Joystream: A Protocol for User Governed Content Platforms

Jsgenesis

March 11, 2019
DRAFT Version 0.1

Abstract

The Joystream protocol attempts to formalize a content platform which 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 incentives 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 process. 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 may be achieved by empowering two user capabilities.

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

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

Contents

1 Preface

The goal of this document is to give a high level overview of design approach for of a new protocol for content platforms. This is an evolving document which is meant as a basis for a 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 substantial across the proposal as a result of different parts being 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 we have. The traditionally constraining institutions of market competition, regulation and litigation, appear simultaneously 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 that platform policy is only incidentally, not primarily, guided by the objectives of the largest platform stakeholder: *users*. Users should be broadly understood to mean anyone who participates 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 [**?**].

- **Voice**

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

- **Exit**

  Users have effective means of creating new platform instances which 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. Firstly, it empowers the voice of users through the threat of a low cost exist. Secondly, it supports the emergence of a diversity of platforms when the underlying interests of stakeholders are irreconcilably incompatible, or where organizational decline as gone sufficiently far.

In order to build in the capacity for such responses, the Joystream protocol is built as follows. Shared platform state lives on an open blockchain consensus system. As a result, it has a public state and history, and which is fully auditable and immutable. The protocol also has secure direct and broadcast communication capabilities integrated with 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, debate or agitation. 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 which allows low cost coordination around how shared assets should be used and regulated, and how the protocol itself should be amended. Importantly this mechanism is self-executing, and thus provides reliable and counterparty free implementation of governance decisions. All of these properties work in concert to support *voice* as response.

The protocol itself is open, is implemented in open source reference software, has 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 schemas or concept definitions, assume the C++ type system.

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

Constant values are displayed with capitalized snake case as follows: FOO_BAR. These are values which cannot be changed through governance, and require a hard fork or consensus upgrade. On some occasions the symbol C will be used to denote some constant which 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, but in what follows it is briefly described what assumptions are made about 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, this would for example be the UTXO set. This would be as distinct 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, adapted to use Proof-of-Stake based voting power for a dynamic validator[1] set. A designated platform token, described further in section **??**, is used by the validators to stake, and it is also the unit in which they are rewarded. There is a growing set of such consensus protocols [**?**, **?**, **?**], and the following general properties of these systems are of importance to the protocol design.

[1] We will refer to a block producing actor as a *validator*, and anyone simply fully validating the chain as a *full validator*.

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

- **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 also should be able to quickly start interacting with the platform, even if the last sign-on was a long time ago, or it 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 will change 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.

  Contrast this with 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. Contrast this with 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*

    In order for secure assertions about the state of the Joystream chain, to be feasible on a different chain, it 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, because there are no reorganization events that can occur. Such events open up the possibility of attacking whatever asset ownership, or general state management, changes executed on the remote chain prior to the reorganization. Dealing with this is complex, and will often involve introducing delays. Moreover, the light client friendliness discussed above also helps in reducing how much information must be submitted to the remote chain light client.

  – *Safe launch and coexistence*

    Finality ensures that the incentive to attack a chain in order 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

[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.

blockchain, as the amount of work (or stake) initially drawn to 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, and 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 is typically enabled by having the transaction validation rules be stored in the state, and run 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 which 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 of itself.

- **Faster iteration:** For the protocol rules to quickly evolve, the process for coordinating 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 which often surround such events, are an additional practical friction. Instead, endogenous upgrading, with an accompanying on-chain immutable record of deliberation, and an 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 tend to constrain the number of developers who securely can contribute to the improvement of the system. This constraint can in the extreme support a kind of 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, in order to be able to validate all blocks. In practice this ends up with an monotonically increasing complexity confronting developers wanting to comprehend and modify the system, which is problematic, as explained.

4.3 Asynchronous Transaction Processing

Certain kinds of transactions may occasionally be required which take a long time to process. Specifically, they may even require more time to process than what is feasible when consuming all the the block time 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 which 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 of 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 found in the 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 time over a given number of blocks. Successfully issuing such transactions within the given limits do not involve marginal outlay for the given actor.

Given that BRAQ will cover the majority of transactions, security would be 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.

Firstly, it dramatically 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 **??**). After this they will be able to immediately interact with the system, within constraints. Users may later earn or purchase tokens in order 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.

Secondly, 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 possible the only feasible, long term mechanism for correcting the externalities associated with long term social cost of transaction processing and state utilization in public chains [**?**].

