Backfeed Protocol - The Objective Layer
draft version - all comments welcome

Matan Field, Jelle Gerbrandy*, Elad Shtilerman, Primavera de Filippi

Abstract

The Backfeed protocol is a full stack operating system for the decentralized ecosystem. The protocol envisions the internal governance of single decentralized autonomous organizations (DAO) as well as the economic or other interactions between multiple DAOs. The grand vision of Backfeed is an alternate economic ecosystem in which the main agents are these decentralized entities powered by smart contracts deployed to a blockchain. In this document we focus on the basic building block of this ecosystem - the way that reputation flows in a system where a group of people vote on the value of a contribution.

1 Introduction

This document is the first in a series of chapters describing the several layers of the Backfeed protocol.

We give formal analysis for the *objective layer* of the Backfeed Protocol or the reputation flow protocol. It consists of a minimal set of design principles that any fully specified protocol should adhere to. The other layers in the Backfeed stack extend the basic protocol in several directions.

The Backfeed Objective Layer Protocol describes an operating system of an organization of agents via a novel reputation system. In this organization, an agent's *reputation* represents her influence in the system, whereas an agent's *tokens* represent the economic possession within the organization. The objective protocol makes the abstraction that governance of the organization can be cast into two atomic actions, *contributions* and *evaluations*. A contribution can be anything which is of value to the organization, typical examples include code snippets, design work or content. An evaluation is an act of appraisal of a contribution done by agents. The basic logic is that once any contribution has obtained enough endorsements via evaluations then the DAO automatically rewards it with, newly minted, reputation and tokens.

*Write to me at `jelle@backfeed.cc`

This document focuses solely on the reputation aspect of the protocol. We first sketch a general formal model of reputation flow in the system. We then formulate a number of precise principles that describe how reputation should flow between different agents. Once the design principles have been laid out, we continue by a general description of the possible reputation flows and discuss the rationale in choosing from this space. A complete solution, satisfying all the requirements, in the form of code or complete mathematical specification is beyond the scope of this document.

2 Problem Formulation

The Backfeed Objective Layer Protocol describes a system that consists of an organization of agents that have *reputation* that represents their influence in the system, and *tokens* that represent the property of the organization.

These agents can initiate certain actions: they can *make a contribution* to the organization, or *evaluate a previous contribution* to help determine the value (in tokens) of that contribution. These actions have effects on the distribution of reputation and tokens in the system - for example, when a contribution gets accepted on the basis of the evaluations of the agents, the contributor may be assigned tokens and reputation on the basis of the value of her contribution.

In this paper, we only focus on the flow of *reputation* - we will leave the discussion of the assignment of tokens out of the equation.

We will represent our system as a labeled directed graph where the vertices are labeled by the reputation distribution over the agents and the edges are labeled with events. These events are typically actions taken by users, such as making a contribution or evaluating a previous contribution.

More specifically, let $N = \{1, ..., n\}$ be a set of agents. A *reputation distribution* is a function $r : N \mapsto \mathbb{R}$ that assigns a real number $r(i)$ such that $0 \leq r(i) \leq 1$ and $\Sigma_{i \in N} r(n) = 1$.

This last condition - that the reputation of the agents sums to 1 - is important: it represents the fact that we are interested in the influence that an agent has in making decisions together with other agents. Reputation, in this case, is not an absolute measure, but a metric that only makes sense *relative to the other agents*

The central idea of the Backfeed Objective Protocol is that events that give information about the behavior of the agents will lead do an adjustment of the reputation in the system. Starting from an initial reputation vector r_0 as "seed", a chain of events $e_0, ..., e_n$ may change the relative reputation of the agents involved.

$$r_0 \xrightarrow{e_1} r_1 \xrightarrow{e_2} r_2 ... r_{n-1} \xrightarrow{e_n} r_n \tag{1}$$

Let a *history*, h, be a sequence $h = e_0, e_1, ...e_n$, where each e_i is an event.

We are interested in defining how reputation flows in a system as the result of a sequence of event. To do that we define a family of functions Φ that map a seed distribution r and a history of events h to a new reputation distribution $\Phi(r, h)$.

So our general model looks like this:

$$(N, E, \Phi) \tag{2}$$

Where N is a set of agents, E a set of events, and Φ a function that maps reputation distributions $r : N \mapsto \mathbb{R}$ and histories $h \in \bigcup_{n\in\mathbb{N}} E^n$ to reputation distributions.

We introduce some notation conventions and abbreviations to make formulating our conditions easier.

With regard to histories, we will not be obsessively precise, and write things like hh' for the concatenation of h with h'; or he and $h(k, c, v)$ for the history h appended with e and (k, c, v), respectively.

