Beh-Raft-Chain: A Behavior-Based Fast Blockchain Protocol for Complex Networks

Li-e Wang, Yan Bai, Quan Jiang, Victor C. M. Leung, *Fellow, IEEE*, Wei Cai, *Member, IEEE*, and Xianxian Li

***Abstract*—By facilitating multiple independent owners to jointly control a distributed network, blockchain can be used to solve the problem of device collaboration in complex networks (e.g., 5G, health care industries) through a distributed consensus mechanism. However, the state-of-the-art blockchain-based solutions cannot meet the demand of high transaction rate for those applications, due to the unavoidable data synchronization cost in decentralized systems. To address this issue, recent research splits blockchain nodes into multiple groups as parallel shardings to improve scalability at the cost of increased communication and storage per node. This paper proposes a fast and secure distributed blockchain protocol to reduce the traffic complexity while enhancing the transaction rates and the capability of fault-toleration. We introduce Proof-of-Behavior (PoB), a behavior-based incentive mechanism, for stimulating honest behavior and neutralizing malicious attacks. We design a blockchain protocol by integrating PoB with Raft, another classic consensus protocol with supervision, called Beh-Raft-Chain. Our approach replaces Practical Byzantine Fault Tolerance (PBFT) with Behavior-based Raft to lower the traffic complexity to O(n) and boost the capability of fault-toleration from n/4 to n/3, where n is the scale of blockchain. In our solution, we weigh all nodes based on their money and behaviors, and then set an adjustment parameter to increase the probability of candidate nodes being chosen beyond only a few nodes with the highest weight, in order to incentivize honest behavior in our mechanism. Our comparative experiments confirm Beh-Raft-Chain's theoretical low complexity and high fault-toleration properties.**

***Index Terms*—Blockchain, Scalability, Trust, Network Security, Performance Optimization, Fault Tolerance, Utility, Complex Network**

I. Introduction

BLOCKCHAIN technology has attracted huge investments from some of the world's leading enterprises because of its security and immutability. For example, health care industries have applied blockchain technology to build electronic health chain (EHC) and medical supply chains (MSC) to revolutionize the interoperability, security, and accountability of electronic health records and health information technology [1]. The 5th-generation (5G) network, which involves massive complex entities and complex network environment, use blockchain technology to verify the integrity protection of mutual information among the complex entities and guarantee the nonrepudiation of interactive behavior [21]. Industry projects have been building blockchain-based IoT platforms in domains such as supply chain, manufacturing automation, and energy grid [22]–[24].

The blockchain system is designed for operating in a decentralized setting and independent of a centralized server or a trusted third party, which is different from traditional distributed databases. A decentralized consensus system, also known as the consensus model, is necessary for all participants to synchronize their data. Most studies [2]–[5] implemented blockchain solutions that use Proof-of-Work (PoW) [6] consensus to probabilistically elect the leader, which employs billions of CPUs to search for hash values [7] and has limitations on the maximum transaction rate. On the other hand, another set of studies [8], [9] adopted permissioned blockchains with Practical Byzantine-Fault Tolerance (PBFT) [11] consensus, which requires every two nodes communicate with each other. The traffic complexity of PBFT is at least O (n^2) and the capability of fault-toleration is up to n/4, where n is the scale of participants. The processing speed is very important to achieve meaningful interoperability in those applications, such as health care industries, 5G network and IIoT. Thus, the limitations of performance and scalability are open challenges with the exponential rise in the number of devices.