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 being positive integers. A combined range length and transaction limit is called a *BRAQ quota*. The model prevents an action in a given block if there are already L such actions in the current to R blocks back. The system must maintain a sequence of positive integers $H = (h_1, \ldots, h_N)$, called the *event list*, for each action, initially set to an empty list. A combined BRAQ quota and such list is called *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 should proceed to normal validation. If the corresponding transaction is subsequently found to be accepted, 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 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 may in some cases be 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 which all have very similar actors. At the same time, it can also be 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 it is 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 could for example be state of a DNS mapping which lives on some other system, like ENS [**?**] or Handshake [**?**]. It may even be desirable to move certain assets from the Joystream blockchain, such as tokens, over onto other blockchain systems. While proposals are being developed to support very general inter blockchain transaction routinely, such as Cosmos [**?**] for tokens and Polkadot [**?**], its 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 used in almost all secure blockchain integration proposals. For each blockchain which 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 of the first. This obviously requires that the given side is expressive and economical enough to support such an on-chain light client. So for example, take the case of wanting to subject an ENS mapping, on Ethereum, to the governance processes on the Joystream blockchain. This would only require deploying a Joystream light client contract on Ethereum which also is set as the owner to the given mapping. Block headers comitted to the Joystream blockchain would have to be submitted to this light client, at the

very least when there were changes in validator sets. A designated party from the Joystream side could be incentivised to regularly do this. This header history in the state of the contract would allow it to be securely convinced of the ENS related signaling actions on the Joystream side through Merkle proofs, since these would be committed to in the corresponding headers[4]. The contract could 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 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 purpose, 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 value to the platform through the token, e.g. purchasing ad placements. There is no platform treasury which would 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 **??**. They are issued by publishing a *token profile* into a registry, called the *token registry*. A token profile includes include key information, such as a standard token symbol, issuer identifier, description and also the ownership state itself, called 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 whomever holds the private key which corresponding the public key. The registration of a key in the balances of a token is referred to as 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 being able to sign a message with a private key that matches the public key corresponding to the origin account. The token registry is a mapping from a *token identifier*, which is a unique positive integer, to the corresponding token profile. The native token has identifier 0.

Notice 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, explained in further detail in section **??**.

6 Member

6.1 Overview

The membership concept is meant unify all platform level participation for the same actor in a way which 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

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

through the same membership, and 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 it supports pro-social investment in actor reputation. Separating this from token holdings is valuable, as it allows for allowing some types 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 **??**).
- **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 **??**), the this is set to ID of screening authority which created membership. If it was established through payment (see section **??**), 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 **??**)
- **Suspended:** Whether member is suspended.
- **Subscription:** If at least one subscription has been entered into, then this is the date and time of this, and the corresponding subscription term ID (see section **??**). Otherwise its blank.

The membership registry is simply a mapping which associated 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 which have indirect proxy quotas.

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

A membership may be established for free, as explained in section **??**, or it may be paid for at 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, called *term ID*.
- **Fee:** Quantity of native token which must be provably burned.
- **Proxy Quota:** Initial quota for membership.
- **Text:** String of capped length (UTF-8) describing the human readable conditions which are being agreed upon.

The platform has a set of terms, called the *active paid membership terms*, which is the terms 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 one greater than the prior active terms, and the initially active terms have an ID of 0. The full history of all such terms that were at one time active is kept in a list, called 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 by making a request, which will burn the required fee from their account, so long as the platform is currently accepting members. This is gated by a platform variable, called the *platform membership gate*, and it can be changed through a proposal.

6.4 Subscription

Subscription terms represent a set of conditions for a prospective subscriber on the platform, and is defined by

Subscription

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

A member which has a subscription which is *active*, that is the current block height it less than the sum of the subscription entry time and duration, is called *a subscriber* in this period.

Just as for 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, be based on these final tallies.

