Oracle consensus in rational environments

Akis Chalkidis akis@ecoc.io

February 2020

Contents

1 Preface

The purpose of this paper is to analyze the protocols for oracles under specific assumptions. Oracle is an entity that has access to real world data and also power to alter the state of a machine, alone (centralized) or after reaching consensus with other oracles (decentralized).

The obvious interesting case is for decentralized oraclization, that is, a group of oracles that cant be trusted and consensus is needed to decide if a value or event is true or false. If it is true then the transaction can take place and the value can be recorder on the ledger (blockchain). But the results of this paper can be used for any system (not only blockchains) of dishonest players, where a consensus is in need and the action of the players are turn based and take place simultaneously (not sequentially).

2 Basic assumptions

Before presenting the protocol, the assumptions about the environment must be named.

1. Oracles behave *rationally*. That means that they want to maximize their profit, so they can follow or (try to) violate the protocol, according to their own benefit. The term "Byzantine" is also used by many, but rational and byzantine participants are not exactly the same. The only motivation for a rational participant is to maximize his *utility*, where utility has a financial meaning. The motivation (and also behavior) of a byzantine participant can be either rational or *malicious*. By malicious we mean participants who are willing to take financial punishment, trying to damage or even destroy the system. In other words the utility for a byzantine participant can be either financial or malicious to the system. Our assumption is that Oracles have rational - but not byzantine - behavior.

2. Oracles can be *faulty*. They still behave rationally, but for a third person their behavior may look irrational. For example, the device that detects the event or extracts the value is dysfunctional, or they have connection problems , losing the ability to take action at a specific round.

3. Actions of all oracles take place simultaneously. That means that the oracles know the full history of actions of the other oracles except of the current round. That makes them unable to consider the actions of other oracles for the current round. They must make their decision without this information.

4. There is an underlying technology, which guarantees the concealed voting of the current round, the immutable recording of their decisions and their payment (payment can be negative, meaning punishment).

5. Because oracles are rational, their mathematical expectation value must be positive (simply put, they expect a profit). Otherwise, they wouldn't participate in the first place.

6. Oracles are intelligent. That means that they have a *non-myopic* view. They try to maximize their profits long term. They can make a decision that does not look optimal short-term but actually the payout is maximized in the long run.

7. There is no "last round", that is, the rounds are infinite (or at least the expectation is infinity).

3 Goals of the protocol

The final goal is that the consumer can get an answer which is correct by a high probability. The consumer is the entity that sets the question. The consumer can be anyone and the evaluation of the result can either:

1. Be decided on chain (for example, extracted automatically by a smart contract function).

2. Be decided off chain by an algorithm and after evaluation the consumer records it on blockchain (centralized version).

3. Be decided off chain and never be recorded on the chain (the value can be used in a centralized application).

4. Be recorded in a traditional database without evaluation, possibly for future use or for extracting statistical results.

   From the above it is clear that the "conclusion" can be reached outside of the protocol. The protocol just outputs the vote and credibility of each Oracle. The evaluation, be it automated or manual or even not even taking place (just saved for future use) can be carried out considering a) the majority of votes b)the ratio of votes c) the credibility of each voter (weight). The credibility "score" of the Oracles can be extracted from its voting history, so it is considered as output.

4 Threats

There are two basic threats for the system, two obstacles that must be overcome:

- Coalitions: Because of the infinite round expectation the Oracles have a motivation to form a coalition. This is because the coalition surplus may be positive. If they are able to form a majority then they can be sure that they will be rewarded each time. So the protocol must prevent coordination of voting, else the produced output will be useless. The oracles will vote regardless of the true value.

- Free-riding: Individual oracles may try to copy the vote of the majority. That way they secure a reward. This degrades the quality of the output.

It must be clear now that because of the above threats the votes must be concealed until all the Oracles finish the voting process. This can be done in a commit - reveal strategy. So at each voting there are two steps , commit (a hashed value) and reveal (revealing the actual vote). Of course the hashed revealed value must match the committed value. The threats are not completely nullified by this strategy alone, as we are going to see. Oracles are asumed to have intelligence and non-myopic logic. Because they have access to the voting history they can make conclusions and form coalitions or try to guess the current round vote of the others and do free-riding. A build-in punishment system in the protocol must incentivise the Oracles to avoid this behavior.

5 Game Theory

The protocol can be formalized in the mathematical field of *Game Theory*. In game theory we are talking about players, rules, utility function of players, action set and strategies (a series of actions). In our case:

- The players are the Oracles, symbolized by o_i
- The utility function is the financial benefit of each , so it is u_i
- The action set at each round is the vote. Because the value is a boolean it is the set $s : \{true, false\}$
- The strategy of an Oracle o_i is a chain of actions σ_i. For each Oracle o_i there is a set of strategies $\sigma_i : \{\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}, ..., \sigma_{i,n}\}$. The cardinality of the set is $|\sigma_{i,n}| = 2^n$ where n is a sequence of rounds, because s is a boolean($|s| = 2$).

The system can be formalized as a game G with the following elements:

- $G :< O, S, U >$
- O is the set of Oracles: $O : \{o_1, o_2, o_3, ..., o_n\}$
- S is the set of possible strategies $\{\sigma\}$ for each oracle o_i.
  $S : \{\sigma_{o_i,1}, \sigma_{o_i,2}, \sigma_{o_i,3}, ..., \sigma_{o_i,n}\}, \forall i \in O$
- $U : \{u_i\}$ is the utility function for player o_i, where $u_i = B_i$. B is the payout function for the players.

Notes:

- The oracles are *rational* players, so their utility function is of financial nature only, that is, $U \leftrightarrow B$. So we do not consider any *malicious* behavior (goal other than financial).

- For each round the function B accepts as parameters the actions of all oracles who belong the the set O for their current round *only*. That means that the payoff b_i, r for each oracle o_i is completely independent from the previous round. Mathematically, $U_{i,r} = B(\sigma_{i,r})$ for each round r.

- Obviously, the strategy that each oracle o_i is going to follow is that to maximize its utility u_i. We already assumed that all players have an non-myopic view. So they are going to take actions and a strategy $\sigma_{i,max} \in S$ that gives the maximum total payoff and not the maximum payoff of the current round only.

  More precisely, for a round r and an oracle o_i the optimal action $s_{i,r}$ is NOT

  $\forall\ s_{k,r},\ \ s_{max,r}$:

  $B(Es_{1,r}, Es_{2,r}, ..., s_{max,r}, ..., Es_{n,r}) > B(Es_{1,r}, Es_{2,r}, ..., s_{k,r}, ..., Es_{n,r})$

  but

  $\forall\ s_{k,r},\ \ s_{max,r}$:

  $\sigma_{i=1}^{r} B(\sigma_1, \sigma_2, ..., s_{max,r}, ..., \sigma_n) > \sigma_{i=1}^{r} B(\sigma_1, \sigma_2, ..., s_{k,r}, ..., \sigma_n)$ (5.1)

  In the first inequation, $Es_{i,r}$ is the *expected* action of all other players o_i because their action of the current action is unknown.

  The second inequation shows the optimal non-myopic strategy for each oracle. This is the strategy that rational participants will follow. Notice that this equation takes into account the whole history (strategy) of the other players. So the oracle gathers information from the previous rounds, has an expectation for other oracles for their future behavior and by this it can form its strategy accordingly.

