similar to that of paid membership and terms, there is an analogues concept of active terms, terms record, and a gate.

While members can establish subscriptions at any time, the time line is divided into *subscription periods*. In each period, a cumulative count of the total amount of fees burned for subscriptions is maintained. At the end of each period, payouts to relevant parties such as publishers, for example, are based on these final tallies.

7 Roles, Staking and Slashing

A *role* is a membership status having a fixed number of varieties, which gives access to a range of different rights and correspondingly confers a range of responsibilities. A given member may occupy multiple roles simultaneously.

An actor may be required to lock up a certain amount of native tokens for some time under certain conditions, and this is referred to as *staking*. Typically, this is in the context of participating in some role or performing some action. In some roles, it is possible to raise or lower the staked balance on an ongoing basis within context-specific limits. There will often be a time before a change in the staked amount of tokens is counted toward the actual total staked balance. For increases, this is known as *bonding period* and for decreases *unbonding period*. If a stake reduction leads to the actual staked balance dipping below the minimum required for a given role at that time, then a full stake balance reduction is automatically initiated, and no increases can be initiated until the unbonding period is over. Tokens that are in one of these two periods are referred to as *in flight*. Tokens staked or in flight cannot be reused to stake in another context at the same time.

Under certain scenarios, it may be possible for a member to lose all or a part of their stake; this is referred to as *slashing*.

8 Rewards

All staking is *rewarded* in tokens paid out directly to the member account and, if no account exists, then the membership key will be used to credit an account with the same key. These payouts will come at the end of some corresponding time interval and are comprised of two components. The first component, known as the *compensation payoff*, is of the form xr^T, where x is the staked quantity, r is a global nominal per block interest rate, and T is the number of blocks in the given period. If the staked quantity has varied over the period, then x will represent the average time weighted quantity. The second component, referred to as the *earned payoff*, is related to the particular activity or role that was undertaken in the given period. Hence, the total reward to a given stake in a given period may be written as

$$\frac{\sum_i x_i \Delta_i}{T} r^T + C$$

where x_i is the quantity staked in the i-th sub period, which itself lasted Δ_i blocks, and C is the earned payoff.

9 Governance

9.1 Council

9.1.1 Terms

The governance process is divided into a sequence of periods known as *terms*, and the first term is known as the *bootstrap term*. Each term, with the exception of the bootstrap term, has a corresponding *council*, which is a set of staked members responsible for voting over submitted *proposals*. Proposals are bids to execute given operations on the platform state in order to serve some contemplated end.

9.1.2 Elections

The council for a given term is established through a *council election*, where all members have the opportunity to place weighted backing behind prospective councilors who announce their bid for council membership and put up their own corresponding stake. The election is conducted toward the end of a term, and current council members are expected to carry out their responsibility of dispatching proposals throughout their term. The number of council positions in the next term is always set to a given number in the preceding term referred to as the *council size*. This may be altered through a proposal. A new number goes into effect at the start of the next term.

The election has four stages.

- **Announcements stage:** Members get to announce their candidacy for the council. They need to put a minimal amount of stake to be able to do this, which is known as the *council staking limit*. This limit may be altered through a proposal, and the new limit may come into effect at the start of the next term. When not in the bootstrap term, there may already be existing council members who are also welcome to announce for the next term, which denotes *extending candidacy*. A council member may in this case reuse their existing stake, which is referred to as *transferring council stake*. In this case, they may have to adjust their staking amount to satisfy a staking limit that may have been altered. There is an upper limit to the number of candidates that may be voted upon and is termed the *council candidacy limit*. This limit may be altered through a proposal, and the new limit may come into effect at the start of the next term. If there are more candidates satisfying the staking limit than this limit, then candidates are ranked based on the staked amount. The set of candidates who actually end up being eligible after this constraint is applied and termed *candidate pool*.

- **Voting stage:** All members, including current and prospective council members, *back* candidates for the next term. Backing refers to staking in support of the candidacy of a member in the candidate pool. Each backing is a sealed commitment to a particular candidate, where the seal is simply salted hashing the candidate identifier. A member may reuse tokens already staked behind an existing council member to a member of the candidate pool, which is referred to as *transferring backing stake*.

- **Reveal stage:** All members who backed candidates must submit their salts to *reveal* the candidate in each of their backings. At the end of this period, all backings are tallied for all members in the pool, and all backings without a corresponding revelation are ignored. Regarding considering the council members during tallying, the candidate pool members are ranked in terms of the total amount of stakes across all revealed backings in their favour. All those outside the council size are discarded and, if the council size is not filled, then one can re-enter the announcement stage.

- **Grace stage:** All stake introduced in the given term backing a losing candidate or not revealed, is immediately unstaked. This is done so that all new stake is able to exit the consequences of an unfavourable council. At the end of this period, a new term begins with a new council. This is also is the start of the unbonding period for all stake that does not continue in the next term.

9.1.3 Rewards

Voters do receive an earned payoff; however, council members receive a payoff proportional to the rate of participation in processing proposals, which is is of the form $p_i u^{\mathsf{CM}}$, where p_i is the *participation rate*, i.e., rate of non-abstention (and thus revealed) votes over all votes of the i-th council member, and u^{CM} is a base reward across all council members for the given period.

9.2 Proposals

9.2.1 Types

At any given time, there exists a finite set of different proposal types. Each type is a value for each of the following properties that apply to all proposals of the given type.

- **Quorum:** The percentage of the council participants who must vote affirmatively in order for the proposal to pass.
- **Threshold:** The minimum percentage of quorum that must vote for a given alternative for it to pass.
- **Constitutionality:** The number of council periods in a row that must confirm the proposal for it to pass.

A proposal type takes the form of one among three different types of propositions. The *binary proposition* is a simple pass or reject, the *multiple choice proposition* require selecting one among at least two different affirmative alternatives or rejection and, lastly, the *ranked choice proposition* requires providing a total ranking of a finite set of alternatives or rejection.

