Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network

Chapter · May 2018

DOI: 10.1007/978-3-319-91458-9_41

4 authors, including:

Fangyu Gai
University of British Columbia - Okanagan
17 PUBLICATIONS **172** CITATIONS
SEE PROFILE

Baosheng Wang
National University of Defense Technology
94 PUBLICATIONS **712** CITATIONS
SEE PROFILE

Wei Peng
National University of Defense Technology
108 PUBLICATIONS **1,614** CITATIONS
SEE PROFILE

Proof of Reputation: A Reputation-based Consensus Protocol for Peer-to-Peer Network

Fangyu Gai, Baosheng Wang, Wenping Deng✉, and Wei Peng

School of Computer, National University of Defense Technology, Changsha, China
wpdeng@nudt.edu.cn

Abstract. The advent of blockchain sheds light on addressing trust issues of peer-to-peer networks by providing a distributed tamper-resistant ledger. Beyond cryptocurrencies, it is believed that blockchain can also be used to protect other properties such as reputation. Most of the existing studies of enhancing reputation systems using blockchains are built on top of the bitcoin-like blockchains, so they are inherently constrained by the low-efficiency and high-consumption of the underlying blockchain. To fill this gap, we present a reputation-based consensus protocol called Proof of Reputation (PoR), which guarantees the reliability and integrity of transaction outcomes in an efficient way. In PoR, we let reputation serves as the incentive for both good behavior and block publication instead of digital coins, therefore no miners are needed. We also implement a prototype and our scalability experiments show that our protocol can scale to over a thousand participants in a peer-to-peer network with throughput of hundreds of transactions per second.

Keywords: blockchain, consensus protocol, reputation system, peer-to-peer network

1 Introduction

Since 2009 Bitcoin was proposed by Nakamoto [1], more than 250 similar altcoins have been proposed. Blockchain, the core technology behind Bitcoin, is considered to benefit not only economic but also politics, healthcare, supply chain and scientific domains [2]. A blockchain is a distributed ledger that cryptographically links a chain of blocks, which records a set of time-ordered transactions. With this emerging technology, a trustless environment can be created for distributed applications. Most blockchain applications running depends on coins as the incentive for miners to produce blocks, such as Bitcoin and Namecoin [3]. In these applications, the security of the blockchain will be affected if miners are not paid [4]. In fact, miners and coins are not necessary, as long as the provision of appropriate incentives to maintain the security of underlying blockchain.

Reputation can be defined as the rating of a member's trustworthiness by others [5]. In peer-to-peer (p2p) networks, reputation systems are applied to drive the ability for each participant to trust one another and facilitate a successful

interaction [6]. The existing studies can be divided into two kinds: the centralized and the distributed, both of which have obvious drawbacks. While centralized reputation systems (i.e. eBay online auction site [7]) can grasp the overall reputation of each participant in the context of their own use case, the single-point-failure problem is inevitable. On the other hand, in distributed methods, participants only catch partial reputation evidence. To assess the trustworthiness, they need both direct experience and indirect opinion of its peers, which is inefficient. Worsely, the effective communication, and sharing of unmodified reputation evidence remain unsolved [8].

Actually, reputation and blockchain can be a good combination: reputation serves as the incentive and the blockchain keeps the reputation records safe in turn. To achieve this purpose, we propose a reputation-based consensus protocol called *Proof of Reputation* (PoR), which provides a distributed ledger of reputation. We hope to build a decentralized protocol that records a history of transaction outcomes without central third parties involved. The protocol is used in the permissioned blockchain, where an access control layer is built into the blockchain nodes. Participants can authenticate with each other using asymmetric cryptography [9]. The reason why we choose permissioned blockchains is that reputation is inherently tied to identity and needs time to accumulate. In PoR, each participant maintains a distributed ledger of reputation evidence, which achieves consistent with the consensus algorithm. Compared with the *Proof of Work* (PoW), the consensus algorithm of Bitcoin, our protocol uses the reputation as the incentive, which is cost-efficient since there is no miners or hash power being consumed during block competition. The cryptographical nature of blockchain can protect the integrity and reliability of the reputation evidence. Theoretically, any p2p applications and decentralized reputation systems (e.g. second-hand dealing, social networking, service sharing etc.) can utilize our protocol as a reputation layer to objectively and securely record transactional history, based on which each participant's reputation can be evaluated without being manipulated by third parties.

The rest of this paper is organized as follows: In Section 2, we review the earlier work related to our work. We define the problem and threat model in Section 3. The rationale for its design and details are presented in Section 4, and the implementation of the protocol and experiment results are discussed in Section 5. Finally, Section 6 concludes the paper with a discussion of the future work.