  Actually, the above two inequations will be equivalent if:

  - The oracles couldn't have access to the voting history of other oracles.

  - The oracles have a myopic view.

  - For any other reason that could prevent them to form coalitions to coordinate actions in the future rounds.

  - The rounds are not infinite but have a known final round.

  If any of the above conditions hold then optimal action of the oracles will be equivalently, making given payoffs equal by any of the two. But because the above conditions do not hold , any oracle can form or enter a coalition. Because some coalitions

have positive surplus, all rational players will try to create or join one. In the next chapter we are going to talk about coalitions.

- The payout function B must have the property to discourage forming coalitions. To achieve that goal the payoff function B must be such that brings any coalition surplus cloze to zero. Let's symbolize the set of all possible coalitions as C and the surplus of a coalition i as Cs, i. So the payoff function B must have the property:

  $\lim_{r->\infty} Cs, i = 0 \quad \forall i \ \in C \quad (5.2)$

6 Coalitions

We can define a coalition C_i as a set of oracles $O_c \subseteq O$ and $|O_c| \geq 2$ who seek to improve their benefit $\sum_{i=1}^{\infty} B$. For this to happen, obviously, the surplus must be positive: $C_{s,i} > 0$, where $C_i \in C$. The surplus is the payoff on top of the sum of each individual participant in the coalition:

$$C_{s,i} = \sum_{i=1}^{\infty} EB_{C_c} - (\sum_{c=1}^{\infty} EB(o_{1,c}) + \sum_{i=1}^{\infty} EB(o_{2,c}) + ... + \sum_{i=1}^{\infty} EB(o_{|C_i|,c})) \Leftrightarrow$$

$$\Leftrightarrow C_{s,i} = \sum_{i=1}^{\infty} EB_{C_i} - (\sum_{i=1}^{|C_i|} \sum_{i=1}^{\infty} EB(o_{i,c}))$$

Because $C_{s,i} > 0$, we have:

$$C_{s,i} > 0 \Rightarrow \sum_{i=1}^{\infty} EB_{C_i} - (\sum_{i=1}^{|C_i|} \sum_{i=1}^{\infty} EB(o_{i,c})) > 0$$

$$\Leftrightarrow \sum_{i=1}^{\infty} EB_{C_i} > \sum_{i=1}^{|C_i|} \sum_{i=1}^{\infty} EB(o_{i,c}))$$

EB_o is the *expected* payoff for each of the coalition members and EB_C the total payoff of the coalition. So we set:

$EB_C = \sum_{i=1}^{\infty} EB_{C_i}$ and $EB_o = \sum_{i=1}^{\infty} EB(o_{i,c})$. Finally we have:

$$EB_C > \sum_{i=1}^{|C_i|} EB_o \quad (6.1)$$

The payoff function $B()$ must have the property of $C_{s,i} - > 0$, so $B()$ function must set that way that

$$\lim_{r->\infty} EB_C = \sum_{i=1}^{|C_i|} EB_o \qquad \text{where } r \text{ is the number of rounds.} \quad (6.2)$$

Let's step back a moment and think how many coalitions are possible when the number of oracles are n. From combinatorics we know that for a set with cardinality n the possible coalitions are 2^n. But we are not interesting for coalitions with only one player or the empty set $\{\emptyset\}$. Substructing these we get $|C_n| = 2^n - n - 1$. (6.3)

Proving this equation by mathematical induction is easy:

- For $n = 2$ we have $|C_2| = 2^2 - 2 - 1 = 4 - 3 = 1$ which is true (only one coalition is possible, namely $\{o_1, o_2\}$
- We suppose that $|C_k| = 2^k - k - 1$
- We must prove that the equation holds for $n = k + 1$. So we must prove that:

  $|C_{k+1}| = 2^{k+1} - (k + 1) - 1$

  The total number of coalitions $|C_{k+1}|$ is the number of coalitions $|C_k|$ and in addition the $k + 1$ element can create $|C_k|$ more because it can join all existing coalitions. In addition it can create coalitions of any other element and itself: $\{\{o_1, o_{k+1}\}, \{o_2, o_{k+1}\}, ..., \{o_k, o_{k+1}\}\}$. So we have:

  Proof: $|C_n| = |C_{k+1}| = |C_k| + |C_k| + |\{\{o_1, o_{k+1}\}, \{o_2, o_{k+1}\}, ..., \{o_k, o_{k+1}\}\} = 2 * |C_k| + (k) = 2 * (2^k - k - 1) + k = 2 * 2^k - 2k - 2 + k = 2^{k+1} - k - 2 = 2^{k+1} - k - 1 - 1 = 2^{k+1} - (k + 1) - 1$

  QED

  Unfortunately, equation (6.3) has a complexity of $\mathcal{O}(2^n - n - 1) = \mathcal{O}(2^n)$ That means that for even a small number of oracles $|O| = n$ the combinations of coalitions is extremely large. The protocol must be build that way to incentivize forming coalitions of any size.

7 Individual oracle strategy

7.1 Motivation

Before taking any decision for the protocol architecture we must analyze the rational behavior of oracles. An oracle, in his effort to maximize his profit, will try to join a coalition which has the maximum surplus, the maximum benefit for the oracle. This is because while the game is NTU (has non transferable utility) the reward splits equally to each winner. That is, no side payment is possible. In a coalition an oracle has greater chance to be in the majority and get a reward.

But why the coalitions have a surplus? It is very easy to prove that. It is based on fact that the probability for a single round for the oracle to get the value of the event faultly is positive, $p_{sf} > 0$. Belonging to a coalition will always get the reward, while voting honestly they will get the reward at each round with a probability $1 - p_{sf}$ and a punishment (or at least zero reward) with a probability of p_{sf}.

Unfortunately, there is a second reason for oracles to join a large coalition. Even in case that there is no error , meaning $p_{sf} = 0$, it is better for them to be in a coalition. This is because in real world to extract the value of the event has a cost (run a server, pay a third party to consume its API, maintain the infrastructure, develop and update software and so on). But in a coalition always vote is the same (be it true or false, it does not matter), so the oracles can drop the running costs and just vote the same value over and over.

Of course, to join a coalition, an oracle first must detect one. If it detects many coalitions the oracle is going to join the one which gives the highest probability to form the majority, in other words the coalition which have the most members.

7.2 Assumptions

First, we are going to analyze how an oracle detects coalitions. The only information available for each oracle is the history of votes of the other oracles and the real values, which are booleans, of the previous events. By "real" we mean the value that the orale has detected from voting of previous rounds. Our assumptions are the following:

1. All values and votes are booleans
2. There is perfect information of voting history because the votes are recorded on an immutable ledger
3. The actual value of each event (round) is known by the oracle , but this information is imperfect: the oracle may be unable to know the correct value of some events (or even all events if there is a serious malfunction in its infrastructure)
4. Oracle expects that the other oracles are acting rationally (see §2.1)
5. Side communication channels do not exist, that is, sending and receiving information to interact with other oracles is impossible. This insures the NTU property.
6. The probability for each oracle to be faulty is low. This is a logical assumption to make. By low we mean much lower than 50%.

7.3 Indicators

The indicators for an oracle to spot some other oracle which possibly belongs to a coalition are:

- the oracle's history shows that the ratio of its votes are against the correct values with a very high frequency. If this is the case the probability that these oracles vote "wrongly" (against real values) by purpose is high.