Aiming at the performance and scalability limitations, Luu. *et.al.* [10] proposed ELASTICO, a secure sharding protocol, which is scalable at the price of lower fault toleration. The key

Manuscript received October 1, 2019; revised January 25, 2020 and February 25, 2020; accepted March 25, 2020. Date of publication April 2, 2020; date of current version July 7, 2021. This work is supported in part by the National Natural Science Foundation of China under Grants 61662008, 61672176, 61502111, 61941201, and 61902333, in part by the Guangxi "Bagui Scholar" Teams for Innovation and Research Project, in part by the Guangxi Collaborative Innovation Center of Multi-source Information Integration and Intelligent Processing, in part by the Guangxi Talent Highland Project of Big Data Intelligence and Application, in part by the Research Fund of Guangxi Key Lab of Multi-source Information Mining & Security (No. 19-A-02-02), in part by the Guangxi Natural Science Foundation (2018JJA170082), in part by the Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), and in part by the National Science Foundation (NSF) under Grant 1921576. Recommended for acceptance by Dr. Yulei Wu. (*Corresponding author: Xianxian Li.*)

Li-e Wang, Quan Jiang, and Xianxian Li are with the Guangxi Key Lab of Multi-source Information Mining & Security, The College of Computer Science and Information Engineering, Guangxi Normal University, Guilin 541004, China (e-mail: wanglie@gxnu.edu.cn; 752074774@qq.com; lixx@gxnu.edu.cn).

Yan Bai is with the School of Engineering and Technology, University of Washington Tacoma, Tacoma, WA 98402 USA (e-mail: yanb@uw.edu).

Victor C. M. Leung is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Department of Electrical and Computer Engineering, the University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: vleung@ieee.org).

Wei Cai is with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Guangdong 518172, China and also with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen, China (e-mail: caiwei@cuhk.edu.cn).

Digital Object Identifier 10.1109/TNSE.2020.2984490

idea is to construct a two-layer consensus. In the first layer, ELASTICO divides the blockchain into shardings with fixed size *c* to make the protocol scalable. The reason is that the size of the sharding does not grow with the size of the blockchain and the number of shardings increases with the size of the blockchain. In the second layer, ELASTICO applies PBFT with the presence of byzantine adversaries for local consensus [11] in each sharding to tolerate byzantine adversaries of up to one-fourth of the total computational power. However, ELASTICO is not efficient in its network messages which are at least quadratic even cubical to the size of sharding as partial and almost quadratic to the size of blockchain as a whole. Furthermore, its capability of fault-toleration is only proportional to one quarter of its nodes. However, the fault tolerance effect of this protocol is not ideal; some of the original issues, such as scalability, are still present. To improve the performance of fault tolerance and optimize the number of network messages under the ELASTICO protocol, this paper designs a new protocol called Beh-Raft-Chain which uses the incentive mechanism based on behavior for weighing users and applies the consensus protocol Raft [12] with supervision to lower network messages and increase fault-toleration. Our goal is to seek a protocol for the open, permissionless network wherein participating processors have no pre-established identities and their consensus interaction messages are linear with high fault-toleration. The solution which uses classic Raft consensus protocols [12] cannot work in an open environment like blockchain because of two fundamental challenges. First, the Raft consensus protocol is suitable for networks with only honest nodes, which do not exist in open environments like blockchain. Second, the Raft consensus protocol, which is designed for a closed cluster, reaches consensus when any majority of their participants are accepted. Thus, the scale of participants in the protocol cannot grow sufficiently to ensure the necessary level of transaction throughputs of a network. For the first problem, we seek compensation by designing a behavior-based Raft consensus protocol in blockchain with the presence of byzantine adversaries or malicious nodes by weighing users based on their behaviors, called Beh-Raft for short. The key idea in the approach is to distinguish honest nodes from malicious nodes according to their weights which are driven by the fundamental forces of rewards and punishments for normal or malicious behaviors. Thus, we can minimize the impact of malicious nodes by electing based on the weight in the Raft protocol. We also designed a supervisory mechanism to guarantee the security of consensus. For the second problem, inspired by ELASTICO proposed in [10], we apply RAFT to the sharding-based consensus protocol which has a fixed and small number of members for guaranteeing the scalability of our consensus protocol. We also modify identity setup and committee formation based on weight so as to keep the protocol secure. To our knowledge, we provide the first behavior-based Raft protocol with supervision integrating sortition for open blockchains which linearly gain throughput and high security while tolerating byzantine nodes in numbers up to one third of the total number of network nodes.