2 Related Works

The reputation systems in p2p networks have been studied in the literature for decades. One of the first reputation systems of p2p networks is proposed by Gupta et al [5]. It is also the most complete and effective solution, although the drawbacks are evident. Other studies also made great contributions to reputation systems [10] [11] [12], but none of them are truly decentralized. In recent few

years, integrating reputation with blockchain sheds light on this area and is under active research.

Carboni [13] describes how a decentralized feedback management system can be built on top of the Bitcoin blockchain. The author explains that online reputation has the same requirements of electronic money: (1) can be expressed as a numerical variable; (2) its value is agreed by every participant and cannot be manipulated by third parties. The proposed system records the interaction feedback during payment, leaving the computation of the actual reputation to third-party applications. Dennis et al. [14] [15] propose a novel reputation system based on the blockchain, which aims to solve the majority of issues remaining in current reputation systems. They built an entirely new blockchain to store reputation data and utilized merge mining from Bitcoin network to prevent 51% attack. However, their blockchain needs miners to produce and verify the blocks, and a possible drawback is requiring the users to be online for the miners to verify the transaction. Buechler et al. [16] propose a reputation algorithm named net flow convergence and a decentralized system that records reputation evidence. The algorithm is to detect fraudulent behavior by looking at the network flow of the Bitcoin. The system, which is implemented in three layers, analyzes the structure of the underlying transaction network and builds a history of transaction outcomes by utilizing smartcontract. Schaub et al. [17] present a trustless, decentralized, and anonymity preserving reputation system based on blockchain for e-commerce applications. The system utilizes Proof of Stake blockchain to keep the reputation system consensus, and meanwhile, it allows customers to submit ratings as well as textual reviews.

All of these studies use blockchain as a decentralized database storing reputation evidence. Meanwhile, these systems rely on blockchain mining to motivate miners to publish blocks, during which a great amount of computing power is consumed. Furthermore, they are subject to the underlying blockchain, and therefore they are facing the same challenges as the Bitcoin blockchain, such as limits on data storage, slow writes, limited bandwidth, and endless ledger [17].

A different study called *Trustchain* is proposed by Otte et al. [18], which is to build a permission-less tamper-proof and Sybil-resistant blockchain for storing transaction records of participants. Compared to traditional blockchains, each block in Trustchain contains only a single transaction record and Trustchain blocks together form a directed acyclic graph (DAG). They also propose the NetFlow accounting mechanism to prevent Sybil attack. Similar to our work, there are no miners in Trustchain, where two agents in a transaction produce their block and store it locally. Nevertheless, participants in Trustchain have less motivation to produce blocks and they do not share a global picture of reputation in the network.

3 Problem & Threat Model

3.1 Problem Definition

The aim of this study is to solve the problem of reputation agreement in p2p networks by providing a distributed ledger of reputation. Unlike Bitcoin's consensus protocol, known as PoW, there is no miner nor bitcoin in PoR. Therefore, nodes do not have to compete for the right to write the block. In our protocol, the node who writes the block is the one who has the highest trust value in the pre-committed block. Reputation serves as the incentive, because in order to increase its overall trustworthiness, the participant urges to write the block into the blockchain when it has the highest trust value in this block.

Noticeably, during the consensus process, there are no complex mathematical problems to be solved, which means our protocol is cost-efficient. Furthermore, we do not have to worry about the double-spending problem, because reputation is an overall status of a node after a number of transactions, which can not be spent or transferred. We make three assumptions about the underlying network and nodes involved in the network:

1. **Enrolment Control:** Our consensus protocol requires an access control layer built into the blockchain nodes, which is known as "permissioned blockchain". Candidates need to go through an enrolment process before joining the network. Each candidate has a pair of cryptographical key (like Bitcoin) used for authentication and digital signature. The public key will be submitted to the registry as the participant's identity, the hash of which will be considered as its ID, and then the registry will broadcast this key so that other participants can authenticate.
2. **Secure Channel:** For simplicity, we assume a secure broadcast channel avoiding man-in-the-middle attack (MITM) [19]. It means that no third parties can intercept or modify messages. Each pair of participants can authenticate each other reliably.
3. **Quick Bootstrap:** In Bitcoin, a new node needs to take more than 3 days to download the whole blockchain from its adjacent peers, verify it for bootstrap. Contrarily, a new PoR node can boot up instantly by requesting a trust ranking list from the registry. Details will be discussed in Section 4.6.

On the other hand, we do not make any assumption that nodes are reliable during protocol runs. That means the impact of Byzantine nodes is not within the scope of our discussion.