7 Roles, Staking and Slashing

A *role* is a membership status, of which there are 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 native tokens over some period of 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 period of time before a change in the staked amount of tokens is counted towards the actual total staked balance. For increases this is called *bonding period*, and for decreases it is called *unbonding period*. If a stake reduction leads to a 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 part of, their stake, and 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, 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 they are made up of two components. The first component, called 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 simply let x represent the average time weighted quantity. The second component, called 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 towards the end of a term, and current council members are expected to carry out their responsibility of dispatching proposals through their term. The number of council positions in the next term is always set to be a given number in the preceding term, called the *council size*. This may be altered through a proposal, and 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 have to put up some minimal amount of stake to be able to do this, called the *council staking limit*. This limit may be altered through a proposal, and the new limit goes into effect at the start of the next term. When not in the bootstrap term, there may already be existing council members, and they are also welcome to announce for the next term, which is *extending candidacy*. A council member may in this case reuse their existing stake, which is called *transferring council stake*. In this case, they may have to adjust their staking amount to satisfy a staking limit which may have been altered. There is an upper limit to the number of candidates which may be voted upon, called the *council candidacy limit*. This limit may be altered through a proposal, and the new limit goes 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 staked amount. The set of candidates who actually end up being eligible after this constraint is applied is called *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 of the candidate identifier. A member may reuse tokens already staked behind an existing council member, to a member of the candidate pool, which is called *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. To decide the council members during the tallying, the candidate pool members are ranked in terms of the total amount of stake across all

revealed backings in their favour. All outside of the council size are discarded, and if the council size is not filled, then one re-enters the announcement stage.

- **Grace Stage:** All stake introduced in the given term which was backing a losing candidate, or was not revealed, is immediately unstaked. This is so all properly new stake should be 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 which 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, hence it is off 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 as a value for each of the following properties, which apply to all proposals of the given type.

- **Quorum:** The percentage of the council participants which must vote affirmatively in order for it to pass.
- **Threshold:** Minimum percentage of quorum which 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 may begin submitting sealed votes on the proposition, in what is called 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, at which point the proposal is automatically rejected by the council.

The next stage allows the council members to submit revelations for their prior sealed votes, and its called 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 is 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 which voted 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 will immediately be put into effect, however, many proposal types will have some applicability pre-condition which must be satisfied for it to be valid. If this does not hold at this time, then the proposal is simply discarded. This may happen in scenarios where the state of the platform changes in some way which was not foreseen by the initial proposal sponsor or council.

In all scenarios 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, called *working groups*, of which there are 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, the *group lead* and the *worker*. Leads are elected by the council through the proposal system, and the lead is 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 happens - 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, called 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 three 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, 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, are referred to as *replacement* and *eviction* respectively.

In all cases, candidates for a group lead role come from a list on the platform called for the given group called 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 more than this number of potential candidates have staked, 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 **??**, it should be possible to onboard users who do not hold any tokens. This activity, which is called *screening*, allows the platform to accommodate new users who can try the platform within certain constraints. However, the combination of no robust identity system and allowing users to consumer resources without paying on the margin creates its own set of new problems. These resources need not only be in the form of tangibles like compute and storage, but also 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 by 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 sort of cat and mouse dynamic with such abuse, in order to reduce the feasibility and cost of attacks.

This is achieved by two measures, firstly, there is a capacity to block access to the platform in various ways. This activity, which is called *suspending*, specifically prevents the relevant actor from being able to participate on the platform through base membership actions. Secondly, when anyone is onboarded through screening, the corresponding platform actor who onboarded them is recorded, and can be punished for a long time into the future, with a suitable bonding period, if abuse is uncovered.

10.2 Working Groups

This work is parceled out into two complementary working groups, the screening and the curation working group. 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 all 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, is entirely exogenous to the protocol. The set of actual policies, and corresponding tools, in production at any given time are expected to be the result of ongoing coordination, based on policy constraints from the council and group lead. Obvious mechanisms which could be employed are

(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, as prospective member must accept a set of terms, called the *screening terms*, 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.

[5] In principle it is probably advisable to keep the screening group small, even perhaps 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.

The terms to be used at any given time are called the *active screening terms*, and they are changed through the proposal system.

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

10.4 Suspension

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

- **Individual:** A specific member can be suspended by setting the appropriate membership field.
- **Group:** A proposal can be submitted which allows 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 which allows for simultaneous suspension of all members.

10.5 Rewards

All working group participants have some given group based periodic payoff, set by 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 system with communication and resource constraints of a browser environment.
- **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 economise on cost, and be sensitive to platform policy about 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 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 can accurately determine 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 which 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 taken, where most good conduct is expected based on risk of governance based sanctions on deposited bonds, and losing reputational capital which give. 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, called a *storage tranche group*, who all 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 which is required.
- **Out Speed:** The number of Mbps out which are required.
- **Stake:** The amount of stake required.
- **Duration:** The number of blocks service must be provided from initiation block.
- **Fixed Reward:** Quantity of native token earned per unit of time.
- **Variable Reward:** Quantity of native token 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 which are being agreed upon.
- **Active:** Whether tranche is actually possible to use, that is assign new members to.