We claim the following contributions**: 1) Behavior-based Incentive Mechanism**: to prevent Sybil attacks, we design Proof-of-Behavior (PoB), a behavior-based incentive mechanism, by electing honest nodes as a leaders/miners; **2) Sublinear Communication**: we design a behavior-based Raft consensus protocol instead of the PBFT consensus protocol, which allows others to verify and accept the identity of a processor excepting PoW solutions in Bitcoin; **3) Intra Consensus with Supervisory Mechanism:** we achieve consensus by applying sharding -based protocol to ensure scalability and the probability of consensus failure will be guaranteed by the supervisory mechanism; **4) Secure Partition:** to prevent an adversary from targeting committee members, we built on the Cuckoo rule [18], [19] and improve it to provably protect against targeted attacks; **5) Decentralized Bootstrapping**: Beh-Raft-Chain is designed for public chains allowing open membership; **6) Weight-based Sortition:** To guarantee the sustainability of Beh-Raft-Chain, we design a weight-based sortition algorithm to elect the leadership. To the best of our knowledge, Beh-Raft-Chain is the first blockchain protocol for integrating Raft protocol with supervisory and PoB mechanism.

II. Problem Statement

Storage and consensus are two challenges to the scalability of the public chain introduced by the total transaction volume and the number of independent participants. This paper focuses on solving these two issues.

A. Problem Definition

We assume these are some external nodes sent the set of transactions to our protocol which include inputs and outputs. We can verify their validity by checking their issuer's signature. We partition the set of transactions into *k* non-intersect blocks. Let $x_{i,j}$ denotes the *j*-th transaction in the *i*th block. For all nodes participating in consensus, we use an externally specified constraint function $f \rightarrow \{0, 1\}$ wherein 1 presenting *valid* and 0 presenting *invalid* of each transaction to reach a consensus. Consider a peer-to-peer network with *n* nodes and *b* nodes controlled by a byzantine adversary where we seek a Beh-Raft protocol that runs between the processors which output a set *X* containing *k* ($k = n/c$) separate subsets or sharding $X_i = \{x_{i,j}\}(1 \leq j \leq |X_i|)$ such that the following conditions hold under the preconditions that 51% of identity-less processors be honest nodes:

—*Agreement*. Honest processors agree on X_i with a probability of 100% and malicious nodes agree on X_i with a probability of a given security parameter *p*.

—*Validity*. The agreed shard X_i satisfies the specified constraint function *C*, i.e., $\forall i \in \{1 \ldots k\}, \forall x_{i,j} \in X_i, \quad f(x_{i,j}) = 1$.

—*Scalability*. The number of shardings *k* grows linearly with *n*.

—*Efficiency*. The computation and bandwidth of each processor are not affected by *n* and *k*. The per-node communication and computation complexity are O(*n*) and the per-node storage complexity is O(*s*/*k*), where *s* is the total number of transactions.

B. Network Model

We apply the standard synchronous model for the underlying network, which is adopted by most public blockchain

protocols. To facilitate the reading, we applied the same assumption as defined in prior works [10], [16]: the honest nodes in the network are well connected and their communication channels are synchronous.

We apply the sharding framework of ELASTICO. The difference is that we replace PBFT by Behavior-based RAFT to boost the fault tolerance of the system. Only 1/2 of the nodes are required to be honest nodes in Raft while PBFT requires 2/3. Furthermore, RAFT implements consensus by assuming the leader is unconditionally trusted. Thus, our approach enhances ELASTICO investigating about how to apply the Raft consensus in a practical setting with malicious nodes or byzantine nodes while keeping it secure. The first significant distinction is that we require the honest processors to be in exact agreement, while the malicious nodes to be in probabilistic agreement. The reason is that the processors which are honest or malicious can be checked externally in our case; each honest processor can check if the behaviors of neighboring processors are normal or malicious and reward or punish them by adding or subtracting their weight value. Thus, we assume that malicious nodes may pretend to be normal for gaining the reward of weight value. The second distinction is that our incentive behavior mechanism can ensure that the agreed value is the input of the honest processors only, while the ELASTICO cannot.