We formalize the problem of designing a reputation consensus protocol for permissioned blockchains as follows. Assume N participants have registered themselves to join the network, and an individual participant is presented by $p_i, i \in N$. Each participant stores others' public keys locally so that it can authenticate identities and verify transactions. Each transaction generated by p_i toward p_j is represented by an real number $x_i^j \in \mathbb{R}$ signed by p_i's private key, denoted as $Sig(x_i^j)$. The transaction from the same pair can only be included once in one block, which means each block contains $N(N-1)$ transactions at

most, and a threshold $\lambda \leq N(N-1)$ can be adjusted to limit the number of transactions in one block.

3.2 Threat Model

Although we assume that nodes communicate through a reliable authenticated point-to-point channel, the network can still be damaged by selfish behaviors, malicious attacks, and even unintentional misconfiguration. Especially, the reputation-based protocol itself can be an attractive target for attackers. Here we discuss several Potential attacks [20].

1. ***Bad-mouthing attack*** is to provide dishonest recommendations to deframe good nodes, which is the most straightforward attack [21]. In our protocol, malicious nodes can continuously give bad comments to a specific node or all other nodes to improve his trustworthiness ranking in a block in return.
2. ***Replay attack*** attempts to reuse transactions and replay them in order to increase the impact of the same transaction. By carrying out this attack, a malicious participant can claim he has been involved in a transaction that is profitable for him multiple times. It can also be used to undermine hostile participants.
3. ***On-off attack*** is referred to irregular behaviors of attackers, which means that malicious nodes can perform well or badly alternatively in order to remain undetected while causing damage.
4. ***Sybil attack and newcomer attack***, which was first described by Douceur [22], is harmful to almost all p2p networks. It can be described that attackers "legally" create more than a single ID. If one ID gets low reputation by performing bad behaviors, it switches to a new ID and starts over.

Further, since every participant can publish arbitrary blocks, malicious nodes can rewrite all the transactions inside or publish a fabricated block. Nonetheless, provided proper security mechanisms, high attack cost, and low reward, there is little incentive for attackers to attack a reputation system [23].

4 the Proof of Reputation Protocol

In this section, we present the Proof of Reputation (PoR) protocol, which serves to provide global reputation agreement in a p2p environment.

4.1 Design Overview

The core idea of PoR is to ensure that every one of participants holds an agreed ledger, recording the rate of each transaction. To achieve this purpose, three questions must be answered:

1. How to keep the ledgers consensus? The answer to this question is similar to that of Bitcoin. Bitcoin applies the PoW protocol, where the first processor that has solved the mathematical problem gains the right to publish

the block, and others can openly verify the block. Likewise, in the PoR protocol, the one who has the highest trust value in a group of transactions can package them into a block and publish it. The verification of the block is also open to other participants.

2. How to motivate participants to publish blocks? Compared with Bitcoin, there is no block reward or transaction fee as incentives in the PoR. We use trustworthiness instead. However, trustworthiness, which is entirely different with coins, can not be spent or transferred. It is a kind of reputation which indicates that participants with high trust value can provide better services. In each round, only the one that has the highest trust value can publish the block. Publishing the block can undoubtedly increase the overall trust rank of the publisher.

3. How to prevent transactions or blocks from being modified? The Bitcoin network is entirely open to every machine that has a client installed on it. Modifying the transactions consumes huge hash power, which is known as *51% attack* [24]. On the contrast, the PoR protocol is applied in the permissioned blockchain, where all participants are registered with their public keys with private keys stored locally, and every participant has a copy of all public keys. Therefore, if some malicious nodes attempt to fabricate identities, it will be detected instantly. Also, the Merkle Tree [25] is used in forming the transactions in the block, which can protect the integrity of transactions.

In each round, the PoR protocol runs the following 3 steps:

1. ***Broadcasting Transactions.*** At the end of each interaction, a piece of feedback will be generated by the service requestor, recording the rate of the service. Then the requestor will broadcast this message together with its signature. Other nodes that have received these messages will verify and store them in the memory.
2. ***Building Block.*** When the number of transactions reaches the threshold, the node stops receiving transactions and starts to calculate a ranking list according to each service providers' scores recorded in this set of transactions. If the top of the list is the node itself, it constructs a block and publishes it after signing with its private key.
3. ***Verifying Block.*** Nodes receiving a block will recalculate the ranking list to check if the sender has the highest trust value. They also need to verify the signature of each transaction using the signer's public key. After verification, the nodes will append the block into the blockchain and prepare for the next round.