- On wrong voting rounds the corresponding reward result is positive with a very high frequency. This is to exclude the possibility that the suspected oracle is *faulty* (continuously imperfect information, making it unable to see the correct value of the events).

- Considering the above, the oracle must know the true value of the events. Unfortunately it can not be sure because we assumed that this information is imperfect, that is, there is a probability that the oracle who does the detection is itself *faulty*. It must also try to compute the probability that it constantly gets wrong values (it malfunctions). So the oracle must also consider its own history.

7.4 Coalition detection

On first glance the above indicators are useless: each oracle can not be sure that it is faulty, so it can not conclude if the other oracles belong to a coalition or they are just honest and vote the correct value. There is one point though that breaks the symmetry to this dilemma: the starting point of history. We have assumed that side communication between oracles,that is, communication outside of the system is impossible (§7.5). Consequently, in the first rounds coalitions are non-existent. The only way to transfer information is voting. Constantly wrong voting signals to other oracles that it wishes to establish a coalition. It is like a beacon , which signals each round, taking a punishment. After many rounds it may succeed, and in the long run, if successful, the alliance surplus is going to cover and exceed oracle's starting losses. So the oracle can safely assume that the first rounds run without any formed alliance.

The above logic stands because of §7.6. If each oracle had a probability of near 50% to be faulty then even the start of history can't be used. But if each oracle has so high chance to be faulty then the system is useless anyway. At $p_f = \frac{1}{2}$ faulty oracles the system entropy is very high and no useful information can be extracted anyway. In practice, to assume that most oracles are not faulty is a true fact.

The urgency of joining a coalition is higher if the oracle believes that it is faulty very often. This will make it willing to copy the votes of the previous winners , which do not change their votes (they always vote "true" or "false"). We can name this strategy as *information free-riding*. The oracle just completely ignores the value that believes is the correct one and copies the vote of the winners of previous rounds.

Let's be more concrete. The first step for an oracle to detect a coalition is to determine for itself if it is constantly faulty or not. His only chance to find that is at the initial rounds, where it is sure that there is no formed alliance yet. We symbolize with p_{cf} the probability of an oracle to be constantly faulty and p_{sf} the probability to be faulty only for a single round. We also assume that total number of oracles $|O|$ are not very low. Under this , the probability for the oracle to have a serious malfunction and be constantly faulty can be computed using the binomial distribution because the value is a boolean and

the actions are taken in rounds (discrete distribution). The oracle does not have a motivation to vote wrongly at first rounds because it will be punished. Only if it wants to initiate a coalition will try that, but in that case it will know it intentionally vote wrongly. Let's assume that it will behave normally. The first k rounds the oracle can safely assume that no large coalition has formed. So, if it is not constantly faulty, it must vote correctly almost all times. Almost, not all times, because $p_{sf} > 0$

Using the cumulative distribution function (CFD) of the binomial distribution we have:

$$CFD(c; k, 1 - p_{cf}) = P(X \leq c) = \sum_{i=1}^{c} \binom{k}{i} (1 - p_{sf})^i p_{sf}^{k-i} \quad (7.1)$$

- c is the limit of correct guesses
- k is the last round number that the oracle believes it is safe to include
- p_{sf} is the probability each round the oracle gets wrong value (much lower than $\frac{1}{2}$)
- X is the number of correct guesses
- $P(X \leq c)$ is the probability that the correct guesses are lower than or equal the limit c

The *expected* value, mean, for the correct guesses of the binomial distribution is $E[X] = k*(1-p_{sf})$, while the variance is $Var(X) = k*p_{sf}*(1-p_{sf})$

By theory we know that the binomial *confidence interval* relies on approximating the distribution of error about the binomially-distributed observation P as in normal distribution. Using the normal approximation, $P(X \leq c)$ is estimated as: $\frac{c}{k} \pm \frac{z}{k}\sqrt{\frac{c(k-c)}{k}}$ (7.2)

where,in our case, $z = 1 - \frac{p_{cf}}{2}$ (7.3)

Combining (7.1),(7.2) and (7.3) we get

$$P(X \leq c) \approx \frac{c}{k} \pm \frac{z}{k}\sqrt{\frac{c(k-c)}{k}} \Rightarrow \sum_{i=1}^{c} \binom{k}{i} (1-p_{sf})^i p_{sf}^{k-i} \approx \frac{c}{k} \pm \frac{1 - \frac{p_{cf}}{2}}{k}\sqrt{\frac{c(k-c)}{k}} \Leftrightarrow$$

$$k\sum_{i=1}^{c} \binom{k}{i} (1 - p_{sf})^i p_{sf}^{k-i} - c \approx \pm\ (1 - \frac{p_{cf}}{2}\sqrt{\frac{c(k-c)}{k}}) \Rightarrow \frac{p_{cf}}{2}\sqrt{\frac{c(k-c)}{k}} > 1 -$$

$$k\sum_{i=1}^{c} \binom{k}{i} (1-p_{sf})^i p_{sf}^{k-i} + c \ \vee\ \frac{p_{cf}}{2}\sqrt{\frac{c(k-c)}{k}} < 1 + k\sum_{i=1}^{c} \binom{k}{i} (1-p_{sf})^i p_{sf}^{k-i} - c \Leftrightarrow$$

$$1 - k\sum_{i=1}^{c} \binom{k}{i} (1-p_{sf})^i p_{sf}^{k-i} + c < \frac{p_{cf}}{2}\sqrt{\frac{c(k-c)}{k}} < 1 + k\sum_{i=1}^{c} \binom{k}{i} (1-p_{sf})^i p_{sf}^{k-i} - c$$

$$\Leftrightarrow \frac{2\sqrt{k}}{\sqrt{c(k-c)}}(1 - k\sum_{i=1}^{c} \binom{k}{i} (1-p_{sf})^i p_{sf}^{k-i} + c) < p_{cf} < \frac{2\sqrt{k}}{\sqrt{c(k-c)}}(1 + k\sum_{i=1}^{c} \binom{k}{i} (1 -$$

$$p_{sf})^i p_{sf}^{k-i} - c)$$

$$\frac{2\sqrt{k}}{\sqrt{c(k-c)}}(1 - k\sum_{i=1}^{c}\binom{k}{i}(1-p_{sf})^i p_{sf}^{k-i} + c) < p_{cf} < \frac{2\sqrt{k}}{\sqrt{c(k-c)}}(1 + k\sum_{i=1}^{c}\binom{k}{i}(1-p_{sf})^i p_{sf}^{k-i} - c) \quad (7.4)$$

From (7.4) the oracle can compute the probability to be constantly faulty p_{cf} with high accuracy. The restrictions for (7.4) to be useful are: $c \neq 0$, $c \neq k$ and k is not very small. Also, c, k and p_{sf} must be known to the oracle for computing p_cf. The correct number of voting, c, is known only because of the fact that there are no coalitions in the k starting rounds, so the majority of votes shows the correct value (we already assumed that p_{sf} is low and far lower than $\frac{1}{2}$). Finally, $c \neq k$ and $c \neq 0$. In either of those cases that $c = 0$ or $c = k$ the (7.4) cant be used. Normal approximation interval can't be used where $\hat{p}$ is close to 0 or 1, that is, when $\frac{c}{k} \approx 0$ or $\frac{c}{k} \approx 1$ These two restrictions, while invalidating (7.4), are not actually a problem. In case that $c = k$ and k is not very small, (7.4) is not needed at all. The oracle can safely conclude that it is not constantly faulty. The probability that an oracle is constantly faulty and correctly guessed all votes by chance is $\frac{1}{2^k}$. For a normal value of k, let's say the first 20 rounds ($k = 20$) the probability is less than one in a million. And the case that $c = 0$ should never arise in practice. A constantly faulty oracle should be correct on an average of half the cases. The probability to always be "unlucky" is again $\frac{1}{2^k}$, which is improbable as we have already seen.

