The dRanking Protocol

Ryan J. Kung
ryankung@ieee.org

February 11, 2023

Abstract

This article introduces a reputation system for monitoring and evaluating the performance of nodes in a structured p2p network. The system is designed to promote good behavior and prevent cheating among nodes in the network. The reputation system is based on individual local rankings for each node, as well as global rankings generated through random sampling. The local rankings take into account the behavior of each node within its own network, while the global rankings provide a broader view of the behavior of nodes in the entire network.To ensure the robustness of the network, the reputation system uses reward proofs and punishment proofs. Nodes with good behavior are rewarded, while nodes with bad behavior are penalized. This helps to create an environment where nodes are incentivized to behave properly, and cheating is discouraged.

Preview

1 Introduction and Motivation

The reputation system proposed in this article addresses the challenge of evaluating node performance and preventing cheating in structured p2p networks. It monitors node behavior through local and global rankings and uses reward and punishment proofs to incentivize proper behavior and discourage cheating. The system operates under the assumption of the Byzantine generals, where at least 2/3 of the nodes are honest [**?**]. This helps to promote a healthy and robust network and provide secure and efficient services to users.

1.1 Local Ranking

Ranking protocol is inspired by Edonkey's [1] Ranking Queue and uses a similar approach to monitor the performance of nodes in the network. The goal is to prevent cheating and denial of service and to maintain a healthy and robust network.

We builds a measurement and local ranking system by establishing a mutual scoring system among nodes. The measurement system takes into account several metrics, including the success rate of requests sent, the validity rate of requests received, and the total number of successful interactions. These metrics help to provide a comprehensive view of a node's behavior and reliability.

By having each node independently maintain the scores of surrounding nodes, by creates a decentralized and distributed local ranking system. This system allows for a more accurate and comprehensive view of a node's behavior and reliability, as it takes into account the observations of multiple nodes in the network.

1.2 Global Ranking

After obtaining the local ranking, we use a random sampling method to obtain the global ranking. The random sampling method is based on a decentralized random number oracle. The use of a decentralized random number oracle helps to ensure the fairness and impartiality of the global ranking. This helps to prevent any biases or manipulations in the global ranking, ensuring that nodes are evaluated fairly and accurately.

1.3 Reputation

The ranking protocol uses the reputation system to incentivize and ensure fair local and global rankings. The global ranking is generated through fair random sampling of the local ranking.

The reputation system rewards nodes for good behavior and punishes nodes for bad behavior. This helps to incentivize nodes to behave properly and discourage cheating. The reputation system is designed to maintain a healthy and robust network by promoting fair and honest behavior among nodes.

2 Related Work

In the field of peer-to-peer networks, the edonkey Ranking Queue is a notable mechanism for measuring and evaluating the performance of nodes in the network. The Rings network uses a similar approach to the edonkey Ranking Queue to implement its local ranking system.

One related work to the Rings network is the eDonkey network, which uses the edonkey Ranking Queue as its mechanism for evaluating node performance. The eDonkey network was one of the first peer-to-peer networks to use a mutual scoring system among nodes to evaluate node performance and prevent cheating or denial of service.

Another related work is the Bittorrent network, which uses a mechanism called "choking" to prevent cheating or denial of service. The Bittorrent network evaluates node performance based on the speed and reliability of data transfers, and nodes that perform poorly are "choked" or restricted from receiving data from other nodes.

Overall, the Ranking protocol is inspired by the edonkey Ranking Queue and other related works in the field of peer-to-peer networks. By using a mutual scoring system among nodes, the Rings network aims to provide a secure and efficient peer-to-peer network that can accurately evaluate node performance and prevent cheating or denial of service.

3 Sampling

The dRanking protocol effectuates a transformation from local ranking to global ranking through a random sampling procedure, which can be decomposed into four phases:

1. Generation of a random seed:

A random seed is generated to guarantee the randomness of the sampling procedure. The random seed is typically generated through a decentralized oracle to ensure a lack of centralization and bias in the generation process.

2. Determination of sampling targets: Sampling targets are determined based on the local rankings and the randomly generated seed.

Algorithm 1 Systematic Sampling

1: $K \leftarrow \lfloor \frac{2^n}{2^m} \rfloor$
2: $r \leftarrow$ random number between 0 and 1
3: $s \leftarrow \lfloor K \cdot r \rfloor$
4: Select elements at positions $s, s + K, s + 2K, \ldots$

We use simple Systematic samping algorithm 1 here, where n is the number of bits in the random number and m is the desired range of the random sample. The final result is a systematic sample of elements within the range (0, 2^m).