4.2 Broadcasting Transactions

We define the participant that provides a certain service during an interaction as the *provider*, while another participant that rates the service as the *rater*. During the interaction, the *provider* sends the requested service, such as a bunch of data, signed by its private key, along with the hash of the data and a timestamp. After verifying the integrity of the data by checking the recalculating the hash, the

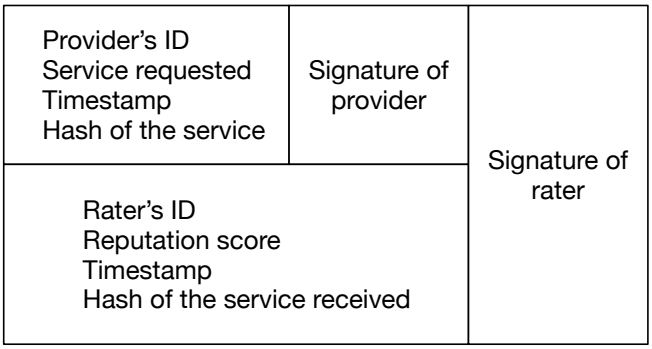

Fig. 1. A diagram of the format of a *transaction*.

rater produces a *transaction* consisting of the *reputation score*, a timestamp, the hash of the received data, and its digital signature. Next, the *rater* broadcasts the *transaction* to the network. A diagram of the format of a *transaction* is illustrated in Fig. 1. the *reputation score*, represented by x_i^j, can be expressed either in a binary way($x_i^j \in \{0, 1\}$, i.e., p_i rates 1 if it is satisfied with the service and 0 otherwise), or a real number in a cointinuous range($x_i^j \in [0, 1]$) to represent the degree of satisfaction.

An example of this step is shown in Fig. 2, where p_1 is the *rater* and p_5 is the provider. At the end of the interaction, p_1 is satisfied with the service and rates it with 1. Then the *transaction*, which is represented as $Sig(p_1 \rightarrow p_5 = 1)$, of this interaction is generated, and sent to the other participants by p_1.

4.3 Transactions Filter

Malicious nodes can spread forged *transactions* and *transactions* with dishonest rating. To prevent this, we need filter *transactions* before packaging them up.

Our algorithm to filter *transactions* is depicted in Algorithm 1. More specifically, when a *transactions* comes, its signature will be tested whether it comes from an authenticated participant. Next step is to check if the same pair of *rater* and *provider* have appeared in the current block. If so, the *transaction* will be replaced by the later one. For example, assume a *transaction* $Sig(x_5^1)$ has already been verified, then if another $Sig(x_5^1)$ comes, it will replace the former one. The purpose of keeping only one *transaction* of the same pair in one block is to reduce the impact of *bad-mouthing attack*. Finally, when the number of filtered *transaction* reaches the threshold λ, this set of *transactions* would be packaged up to conduct an alternative block, while other *transactions* will remain in the memory pool for the next round.

4.4 Block Publication

With a set of verified *transactions*, there would be two steps to construct an alternative block.

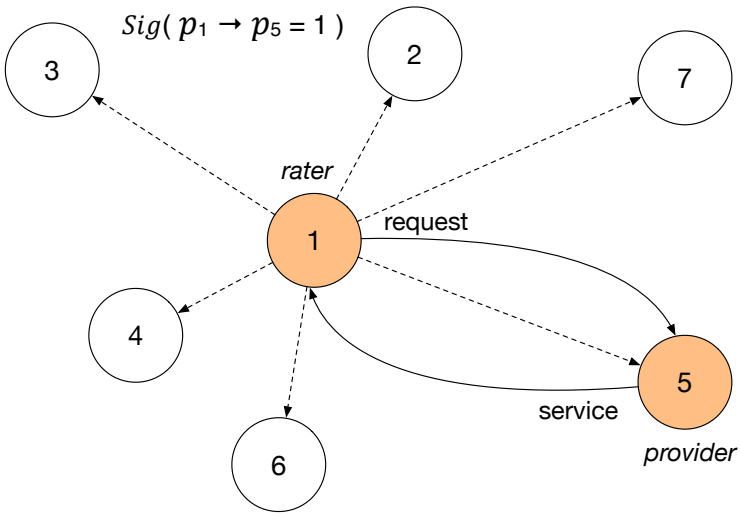

Fig. 2. Broadcasting transactions step of p_1 rating the service of p_5.