As a final note, the lower the probability of p_{sf}, the higher the accuracy of (7.4) for p_{cf}. This is good because we assumed that p_{sf} is low.

In the case that the oracle detects that itself is constantly faulty it has the following options:

(a) Try information free-riding. Fortunately this is not an option in our system because the vote of the current round is always concealed.

(b) Try to detect and join a coalition. This is the only option it has if it wishes to stay in the system.

(c) Leave the system because it believes that his expected future payoff will be negative ($EB(O_{i,\infty}) < 0$).

Because an oracle, which is constantly faulty, degrades the output quality we don't want them to stay in the system. Later we are going to examine how we can prevent (7.4.b)

Now let us examine the common case that the oracle is not constantly faulty. His strategy is simple enough:

(d) If it detects a winning coalition, that is, a coalition that its surplus is positive, it will join it.

(e) Otherwise it will vote at each round honestly because it believes there are no coalitions, so voting honestly will reward him most of the time. As long as the reward expectation is higher than the penalties it will remain in the system($EB(O_{i,\infty}) > 0$).

(f) Finally, It can try to become a leader, forming a coalition. In that case, after the kth round it will start voting against the real values on purpose. It knows that it will take a financial punishment at each round. This is an option for the oracle only if it believes that the future dividends of the alliance surplus are going to exceed the starting penalties.

Obviously, as architects of the system we don't want (7.4.d) and (7.4.f) ever happen. We want to permanently ban constantly faulty oracles and to prevent functional oracles to create or join coalitions. At **System Architecture** we are going to analyze how the system can achieve that. But first we must analyze coalitions strategy.

8 Coalition consensus

There are three phases for a coalition: The creation phase, the transition phase and the preserving phase.

8.1 Creation phase

The creation phase is the phase where an initiator, which we call "leader", signals to all other oracles to follow him at each round. Communication side channels do not exist but the leader needs to broadcast only one bit of information ("join me!"). It can do that by repeatedly voting against the correct value. Or, alternatively, it can not vote at all (or reveal a value that does not match the committed value). Of course, all these cases are equally handled by our protocol, they are considered wrong values. We must first try to compute the expected loss that the leader and his followers will have until coalition reaches the majority. Here we can safely assume that the oracle got the decision to be leader because its computed probability to be constantly faulty is very low ($p_{cf} \approx 0$). He must broadcast a strong signal. How strong is his signal and how many rounds it needs to vote wrongly it depends on the probability p_{sf} to be faulty for a round. First, there is a "noise" from the other oracles who are faulty for the current round. Each oracle who tries to detect a leader will consider the probability p_{sf}, the number of oracles $|O_{n-1}|$ and the current wrong voters of the previous round, lets say w. The expected value for w is of course:
$E[w] = \frac{p_{sf}}{2}|O_{n-1}|$, because half of the faulty oracles are going to give the wrong value on average.
The larger the value w the more probable is that at least one oracle tries to be a leader. Let $\Delta w = w - E[w] > 0$ Δw is an indicator that some oracles try to form a coalition.

We define as *signal strength* and symbolize by ss the probability that an oracle wrongly votes on purpose. This indicator alone is enough for an ora-

cle to decide if another oracle initiates a coalition. If for an oracle $ss \approx 1$, that means that this oracle tries to be a leader. The serious problem that a leader faces is the fact that his signal will break if it became faulty for a round. At this round, accidentally may vote the correct value, weakening its signal. When faulty at the current round, it may vote the correct value by $\frac{p_{sf}}{2}$.
We must compute the signal strength ss. For the first round, $r = 1$, the probability to be faulty is p_{sf} and constantly faulty p_{cf}. So to choose the wrong value by fault it has a total probability of

$$\frac{p_{sf}}{2} + \frac{p_{cf}}{2} - \frac{p_{sf}}{2}\frac{p_{cf}}{2} = \frac{p_{sf} + p_{cf}}{2} - \frac{p_{sf}p_{sf}}{4} = \frac{2p_{sf} + 2p_{cf} - p_{sf}p_{cf}}{4}$$

We can symbolize the signal strength of the first round of detection as ss_1.
$ss_1 = 1 - (\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4}) \Leftrightarrow ss_1 = \frac{4-2p_{sf}-2p_{cf}+p_{sf}p_{cf}}{4}$
For the second, third,... ith round the signal strength is the same at each round:
$ss_i = \frac{4-2p_{sf}-2p_{cf}+p_{sf}p_{cf}}{4}$
Because the rounds are linearly independent, the signal strength of *sequently wrongly voting* on purpose that an oracle sends to others after r rounds is:

$\overline{ss_{1,r}} = \overline{ss_1}\ \overline{ss_2}\ \overline{ss_3}...\overline{ss_r} = \prod_{i=1}^{r} \overline{ss_i} = (\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r$
$ss(p_{sf}, p_{cf}, r) = 1 - \overline{ss_{1,r}} = 1 - (\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r$
$ss(p_{sf}, p_{cf}, r) = 1 - (\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r \quad (8.1.1)$

Equation (8.1.1) is the most important for our protocol. It will be used to form the payoff function $B()$. It is worth standing and analyzing it more. $ss()$ depends on r, p_{sf} and p_{cf}. Let's examine the monotonicity of the function $ss(p_{sf}, p_{cf}, r)$ according to its inputs. To prove the monotonicity we must find the derivative function for each of its arguments.

Obviously, $\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4} < 1$. Indeed:
$0 < p_{sf} < 1,\ \ 0 < p_{cf} < 1$
$p_{sf} < 1 \wedge p_{cf} < 1 \Rightarrow p_{sf} + p_{cf} < 2 \Leftrightarrow 2p_{sf} + 2p_{cf} < 4$

Also $p_{sf} > 0 \wedge p_{cf} > 0 \Rightarrow p_{sf}p_{cf} > 0 \Leftrightarrow -p_{sf}p_{cf} < 0$
$2p_{sf}+2p_{cf} < 4 \wedge -p_{sf}p_{cf} < 0 \Rightarrow 2p_{sf}-2p_{cf}-p_{sf}p_{cf} < 4 \Rightarrow \frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4} < 1$
QED
We are going to study the monotonicity to all arguments of $ss(p_{sf}, p_{cf}, r)$

$\frac{dss}{dp_{sf}} = \frac{(1-(\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r)'}{dp_{sf}} = 0 - (\frac{2*1+2*0-0*1}{4})^r = -(\frac{1}{2})^r < 0$
$\frac{dss}{dp_{sf}} < 0 \quad (8.1.2)$

The derivative function is always negative so $ss()$ is *monotonic* because it strictly decreasing as p_{sf} is increasing.
Similarly, $\frac{dss}{dp_{cf}} = \frac{(1-(\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r)'}{dp_{sf}} = 0 - (\frac{2*0+2*1-0*1}{4})^r = -(\frac{1}{2})^r < 0$
$\frac{dss}{dp_{cf}} < 0 \quad (8.1.3)$

$ss()$ also decreases monotonically as p_{cf} is increasing.
The result is that the closer to zero the probability for the oracle to be faulty, the stronger the signal on wrong voting. In the edge case that $p_{sf} = p_{cf} = 0$ (8.1.1) gives $ss(p_{sf}, p_{cf}, r) = 1$) That means that even from the first voting against the real value an oracle can be sure that this oracle wants to be a leader and start an alliance with probability 100%. This is logical, of course, because a non faulty rational oracle is willing to take punishment only if it has a reason (to signal to others).