9.2.2 Life cycle

A proposal, of a given type, is first created by a member referred to as the *proposal sponsor*. The sponsor has to back the proposal with a given amount of stake. This amount may be altered through a proposal, and the new amount goes into effect at the start of the next term. Once a proposal has been created, the council members can start submitting sealed votes on the proposition in what is known as the *voting stage*. The sealed vote can be one among

- **Abstention:** Signals presence, but unwillingness to cast judgment on substance of vote.
- **Reject:** Against proposal.
- **Affirm:** Pass an alternative or a ranking for binary, multiple-choice, and ranked-choice propositions, respectively.
- **Slash:** Against the proposal, and slash proposal stake.

This stage ends on the earliest of the following events

(a) All council members have submitted sealed votes.

(b) Time since proposal creation has exceeded a designated platform parameter. This amount may be altered through a proposal, and the new amount goes into effect at the start of the next term.

(c) End of the given term, the point at which the proposal is automatically rejected by the council.

The next stage allows the council members to submit revelations for their prior sealed votes and is referred to as the *revelation stage*. At the end of this stage, the tallying commences, which works as follows.

If all revealed votes are slashes, then the proposal is rejected and the proposal stake slashed. To clear the quorum requirement, the percentage of council members with revealed votes must be no less than the quorum value for the given proposal type. To clear the threshold requirement, the percentage of council members voting in favour of the proposition must be no less than the threshold. For multiple and ranked-choice propositions, this is interpreted to mean the number of votes in favour of the most popular choice or single ranking, respectively.

When a proposal passes, it needs to immediately be put into effect; however, many proposal types have some applicability pre-condition that must be satisfied for it to be valid. If this does not hold at this time, then the proposal is simply discarded. This can occur in scenarios where the state of the platform changes in a way not foreseen by the initial proposal sponsor or council.

In all scenarios given above, an archived record of how a proposal was processed will be left for future inspection.

9.3 Working Groups

All non-validator service-provider roles on the platform are organized into domain specific groups termed *working groups*, which comprise the following.

1. **Membership screening:** Grant membership status to members while trying to avoid Sybil.
2. **Membership curation:** Monitor membership base for abuse and Sybil attacks.
3. **Content:** Curates and manages the availability and integrity of content in the content directory.
4. **Storage and distribution:** Stores and distributes static data to consumers on demand.
5. **Live streaming (in future draft):** Distributes dynamic video data to consumers.
6. **Discovery:** Provides standard discovery services over the content directory to consumers, i.e., search and recommendations.
7. **Software development:** Develops and deploys all software assets of the platform, including consensus code and user facing applications.
8. **Content finance:** Curates content finance market and adjudicates project disputes.
9. **Advertising:** Polices the advertising market on the platform.
10. **Communications (in future draft):** Administrates the on-chain forums, messaging channel governance, and user support inquires.

9.3.1 Leads and workers

Each group has two distinct types of roles, namely the *group lead* and the *worker*. Leads are elected by the council through the proposal system and are responsible for populating and managing other roles in the given group as well policing the conduct of group workers. Leads can also be replaced or evicted and, if the latter, or if there is no lead to begin with, then no group workers can perform any platform-level actions until a new leader is installed. The particular rights and privilege of a worker is entirely dependant on the group in question.

Each group pays out a reward to all group members at a given interval referred to as the *group payout period*, which is a platform parameter distinct for each group. This may be altered through a proposal, and the new value goes into effect at the start of the next period.

9.3.2 Installing, replacing, and evicting leads

There are two scenarios under which a new group lead may be introduced in a given group: in the case where there is no existing lead, which is only the case while bootstrapping, and is referred to as *installation*; when an existing lead is leaving their position by their own initiation or alternatively by initiative from the council, and a new one is to be introduced. These are referred to as *replacement* and *eviction*, respectively.

In all cases, candidates for a group lead role come from a list on the platform for the given group, referred to as the *group lead candidate list*. Members can enter the list by staking the amount required to hold the group lead position for the given group. Each list is a fixed size and, if the number of potential candidates who have staked is greater, then inclusion is determined by how much has been staked beyond the limit, although the actual staked amount when introduced a lead is always the staking limit.

10 Membership Screening and Curation

10.1 Overview

As mentioned before, for example, in section 6, it should be possible to onboard users who do not hold any tokens. This activity, referred to as *screening*, allows the platform to accommodate new users who wish to try the platform within certain constraints. However, the combination of no robust identity system and allowing users to consume resources without paying on the margin creates its own set of new problems. These resources need not only be in the form of tangibles such as compute and storage, but can also be things like peer member attention. Not only does this result in wasteful overutilisation but, perhaps, more importantly, it allows platform participants to shape their own payouts at a low cost, since resource use is often tied to payouts. For example, viewing content will divert a larger share of payout to the publisher.

The Joystream protocol also suffers from these ills in principle, at least under a certain set of policy choices. There is no final solution to this without dealing with the fundamental identity and pricing preconditions. None the less, it is critical to equip the platform with a baseline capacity to engage in the presumed cat-and-mouse dynamic with such abuse to reduce the feasibility and cost of attacks.

This is achieved by two measures: first, there is a capacity to block access to the platform in various ways. This activity is termed *suspending* and specifically prevents the relevant actor from being able to participate in the platform through base membership actions. Second, when anyone is onboarded through

screening, the corresponding platform actor who onboarded them is recorded and can be punished for a long time in future, with a suitable bonding period if the abuse is uncovered.

10.2 Working groups

This work is parceled out into two complementary working groups, the screening and curation working groups. Both groups allow all members to engage in the same set of group activities, except for the lead, which is also involved in normal group lead activity.[5]

10.3 Screening

Participants in the screening working group can engage in the screening process and are in that capacity referred to as a *screening authority*.

The precise steps involved in screening, which is an offchain process, are entirely exogenous to the protocol. The set of actual policies and corresponding tools in production at any given time is expected to be the result of an ongoing coordination, based on policy constraints from the council and group lead. Obvious mechanisms that can be employed comprise the following:

(a) Email or social media confirmation.