Algorithm 1: Transaction Filter Algorithm

Input: A set of *tansactions*, represented as $T = \{t_1, t_2, t_3, ..., t_m\}$ where m is the number of the set.
Output: Filtered *transactions*

```
FilteredNumber = 0;
FilteredTransactions = {};
for t_i in T do
    if VerifySig(t_i) then
        if the rater and the provider of t_i
        have appeared in FilteredTransactions then
            update(FilteredTransactions);
        else
            if FilteredNumber < λ then
                FilteredTransactions.append(t_i);
                FilteredNumber+=1;
            else
                break;
        else
            drop(t_i);
```

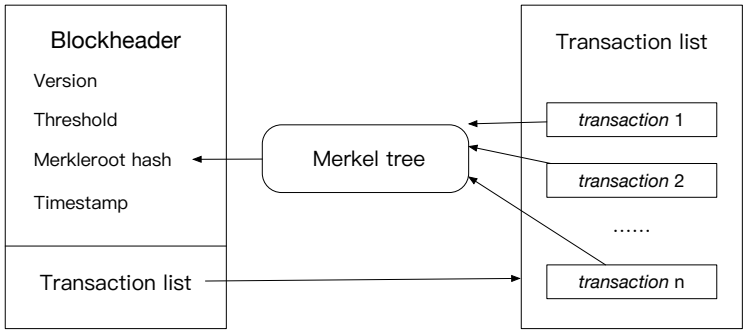

Fig. 3. Data structure of the block.

1. Trustworthiness Evaluation. From the set of verified *transactions*, a participant can extract each *provider*'s action log, which can be represented as $l_j = s_1, s_2, s_3, ..., s_n$ for the *provider* p_j, where s_i is the ith *score* of l_j and n is the number of the *scores*. We can use a certain trustworthiness evaluation model to squash the log into a trust value. For simplicity, we use the sigmoid function to evaluate the trustworthiness in our protocol (Other trust evaluation algorithms can also be used, i.e., EigenTrust [12]). Assume that $s_i \in \{1, 0, -1\}$ for *good*, *general*, *bad* actions respectively. Then the trustworthiness of p_j can be calculated as follows:

$$trust^j = \frac{1}{1 + e^{-\phi \sum_{i=1}^{n} s_i}} \tag{1}$$

Accordingly, ϕ is simply the adjustable parameter. With this measure, each *provider* will be assigned a trust value, and a *ranking list* is generated. Then the participant needs to check if itself is the *provider* and it has the highest trust value. If so, it moves on to the second step to publish the block, otherwise, it has to give up this round and prepare the next round. The reason why only the *provider* with the highest trust value can publish the block in each round is that this mechanism can motivate participants that have the highest trust value in each round to publish the block since it will increase their trust ranks.

2. Block Construction. The way we consider to construct a block is comparable to Bitcoin, but with different data structure. Details are shown in Fig. 3. Merkle tree is also introduced when constructing the block to facilitate transaction searching. The threshold is to control the number of *transactions* contained in a single block. In addition, the participants who publish this block need to digitally sign the block and add its signature into the blockheader for authentication.

After these two steps above, the block will be committed and spread to the network.

4.5 Block Verification

Every block received by participants need to be verified before appended to the blockchain. The verification process is presented in Algorithm 2. First, the signature of the block should be from an authenticated participant, so be the signatures of each *transactions* in the block. Next, a trustworthiness ranking list will be recalculated based on the checked transactions. Finally, it will be checked that whether the provider on top of the list is same as the sender of the block.

Algorithm 2: Block Verification Algorithm

```
Input: currentBlock
Output: block validity
rankingList = {};
if !verify(currentBlock.signature) then
    return false;
if !verify(currentBlock.transactionList.signatures) then
    return false;
rankingList = calculateRankingList(currentBlock.transactionList);
if (currentBlock.signature is not matched to the public key of rankingList.top)
then
    return false;
return true;
```

After verification, the block it will be cryptographically linked to the blockchain and wrote to the disk. If any *transactions* in the memory pool also exist in the newly written block, they would be removed.

4.6 Cost Analysis & Quick Bootstrap

The space cost of a single block is closely related to the number of *transactions* controlled by λ, which can also be utilized to control the average time a block is generated according to the number of participants in the network. A block with no *transactions* would be about 80 bytes, which is close to that of Bitcoin. A single *transaction* costs about 100 bytes. If we suppose $\lambda = 100$, a single block costs no more than 10KB. Since blockchain is an append-only data structure, the space cost grows linearly with time. However, because the number of participants is limited in permissioned blockchains, the space cost is far less than Bitcoin blockchain.