Finally, lets examine the monotonicity of $ss()$ as r changes. We expect, as r increases, the signal strength $ss()$ to increase also. Indeed:
$\frac{dss}{dr} = (\frac{(1-(\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r)}{dr})' = (\frac{-(\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r}{dr})'$
But $\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4}$ does not depend on r, so we can replace it. Let $\omega = \frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4}$. We have
$\frac{dss}{dr} = (\frac{-\omega^r}{dr})' = -(\frac{e^{r\ln\omega}}{dr})' = -e^{r\ln\omega}(\frac{r\ln\omega}{dr})' = -e^{r\ln\omega}\ln\omega(\frac{r}{dr})' = -e^{r\ln\omega}\ln\omega * 1 = -e^{r\ln\omega}\ln\omega$
Additionally, $e > 0 \Rightarrow e^{r\ln\omega} > 0$. Also, we have already proved that $\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4} < 1$. We have: $\omega = \frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4} < 1 \Rightarrow \omega < 1 \Leftrightarrow \ln\omega < \ln 1 \Leftrightarrow \ln\omega < 0$
$e^{r\ln\omega} > 0 \wedge \ln\omega < 0 \Rightarrow e^{r\ln\omega}\ln\omega < 0 \Leftrightarrow -e^{r\ln\omega}\ln\omega > 0 \Rightarrow \frac{dss}{dr} > 0$ (8.1.4)

(8.1.4) shows that the derivative of ss to r is always positive , so it is monotonically increasing. This, of course, is what we expected. Each wrong voting increases the strength signal.

Let's see an example. Suppose that for each round an oracle has a probability to be faulty about 15% and to be constantly faulty at 3%. Additionally, the oracle votes against the correct value three times in a row ($r = 3$). What is the probability that this oracles signals to form a coalition? From (8.1.1) we have:
$ss(p_{sf}, p_{cf}, r) = 1-(\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r \Rightarrow ss(0.15, 0.03, 3) = 1-(\frac{2*0.15+2*0.03-0.15*0.03}{4})^3 \Leftrightarrow$
$\Leftrightarrow ss(0.15, 0.03, 3) = 1-(\frac{0.3+0.06-0.0045}{4})^3 \Leftrightarrow \Leftrightarrow ss(0.15, 0.03, 3) = 1-(0.21375)^3 \Leftrightarrow$
$\Leftrightarrow ss(0.15, 0.03, 3) = 1-0.009766037109375 \Leftrightarrow \Leftrightarrow ss(0.15, 0.03, 3) = 0.990233962890625$
That means that the oracle with a probability of 99% tries to start an alliance. The "doubt" that it is faulty is only 1%.

We can improve more (8.1.1). So far we assumed that the voting is independent each round, but this is not completely correct. If the oracle is constantly faulty then linear independence assumption is not very sound. For our example, the probability of the continuously faulty oracle, which also was unlucky for three consequent rounds is simply:
$ss(p_{cf}, r) = p_{cf}(\frac{1}{2})^r = 0.03 * 0.5^3 = 0.03 * 0.125 = 0.00375 = 0.375\%$
In our example (8.1.1) holds. But it could be the case that $p_{cf}(\frac{1}{2})^r > \overline{ss_{1,r}} = \frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4}^r$ To be more precise we must get the minimum value of the two expressions, as we expect the rational oracles will do. So, finally we replace (8.1.1) (with 8.1.5)

$ss(p_{sf}, p_{cf}, r) = min\{1 - (\frac{2p_{sf}+2p_{cf}-p_{sf}p_{cf}}{4})^r, 1 - p_{cf}(\frac{1}{2})^r\}$ (8.1.5)

The leader is going to take punishment for the first k rounds until it sees that the majority follows him (they copy his value). At this point he will switch to the next phase - the transition phase. As for the followers the initial punishment for them will be less than the leader because they will join after some rounds (when they are almost sure about the leader). Their best strategy - their *dominant strategy* - is to follow the leader. For the leader, the negative payoff must be so high from the protocol as to prevent him to initiate the coalition. That is ,

$\sum_{i=1}^{k} B_{O_l} - \sum_{i=k+1}^{\infty} EB_{C_{O_l}} < 0$ (8.1.6),

where B_{O_l} is the initial punishment and$EB_{C_{O_l}}$ the expected dividends of the coalition surplus for the leader.

In theory the above inequation has no solutions. This is because the rounds are infinite, meaning that the total dividends of the coalition surplus is infinite, so it can always cover the initial punishment. Fortunately there is a clever strategy, namely NBP (the term is ours), which can solve this problem.

8.2 Transition phase

The leader can detect the number of its followers very easily, applying the equation of (8.1.5) for each of the other oracles. When it is convinced that there is majority it will *shift* his vote from voting against the real value to voting the real value. This will happen only for a single round. This is a bit of broadcasted information meaning "everyone switch now to this constant value". The leader could not vote or illegally vote achieving the same result. In the next round, all other members are going to see the shifting(except the faulty ones). They can follow immediately or wait some rounds. For the oracles who are going to wait, the probability that the leader did not intentionally switched phase but was faulty is given by the equation (8.1.5), where $r = 1$ and $p_{cf} = 0$(oracles can safely conclude now that the leader is not constantly faulty):

$ss(p_{sf}, p_{cf}, 1) = min\{1 - \frac{2p_{sf}}{4}, 1\} = 1 - \frac{p_{sf}}{2}$ (8.2.1)

At each round c , r increases only if the leader's vote is the same as the real value. In other words, skipping the rounds at witch the leader's vote are against the real value, the probability that the leader has switched can be computed using (8.1.5) where $\mathrm{p}_(cf) = 0$:

$\mathrm{ss}(\mathrm{p}_{sf}r) = min\{1 - (\frac{2p_{sf}}{4})^r, 1\} = 1 - (\frac{p_{sf}}{2})^r$ (8.2.2),

under the condition that the leader never changed its vote and r is the number of rounds without computing the rounds that the leader's vote was against real value. In a few rounds $ss(p_{sf}r) \approx 1$, at which point all oracles will be already switched. This finalizes the transition state and coalition is established.

8.3 Preserving phase

After establishing a coalition the members always vote the same value regardless the real value (always true or always false). Actually, they don't even need to get the value from the real world. The other oracles who belong to the minority and didn't join yet will receive financial damage each time the correct value will be different from the dominant alliance vote. So they are punished for being honest. Their dominant strategy is to join the alliance. Actually, because oracles are acting rationally, they are going to join in the second round after the transition finish. For a large number of oracles $|O_n|$ they can detect the alliance with high probability even before the coalition enters the final phase. Using (8.1.5) for each other oracle they can conclude with a probability very close to $p_a = 1$ that the alliance has taken place:
$p_a = 1 - \prod_{i=1}^{c}(1 - ss(p_{sf}, p_{cf}, r_c)) \approx 1$ (8.3.1),
where $c = |C|$ are the possible coalition members (the oracles who constantly vote against the real value).