All are kept in a mapping called 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 will continue to be 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 should be considered active.
- **Paused:** Whether this role is currently paused.

All such profiles live in a mapping from the ID to the profile, called 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.

Notice 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 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 would be data objects which are behind paywalls.
- **Accounting Procedures:** Some objects may require some kind of accounting or cleanup process as a result of the accessing the data. This could for example be to record reliable access statistics for media content, i.e. view count.

The first requirement is about reducing cost, the latter two is 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 must 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 which can be used with this data type. If empty, then any tranche may be used.
- **Active:** Whether objects of this type can be added at this time.

All available types are kept in mapping, called 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. As an example, it may be the case that whenever a particular publisher uploads a media item, there will be 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 which allows for 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 data.
- **Liaison:** ID of storage provider which accepted 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 provider.

All such relationships are kept in the *storage relationship registry*, which maps the ID to the relationship. Analogous concepts for distribution, called *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 for member downloading.
- **Distributor:** ID for distributor which distributes the content.
- **Initiated:** Date and time when session was initiated.
- **State:** Either *started* or *ended*.
- **Transmitted:** Amount of bytes of data actually downloaded.

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

11.6 Uploading

A normal uploading flow goes 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 the
   (a) the CID is not already in 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, it will result in the creation of a data object record, where the liason and storage object relationships are automatically assigned. The latter all have an initial status of no 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 should 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 should then proceed to accept the upload.
3. The storage provider will check
   (a) the payload matches the CID
   (b) has the right size
   (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 which have a storage relationship with the given record. As they receive the payload, each has a responsibility to alter the readyness of their storage relationship. Any relationship which is still not ready after some defined period of time from the record being 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 which have corresponding storage relationships with ready status to acquire a copy of the payload.

11.7 Downloading

A normal downloading flow goes as follows [8].

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

   (a) the CID corresponds to a data object record

   (b) the data object record has at least one distributor relationship which 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 [**?**]. 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 terminates the exchange. At this point, the distributor will submit a new transaction 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 updating what are assigned to what data. These may be initiated and coordinated through off-chain messaging protocols, and 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 happen 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 off 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. Importantly, it does not contain the primary media itself, that is stored in separate off chain infrastructure described in the section on storage and distribution (section **??**).

12.2 Working Group

The working group has two roles, lead and member. Members are responsible for executing the following functions

- **Curation:** Ensuring that the underlying content media and metadata correspond to each other, e.g. that display assets are correct, or that 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 profiles 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

- **Category:** The type of content
- **Payload:** Data directory identifiers for media and metadata.
- **Owner:** A publisher or content project (see section **??**) who has control over the content, and has rights to value.
- **Monetization Policy:** How end users must pay to access content, among which 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 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:** 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, others with changing attribution or ownership information, and as a result redirecting 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 set promptly. In order to alleviate the resulting processing and bandwidth costs this would impose, there is a designated set of discover nodes which provides these services.

13.2 Working Group

The working group is made up off members which all run nodes that run a discovery provider service, 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 cases 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 that would mean, as different discovery providers are free to pursue their own ranking and discovery policy. Reasonable incentives for good behavior can be generated by having user clients keep local statistics about the success rate of various providers, by measuring user behavior, which would drive more traffic to better providers. Likewise, the incentive to generate more traffic can be generated by giving a provider the privilege of displaying some in place advertisement which is not hooked into the normal advertising ecosystem. The inclusion of minimal telemetry feedback to providers can also help providers 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 should 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 they rely on third party revenue sources which do not have incentives that are guided by the interests of all protocol stakeholders.

To address this, the platform has a dedicate working group of contributors who are rewarded for providing these sorts of services, and which 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 could 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 becomes a security risk, and it is not formally accountable to protocol stakeholders. Socially desirable changes may be rejected, or adopted too slowly, and changes which are undesirable may be adopted. While fork based exit is in principle an option, it is also costly. 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. Merge commits are in this respect particularly sensitive, as this is typically the way new changes make their way into production 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, like merging, are done 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 that 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, a there is a set of standards for defining and conducting deterministic builds, and software updates happen 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 allow them to perform write operations directly. While anyone can in principle contribute 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 to 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 all fully secured by the platform itself, by having the Git processing rules embedded in consensus itself. Opening issues and pull requests is open to any platform member, but only a working group member with suitable permissions can actually moderate issues or merge requests, respectively.