To boost setup, new participants do not have to download the whole blockchain. Instead, they can simply get a ranking list from the registry. This ranking list is similar to that described in 4.4 except that it is calculated from the whole blockchain. The ranking list is signed by the registry and expressed in JSON format. An example of it is shown below.

```
{
  "number":128
  "rankingList": {
    "rankingitem": [
      {"id": "62923F...CF0629", "value": "0.84"},
      {"id": "730F75...82848F", "value": "0.74"},
      ...
      {"id": "27B2C1...B93255", "value": "0.92"}
    ]
  }
  "rankingListHash":"7520149...60265B"
  "rankingListHashSignature":{
    "r":"F0AB42...1D6875"
    "s":"4C556E...31C42B"
  }
  "time":1496374331
}}
```

4.7 Security Analysis

After the details of the protocol design, we provide security analysis for how PoR prevents potential threats and works securely. Since we have already assumed that our protocol is deployed in permissioned blockchains where participants can interact in a secure channel and can authenticate with each other, but the trust-based protocol itself is easily targeted by potential attackers.

Malicious participants may perform *bad-mouthing attack* by rating the *provider* dishonestly. In PoR, the *transaction* of the same pair of a *rater* and a *provider* can only appear once in one block. Therefore, even though the *rater* is malicious, its dishonest recommendation has limited impact on the *provider*. Moreover, it can effectively resist *Replay attack*. In *on-off attack*, attackers have irregular behaviors. In our protocol, the distributed ledger of reputation can provide a trust ranking list, which is calculated from all the *transactions* recorded in the blockchain. This ranking list indicates the overall trustworthiness of each participant, so the attackers performing *on-off attack* will be reported by the *rater* resulting in a lower rank. If a participant has a low rank, they may want to switch to another ID and start over, which is known as *Sybil attack* or *Newcomer attack*. The access control layer built on the blockchain can increase the cost of creating a new ID to prevent this kind of attack.

5 Experiments & Evaluation

In this section, we carry out several experiments and performance measurements based on a prototype implementation of the PoR protocol. The goal of our evaluation is to quantify the overhead and scalability of our protocol when deploying it in a p2p network.

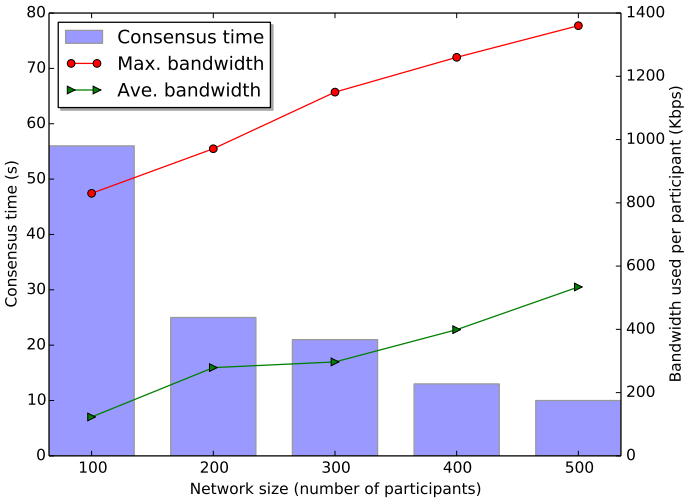

Fig. 4. Consensus time and bandwidth of PoR with different network sizes.

5.1 Experiment Setting

We developed a prototype implementation of PoR in the Python 2.7 programming language. Then we virtualized participants by running the protocol on Docker [26], which is an open source software technology providing containers. We also used Rancher [27], which is a container management platform, to manage our containers so that we can easily create p2p networks of different sizes and measure the performance of our protocol. We vary the number of participants in the network from 100 to 500, using 3 servers with load balancing strategy. Each server has an E5-2640 CPU with 16 cores and 64GB of memory.

5.2 Performance Evaluation

The ability to log *transactions* in a light-weight, scalable and efficient manner is key to a transactional p2p network. In our experiments, we measure the bandwidth consumptions per node of PoR, consensus time, production time, as well as the throughput in different settings.

Scalability of PoR. We start with a network of 100 participants, then we adjust the network size 4 times, from 100 participants to 500 participants. Meanwhile, we set the number of transactions in each block fixed to 1000. We quantify the average time to reach consensus, maximum and average bandwidth consumed at each participant. The experimental results are shown in Fig. 4.