Consequently, all oracles - except the constantly faulty, which can't detect the behavior of other oracles - are going to join the alliance, forming the *Grant Coalition*. After that point the output will be the same every time. This is the *equilibrium* point for the rational oracles, at which the output is of no value, rendering the system completely useless.

It is clear that the system must never reach this state. The architecture must prevent that from happening.

9 Architecture of the system

9.1 Introduction

The goal of our system is to produce the most quality output for the consumer(§3). The two threats that can degrade (free-riding information) or render useless the output (coalition) (§4) must be successfully countered.

Free-riding information can be easily prevented by the *commit - reveal* strategy. A pair of a commit and a reveal is defining a *round*. The committed value is the resulting hash of the vote of some algorithm that is common for all oracles. The prehashed message does not contain only the vote (because precomputing only the two possible combination will reveal the vote before concealing). The prehashed value contains:

- The unique ID of the oracle. This is to prevent other oracles to copy the hashed value. Without the unique ID any oracle can copy and commit the hashed value (because in real world the systems are not synchronous), copying that way the vote of another oracle. Including the unique ID

invalides the hash for all other oracles. For blockchains the public address can be used as an ID.

- A fixed length, long in bits, one time random value. This is to prevent other oracles to successfully brute-force the prehashed value. The protocol must consider as illegal a prehashed value that uses the same random value for a second time. The reason is obvious, using the same value actually reveals the vote before the concealing step.
- a bit (boolean). This is the vote (1 for "true" and 0 for "false").

From the above we can be sure that no free-riding takes place as copying votes of the current round is impossible, under the assumption that side communication is impossible(§2.3).

Countering the forming coalitions or dissolving existing ones is much more challenging. At first glance inequation (8.1.6) is disheartening. The dominant strategy of the oracles, even the constantly faulty ones, is to join a coalition. The equilibrium point is the establishing of the Grant Coalition, at which point the output is completely useless. And because of the infinite rounds assumption no starting penalty can prevent them to reach this point.

Someone could argue that the system could decide with a high probability that a coalition is formed when the majority of the votes never change. This would be true if the real value , be it true or false is not close to 100%. Unfortunately in real world the consumer can set a long series of events for validation with probability of same value being close to $p = 1$. For example, asking assurance for the quality of products. The product can have a change of only 1/1000 or even 1/10000 to be dysfunctional. So recording same sequental value of a thousand or more rounds would be valid but detected as invalid with this logic. The system can not judge from this fact alone because it does not know the real value of each event. Only non-faulty oracles can read the real value.

Fortunately there is a strategy to counter this , the *Nobody is Perfect strategy*, which we are going to refer as *NBP* for short.

9.2 Nobody is Perfect (NBP) strategy

NBP is an unorthodox, at least at first glance, method that can prevent players from cheating.Inequation (8.1.6) looks impossible to be beaten because when we formed it we assumed that oracles are rewarded each time their vote is "correct" (it agrees with the majority). Punishing (or at least not rewarding) oracles for winning voting, *sometimes*, may look irrational but it is very rational at a closer look. Because nobody is perfect ($p_{sf} \neq 0$), it is highly improbable that an oracle will vote correctly 100% for all times. In a large sample of rounds it deviates too much from the expected value , which is $E[X] = k(1 - p_{sf})$, where k is the sample size(number of rounds), as we have already seen.

As an example from real life, weak students when cheating they do not answer correctly all tests but do mistakes on purpose. This is to convince their teacher that they didn't use external help. They want their performance to look real, because it is closer to the expected one.

We are going to examine the above more formally. We assume that the oracle can be faulty each round with a probability $p_{sf} > 0$. Under the assumption that an oracle is honest it has a probability to vote for the real value $p_c = \overline{p_{sf}} = 1 - p_{sf}$ and to vote correctly for all r rounds:

$p_{c=r} = \prod_{i=1}^{r} (1 - p_{sf}) = (1 - p_{sf})^r$ (9.2.1)

For example, if $p_{sf} = 5\%$ for a random sample of $r = 100$ rounds the probability that the oracle is voting correctly without being in the alliance is:
$p_{c=100,r=100} = (1 - p_{sf})^r = (1 - 0.05)^{100} = 0.95^{100} = 0.00592$ We can conclude that with a probability 0.592% the oracle was lucky, getting all answers correctly and with a probability $100\% - 0.592\% = 99.408\%$ the oracle is cheating (being a member in a coalition).

We are going to punish each oracle for being "too lucky". The punishment will depend on the values r and p_{sf} and also the expected performance, which is based at the value of $E[c] = r(1 - p_{sf})$. In the above example, $E[c] = r(1 - p_{sf}) = 100(1 - 0.95) = 95$. So, after r rounds the protocol must remove any excess reward the oracle got from the expected reward. Better still, the reward will be computed but not given each round until the end of the sample r. We are going to call this batch of sequential rounds a *session*.

We are going to introduce a constant for the system, *threshold* th, which is a limit of the probability of equation(9.2.1). Upon this parameter and the parameter p_{sf} that represents the probability of each oracle to be faulty in a round(cant get the real value) the system can compute r. Each session consists of sequential rounds r. So r can be computed from (9.2.1) replacing $p_{c=r}$ with the threshold th:
$p_{c=r} = (1 - p_{sf})^r => th = (1 - p_{sf})^r \Leftrightarrow ln_r th^r = ln_r(1 - p_{sf})^r \Leftrightarrow rlnth = ln(1 - p_{sf}) \Leftrightarrow r = \frac{ln(1-p_{sf})}{lnth}$
Example: We decide that an oracle can be lucky with a probability $1/500$. So $th = 1/500 = 0.002$. We also believe that $p_{sf} = 7\% = 0.07$. So the number of rouns r for each session must be: $r = \frac{ln(1-p_{sf})}{lnth} = \frac{ln(1-0.07)}{ln0.002} = \frac{ln(0.93)}{ln0.002} \approx \frac{-0.072570692834835}{-6.21460809842219} \approx 85.635$ r is an integer so $r = \lceil 85.635 \rceil = 86$ That means that each session must consist of 86 rounds. Consequently:
$r = \lceil \frac{ln(1-p_{sf})}{lnth} \rceil$ (9.2.2)

Expected reward for each *session* must also be computed(without the final excess removal, without NBP strategy). Let's symbolized it as $E[r_s]$. The expected reward, also called *dividend*, for each oracle which is not constantly

faulty , for a random round i is $E[r_i] = (1 - s_{pf})\frac{R}{|O|(1-sp_f)} = \frac{R}{|O|}$. Here, R is the external reward of each round (the consumer of the service gives this reward) and $|O|$ the number of oracles. We expect $|O|(1 - sp_f)$ oracles to be non-faulty and get the dividend from R and $(1 - s_{pf})$ the probability of the oracle to be non-faulty, so it is the probability it gets the dividend for the round i. For a session, the expected reward for an oracle that is not misbehaving and is not constantly faulty is: $E[r_s] = rE[r_i] \Rightarrow E[r_s] = \frac{rR}{|O|}$ (9.2.3)
(9.2.3) provides us the *expected* reward for each oracle for the whole session. Our NBP strategy must remove any excess reward from the actual reward. At the last round of the season the protocol will give $min\{r_s, \frac{rR}{|O|}\}$(9.2.4),
where r_s is the total reward session that the oracle would receive if no NBP strategy was present. It is possible that the reward R each round is different because different clients are willing to pay different amounts and the system accepts them. In this occasion we can simple replace R with the mean reward of the session $\overline{R}$.