3. Sampling process: The elements in the local

Preview

ranking are selected as a sample based on the determined sampling targets.

Due to the potentially large number of nodes in a DHT [2] network, it may not be possible to exactly locate the target in the sample processing stage. As a result, the sample processing becomes an approximation process, where the DHT network uses a lookup algorithm to find the peer closest to the sample target and provides proof of proximity.

This is because the number of nodes in a DHT network can be extremely large, and it may not be feasible to store information about every node in the network. The lookup algorithm helps to mitigate this issue by finding the closest peer to the sample target, and the proof of proximity helps to ensure that the peer found is indeed close to the target.

Algorithm 2 Proof of Proximity using Bloom Filters [3]

Input: Target element x, Bloom filters $BF_1, BF_2, \ldots, BF_n$ of nodes $N_1, N_2, \ldots, N_n$ in the DHT network
Output: Node N_i with the closest proximity to target element x
Initialize $max_bits = 0$ and $closest_node$ = null
for each node N_i in the DHT network **do**
    $bits$ = count(bits==1) inBF_i correspond to x
    if $bits > max_bits$ **then**
        $max_bits = bits$
        $closest_node = N_i$
return $closest_node$

We use Bloom filters to provide proof of proximity in a DHT network. By representing the elements stored at each node as a set of bits in a Bloom filter, the proximity of nodes to a target element can be estimated based on the number of common bits in their Bloom filters. The node with the highest number of bits set to 1 in its Bloom filter is considered the closest node to the target element, and the number of common bits can be used as proof of proximity.

Algorithm 2 shows how using Bloom filters to provide proof of proximity in a DHT network, This algorithm takes as input the target element x and the Bloom filters of the nodes in the DHT network. It counts the number of bits set to 1 in each Bloom filter that correspond to the target element, and selects the node with the highest number of bits set to 1 as the node with the closest proximity to the target element. The output is the node with the closest proximity to the target element, which can be used as proof of proximity.

4. Validation of sampling results: The sampling results are validated to confirm their representativeness of the true order of elements in the queue.

We use Kolmogorov-Smirnov(K-S) test to check the result data of sampling is normally distributed. The KS test works by comparing the empirical cumulative distribution function (CDF) of the data with the theoretical CDF of the normal distribution. K-S is described as: Let $F_n(x)$ be the empirical cumulative distribution function (CDF) of a sample of size n, and let $F(x)$ be the theoretical CDF of the normal distribution. The Kolmogorov-Smirnov test statistic is defined as:

$$D = \sup_x |F_n(x) - F(x)| \tag{1}$$

where sup represents the supremum, or the least upper bound. The value of D measures the maximum difference between the empirical and theoretical CDFs, and is used to determine whether the data is normally distributed.

The null hypothesis of the KS test is that the data is normally distributed, and the alternative hypothesis is that the data is not normally distributed. The test statistic D is compared to critical values from the KS distribution to determine whether to reject or fail to reject the null hypothesis. If the test statistic is greater than the critical value, the null hypothesis is rejected, and the data is considered not to be normally distributed.

And the algorithm can be present as **??**

This algorithm calculates the empirical and theoretical CDFs for the sample data and the normal distribution, and calculates the test statistic D as the

Preview

Algorithm 3 Kolmogorov-Smirnov Test

1: **Input:** Sample data $x_1, x_2, \ldots, x_n$
2: **Output:** Test statistic D and decision on normality of data
3: Calculate the empirical cumulative distribution function (CDF) $F_n(x)$ for the sample data
4: Calculate the theoretical cumulative distribution function (CDF) $F(x)$ for the normal distribution
5: Calculate the test statistic D using the formula:

$$D = \sup_x |F_n(x) - F(x)| \tag{2}$$

6: Compare the test statistic D to critical values from the KS distribution
7: **if** $D \leq critical_value$ **then**
8: **return** "Data is normally distributed", D
9: **else**
10: **return** "Data is not normally distributed", D

maximum difference between the empirical and theoretical CDFs. The test statistic is then compared to critical values from the KS distribution to determine whether the data is normally distributed or not. The algorithm returns the test statistic and a decision on the normality of the data.

4 Gaming

Let's consider the scenario where individuals are able to repeatedly calculate their global rank until they are satisfied with the outcome. This raises two questions: 1) Will the repeated sampling by the sampler result in a significant increase in network traffic, and 2) Will the sampled individuals be willing to disclose their accurate local rank.