The maximum bandwidth refers to the bandwidth used when the participant receiving or delivering a published block, while the average bandwidth is the bandwidth consumed when processing transactions. The results show that

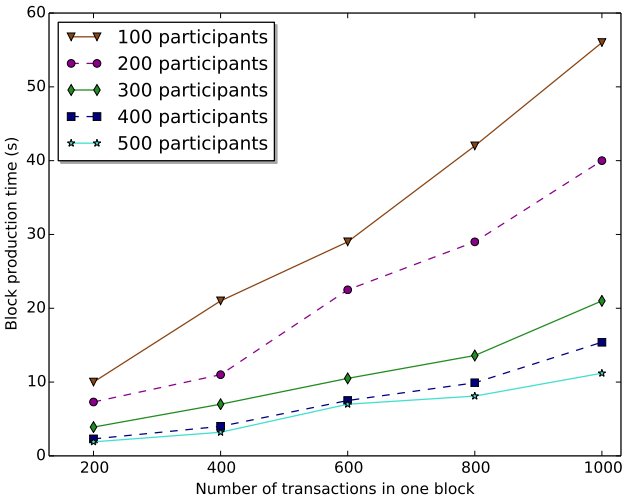

Fig. 5. Average time to produce a block with different block sizes.

both the maximum and average bandwidth increase linearly (from 820Kbps to 1380Kbps and 160Kbps to 470Kbps, respectively) as we enlarge the network size (100 to 500 accordingly). This is because more participants joining the network, more messages being transferred in the network. In contrast, the consensus time is shorter (e.g., 56s in 100 participants to less than 10s in 500 participants) since more transactions are generated, each participant requires less time to publish the block. These results show the average time of PoR to reach consensus is less than a minute, which is much shorter than that of Bitcoin blockchain (usually 1 day). In addition, the average bandwidth is less than 1Mbps, which is negligible in most p2p networks.

Production time. We next vary the number of transactions in a single block from 200 to 1000 to measure the average time it takes to produce a block with different network sizes. In our experiment, we measured the production time in different network sizes ranging from 100 to 500 participants.

Fig. 5 depicts that longer time is consumed if the block size is larger since a participant needs to receive more transactions to package them up. Meanwhile, we observe that smaller network has longer block production time and it grows faster as the block size goes up. This is because smaller network generates fewer transactions per unit time. When the block size is set to 1000, the production time of the network with 100 participants is less than a minute, and in the network with 500 participants, the production time is less than 10 seconds. This indicates that the PoR protocol is far more efficient than bitcoin-like ones, where it takes about 10 minutes to produce a block. In addition, the results show that

we can adjust the time a single block consumed by changing the block size according to the requirement of upper applications.

Throughput. We have the same definition of throughput in [28], where it is defined as the number of transactions committed per unit of time. In this experiment, we also vary the block size from 200 to 1000 transactions and the network size from 100 to 500 participants. The results are plotted in Fig. 6.

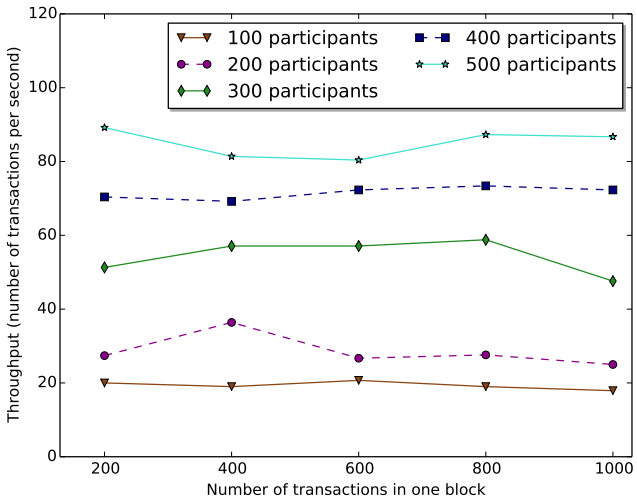

Fig. 6. Throughput with different block sizes.

From Fig. 6 we can see that the throughput has an increment when more participants joining the network, the reason of which is obvious. Meanwhile, the throughput fluctuates slightly around a certain value (i.e., 20, 30, 50, 70, 85 transactions per second respectively in different network sizes) as the block size increases. The most likely explanation for this is that although a participant can commit more transactions in larger-size blocks, it takes more time to produce one. These results indicate that the throughput of PoR hinges on the number of participants involved in the network. Even in a thousand-level network, it is possible to process hundreds of transactions in one second, which is far more efficient than the throughput of bitcoin-like protocols.

In summary, the experiments confirm the expected scalability of PoR's transaction throughput and bandwidth usage. The performance of our protocol largely depends on the network size and the block size, which gives it flexibility to adjust various p2p environment. In addition, the proposed protocol can also be deployed on IoT devices since low bandwidth and low computation resources are consumed.

