Consensus V1 Lite Protocol Specifications

This document introduces the initial consensus protocol, specifically designed for the Nebulae Oracle. While the primary focus of this discussion is on price feeds, the protocol's general principles can be applied to a variety of other use cases. The document first covers the Proof of Stake (PoS) protocol, then extends into a hybrid Reputation System.

PoS Protocol (e.g. SchellingCoin)

The SchellingCoin mechanism is one plausible consensus algorithm for a decentralized oracle network. By leveraging game theory, it incentivizes participants to deliver accurate data. The pros and cons of this mechanism are as follows:

Pros:

1. **Honesty Incentivization:** Participants who provide information close to the majority or median value are rewarded, thereby encouraging accurate reporting.
2. **Simplicity and Efficiency:** The mechanism is intuitive and straightforward to implement, requiring minimal computational and communication overhead.
3. **Sybil Attack Resistance:** The need for participants to stake tokens to participate makes it costly for an attacker to spawn multiple identities to manipulate the outcome.

Cons:

1. **Coordinated Attack Vulnerability:** A colluding group of malicious participants holding a significant portion of staked tokens could manipulate the consensus output.

2. **Majority Dependence:** The mechanism assumes that the majority of participants are honest, which if untrue, can lead to inaccuracies.
3. **Limited Scalability:** The mechanism may struggle to handle a large number of participants or frequent updates due to increased time and resource requirements.

To mitigate these limitations and address potential security risks, we propose enhancing this protocol by integrating a Reputation System.

Reputation System Plugin

The Reputation System complements the PoS mechanism by assigning reputation scores to participants based on their historical accuracy. Nodes with higher reputation scores have more influence on the consensus outcome, thereby mitigating the impact of a group of high-stake malicious participants.

Pros:

1. **Enhanced Security:** The PoS and Reputation System hybrid approach increases the cost and complexity of launching a coordinated attack, thus enhancing network resilience.
2. **Long-Term Honesty Incentive:** The Reputation System encourages participants to maintain a history of honest and accurate reporting.
3. **Sybil Attack Resistance:** Both the PoS and Reputation Systems make it costly for an attacker to spawn multiple identities to manipulate the consensus outcome.

Cons:

1. **Increased Complexity:** The hybrid approach adds complexity to the consensus algorithm, potentially requiring more development and maintenance efforts.
2. **Potential Centralization:** A small group of high-reputation, high-stake participants could exert disproportionate control over the consensus process.
3. **Bootstrapping Challenges:** New participants with low initial reputation scores may struggle to gain influence, potentially discouraging diversity and new entrants.

Technical Specifications

Assume we have a set of oracle nodes, O, with n nodes each specified by O_i. Each node reports the price as p_i for simplicity. The state of each node in each round is as follows:

$$O_i = \begin{cases} s_i & \text{stake of the node} \\ r_i & \text{reputation of the node initialized to zero} \\ p_i & \text{reported value for this feed round} \end{cases}$$

We define these notations:

$$\begin{array}{ll} S & \text{set of oracle stakes} \\ R & \text{set of oracle reputations} \\ P & \text{set of oracle reports} \end{array}$$

1. To become an oracle, each node needs to stake at least min_stake amount of tokens/TONCoin and join the network.
2. Nodes have to participate in requests and report the price p_i accurately.
3. (Experimental) The consensus value is determined as the weighted median:

   $$consensus\_value = WeightedMedian(P, R)$$

   where the $WeightedMedian$ is defined as feed p_k that satisfies:

   $$\sum_{i=1}^{k-1} r_i \leq \frac{1}{2} \quad \text{and} \quad \sum_{i=k+1}^{n} r_i \leq \frac{1}{2}$$

   Reputation scores are initialized to zero and normalized before calculating the consensus.
4. We define a threshold config parameter, t, that indicates the allowed deviation from the exact value. Nodes within $\pm t\%$ of the consensus value are considered accurate reporters; others are deemed malicious. The precision of the node pr_i and the accuracy of the node acc_i are calculated as:

$$dev_i = \frac{|p_i - consensus\_value|}{consensus\_value}$$

$$pr_i = 1 - dev_i$$

$$acc_i = \begin{cases} 1 & dev_i \leq \frac{t}{100} \\ 0 & o.w. \end{cases}$$

5. We define the following parameters for our protocol:
    a. A: Total award of the round, distributed fairly between participants according to their stake, reputation, and precision:

$$a_i \propto s_i, r_i, pr_i$$

    b. $P_{malicious}$: Proportional slash amount for a participant's malicious behavior.
    c. $P_{inactive}$: Proportional slash amount for an inactive participant.
    d. h: Reputation weighting factor to balance between recent and past performance, where $0 \leq h \leq 1$. A higher h gives more importance to recent activity.
6. After distributing the awards, we update the reputation score of the nodes:

$$r_i = h * acc_i + (1 - h) * r_i$$

Mitigating Potential Risks

One potential risk arises if all oracles act maliciously and report incorrect data at the start of the protocol. This issue can be mitigated by employing a bootstrapping method where we start the network with a group of trusted oracle nodes that have been carefully vetted. As the network grows and more nodes join, the influence of these trusted nodes gradually diminishes, and the system becomes more decentralized.