What is the benefit of this strategy? The benefit for the system is that it decreases the expectation of the reward of future rounds when it belongs to a coalition , putting the limit at the expected reward when the oracle is honest. That has the property of zeroing the coalition surplus. That is, (9.2.4) brings the desired result for (5.2): $\lim_{r->\infty} Cs, i = 0 \ \ \forall i \ \in C$
Examining (8.1.6) under NBP strategy we can easily see that now that it holds every time. Now $\sum_{i=k+1}^{\infty} EB_{C_{O_l}} = 0$ because the expected coalition surplus is zero. Also, $\sum_{i=1}^{k} B_{O_l}$ is negative because the leader or followers receive punishment at creation and switching phases. This brings the fact that (8.1.6) inequality is true every time. Any rational oracle, and by our assumption (§2.1) all oracles, will NEVER try to form or join a coalition. With NBP strategy the coalition threat is successfully countered.

9.3 Rewards and punishment

Before forming the payoff function $B()$ a sum up of rewards and punishment is necessary. Payoff function must be such that:

1. prevents constantly faulty oracles from participating(§7.4b). It must inflict heavy penalties or ban completely these oracles. We prefer penalties and not a permanent ban because there is always a small error of false positive, especially in cases that the threshold th is very high. The choice (and risk) is left to each oracle. Anyway, constantly faulty oracles will receive heavy penalties forcing them to finally stop.

2. prevents oracles from joining a coalition, that is, to become followers(§7.4d).

A combination of punishment at creation phase and NBP strategy, which zeroes the expected coalition surplus must be used.

3. prevents oracles from starting a coalition (leaders, §7.4f). Similarly to above, a combination of punishment at creation phase and NBP strategy is needed.

4. For all non constantly faulty honest nodes the total expected profit must be positive. In other words, they must have a profit. If this is not the case then the oracles will leave the system.

It is time to examine the payoff function $B()$.

9.4 Formalizing the payoff function

The payoff function must achieve all the above §9.3.1-§9.3.4 The arguments of B() are the inputs of the system th, R_{min},p_{sf}. R_{min} is the minimum reward the system accepts from the clients because it must cover the operating costs of the oracles. A client may be willing to provide rewards R greater than R_{min} if he is in a hurry and there is a queue, that is, he needs priority over other clients. But in any case for any round i, $R_i \geq R_{min}$. We are going to examine more R_{min} at system core architecture.

Additionally, r is an input also. r is computed from (9.2.2):
$r = \lceil \frac{ln(1-p_{sf})}{lnth} \rceil$
Because constantly faulty or potential leaders must be urgently countered, the punishment must be devastating enough. We are going to follow the *exponential* method. Every time that sequentially there is a wrong voting the penalty increases exponentially. Starting, no guarantee deposit is needed. After the first mistake, a guarantee must be deposit, which depends on probability p_{sf}. A high p_{sf} of the system must be more tolerable while a low p_{sf} must inflict heavier punishment. We must not forget our restriction of §9.3.4, which states that the reward expectation of the honest nodes must be positive, $E[s_i] > 0$, else they will abandon the system.
For a honest node the probability that it will be unlucky one time is p_{sf}, two times p_{sf}^2 and generally n times in a row p_{sf}^n. Also, without punishment it expects an average reward for the session $E[r_s] = \frac{r\overline{R}}{|O|}$(9.2.3) $E[r_s] > 0$ is positive. But the expectation must be positive for the honest oracles even after the subtracted punishment. If PU_s is the total punishment we have:
$E[r_s] - PU_s > 0 \Rightarrow \frac{r\overline{R}}{|O|} - PU_s > 0 \Leftrightarrow PU_s < \frac{r\overline{R}}{|O|}$ (9.4.1)
This should be our limit. We set $PU_1 = 0$, meaning that for the first error we just tolerate the oracle. But for two sequentially wrong votes we set $PU_2 = \frac{R_2}{|O_c|s_{pf}}$ and for e sequential errors we set $PU_e = \frac{R}{|O_c|s_{pf}^{(e-1)}}$. The guaranteed amount must always cover this value before each new round starts. At each round of a sequential wrong vote a new deposit is made to cover PU_e. A correct voting

releases the exceeded guarantee back to the oracles and sets PU counter to zero. Obviously, the worst case scenario for a honest node is to be faulty at all rounds (an unlike scenario). The expected punishment for this is:

$E[PU_r] = p_{sf}^r \frac{\overline{R}}{|O_c| p_{sf}^{(r-1)}} + \sum_{i=2}^{r} ED_i$, (9.4.2)

where p_{sf}^r is the probability that the honest oracle is unlucky (faulty) at each round, $\overline{R}$ is the mean reward or rounds and $|O_c|$ the oracles that vote correctly and get dividends from the reward at each round. Using (9.4.2) we must prove that $maxE[PU_e] < \frac{r\overline{R}}{|O|}$ to be sure that our protocol is not too devastating for the honest nodes, forcing them to leave the system.

Also the punishment includes the fact that no dividents will be given after the first wrong answer so this also adds to the punishment. We symbolize the expected dividends as $\sum_{i=2}^{r} ED_i$ in (9.4.2)

Finally, $|O_c| = \overline{s_{pf}}|O|$

We must prove that $maxE[PU_e] < \frac{r\overline{R}}{|O|}$ (9.4.3). We have:

$$maxE[PU_e] = E[PU_r] = p_{sf}^r \frac{\overline{R}}{|O_c| p_{sf}^{(r-1)}} + \sum_{i=2}^{r} ED_i = p_{sf} \frac{\overline{R}}{|O_c|} + \sum_{i=2}^{r} p_{sf} \frac{\overline{R}}{|O_c|} =$$

$$p_{sf} \frac{\overline{R}}{|O_c|} + (r-1) p_{sf} \frac{\overline{R}}{|O_c|} = r p_{sf} \frac{\overline{R}}{|O_c|} = p_{sf} \frac{r\overline{R}}{\overline{p_{sf}}|O|} = \frac{p_{sf}}{1-p_{sf}} \frac{r\overline{R}}{|O|} \quad (9.4.4)$$

Our assumption is that the probability p_{sf} is lower that 50% (§7.2.6). Consequentaly,

$p_{sf} < \frac{1}{2} \Leftrightarrow 2p_{sf} < 1 \Leftrightarrow p_{sf} + p_{sf} < 1 \Leftrightarrow p_{sf} < 1 - p_{sf} \Leftrightarrow \frac{p_{sf}}{1-p_{sf}} < 1 \Leftrightarrow$

$\frac{p_{sf}}{1-p_{sf}} \frac{r\overline{R}}{|O|} < \frac{r\overline{R}}{|O|} \Rightarrow maxE[PU_e] < \frac{r\overline{R}}{|O|}$

QED

We proved that our payoff function complies with (§9.3.4). Honest nodes can stay in the system. But attackers (potential leaders) and constantly faulty oracles must be punished heavily, forcing them to leave our system. Indeed , (9.4.3) guarantees this. They key to understand this is p_{sf}^r. Lets analyze the limit for three cases, for honest nodes, attackers and constantly faulty oracles:

- Honest oracle: $\lim_{r->\infty} p_{sf}^r = 0$, because $p_{sf} < 1$
- Attacker(leader): 1 , because he voluntarily chooses to vote wrongly $p_{sf} = 1$
- Dysfunctional oracle: Because it has no access to the real value it will vote correct on average half of the times. For long session, r large, its financial expectation is negative, so rationally behaving it will leave the system.