We define

$$\Phi_{h'}(r, h) \equiv \Phi(r, hh') \tag{3}$$

The *reputation gain of i from h' with respect to (r, h)* as:

$$\Delta_{h'}(r, h)(i) \equiv \Phi_{h'}(r, h)(i) - \Phi(r, h)(i)$$

An *evaluation* is a event in which an agent k evaluates a contribution c as having value v. So, given a set of agents N, a set of contributions C and a set of possible values V, a contribution is an element $(k, c, v) \in N \times C \times V$. To keep the analysis manageable, in this paper we almost exclusively focus on the case where $V = \{0, 1\}$ - i.e. that evaluators give a simple yes/no evaluation. Moreover, we assume that each agent only votes *once* for each contribution - in other words, for each agent i and contribution c, we assume that each history contains just a single evaluation event of the form (i, c, v).[1])

Although in this document, we will be concerned only with 'evaluation events', we can consider other events as well. For example, a 'contribution event' where agent i makes a contribution c represents an important element of the Backfeed protocol that can lead to reputation or tokens flowing the the contributor. Also, an event in which a 'new agent enters the system' would make our analysis more complete. We will leave these as objects for a future version of this document.

[1]The condition can be reformulated allowing for changing votes as well, but for now we will keep this simple.

3 Reputation Flow

We will now formulate - in a formally rigorous way - a number of properties that a reputation flow function Φ should satisfy.

3.1 Reward consensus voters

Reputation should flow to an early evaluator in the case that a later evaluator voted the same as her.

reward consensus voters:
if (i, c, v) is an evaluation in h and $i \neq k$ then $\Delta_{k,c,v}(r, h)(i) \geq 0$

3.2 Reward consensus voters proportionally

Previous voters should, *ceteris paribus*, be rewarded in proportion to their reputation. The more reputation you have, the more reputation you gain, and the less you have, the less you can gain.

reward proportionally and linearly:
If (i, c, v) and (j, c, v) are evaluations in h, then $\Delta_{k,c,v}(r, h)(i) \geq \Delta_{k,c,v}(r', h)(j)$ if and only if $\Phi(r, h)(i) \geq \Phi(r', h)(j)$

There is an alternative formulation of this principle, in which we do not reward the voters on the basis of their current reputation $\Phi(r, h)(i)$, but on the basis of their reputation at the moment they voted.

reward proportionally on the basis of stake:
Let $h = (e_0, ..., (i, c, v), ..., (j, c, v), ..., e_n)$. Then $\Delta_{k,c,v}(r, h)(i) \geq \Delta_{k,c,v}(r', h)(j)$ if and only if $\Phi(r, (e_0, ..., (i, c, v)))(i) \geq \Phi(r, (e_0, ..., (j, c, v))(j)$

However, these conditions may be too strong - the it embodies the 'rich get richer' principle, where we reward the agents with a lot of reputation more for the same action then an agent that has less reputation. We can weaken the principle by only requiring that if two agents i and j with i having more reputation than j, act in the same way will not result in a situation where j has more reputation than i.

reward proportionally:
Let $h = (e_0, ..., (i, c, v), ..., (j, c, v), ..., e_n)$. Then $\Phi_{k,c,v}(r, h)(i) \geq \Phi_{k,c,v}(r', h)(j)$ if and only if $\Phi(r, h)(i) \geq \Phi(r', h)(j)$

3.3 Contribution Independence

The reputation flow resulting from the evaluation of a contribution c should be independent of any previous evaluation of contributions other than c.

contribution independence:
If h and h' are such that if, for each i and v, it holds that $(i, c, v) \in h$ iff $(i, c, v) \in h'$, and if $\Phi(r, h) = \Phi(r', h')$, then $\Phi_{k,c,v}(r, h) = \Phi_{k,c,v}(r', h')$

3.4 Time Independence

The reputation flow as the result of an evalution to or from a previous voter i should not rely on when i has voted. Or, generalized, the *order* of the previous evaluations does not matter for calculating the reputation flow that results from the current evaluation.

time independence:
If h and h' are permutations of each other, and $\Phi(r, h) = \Phi(r', h')$, then $\Phi_{h''}(r, h) = \Phi_{h''}(r, h)$

The reason we include this property is not because we think all protocols *should* behave like this. However, a Φ-function that is time independent greatly simplifies both the logic as well as the implementation of the protocol - it simply means we have to do a lot less of 'administration' to calculate the values of $\Phi(r, h)$.