(b) CAPTCHA, or other labeling, or perceptual tasks.

(c) Onboarding delays.

(d) Manual human confirmation.

As part of being screened, a prospective member must accept a set of terms referred to as the *screening terms* and defined as

Screening terms

- **ID:** Unique integer identifier.
- **Proxy Quota:** Initial quota for membership.
- **Text:** String of capped length (UTF-8) describing the human readable conditions which are being agreed upon.

The terms to be used at any given time are known as the *active screening terms* and can be changed through the proposal system.

A new member can be added through screening authority by presenting a signature of the current active terms (or similar) using a key that can also be used as a basis for the membership. When this happens, the ID of the authority is also included in the membership (see section 6.2).

[5] In principle it is probably advisable to keep the screening group small, even perhaps to a single actor at any given time, in order to keep the set of stakeholders in this capacity limited. This also makes a longer unbonding and staking period more economical.

10.4 Suspension

The organization and principles of this working group are identical to the screening working group. There are three distinct forms of suspension that can be initiated, namely

- **Individual:** A specific member can be suspended by setting the appropriate membership field.
- **Group:** A proposal can be submitted allowing for the simultaneous suspension of all members who have been added through screening from a given screening authority within a given window of time in the past.
- **Lock down:** A proposal can be submitted allowing for simultaneous suspension of all members.

10.5 Rewards

All working group participants have some given group-based periodic payoff set by the council.

11 Data storage and distribution

11.1 Overview

Reliably storing and distributing off-chain static data at scale is one of the primary requirements of the platform. This includes data such as media content, metadata, applications and static assets, and private (encrypted) member data such as preferences and statistics. Specifically, this subsystem should satisfy a range of objectives

- **Consumption:** Members can
  (a) upload content, respecting certain platform quota limits, for free and expect permanence and distribution.
  (b) securely download data, respecting certain platform quota limits, for free.
  (c) fully utilize a system with communication and resource constraints of a browser.
- **Paying:** Quota restrictions can be augmented as a result of paying (e.g. through subscription) or discretionary changes made by platform governance.
- **Fault tolerance:** Storage has some level of fault tolerance for all content, which ensures that single actor faults do not lead to permanent data loss. This replication should economize on cost and be sensitive to platform policy regarding the desired rate of tolerance.
- **Dynamic:** The platform should dynamically be able to be and remain effective at distributing content in concert with changes in total volume of downstream demand and where this demand is located.
- **Privacy:** View counts and quota limits are maintained without leaving a permanent public history of what end user viewed what and sharing viewing activity with as few counterparties as possible.

- **Ad Awareness:** It may be desirable to interleave media distribution with a dynamic advertising system and, in order to support this, the infrastructure must know when to inject what ad, to whom, and how long to block until normal distribution is resumed.

- **Low Bar:** Barrier to entry for aspiring service providers is not too high, for example, in the requirement of lots of hardware or stake.

The key problem in organizing this part of the platform is correctly enforcing rewards and punishments on the infrastructure providers. This requires that one accurately determines whether they are behaving correctly. At the core of this assessment lies the problem of how to adjudicate disputes over the timeliness and integrity of response to queries. It is fundamentally not possible to furnish direct purely cryptographic proofs about such claims; hence, the fallback of most alternatives is to construct mechanisms that provide game theoretic assurances of honest and reliable conduct. These alternatives most often attempt to be open commodity markets, where buyers and sellers have temporary relationships. Here, a much simpler approach is utilized, where most good conduct is expected based on risk of governance-based sanctions on deposited bonds, and losing reputational capital. Honesty of the platform, and thus its governance outcomes, is assumed to be reliable due to being a long-term player.

11.2 Working Group

The working group is made up of the following roles

- **Lead:** Has the normal group-lead responsibilities, but also performs two additional coordinating functions, namely

  (a) Produces policy for how different data should be distributed at any given time.

  (b) Receives and processes errors about misconduct or unavailability among group members.

- **Storage:** Stores a copy of some subset of data in the data directory and replicates to peers and distributors upon request.

- **Distributor:** Distributes data in the data directory on demand to members.

11.3 Storage providers

A storage provider is member of a group of providers termed *storage tranche group* who share the same participation terms. A tranche is defined as follows:

Storage provider Tranche

- **ID:** Unique integer identifier.
- **Capacity:** The number of bytes of storage capacity required.
- **Out speed:** The number of Mbps required.
- **Stake:** The amount of stake required.
- **Duration:** The number of block service to be provided from the initiation block.
- **Fixed Reward:** Quantity of native token earned per unit of time.
- **Variable Reward:** Quantity of native tokens earned per unit of date stored.
- **Slots:** Total number of roles available in this tranche.
- **Terms:** String of capped length (UTF-8) describing the human readable conditions to be agreed upon.
- **Active:** Whether tranche is actually possible to use, that is, assign new members.

All these are kept in a mapping known as the *tranche registry*, which maps tranche ID to the corresponding tranche. Even when a tranche is no longer active, or will be used in the future, it continues to be a part of the registry.

The participation of a member in the role as a storage provider is represented by a *tranche membership* and is defined as follows:

Tranche membership

- **Member:** Membership ID.
- **Tranche:** Tranche profile ID.
- **Started:** Block height where role started.
- **Availability:** Amount of bytes still available.
- **Active:** The block at which this provider is to be considered active.
- **Paused:** Whether this role is currently paused.

Such profiles live in a mapping from the ID to the profile termed the *storage providers*. A member is only considered a provider when the corresponding profile is registered in this map, and this happens through the group lead. The active status of a membership is meant to afford some time between when the provider has entered the role and when there is a full obligation to provide service. The pausedness of a membership is meant to allow discretionary pause in inbound storage requests due to extraordinary circumstances and can be invoked by the member or lead directly.

Note that the same member may participate as a storage provider multiple times and under different tranches.

11.4 Distributors

Distributors are organized in a very similar way to storage providers, with analogous concepts such as *distributor tranche group*, *profile*, *registry* and *membership*, with suitable minor alterations. In particular, tranches will also include information about geographically bound latency guarantees and the number of simultaneous upstream connections.