6 Conclusion and Future Work

We present PoR, a reputation consensus protocol for p2p networks, which is derived from the idea of blockchain technology. The contribution of our study is to provide a distributed ledger of reputation so that each participant can share a global view of reputation. We have implemented a prototype of our protocol and experimental results demonstrate that PoR can be a suitable component to for transactional applications to make reputation based decisions. More generally, our study has shown the potential of reputation to serve as the incentive in a consensus protocol.

As future directions, we would extend our work to build a reputation based system which contains access control, identity management, and other security strategies. Additionally, we would test our system in a broader area, such as participants being deployed in different continents, for performance evaluation.

References

1. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
2. Melanie Swan. *Blockchain: Blueprint for a new economy.* Ö'Reilly Media, Inc.", 2015.
3. Harry A Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, and Arvind Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. In *WEIS*, 2015.
4. Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 154–167. ACM, 2016.
5. Minaxi Gupta, Paul Judge, and Mostafa Ammar. A reputation system for peer-to-peer networks. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 144–152. ACM, 2003.
6. Ali Aydin Selcuk, Ersin Uzun, and Mark Resat Pariente. A reputation-based trust management system for p2p networks. In *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*, pages 251–258. IEEE, 2004.
7. Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. In *The Economics of the Internet and E-commerce*, pages 127–157. Emerald Group Publishing Limited, 2002.
8. Prashant Dewan and Partha Dasgupta. Securing reputation data in peer-to-peer networks. In *Proceedings of International Conference on Parallel and Distributed Computing and Systems PDCS*, 2004.
9. David Shrier, Weige Wu, and Alex Pentland. Blockchain & infrastructure (identity, data security). Technical report, Retrieved 27-11-16, from http://cdn. resources. getsmarter. ac/wp-content/uploads/2016/06/MIT_Blockain_Whitepaper_PartThree. pdf, 2016.
10. Yao Wang and Julita Vassileva. Trust and reputation model in peer-to-peer networks. In *Peer-to-Peer Computing, 2003.(P2P 2003). Proceedings. Third International Conference on*, pages 150–157. IEEE, 2003.

11. Stefan Schiffner, Sebastian Clauß, and Sandra Steinbrecher. Privacy, liveliness and fairness for reputation. *SOFSEM 2011: Theory and Practice of Computer Science*, pages 506–519, 2011.
12. Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.
13. Davide Carboni. Feedback based reputation on top of the bitcoin blockchain. *arXiv preprint arXiv:1502.01504*, 2015.
14. Richard Dennis and Gareth Owen. Rep on the block: A next generation reputation system based on the blockchain. In *Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for*, pages 131–138. IEEE, 2015.
15. Richard Dennis and Gareth Owenson. Rep on the roll: a peer to peer reputation system based on a rolling blockchain. *Int J Digital Society (IJDS)*, 7(1):1123–1134, 2016.
16. Matthew Buechler, Manosai Eerabathini, Christopher Hockenbrocht, and Defu Wan. Decentralized reputation system for transaction networks. Technical report, Technical report, University of Pennsylvania, 2015.
17. Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie. A trustless privacy-preserving reputation system. In *IFIP International Information Security and Privacy Conference*, pages 398–411. Springer, 2016.
18. Pim Otte, Martijn de Vos, and Johan Pouwelse. Trustchain: A sybil-resistant scalable blockchain. *Future Generation Computer Systems*, 2017.
19. Franco Callegati, Walter Cerroni, and Marco Ramilli. Man-in-the-middle attack to the HTTPS protocol. *IEEE Security & Privacy*, 7(1):78–81, 2009.
20. Yan Sun, Zhu Han, and K. J. Ray Liu. Defense of trust management vulnerabilities in distributed networks. *IEEE Communications Magazine*, 46(2):112–119, 2008.
21. Chrysanthos Dellarocas. Mechanisms for coping with unfair ratings and discriminatory behavior in online reputation reporting systems. In *Proceedings of the Twenty-First International Conference on Information Systems, ICIS 2000, Brisbane, Australia, December 10-13, 2000*, pages 520–525, 2000.
22. John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.
23. Audun Jøsang and Jennifer Golbeck. Challenges for robust trust and reputation systems. In *Proceedings of the 5th International Workshop on Security and Trust Management (SMT 2009), Saint Malo, France*, page 52, 2009.
24. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, pages 436–454, 2014.
25. Michael Szydlo. Merkle tree traversal in log space and time. In *Eurocrypt*, volume 3027, pages 541–554. Springer, 2004.
26. Docker. `https://www.docker.com`.
27. Rancher. `http://rancher.com`.
28. Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42. ACM, 2016.