3.5 Dynamic calculations

We say that a protocol Φ is *dynamic* if the reputation flow only depends on the current reputation distribution.

dynamic flow:
A protocol Φ is *dynamic* iff for each r and r', if $\Phi(r, h) = \Phi(r', h)$, then $\Phi_e(r, h) = \Phi_e(r', h)$

The mirror image of this property is a more 'static' picture, in which reputation flow towards an agent as an effect of, for example, her vote for a contribution c, depends on her reputation at the moment that she voted - instead of of her current reputation.

static flow:
Define $r_c^{Static}(r,(e_0,...,e_n)(i)$ be equal to $\Phi(r,(e_0,...,e_m))$ if $e_m \sim (i,c,v)$ is in h, and set it fo $\Phi(r,h)$ otherwise.
A protocol Φ is *static* iff for each r and r', if $r_c^{Static}(r,h) = r_c^{Static}(r',h)$, then $\Phi_e(r,h) = \Phi_e(r',h)$

3.6 Split Insensitivity

We envisage the system to be used in environments where users cannot be identified. Reputation flow is potentially vulnerable to a *Sybil attack* in which agents try to "game the system" by creating many agents. This is something we want to protect against.

Split insensitivity means that, everything else remaining equal, it should not matter that the effect on the system of two voters voting as a block (that is, always subsequently and with corresponding votes), the effect on their combined reputation is the same as the effect on the reputation of a single agent that has the same reputation as the two combined.[2]

To express split insensitivity, we need some notation. Let (r,h) be a history for N, and i an agent in N. We say that i has split into i_1 and i_2 in (r',h') iff (r',h') is a history over $N/i \cup \{i_1,i_2\}$, r' is just like r, with $r(i) = r(i_1) + r(i_2)$, and h' is like h, but with each (i,c,v) replaced with $(i_1,c,v)\cdot(i_2,c,v)$.

If this is the case, we write $(r,h) \sim_{i/\{i_1,i_2\}} (r',h')$

strong split insensitivity:
if $(r,h) \sim_{i/\{i_1,i_2\}} (r',h')$ then $\Delta(r,h)(i) = \Delta(r',h')(i_1) + \Delta(r'_0,h')(i_2)$

split resilience:
if $(r,h) \sim_{i/\{i_1,i_2\}} (r',h')$, then $\Delta(r,h)(i) \geq \Delta(r',h')(i_1) + \Delta(r'_0,h')(i_2)$

[2]Our users cannot split (or merge). A more 'direct' model of a Sybil attack represents also how new users enter the system (and should make it clear that it is not lucrative to have lots of new agents instead of just a single one. We can add 'new-user'-events to our history and take it from there.

Split sensitivity is a strong protection against Sybil attacks: it says that the fact that the influence an attacker has on the reputation flow depends only on the amount of reputation he controls, but is independent of the number of agents he controls. Split resilience is a weaker condition which states that an agent is better off voting as a single agent then by splitting his reputation to two or more agents.

3.7 Incentivize Voting

Those who vote should earn in confront of those who do not vote.

punish inactivity:
If h is a history in which i has voted on c, but j has not, then $\Delta_{k,c,v}(r,h)(i) > \Delta_{k,c,v}(r,h)(j)$

3.8 Incentivize early voting

In the case of equal votes two evaluators with the same reputation evaluating consecutively with the same value, the first should be better off - or at least not worse off -than the second. This is an incentive for evaluating early.

Incentivize early voting:
If $\Phi(r,h)(k) = \Phi(r,h)(l)$ then $\Phi_{(k,c,v)(l,c,v)}(r,h)(k) \geq \Phi_{(k,c,v)(l,c,v)}(r,h)(l)$

3.9 Regain your stake

It should be possible for any evaluator to regain at least the reputation put at stake upon evaluating. In other words, if we have a system where users pay a fee for voting, then it should always be possible to regain that penalty.

regain your stake:
For each history h, if $\Delta_{(k,c,v)}(r,h)(k) < 0$, then there is a history h' such that $\Delta_{(k,c,v)h'}(r,h)(k) \geq 0$

3.10 Speed of flow

[try to express something about how quickly reputation flows in the system - i.e. about 'how many rounds need to pass minimally to double your rep'. Concepts

like "stability", "viscosity" may be useful. This would be a measure of resilience: if there is an attack, it can only be slow. Concepts like "stability", "viscosity" may be useful.]

3.11 Reward more for controversial votes

Some contributions are easier to evaluate than others. For example, it is easy to reach consensus about the truth of '1+1=2', and therefore, voters should get rewarded less for voting on such decisions than voters that 'stick out their neck' when voting on more difficult decisions.