11.5 Data directory

The platform maintains state about the available data, and how it is distributed, in the *data directory*. All data objects stored correspond to one among a finite set of *data object types*. Each type is meant to capture the following storage and distribution requirements for some broader family of objects:

- **Infrastructure requirements:** By allowing a range of guarantees about permanence and performance, which better aligns with the underlying requirements of different data objects, one can allow better resource utilization.
- **Access policy:** Some objects may only be accessible to a given member for certain periods of time, if at all. The obvious example will be data objects behind paywalls.
- **Accounting procedures:** Some objects may require some kind of accounting or cleanup as a result of accessing the data. This can, for example, be used to record reliable access statistics for media content, i.e., view count.

The first requirement is about reducing cost while the latter two about making the same infrastructure parametric and thus reusable for a wide range of purposes. A data object type is specifically designed as follows:

Data object type

- **ID:** Unique integer identifier.
- **Description:** Human readable description.
- **Size limit:** When set, represents the maximum number of bytes.
- **Replication factor:** Number of copies for data objects of this type that need to exist in the storage system at any time. The simplest interpretation for this is the minimum number of storage providers that must replicate the data object.
- **Storage tranches:** Set of tranche IDs that can be used with this data type. If empty, then any tranche can be used.
- **Active:** Whether objects of this type can be added at this time.

All available types are kept in mapping termed the *data object type registry*, which maps the ID to such a type. Admissible storage tranches are in part kept on the platform to allow automatic assignment of storage providers to any new data object of a given type. For the same reason, there is no commitment to the distributor tranche because distribution requirements will often depend on internal details about the data object itself, which are hard to fully describe.For example, it can be the case where whenever a particular publisher uploads a media item, there are substantial inbound download requests from a corresponding area. This type of policy rule is better left to discretion to setup rather than an automatic consensus rule.

Each data object is defined as follows:

Data Object

- **CID:** A content identifier that allows secure authentication of the data under some implicit chunking schema.
- **Type:** Data type ID.
- **Size:** Number of bytes occupied by data.
- **Added:** Date and time for original upload event.
- **Origin:** ID of member who uploaded the data.
- **Liaison:** ID of storage provider that accepted the initial upload.
- **Liaison judgement:** One among *pending*, *rejected*, and *accepted*.

All records are kept in the *data object registry*, which is a mapping from the CID to the corresponding record. The fact that a storage provider is storing a data object is represented by a *data object storage relationship*, which is defined as follows:

Data object storage relationship

- **ID:** Unique integer identifier.
- **CID:** CID of data object.
- **Storage:** ID of storage provider which should store object.
- **Ready:** Whether the service relationship is ready to be honored by the provider.

Such relationships are kept in the *storage relationship registry*, which maps the ID to the relationship. Analogous concepts for distribution that are referred to as *data object distribution relationship* and *distribution relationship registry* also exist.

Lastly, a member downloading a data object is represented by a *download session*, which is defined as follows:

Download session

- **CID:** ID for content.
- **Consumer:** ID for member downloading.
- **Distributor:** ID for distributor that distributes the content.
- **Initiated:** Date and time when the session was initiated.
- **State:** Either *started* or *ended*.
- **Transmitted:** Amount of bytes of data actually downloaded.

Such sessions are kept in the *download session registry*, which maps the ID to the session. Please see the discussion (section 18) on how to address obvious scale and privacy problems introduced by sessions and the registry.

11.6 Uploading

A normal uploading flow is as follows [6]:

1. The user issues a transaction to create a new data object record by providing a CID for the underlying payload as well as information about its size and type ID. The transaction is only valid if
   (a) the CID is not already in the data registry.
   (b) the size respects data type size limit.
   (c) data type is active.
   (d) sufficient storage capacity is available among active, non-paused, storage providers within the set of tranches available for data type.
   (e) uploader has no other data object records with pending liason judgment.

   If so, this will result in the creation of a data object record, where the liason and storage object relationships are automatically assigned. All the latter have an initial status of not being ready. The liason judgment is set to pending. If the record remains pending over some time limit, then the record goes away, and the lead is needed to inspect any received error reports.
2. The user connects to the liason and requests to upload the data by referencing the data object record. The storage provider can validate the request by reading the platform state and can only then proceed to accept the upload.
3. The storage provider will check
   (a) the payload matches the CID,
   (b) has the right size, and
   (c) passes an upload filter[7]

[6] This is a highly informal description which will be fleshed out in future drafts.
[7] TBD.

If either fails, the judgment will be set to rejected for the given reason. If all pass, the judgement will be set to accepted, and the corresponding storage relationship will be set to ready.

4. The liason must accept incoming replication attempts from all other providers with a storage relationship with the given record. As they receive the payload, each has a responsibility to alter the readiness of their storage relationship. Any relationship that is still not ready after some defined period of time from the time the record has been added is considered a failure. At this point, the lead must inspect any possible reported errors and adjudicate.

5. When the time limit for replication by all storage providers has been exceeded, the lead creates distribution relationships for the record based on local policy.

6. The distributors corresponding to the new relationships can connect to the storage providers that have corresponding storage relationships with ready status to acquire a copy of the payload.

11.7 Downloading

A normal downloading flow is as follows [8]:

1. The downloader issues a transaction to create a download session by providing the relevant CID. The transaction is accepted once the following holds:

   (a) the CID corresponds to a data object record

   (b) the data object record has at least one distributor relationship that is ready

   (c) the access policy of the data type accepts the request

   If so, then a session is added to the session registry, which is in the started state and where the distributor has been chosen from the available ready set.

2. The downloader connects to a host corresponding to the assigned distributor and authenticates and finally provides a verifiable reference to the new session.

3. The distributor sends verifiable (based on Authority Key) data chunks in response to requests from the downloader in a tit-for-tat exchange. The downloader sends to the distributor a signature over the claim that a certain total amount of data has been transmitted over the lifetime of the session, analogous to payment channels [10]. To avoid latency, there should be some amount of pipelinining.