As an example for the attacker , if $p_{sf} = 5\%$, voting wrongly 4 times in a row ($r = 4$) will inflict him a punishment of loosing the 4 dividends of these rounds plus:

$PU_4 = 1^4 \frac{\overline{R}}{|O_c| 0.05^{(4-1)}} = 20^3 \frac{\overline{R}}{|O_c|} = 8000 \frac{\overline{R}}{|O_c|} = 8000\overline{D}$. That means that it must pay 8000 dividends! Clearly attacks are now financially suicidal. We can formalize our payoff function , combining (9.2.4) and (9.4.2).

$B(th, R_{min}, p_{sf})$ (9.4.4):

- Number of rounds for each session: $r = \lceil \frac{ln(1-p_{sf})}{lnth} \rceil$ (9.2.2)
- NBP strategy: $min\{r_s, \frac{rR}{|O|}\}$(9.2.4)
- Punishment $PU_e = p_{sf}^e \frac{\overline{R}}{|O_c| p_{sf}^{(e-1)}} + \sum_{i=2}^{e} ED_i$ (from 9.4.2)

9.5 Core Architecture

We are finally ready to provide the description of the whole system.

9.5.1 Inputs

There are some necessary inputs which our system needs to implement all the above. The inputs are:

1. th, the threshold which is a limit of the probability of equation (9.2.1)
2. R_{min}, the minimum reward the system accepts from the client
3. p_{sf_0}, the initial known probability that an oracle can by faulty at each round. This information is crucial and must be very close to the real one
4. z and e , where z is the level of confidence and e is the maximum acceptable error. See Determining sample size. This values will be used to set an interval that will recompute the probability p_{sf} at each epoch. Epoch is a number of rounds, divisible exactly by r (contains an integer value of sessions)

9.5.2 Initial computations

Initially, our system computes the values r and ep.

- r is the number of rounds each session has and is computed by (9.2.2)
- ep is computed using the formula $ep = \frac{p_{sf_0}(1-p_{sf_0})z^2}{e^2}$
- ep are the rounds of the epoch and must be divisible by r. So ep must be rounded up, $ep \equiv 0(\mod r)$

ep represents the size of the sample that is enough to recompute p_{sf} . It must be recomputed for two reasons:

- The initially estimated probability p_{sf_0} may deviate much from the really one. In this occasion serious problems may arise. Recomputing p_{sf} using last epoch data can correct the problem.
- In real environments conditions may change. The probability may change from external events. Recomputing helps the system to update the value closer to the real one.

9.5.3 Process

The process that our system functions is the following:

- The clients set a sentence, pay a reward and are waiting the output of the system, which is an array of booleans (votes). The clients also have access to the ledger, reading the performance history of each oracle. From these two they can learn (or convince others) that the questionable event is true or false.
- The system checks if their reward isn't lower that than the minimum. If it is, it returns it and rejects the request.
- On many requests the system gives priority to the clients with the higher reward.
- Oracles are voting simultaneously , following a commit - reveal scheme, according to (§9.1)
- If the vote of the oracle is invalid or belong to the minority it must deposit a guarantee to the system. Sequential wrong voting increases the amount deposit of deposit each round, according to $B()$ punishment function (§9.4). A correct voting resets the punishing counter.
- Income from punishments stay in the system, they are not given as dividends to other oracles. This avoids many types of attacks on honest nodes as it removes any motivations.
- The reward is computed at each round and is equally divided between the oracles who form the majority. The reward is not given at each round but accumulated and released after the last round of the session.
- At the end of each epoch, p_{sf} is recalculated simply by getting the ratio of wrong to total votes of the last epoch. It is possible that epoch duration consists of only one session.
- If the rewards for an oracle exceed the expected value for the session the excess is slashed and kept by the system: Nobody is Perfect Strategy (§9.2)
- Always dividends are given according to the payoff function (9.4.4)

- Payments outside of the system are completely forbidden. The same is true for payments between oracles. On any of this cases all related oracles are banned immediately. All guarantees of them are confiscated.
- New oracles can join only at the first round of a session.

The above complete the protocol. We hope that we mathematically proved that this protocol achieve all goals (§3), under the assumptions we made at (§2).

10 Conclusion

We provide a system and a protocol that forces the rational oracles to be honest. Honest oracles combined with the fact that p_{sf} is not high, produces a quality output, where the consumer can reach a conclusion for an event with very high probability of accuracy.

For completeness we must criticize our system. Our system fails when:

- real probability of error is much different than the initial one. This can be corrected at the next epoch
- real probability of error is very close to 50%
- oracles can communicate outside of the system (side channels). That way the system is not NTU any more and there are Nash equilibrium points that render the output useless
- many oracles do not have rational behavior in the system, they act maliciously (they are willing to take large financial damage to degrade the system output). The oracles may look irrational but are rational if they are payed by someone OUTSIDE of the system (bribing), which has the benefit to pass a wrong value through oracle consensus if the briber is willing to pay a high price. In this occasion the oracle behavior is rational outside of the system. This is especially true if it happens only for one specific round. In any case, we are talking about oracle *bribing* without the system being able to detect it.

Our system is perfectly designed and produces the optimal quality output(as much as s_{pf} allows it) in the case that:

- all our assumption (2.1 - 2.7) hold.
- real $p_{sf_{real}}$ is known by the system with high accuracy, that is, $p_{sf_{real}} \approx p_{sf_0}$. If the real probability of each oracle to be faulty for a round is far from the value set, then the $B()$ payoff function will not be correct, either over punishing the oracles, making their operation unprofitable as expected benefit will be negative and forcing them to leave the system,

or under punishing them, allowing them to form and preserve coalitions. Under the above assumptions we have proved that our system can produce the best output, with the limitation of course of p_{sf}. The closest the p_{sf} to $\frac{1}{2}$, the lowest the usefulness of the output.

References

On Harsanyi Dividends and Asymmetric Values,
https://www.researchgate.net/publication/336722902_On_Harsanyi_Dividends_and_Asymmetric_Values

Determining sample size; how to calculate survey sample size
https://www.researchgate.net/publication/322887480_Determining_Sample_Size_How_to_Calculate_Survey_Sample_Size

Binomial proportion confidence interval
https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval

The free-rider problem
https://en.wikipedia.org/wiki/Free-rider_problem

Entropy - Information Theory
https://en.wikipedia.org/wiki/Entropy_(information_theory)

Essentials of Game , Theory Kevin Leyton-Brown and Yoav Shoham , 2008

Cooperative games: core and Shapley value , Roberto Serrano , 2007

On non-transferable utility games with coalition structure
https://www.academia.edu/29499635/On_non-transferable_utility_games_with_coalition_structure

Cooperative Game Theory and Its Application in Localization Algorithms
https://www.intechopen.com/books/game-theory-relaunched/cooperative-game-theory-and-its-application-in-localization-algorithms

An analytical study of the N-person prisoners' dilemma
https://www.researchgate.net/publication/266710753_An_analytical_study_of_the_N-person_prisoners'_dilemma