1protocol: Virtual Workers on Ethereum

Zachary Lawrence
Stanford University
zack@1protocol.com

Axel Ericsson
Stanford University
axel@1protocol.com

May 21, 2017

Abstract

Participation in staking protocols on Ethereum requires capital in addition to computing power. `1protocol` serves as a platform for Ethereum users with idle capital and workers with excess computing power to jointly perform work for staking protocols. Herein, we define `1protocol` and present its immediate applications.

Contents

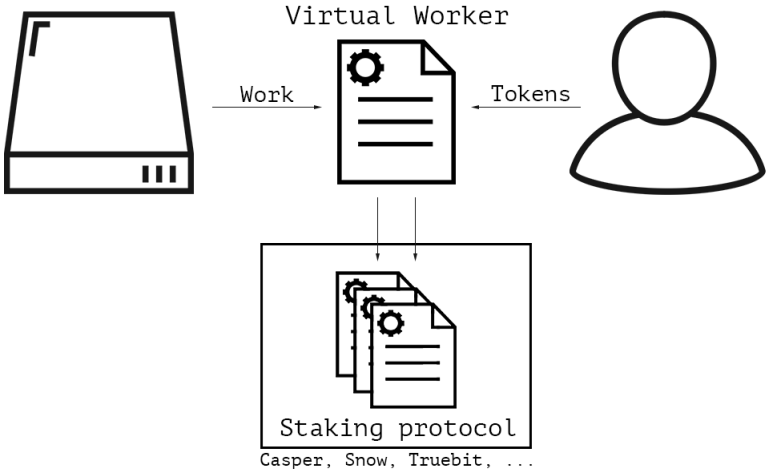
1 Overview

1.1 Staking Protocols

Instead of relying on central servers, protocols on Ethereum can provide digital services by employing decentralized networks of machines (e.g. file storage[1], computation[2], and block production[3]). These protocols ensure correctness using economic incentives; they compensate machines for accurately performing tasks and financially penalize them for violating protocol rules. To enforce financial penalties, these protocols often require worker machines to submit security deposits or *stakes*. As a result, workers lacking capital for deposits often cannot fully utilize their computing power.

1.2 Virtual Workers

`1protocol` enables users with idle capital and workers with excess computing power to jointly perform work for protocols requiring capital. Using `1protocol`, Ethereum users can pool their capital into smart contracts; these smart contracts in turn incentivize workers to leverage the pooled capital to perform work for staking protocols on Ethereum. We term such constructions: *Virtual Workers*.



[1] http://filecoin.io/
[2] https://truebit.io/
[3] https://cdn.hackaday.io/files/10879465447136/Mauve%20Paper%20Vitalik.pdf

1.3 Introducing `1protocol`

Herein, we introduce the `1protocol` project. We begin by defining the on-chain mechanics of `1protocol`: the collection of smart contracts enabling the construction of Virtual Workers. We then present 1client, an extensible client-side software suite supporting the `1protocol` ecosystem. We conclude by discussing `1protocol`'s incentive layer —The Credit Token (CRED)—and immediate applications. Specifically, we outline Virtual Workers for several upcoming Ethereum protocols.

2 Protocol Schematic

In this section, we define the on-chain procedure, enabled by `1protocol`, for constructing Virtual Workers. The procedure involves two parties:

- *Capitalist*: The Ethereum account supplying capital.
- *Operator*: The Ethereum account supplying computing power.

Main Procedure. We define the procedure below.

1. *A Capitalist submits a Virtual Worker Proposal*

   A Capitalist may submit a proposal for a Virtual Worker to `1protocol` in the form of a *Proposal* object.

   `P = {code, min_deposit, hash_linked_resource, seed}`

   `P:` The Virtual Worker Proposal object.

   `code:` EVM bytecode for the proposed Virtual Worker. If an Operator accepts `P`, `1protocol` initializes a smart contract running `code`.

   `min_deposit:` Amount of Credit tokens an Operator must deposit to initiate the Proposal (see Credit Section).

   `hash_linked_resource:` A content-address linking to recommended client side software for the Operator of `P` (see 1client Section).

   `seed:` Arbitrary ERC20 tokens; `1protocol` transfers the tokens to the Virtual Worker after initialization.