4. The exchange ends when either all data has been sent or when the downloader has terminated the exchange. At this point, the distributor will submit a new transactions with the most recent signed consumption statement from the downloader. This transaction will settle both the actual consumed data in the session and the state of the session.

5. Any data type specific accounting policy is executed with reference to the session.

[8] This is a highly informal description which will be fleshed out in future drafts.

11.8 Entry, exit, and distribution policy updates

There are a number of key infrastructure dynamics, such as entry and exit of actors into the roles as distributors or storage providers, and also updates on what is being assigned to what data. These may be initiated and coordinated through off-chain messaging protocols. An honest conduct is expected purely on governance sanctions. This will be described in further detail in the future.

11.9 Policing and data removal

The storage and distribution infrastructure is really a utility for the rest of the platform, hence the removal of data from this infrastructure, for any reason, is under the control of the working group which has control over the use case to which the data corresponds.

11.10 Rewards

All payouts take place at a given group-specific payout interval. The group lead is paid a given amount of tokens regardless of what has occurred. Storage providers and distributors both have a fixed and variable payout component, where the latter is based on the actual stored or distributed data quantities, and the rates are captured in the corresponding tranche profiles. The actual variable base quantities are maintained through the uploading assignment process and data type-specific accounting policies, respectively.

12 Content directory

12.1 Overview

The content directory contains information about the media content available on the platform. More importantly, it does not contain the primary media itself, which is stored in a separate off-chain infrastructure described in the section on storage and distribution (section 11).

12.2 Working group

The working group comprises the lead and the member. Members are responsible for executing the following functions

- **Curation:** Ensuring that the underlying content media and metadata correspond to each other, e.g., in that they display assets are correct or content is in the correct category, etc.
- **Policing:** Adjudicating disputes around the availability, ownership, and attribution of content.
- **Filtering:** Maintaining and developing the filtering technology in place when publishing to the directory.
- **Verification:** Granting privileged status to publishers and content as verified and canonical, which helps in discovery and resolves disputes over the publisher namespace.

12.3 Publisher

A *publisher* is a member who is allowed to publish content in the content directory and is defined as follows:

Publisher

- **Name:** This is separate from membership name and is its own namespace.
- **Description:** A description of the publisher.
- **Brand artifacts:** Data directory CIDs for a set of off-chain artifacts that make up the profile brand identity.
- **Verification status:** Whether the implied identity of the publisher profile matches the actor who is in control of the membership.

12.4 Content

There is a base set of properties for all content on the platform, which includes the following:

- **Category:** The type of content.
- **Payload:** Data directory identifiers for media and metadata.
- **Owner:** A publisher or content project planner (see section 15) who has control over the content and rights to value.
- **Monetization policy:** How end users must pay to access content, among which are being free (with or without advertising), transactional, or subscriber access (only subscribers have access).
- **Dispute status:** Whether some dispute is currently ongoing. This has implications for how end users should engage with the content.

There are a fixed set of primary content categories that are supported at any given time. Each category defines a particular schema for how to define key properties, including

- **Payload format:** How to organize the media payload.
- **Rendering:** Metadata about how to play back or render media.
- **Accessibility resources:** Things that assist a variety of users in consuming the content, such as subtitles or translation metadata, dubbing information, etc.
- **Attribution:** Defines who did what in the process of producing the content.

Content will also have associated social information around engagement, such as view/access rates and counts, likes, and a comment feed.

12.5 Disputes

Any member can submit a dispute about any published content, and these disputes are processed by the working group. There will be a range of different dispute forms, some with the goal of entirely removing content while others with changing attribution or ownership information and, as a result, redirect revenue streams.

12.6 Rewards

All group members are paid fixed amounts per unit of time.

13 Discovery

13.1 Overview

In order for end users to effectively discover relevant content in the content directory, access to the directory is required, and the ability to execute effective processing heuristics across this data needs to be set promptly. In order to alleviate the resulting processing and bandwidth costs, this will impose there is a designated set of discover nodes that provides these services.

13.2 Working group

The working group is composed of members that run nodes running a discovery provider service, as described in the next section.

13.3 Services

There are three types of discover services that are offered by discovery provider nodes

- **Search:** Keyword and filter-based ranked lists of content.
- **Browsing:** Category and filter-based ranked lists of content.
- **Recommendations:** Content identifier-based ranked lists of content.

In all three cases, the response provided by a service provider includes the relevant content as well as Merkle proofs, which allows the client to authenticate the integrity of the response. If required, one can extend this to have automated slashing based on bad proofs sent to clients. There is no way to prove omission in this scheme. There will not be a well-defined definition of what this may imply, as different discovery providers are free to pursue their own ranking and discovery policy. Reasonable incentives for good behavior can be generated by encouraging user clients to keep local statistics about the success rate of various providers or by measuring user behavior, which will in turn drive more traffic to better providers. Likewise, the incentive to generate more traffic can be generated by giving a provider the privilege of displaying in-place advertisement, which is not hooked into the normal advertising ecosystem. The inclusion of minimal telemetry feedback to providers can also help them guide the development of their own discovery policies.

13.4 Rewards

Beyond the incentive described in the prior section, all providers are given a fixed token reward per unit of time.

14 Software development

14.1 Overview

Software development needs to be understood broadly, encompassing all activities surrounding research, production and testing of standards (e.g. analogous to BIPs or EIPs), protocols, algorithms, source code, binaries and other digital software assets, as well as deploying such outputs into production environments. Three aspects of this activity are of importance to the platform.

14.1.1 Financing

Many protocols suffer from the lack of an endogenous financing mechanism, and stateless protocols cannot have them by definition. Even in many stateful protocols, key contributors end up either severly underfunded, which is detrimental to quality and development progress, or relying on third-party revenue sources that do not have incentives that are guided by the interests of all protocol stakeholders.