14.4 Artifacts, Reproducible and Releases

An artifact is a file is the result of some processing of the source material in a project repo commit. This processing may be things like a building, linking and packaging. There is a format for how to describe such processes, and they will always yield fully deterministic outputs. These processes happen 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 build process, where this determinism provides reproducible builds. A release is simply 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 challenger gets reward.

14.5 Automated Testing

There is a format for describing how to run tests off-chain, and the presence of tests metadata will prevent pull requests from going through unless the group lead signs off on them 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 that a new version of some application is available, whether it is backwards compatible - and thus optional, and how to retrieve the application itself.

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

Both types of deployments are triggered by having the group lead submit a proposal based on a release. When it 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 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 description of 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 sale begins, and how long it will 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 are 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 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:** Project 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 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 currently available. There are two families of targeting parameters, audience and session parameters. Audience parameters are those which 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 which are specifics to the context in which the surface is being displayed, so this things 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 parameters available depends on the surface in question.

Targeting values will be shared with advertising authority, in order to both select the correct ad, and also provide confirmation of viewed ads back to the authority, but 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 given surface
- Expiry time
- Maximum number of impressions
- Price per impression
- Funds covering maximum expenditure

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

16.4 Delivery

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

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

When 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. The proof also includes information about how many impressions are derived from surfaces that are tied to publisher content, and what content was actually 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 [**?**], 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 revenue a publisher has a claim on over a given period of time.

16.5 Rewards

The advertising authority receives a payment with two component, 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 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 that is driven through surfaces displaying 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 signing a new message reflecting this and storing 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 do frequent on chain upgrades, the classical blockchain operations model of absolutely never having bugs in production, in perpetuity, may be too costly. Among other costs, this requirement may reduce the speed of improvement and encouraging a set of trusted developer gatekeepers. These costs are possibly dead weight for a protocol which is not intended to emphasize 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 describe model for BRAQ has at least two short comings.

  Firstly, the presence of inevitably imperfect screening based membership introduction (see section **??**), the number of malicious BRAQ instances may silently grow over time. At some point, a concerted attack across such memberships could could make chain throughput unavailable for some period of time.

If such an attack is timed to cover some sensitive time period, it may be that it can automatically cause slashing of a platform member. This problem could be ameliorated by having for example introducing sensible default transaction inclusion policies which favour non-BRAQ based transactions, or simply by including global limits on how many such transactions can be accepted at any given time, perhaps implemented as a BRAQ instance itself.

Secondly, 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 how to come 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 should 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, like 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 any anyone during a challenge period. In a challenge, all validators would actually do the normal ATP processing, and this way adjudicate the final dispute securely. A challenger found to be correct would win the bond.

Another obvious alternative to the entire ATP approach are things like ZkSNARK [**?**] and Truebit [**?**]. However, this would impose severe practical limitations on what processing which would be in scope. For ZkSNARK it would be, at present, infeasible to generate proofs for just about all processing which would warrant ATP to begin with, by assumption. For both approaches one could almost never reuse existing implementations of the processing in question, even if it was 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 that would constitute a functional content platform. Further work on incorporating more mature ideas is needed going forward.

- *Screening and Suspension and Web 2.0 assets*

One of the biggest weaknesses of the current proposal is how feasible it may or may not be to 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 like domains and app store entries. An ecosystem of such screeners could own these assets, conduct their own screening procedure, and capture in the upside that the membership base they brings 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 which occupies dozens of gigabytes in, primarily, commit object blobs. This is not a non-starter per say, but it clearly has costs which must be recognized. There are other approaches, such as OSCoin and the Radicle stack [**?**], 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 of the outlined requirements, and begs the question of how to make the consensus actors accountable to the platform, but has the benefit being cheaper on the platform blockchain.

  Another issue which requires further work is how to safely update the platform in concert with user facing applications which make assumptions about how the platform functions at any given 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 download events results in a set of transactions, and leaves a permanent public history of downloads. This is infeasible from both a capacity and privacy perspective. The alternative which is currently being explored is to offload these transactions into a separate chain with trust model based on fraud proofs, and governance to ensure availability, inspired by the Plasma framework [**?**]. 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 could be introduced by allowing members to use pseudonomous 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 [**?**], such that less total storage space is required for a given level of fault based unavailability risk. This makes the storage costs lower.