2. *An Operator accepts a Proposal*

   To accept a Proposal `P` an Operator must first supply a security deposit greater than `min_deposit`. `1protocol` subsequently deploys a smart contract initialized with `code` and transfers `seed` to the newly deployed contract. We term the newly deployed contract a Virtual Worker.

3. *Live Virtual Worker*

   Once deployed, the Virtual Worker contract exposes its capital, `seed`, to its Operator; in turn, the Operator can leverage the exposed capital to

perform work for Ethereum staking protocols. To incentivize Operators to perform work, Virtual Worker contracts can call several functions exposed by `1protocol`:

`fire()`: Confiscates the Operator's security deposit and recycles the Virtual Worker to Step 1.

`punish(amount)`: Transfers `amount` funds from the Operator's security deposit to the Capitalist.

`reward(amount)`: Transfers `amount` funds from the Capitalist to the Operator.

3 1client

To support the `1protocol` ecosystem, we will release 1client: an extensible collection of client-side software. The software will serve as a tool for Operators and Capitalists to seamlessly interact with `1protocol` contracts. In this section, we outline the software's structure and the features it offers.

3.1 Structure

Consistent with existing Ethereum client implementations, 1client exposes its functionality via an RPC Interface. Atop the RPC Interface, we provide a command line interface (REPL) and a GUI. We additionally include a Python library for developers to write scripts which directly control 1client. Note that 1client ultimately serves as a reference implementation; users remain free to develop custom software for interacting with the `1protocol` ecosystem.

3.2 Features

- **Virtual Work**
  1client enables users to browse and accept pending Virtual Worker Proposals; alternatively, using whitelists (arbitrary user-defined filters for Proposals) users can instruct 1client to automatically accept Proposals. Users can further instruct 1client to immediately execute Operator software after accepting a Proposal.

- **Proposal Analysis**
  1client supplies tools that enable Operators to analyze pending Proposals. Notably, 1client provides: whitelists and profit projections.

- **Hash-Linked Resources**
  Recall that Virtual Worker Proposals can contain HLRs: content-addresses that point to recommended Operator software. Using 1client, Operators can effortlessly retrieve and begin executing such software.

- **Work Pools and Funds**
  Analogous to Virtual Worker Proposals, 1client allows users to construct, browse, join and monitor Work Pools and Funds. We define Work Pool and Fund below.

  Work Pool: A smart contract enabling an arbitrary number of Operators to jointly operate a Virtual Worker.

  Fund: A smart contract enabling an arbitrary number of Capitalists to jointly finance a Virtual Worker.

4 The Credit Token: CRED

To participate as a buyer or seller of `1protocol` services, Operators and Capitalists use our custom ERC20 token: The Credit Token (CRED). Specifically, CRED occupies several roles in the `1protocol` ecosystem:

- Operator staking (see Protocol Schematic)
- Compensating Operators (see Protocol Schematic)
- Funding development of the `1protocol` project
- Usage in future Virtual Worker-compatible protocols developed by One Protocol Inc.

We will present our complete token sale mechanics in a future, stand-alone document.

5 `1protocol` Applications

In this section, we outline Virtual Workers for three upcoming Ethereum protocols: Casper, snow, and Truebit. Our Virtual Workers enable Capitalists and Operators to collaboratively perform work for these upcoming protocols.

5.1 Fragmented Staking

Recall: to accept a Virtual Worker Proposal, an Operator must supply a security deposit. Note that an Operator only has an incentive to accept a Proposal if the capital he will control substantially exceeds the required deposit; otherwise, the Operator could achieve greater returns using his security deposit to work for protocols directly. Thus, formally, we require:

$$\mathtt{P}_{seed} * r > \mathtt{P}_{min\_deposit}$$

Where `P` is an arbitrary Virtual Worker Proposal and $r \in [0, 1]$ is the percentage of the Virtual Worker's returns the Operator recieves as compensation. Further, the Operator's security deposit should cover the Virtual Worker's vulnerable capital: the amount an Operator can destroy before being fired. Thus, we also expect:

$$\mathtt{P}_{min\_deposit} \geq \mathtt{P}_{seed} * v$$

Where $v \in [0, 1]$ represents the percentage of the Virtual Worker's capital that is vulnerable. To satisfy these requirements, our sample Virtual Workers employ a novel mechanism: fragmented staking. We define the mechanism below.