To address this, the platform has a dedicated working group of contributors who are rewarded for providing these sorts of services and are accountable to stakeholders through the governance process.

14.1.2 Development

For a variety of reasons, most software development projects for open stateful protocols end up with an equilibrum where the development process is organized around a canonical collaborative state. By collaborative state, we mean some sort of code base and social collaboration metadata about that code base. For example, the state may often be a Git repository, and the collaboration metadata may be repository hosting service of some kind, or it can be a curated mailing list of patches and discussion. Ultimately, some social consensus process emerges for how changes to this state is made and, almost invariably, this process becomes a source of market power, its integrity becoming a security risk, and is not formally accountable to protocol stakeholders. Socially desirable changes may be rejected or adopted too slowly, and changes that are undesirable may be adopted. While fork based exit is in principle an option, it is also expensive. This results in the following specific requirements:

- *Accountable canonicity*

  The platform provides the means by which everyone securely resolves the same collaborative state at all times.

- *Open contributions*

  Any member should be able to submit a proposal for changes.

- *Direct control*

  The governance process may directly mutate the state at any time.

- *Gated updating and moderation*

  Designated role(s) occupied by a dynamic actor set, subject to governance, have the right to accept contributions or make changes on an ongoing basis.

- *Immutable and secure updating history*

  An immutable history of all state changes. Even a Git repository only gives a history of states, but is devoid of secure information about whether a given commit was introduced by someone who had the right to do so. Merged commits are in this respect particularly sensitive, as this is typically the way new changes make their way into producing source snapshots. Moreover, Git does not include the broader collaborative state.

- *Robust availability guarantees*

  The highest level of guarantee is required around the actual availability of state.

Given this set of requirements, the platform maintains a set of on-chain Git repositories, which include familiar functionality, such-a-pull requests, merging, issue tracking, permissions, and publishing releases and test/CI results. Critical changes such as merging are conducted in consensus using ATP.

14.1.3 Deployment

Deployment is the process of converting a set of source assets into final production assets, like binaries, and distributing these securely, and possibly automatically, to end users. This requires the following:

- *Build authentication*

  Given a commitment to a set of source assets, there must be a reliable, reproducible, and secure way for anyone to determine whether a particular set of production assets are the correct output of this process.

- *Secure updates*

  There must be secure way for users to acquire or run new versions of software binaries.

Given this set of requirements, there is a set of standards for defining and conducting deterministic builds. Software updates occur directly from the platform state.

14.2 Working group

There are two group types: lead and contributor. The lead is responsible for creating and assigning permissions on projects to contributors in order to help them perform write operations directly. While anyone can in principle contribute through their effort to development, only contributors have a recurring reward for their involvement. The working group also has its own messaging channel.

14.3 Project

A project is the following set of items stored on chain:

1. Git repository.

2. Permissions for who, in the working group, is allowed to make write operations, i.e., push and merge into it.

3. A set of open issues and pull requests with a corresponding discussion thread.

4. A set of releases associated with tagged commits.

The normal write operations in Git repositories, such as initialization, pushing, merging, and tagging, are fully secured by the platform itself by having the Git processing rules embedded in the consensus itself. Opening issues and pull requests are open to any platform member, but only a working group member with suitable permissions can actually moderate issues or merge requests.

14.4 Artifacts, reproducible, and releases

An artifact is a file that is the result of some processing of the source material in a project repo commit. This processing may involve processes such as building, linking, and packaging. There is a format to describe such processes, and these will always yield fully deterministic outputs. These processes occur entirely off chain; however, the determinism about the outputs is critical in order to facilitate reliable coordination around the results, in particular around the validity of hash commitments of the artifacts. The most important process is the building process, where this determinism provides reproducible builds. A release is a simple publication of a set of artifacts corresponding to a tagged commit. This can only be done by the group lead and, if the proposed hash commitments turn out to be fraudulent, then they can be challenged through a proposal by anyone. If found to be a correct challenge, then the lead will face sanctions and the challenger will get reward.

14.5 Automated testing

There is a format for describing how to run tests off chain. The presence of tests' metadata prevents pull requests from going through unless the group lead signs on them as suitably passing. Just like in the release process, such sign offs can be challenged.

14.6 Deployment and upgradeability

There are two types of deployments on the platform. Light deployments simply involve updating the platform state to reflect a new version of some application is available, whether it is backward compatible, and thus optional, and how to retrieve the application itself.

The other type of deployment is heavy deployment and also involves some sort of change to the consensus as well. This change does not involve simply changing some platform parameter value, but rather making an exogenous change to the platform state or processing logic itself. Such a deployment may be just a set of consensus changes that support change in the behavior of a single application or a change in the platform that requires a concerted upgrade to a number of different applications simultaneously.

Both types of deployments are triggered when the group lead submit a proposal based on a release. When this proposal is accepted by the council, the application artifacts are replicated to storage and distribution infrastructure from the group lead. Installations on consumer devices to automatically and securely update

are conducted by first consulting the chain to detect the deployment and then connecting to the distribution infrastructure to fetch the payload.

15 Content finance

The platform has a built-in ecosystem and tools for creators to finance the production of new works through flexible crowd funding, which gives backers a stake through a project-specific token. This stake gives the right to engage in governance over the project, a possible share of revenue generated by produced assets, and possibly the option to trade assets in a market on the platform.

15.1 Working group

All working group members, including the lead, have the same two sets of responsibilities

- **Curation:** Curating the project pool for abusive or non-compliant proposals.
- **Arbitration:** Arbitrating disputes in project, where the primary sanctions would be in the form of banning, redirecting project funds, and slashing organizer stake.

15.2 Project life cycle

A project is initially created by the prospective project organizer, and this step involves specifying

- **Description:** Text, visual, and other assets used for the description of the project.
- **Assets and terms:** What final productive assets will be produced, and what claims, if any, do backers have on different assets in terms of use and reward.
- **Funding:** A description of the funding model of the project, including
  (a) How much is being raised, minimum and maximum
  (b) When do the sales begin and how long do they last.
  (c) How much of the total project stake is up.
  (d) What jurisdictions are allowed to participate in funding.
- **Token:** A description of the project token, including
  (a) Name
  (b) Symbol
  (c) Whether it is trade able, if so at what earliest time after end of funding period
  (d) Whether it gives claim on revenue
  (d) Whether it gives claim to govern
- **Claimants:** Token allocations to those with claims against the project due to their involvement in the production process.