We define the *controversiality* of a contribution c in a history h as the Shannon index over the votes on it's values. More precisely, for a history h and a seed reputation r_0, let v*otes*(h, c, v) be equal to $\Sigma\{\Phi(r_0, h)(i) | i$ voted v for c in $h\}$.

$$\text{controversiality}(r_0, h, c) := -\Sigma_{v \in V} \text{ votes } (h, c, v) \cdot \ln (\text{ votes } (h, c, v)) \quad (4)$$

We can then define:

reward controversial votes:
If $\Phi(r, h) = \Phi(r', h')$ and $\text{controversiality}(r, h, c) \geq \text{controversiality}(r', h', c)$, then, for each i, $|\Delta_{k,c,v}(r, h)(i)| \geq |\Delta_{k,c,v}(r', h')(i)|$

3.12 Resilience and Game Theory

The concept of 'resilience' of a protocol function Φ pertains to the possibility of 'gaming' the system, and in particular of the possibility of malign agents to gain reputation no matter what.

Some concepts from Game Theory may be useful here. For example, a 'winning strategy' for an agent is an algorithm that allows her to gain reputation in the system no matter how the other agents vote. We can say that a protocol a minimally resilient if no agent has a winning strategy.

The structure we have so far lends itself very well to a classic game theoretic analysis. We have neither time nor space to develop this in much detail in this document, but the basics will look like this.

We have not talked about who decides on the order in which agents votes. We use a standard hack from game theory and introduce 'nature' or 'chance' as a player for that.

The *reputation game defined by* Φ *and* r_0 then, is a rooted labeled tree. On each even node, nature chooses an agent i. In the next node, which we call an i-node, i either chooses to evaluate a contribution, or passes her turn (i.e. she

chooses not to act at all). We can label each even node n with the value of $\Phi(r_0, h)$ - where h is simply the history of events that brings you from the root node to n.

If we identify the pay-off of the users with the reputation function (i.e. if we assume that players are only motivated by their reputation and not the decisions that are made), then what we have here is a *infinite-horizon extensive-form game.*

Now we can define *a strategy for i in the game given by r_0 and Φ* as a function S that assigns to each i-node s an action $S(s)$.

We can now define the resilience of Φ like this.

weak resilience:
there is no winning strategy for games of arbitrary length: For each r_0 and each history h compatible with S, there is an h' such that hh' is compatible with S, and $\Delta(r, hh')(i) \leq 0$.

somewhat less weak resilience:
Let H be the set of histories compatible with a strategy S for i. Then $\Sigma_{h \in H} \Delta(r, h)(i) \leq 0$

A full game theoretic analysis lies beyond the scope of this document. We will just note here some directions that such an analysis could take that we think are interesting. For example, we make the pay-off function more realistic, and assume that preferences of the players not only depend on their reputation, but also on the actual value that is conferred by the decision process on the given contribution. Another ingredient to add would be "beliefs about beliefs" - where agents base their decisions on what they think that other players will vote.

4 Defining flow functions

We have defined a number of formal conditions on how reputation could flow. Our task is now to explore the space of functions that satisfy these criteria.

4.1 An example of a flow function

We are looking for a definition of $\Phi_{k,c,v}(r, h)$.

Given contribution independence and time indepence, we know that $\Phi_{k,c,v}(r, h)$ depends on not more information than four sets: the set K consisting of the

voter $\{k\}$ only, the set O of agents that have not voted for c yet, the set V of agents that have voted v, and the set $-V$ of agentst that voted on c with a value different from v.

Now to keep the definitions simple, write r' for $\Phi(r,h)$, and write $r'(S)$ for $\Sigma_{j\in S}\Phi(r,h)(j)$

Consider now the following definition, where we have reputation flow from non-voters to previous voters:

definition: flow function

1. $\Delta_{k,c,v}(r,h)(i) = 0$ if $i = k$ (current voter gains nothing)
2. $\Delta_{k,c,v}(r,h)(i) = 0$ if $i \in -V$ (different voters gain nothing)
3. $\Delta_{k,c,v}(r,h)(i) = \gamma \cdot r'(O) \cdot r'(i)/r'(V)$ if $i \in V$ (equal voters get some reputation from non-voters)
4. $\Delta_{k,c,v}(r,h)(i) = (1-\gamma) \cdot r'(O) \cdot r'(i)/r'(O)$ if $i \in O$ (non-voters loose some reputation)

hypothesis: The function $\Delta_{k,c,v}$ defined here satisfies the properties 'split insensitivity', 'reward consensus voters', 'reward proportionally', 'contribution independence', 'time independence', 'punish inactivity', 'incentivize early voting'.

1.

5 Conclusions

[where we say what we have done, and, more importantly, what we want to do]