1. The Virtual Worker divides its seed capital into fragments. When performing work for protocols, it uses each fragment as a separate deposit.
2. The Virtual Worker contract enforces that its Operator can destroy at most a small fraction of its fragments.
3. If an Operator destroys a fragment, the Virtual Worker confiscates a percentage of the Operator's security deposit.

Using fragmentated staking, we can reduce the percentage of vulnerable capital, v, and thus satisfy our requirements.

5.2 Casper Virtual Worker

Ethereum plans to replace its current Proof-of-Work consensus algorithm with a novel Proof-of-Stake algorithm, Casper. The Casper protocol reaches consensus by incentivizing Ethereum users, termed validators, to propose and bet on blocks. To participate as a validator in the Casper Protocol, Ethereum users must supply a deposit of ETH; after supplying a deposit, users can earn rewards producing new blocks.

More practically, in addition to ETH, participation in Casper requires: sufficient hardware, a solid internet connection, and technical prowess. Thus, regardless of the financial benefits, we expect that many ETH holders will decide not to participate. Using our Casper Virtual Worker, ETH holders can trustlessly transfer their capital to Operators; these Operators then earn rewards leveraging the capital to participate in Casper.

Our construction consists of three parties[4]:

- *Capitalist*: The party supplying ETH.
- *Signer*: The party signing blocks.
- *Producer*: The party producing blocks.

Example 1. Let us now define the skeletal procedure followed by our Casper Virtual Worker:

1. A Capitalist supplies the Virtual Worker with an initial investment of ETH.

[4]Signer and Producer both serve as Operators

2. The Virtual Worker divides the ETH investment into fragments; each fragment is used as a deposit in Casper.

3. Whenever Casper elects the Virtual Worker to propose a new block:

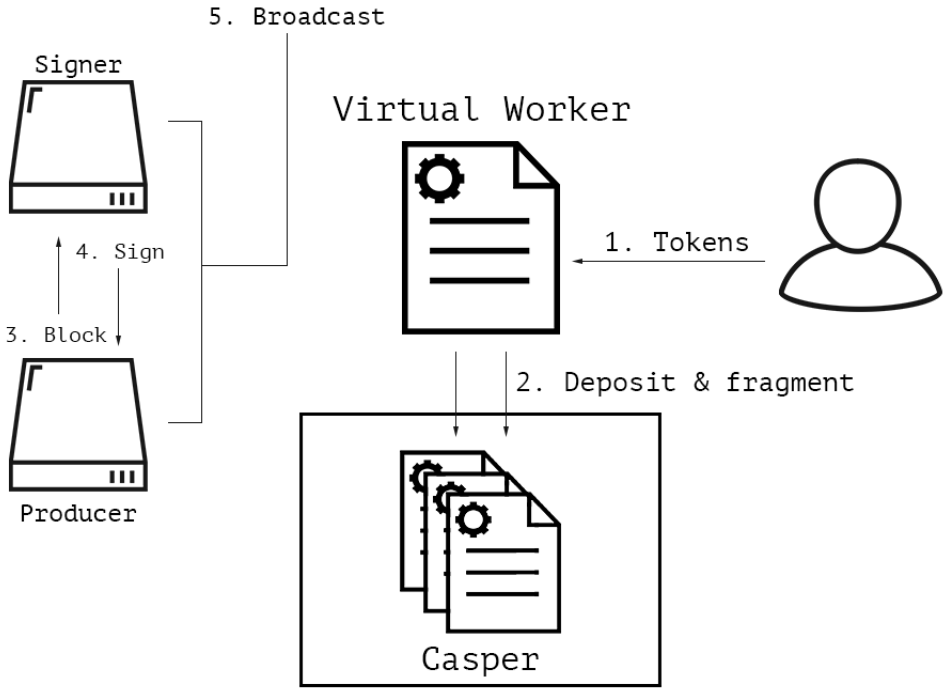
   (a) The Producer constructs a block and requests a signature on the block from the Signer.

   (b) The Signer evaluates the proposed block, signing and returning the block only if it passes his evaluation (e.g. the Signer may limit the rate he signs blocks or only sign blocks with odds below a threshold).

   (c) The Producer publishes the signed block.

4. If the Producer constructs a dunkle or triggers a Casper slashing condition, he loses his security deposit.

5. At the end of the Casper staking period, The Producer and Signer receive compensation relative to the Casper rewards they accrue.