4.1 Rank maximize

In each sampling, it is assumed that the initial attitude of the nodes is neutral, but as the requests increase, they may become negative. A rank n that follows a normal distribution will be produced for each sampling. However, the maximum value of the normal distribution may decrease in subsequent samplings. This means that if the sampler continues to sample, the expected value of the result may decrease.

In this scenario, if the sampler wants to achieve the highest result possible, it may need to adjust the number and timing of samplings based on a comprehensive understanding of the normal distribution. This requires statistical analysis of the mean and variance of the distribution to evaluate different strategies and choose the most optimal one.

$$\max_x \quad n \tag{3}$$

$$\text{subject to} \quad g_i(x) \leq 0, \quad i = 1, \ldots, m \tag{4}$$

$$h_j(x) = 0, \quad j = 1, \ldots, p \tag{5}$$

In this model, x is a vector of variables that represent the actions taken by A, n is the objective function to be maximized, $g_i(x)$ represents the inequality constraints, and $h_j(x)$ represents the equality constraints. The goal is to find the values of x that maximize n subject to the constraints.

4.2 Self-interest

We use a Guess the Median [4] method to build a game between the sampled nodes, for the sampled objects, the sampled nodes need to have enough motivation to participate, which means rewards, but also means the possibility of cheating. Therefore, we only reward nodes that are close to the Median.

In this game, each player has two strategies: to guess a number that is higher than the median, or to guess a number that is lower than the median. Let's denote the strategy of guessing a number that is higher than the median as H, and the strategy of guessing a number that is lower than the median as L.

The game matrix for Guess the Median would then look like this:

In this matrix, the rows represent the first player's

Preview

	H	L
H	0	1
L	-1	0

Figure 1: Two player median

strategy, and the columns represent the second player's strategy. The entries in the matrix represent the outcome for each strategy combination, with 1 representing a win for the first player, -1 representing a win for the second player.

This game has two Nash equilibria points.

Let x_1 and x_2 be the strategies chosen by Player 1 and Player 2, respectively, where x_i is equal to 0 if the player chooses to guess a number less than 50, and equal to 1 if the player chooses to guess a number greater than 50. The Nash Equilibrium is then given by the solution to the following system of equations:

$$x_1 = \arg \max_{x_1 \in \{0,1\}} \min(x_2, 0.5) \tag{6}$$

$$x_2 = \arg \max_{x_2 \in \{0,1\}} \min(x_1, 0.5) \tag{7}$$

As we can see, in the case of mutual ignorance, the Nash Equilibrium will push the result towards the 50% position of the total, which is a safe neutral result for the players. But in fact, nodes on the network each have their own local ranking, so this is not actually a Nash Equilibrium, but a Correlated Equilibrium. In this scenario, we can safely return to the Byzantine assumption that when 2/3 of the people are honest, we will obtain the correct global rank.

4.3 Reward and Slash Proofs

Of course, limiting the number of samples by maximizing profits is an invisible big hand, and we still need visible hard boundaries to control user behavior. Therefore, we will introduce a penalty mechanism. The ranking obtained by the user each time will be locked for a period in the distributed ledger, and during this period it can be reported by any other node and deducted from the ranking, and the reporting node also uses ranking sampling to perform slash.

Both rewards and punishments are based on Sample proofs, and they should contain signatures generated by the sampled node using cryptographic algorithms. For rewards, it will claim its token through the distributed ledger, while for punishments, it will slash others' tokens, both of which can have a lock-up period.

5 Conclusion

Overall, the Ranking protocol provides a comprehensive and effective solution for evaluating node performance and preventing cheating or denial of service in structured peer-to-peer networks. By using a mutual scoring system among nodes and a reputation system that incentivizes proper behavior, the Ranking protocol aims to provide a secure and efficient peer-to-peer network.

References

[1] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):556–565, 1978.

[2] Edonkey Development Team. Edonkey: A decentralized file sharing system. *International Journal of Distributed Systems*, 5(4):280–290, 2003.

[3] D. Karger M. F. Kaashoek I. Stoica, R. Morris and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. `https://dl.acm.org/doi/10.1145/383059.383071`, 2001.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. In *Communications of the ACM*, volume 13, pages 422–426, 1970.

[5] J. von Neumann and O. Morgenstern. The theory of games and economic behavior. *Princeton University Press*, 1944.

Preview