The project itself goes through the following states after being submitted:

- **Review:** Here, it will be reviewed to see if it is acceptable within the policy of the platform at the time.
- **Open:** The project becomes visible to the public, and corresponding communication channels become available, namely a messaging room and a messaging forum.
- **Funding:** Backers can send funds to the project.
- **Production:** The raised funds are deployed to create project assets, with the organizer and backers collaborating through a governance process. In particular, the funding may be released based on milestones.
- **Active:** Productive assets are finalized and distributed through the platform and possibly third-party platforms also. Any revenue generated by assets distributed on the platform directly will automatically pay out claimants based on the given term, possibly subject to governance. Any revenue generated on external platforms must be documented and submitted to the project by the organizer. Failure to do so will trigger a dispute, and possible sanctions, with the arbitrator.
- **Terminated:** The project is over in that backers have no further active influence or claims. The terms will describe when, or if, this can occur.

16 Advertising

The platform has a built-in advertising targeting, auction, and delivery system. It allows advertisers to reach audiences through a competitive bidding process for display time across a variety of surfaces in consumer-facing experiences. A core premise for a well functioning advertising ecosystem is that a substantial number of non-Sybil users are accessing the platform through a well-behaved reference client.

16.1 Working group

The advertising working group just has a single member, which is the lead, referred to as the advertising authority. This role is responsible for running nodes that assist end users in fetching the correct advertising campaigns from the auction system, and also settling the state of campaigns upon completion or termination, while keeping the display information off the chain.

16.2 Surfaces and targeting

At any given time, there is a fixed set of such surfaces available, each with its own set of display, interaction, and targeting constraints. A separate advertising auction is maintained for each surface, where access allocated to the highest cost per impression bid is currently available. There are two families of targeting parameters, audience, and session parameters. Audience parameters are those that allow one to target consumers based on individual characteristics, such as age interval, gender, location, language, consumption history, wallet balance, etc. This information does not go on the chain in clear text, but some of it may be stored in the membership settings' object in encrypted form. Session parameters are those that are specific

to the context in which the surface is being displayed, thus identifying media being viewed or searched for, or category being browsed, etc.

While all surfaces can target based on audience parameters, the type of session parameter available depends on the surface in question.

Targeting values will be shared with advertising authority to both select the correct ad and also provide confirmation of viewed ads back to the authority; however, this viewing information is never published on the chain.

16.3 Campaigns

A campaign is a bid to occupy a certain surface, subject to a particular set of targeting parameters, at a particular price per impression. Specifically, it includes

- Advertiser identification.
- Advertising payload matching constraints of given surface.
- Targeting parameter values matching constraints of the given surface.
- Expiry time.
- Maximum number of impressions.
- Price per impression.
- Funds covering maximum expenditure.

Once a campaign has been submitted, it lives in a pool of campaigns for the given surface until all the funds have been spent; it is then canceled by the advertiser or the platform.

16.4 Delivery

A given advertising surface has a display policy that prescribes when an end user should repopulate an interface (new) content. The following informally describes the steps involved in this process, including a user, an advertising authority, and an advertising server.

- The user sends the signed session and the audience targeting values to the advertising authority.
- The advertising authority filters the campaign pool for the given surface based on targeting values and returns the highest bidding match, with corresponding Merkle inclusion proofs of campaign payload.
- The user fetches the ad from advertising server and receivesthe receipt token in response.
- The user sends a receipt including this token, the campaign identifier, and their own key to the advertising authority.
- The user renders advertisement.

When an advertising authority detects that a campaign has been displayed enough to exhaust the funds locked up, it submits a proof of this to the platform, which also includes information about how many impressions have been derived from surfaces that are tied to publisher content and what content has actually been shown. This proof is made up of all the receipts from the users that have fetched the given ad. It is compressed using the ZkSNARK primitive [11], where the user keys are kept as private inputs. This proof can also enforce limits on how many times any given user can at most have submitted a receipt for a given ad, preventing trivial abuse from a single member. The chain automatically validates the proof and settles the given campaign by burning the funds. The advertiser can cancel the campaign using an analogous process. These proofs are further combined to create proofs about the total amount of advertising revenues a publisher has a claim over a given period of time.

16.5 Rewards

The advertising authority receives a payment with two components: a fixed per unit of time payout and a payout scaled by proportion of campaign settlements for which proofs are actually submitted. Publishers receive a payout from the advertising, not in the traditional staking-based reward, but an unconditional value transfer per unit of time. This amount is set to be a given fraction of the total revenue being driven through surfaces displaying the content tied to their content.

17 Miscellaneous

17.1 Resolving hosts

Frequently, one may need to resolve host nodes corresponding to an actor on the platform identified with a public key. The mapping between this key and the set of active host nodes under the control of the given actor is represented in a DHT. The actor registers mutable records of such hosts under a DHT key corresponding to the public key. When the set of active nodes changes, simply sign a new message reflecting this and store it under the given key. There are many practical implementations of this pattern; for example, IPNS or BEP44 in BitTorrent.

18 Discussion

In this section, various avenues of possible or required research and inquiry are explored.

- *Catastrophic error recovery*

  Given the rich-on-chain feature set of the platform and the aspiration to conduct frequent on-chain upgrades, the classical blockchain operations model of absolutely never having bugs in production, in perpetuity, may be too expensive. Among other costs, this requirement may reduce the speed of improvement and encourage a set of trusted developer gatekeepers. These costs are possibly dead weight for a protocol that does not intend on emphasizing operational availability and integrity above all other objectives. As a result, it may be advisable to investigate various formal and informal protocols that may be relied upon to coordinate recovery among validators from some sort of consensus failure.