Note, while a Capitalist can serve as his own Signer, alternatively, a Capitalist can delegate the task of signing to a trusted Signer such as a friend or reputable

organization. To achieve greater security guarantees, we plan to construct a *Decentralized Signer*: A decentralized set of machines incentivized to only sign blocks that satisfy an arbitrary validator strategy. Further, we additionally plan to construct a *Producer Work Pool*: a smart contract enabling an arbitrary number of Producers to jointly produce blocks.

5.3 Snow Virtual Worker

The snow protocol allows Ethereum accounts to schedule functions to execute as soon as a provided Boolean function evaluates to true. Specifically, Ethereum accounts can submit (Boolean function, Callback function) pairs; snow then ensures submitted callback functions executes as soon as their respective Boolean function evaluates to true. To ensure execution of callback functions, snow assigns worker machines to recurrently execute registered Boolean functions off-chain. To avoid financial penalties, the assigned workers must alert snow as soon as their assigned Boolean functions evaluate to true. In turn, snow executes the respective callback function.

Example 2. Employing fragmented staking, our snow Virtual Worker subdivides its capital into multiple fragments, using each fragment as a seperate deposit with snow. To ensure that the Operator security deposit always covers the Virtual Worker's vulnerable funds, our Virtual Worker only assigns itself to snow tasks with minimal penalties. Formally, the Virtual Worker contract requires:

$$\sum_{i=0}^{N} p_i \leq \texttt{min\_deposit}$$

Where N is the number of snow tasks assigned to our Virtual Worker and p_i is the maximum penalty snow inflicts if our Virtual Worker fails to notify snow as soon as task i's Boolean function evaluates to true. To incentivize the Operator, the Virtual Worker logic ensures:

- If the Operator incurs a penalty on the Virtual Worker by failing to alert snow as soon as a Boolean function evaluates to true, the Operator immediately loses his security deposit.
- The Operator receives a percentage of the returns he earns the Virtual Worker.

Assuming the Operator selects tasks with low penalties and supplies an adequate security deposit, the Operator gains leverage while the Capitalist's vulnerable funds remain insured.

5.4 Truebit Virtual Worker

Truebit enables trustless off-chain computation on Ethereum. Specifically, Ethereum users can submit computational tasks to Truebit; Ethereum miners, termed Referees, then elect machines, termed Solvers, to solve submitted tasks. Notably,

to participate in the lottery, Solvers must supply a security deposit.

Example 3. Employing fragmented staking, our Truebit Virtual Worker subdivides its capital into multiple fragments, using each fragment as a seperate security deposit in Truebit's Solver election. Further, our Virtual worker restricts the number of tasks its Operator can accept so that the Operator's security deposit always covers the Virtual Worker's vulnerable capital. To incentivize its Operator to submit accurate solutions, our Virtual Worker Contract enforces:

- The Operator loses his Virtual Worker security deposit if he submits an inaccurate solution to Truebit.
- The Operator receives a percentage of the returns he earns the Virtual Worker.

5.5 Future Applications

As Casper, snow, and Truebit mature, we will release complete specifications of our Virtual Workers. Further, we plan to construct Virtual Workers for other upcoming protocols along with generalized Virtual Workers: Virtual Workers capable of performing work for multiple protocols.

6 Future

6.1 Next Steps

In aggregate, we plan to initially release:

- A collection of Ethereum smart contracts enabling the construction of Virtual Workers (see Protocol Schematic)
- An extensible software suite enabling seamless interaction with the `1protocol` ecosystem (see 1client)
- An incentives layer powered by a custom ERC20 Token: CRED (see The Credit Token).
- Virtual Workers for protocols on Ethereum (see Applications)
- Supporting constructions for our Virtual Workers: Work Pools, Funds, Decentralized Signers and Producers

6.2 Digital Service Economy

`1protocol` serves as a platform for users with idle capital and workers with excess computing power to seamlessly perform work together for protocols on Ethereum. Using our platform, users can passively gain interest on their idle capital; similarly, workers can gain rewards employing their computing power

to leverage capital. In short, `1protocol` allows Ethereum users to effortlessly obtain rewards contributing capital or computing power; it sanctions a digital economy with all capital and computing power—human and machine–efficiently allocated.