- *BRAQ*

  The currently described model for BRAQ has at least two short comings.

  First, due to the presence of an inevitably imperfect screening-based membership introduction (see section 10), the number of malicious BRAQ instances may silently grow over time. At some point, a concerted attack across such memberships can make chain throughput unavailable for some time. If such an attack is timed to cover some sensitive time period, it may automatically cause slashing of a platform member. This problem can be ameliorated by, for example, introducing sensible default transaction inclusion policies that favour non-BRAQ-based transactions or simply by including global limits on how many such transactions can be accepted at any given time, perhaps being implemented as a BRAQ instance itself.

  Second, the actual quota is not sensitive to payloads involved in actions, only the number of actions. This is not fit for a range of different purposes and can easily be amended.

- *ATP*

  The obvious problem and risk in using ATP is regarding coming up with a safe time period for a given transaction, which may also depend on the payload. Moreover, its not entirely clear how a validator must treat the scenario where the processing is not finished, despite the time being up. The simplest approach is to consider it as any other node failure, such as running out of resources, and simply stop the node.

  There is also the opportunity of saving on processing resources by making the current proposal into a challenge response protocol. In this alternative approach, the final state can be proposed by anyone, subject to some bond, and this proposal can be challenged by anyone during a challenge period. In a challenge, all validators can actually conduct the normal ATP processing and, this way, adjudicate the final dispute securely. A challenger found to be correct will win the bond.

  Another obvious alternative to the entire ATP approach are things like ZkSNARK [11] and Truebit [12]. However, this would impose severe practical limitations on what processing which within scope. For ZkSNARK, it will be, at present, infeasible to generate proofs for just about all processing that can warrant ATP to begin with, by assumption. For both the approaches, one can almost never reuse the existing implementations of the processing in question, even if it is in principle compatible with the approach-specific computing model (e.g. WASM or register machine), which is also not always going to be the case.

- *Governance*

  The current proposal for how governance and elections are organised is relatively naive to obvious problems around liveness, vote buying, and validator censorship. This despite governance being at the heart of the capabilities of the platform and the incentive compatibility of the protocol every other actor faces depends on how effective this process is. Most of the work in this proposal is about how to endow a presumed effective governance system with all the assets constituting a functional content platform. Further work on incorporating more mature ideas is needed.

- *Screening and suspension and seb 2.0 assets*

One of the biggest weaknesses of the current proposal is that it may or may not be feasibleto manage the integrity of the memberships that are established through screening, in particular since it may be difficult to properly identify malicious attacks in production and, even if one could, the corresponding screening authority may no longer be bonded. The most obvious solution to this problem is to make a screening authority the owner of screened memberships and disable them when the authority unbonds. Members could get rescreened with a new authority, using the old keys, in order to preserve their membership. This may coincidentally ameliorate one of the other primary limitations of the platform, due to the state of current infrastructure, which is the inability to properly own assets such as domains and app store entries. An ecosystem of such screeners can own these assets, conduct their own screening procedure, and capture in the upside the membership base they bring to the platform.

- *Software development*

Keeping this entire development process has resource costs, not primarily in throughput capacity but state size. For example, a very mature large open-source project may have a Git repository that occupies dozens of gigabytes in, primarily, commit object blobs. This is not a non-starter, per say, but clearly has costs that need to be recognized. There are other approaches, such as the OSCoin and the Radicle stack [13], which aim to keep the entire process off chain, extend the Git protocol to also incorporate the collaborative assets, and delegate consensus to a trusted set of actors. This does not satisfy all the outlined requirements and begs the question of how to make the consensus actors accountable to the platform; however, it has the benefit of being cheaper on the platform blockchain.

Another issue requiring further work is how to safely update the platform in concert with user facing applications that make assumptions about how the platform functions at any given point of time. This is further complicated by the objective of wanting to use the platform itself, as the infrastructure to do the application updates.

- *Storage and distribution*

The major problem in the current approach is that all user-downloaded events result in a set of transactions and leave a permanent public history of downloads. This is infeasible from both a capacity and privacy perspective. The alternative currently being explored involves offloading these transactions into a separate chain with trust model based on fraud proofs, and governance to ensure availability, inspired by the Plasma framework. [14]. By committing blocks, and in particular state commitments, into the main blockchain, it becomes possible to make positive claims about the current state in the parallel chain at different times, such as the total amount of data a distributor has moved or the total number of downloads for a particular data object. Expired or fraudulent states can be challenged in the same way during an exit period. Privacy can be introduced by allowing members to use pseudonymous identifier in the parallel chain, which can be proven to match a main chain membership through an appropriate ZkSNARK. The disadvantage of this approach is the complexity and latency of main chain state changes. An alternative can be to dramatically change the trust model and rely on data being shared entirely out of band w.r.t. any shared public state, and then having a voting process-based main chain update based on this objective verifiable data.

A secondary objective is to replace full replication with erasure coding for each data object [15] such that less total storage space is required for a given level of fault based on unavailability risk. This

makes the storage costs lower.

References

[1] Albert O Hirschman. *Exit, voice, and loyalty: Responses to decline in firms, organizations, and states*, volume 25. Harvard university press, 1970.

[2] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938, 2018.

[3] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[4] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 51–68, New York, NY, USA, 2017. ACM.

[5] E. Buchman J. Kwon. Cosmos - a network of distributed ledgers.

[6] V. Buterin. Blockchain resource pricing.

[7] Nick Johnson. Ethereum domain name service - specification.

[8] J. Poon. Handshake.

[9] G. Wood. Polkadot: Vision for a heterogeneous multi-chain framework draft 1.

[10] Bitconi Wiki. Payment channels.

[11] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security Symposium*, pages 781–796, 2014.

[12] Jason Teutsch and Christian Reitwießner. A scalable verification solution for blockchains. *url: https://people. cs. uchicago. edu/teutsch/papers/truebit pdf*, 2017.

[13] OScoin. Radicle, 2019.

[14] Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, 2017.

[15] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.