C. Threat Model

We assume that processors controlled by the byzantine or malicious adversary can arbitrarily deviate from the protocol, such as reject responses, send fake messages and so on. All malicious nodes can collude together but they are only effective for the same shard in our Beh-Raft-Chain. Further, we consider that the adversary has full information about the messages transmitted on all links since any processor in the same shard can setup point-to-point communication links to each other. To differentiate the malicious behavior of rejecting responses from the network delay of honest processors, we assume that the network graph between honest processors is connected and the communication channel between honest processors is synchronous. That is to say those other honest processors will receive a message within a bounded delay parameter of Δ seconds. Byzantine nodes can delay even more significantly. Note that in our Raft protocol, an adversary can do nothing but reject responses when a malicious processor is not the leader. There is no effect on the consensus process at all, because our Beh-Raft-Chain can guarantee that the number of malicious processors in each sharding is less than half of the committee size, as shown in Section IV-A. However, if a malicious processor is elected as a leader, an adversary can send fake messages and make the consensus process fail because we design a supervisory mechanism to ensure the security of local consensus. In Section IV, we will show how to ensure that the malicious processor can't be elected as a leader by proposing a PoB mechanism.

During protocol runs, we make the assumption of honest nodes being reliable and failed nodes being considered malicious nodes. Moreover, we use standard majority assumptions for the total computational power of the byzantine adversaries, who corrupt $b < n/3$ of the nodes at any time. Similar to most sharding-based protocols [8], [16], we assume that the adversary can corrupt some nodes at the beginning of the protocol or between each epoch. However, the set of corrupt nodes cannot change within an epoch. And the adversary can corrupt a certain number of honest nodes while maintaining their identities at the final phase of each epoch. In addition, the adversary can run a join-leave attack [13], [19], meaning that it rejoins a small number of corrupt nodes with fresh identities in order to take over one or more committees. However, at any moment, at least two-thirds of the total number of nodes are honest.

D. Challenges

To provide "scale-out" transaction processing capacity competitively without compromising security for permissionless decentralization, we face three challenges: **1) Identities' Acceptance:** we must design an effective mechanism to resist Sybil attacks in a permissionless setting without the use of PoW; **2) Malicious Churn:** a malicious processor can re-enter with a new identity after quitting or simulate many virtual nodes thereby creating a large set of Sybils [13], [14], since there are no inherent identities or external Public Key Infrastructure (PKI) to trust for processors; **3) Secure Partition:** extra efforts are needed to ensure that all processors split into multiple shards uniformly, thus, each shard has the majority honest processors with high weight.

III. Beh-Raft-Chain Solution

In this section, we present Beh-Raft-Chain in detail. The algorithm proceeds in fixed periods called *epochs* and we describe the first epoch as a one-time bootstrapping protocol and the steps used in the subsequent epochs periodically.

A. Bootstrapping and Solution Overview

Inspired by Rapidchain [16] and Algorand [17], we replace the synchronous consensus protocol by integrating the Raft consensus protocol with the PoB mechanism. Specifically, we run the bootstrapping protocol to construct a behavior committee (BC) based on weighted nodes. In the meantime, the weights are generated based on their account balance against Sybil attacks as in Algorand [17] to ensure the honesty of the majority in BC, which is responsible for driving periodic reconfiguration events between epochs. The architecture overview of Beh-Raft-chain is shown in Fig. 1. At the bootstrapping phase, assume that all nodes start with a certain amount of money in their account and they can get a weight based on the money in their accounts. Each processor will be assigned to a local committee, which processes a disjoint set of transactions (or shards). The number of local committees grows proportionally to the total number of nodes in the network, which is $k = n/c$. All local committees with a relatively small number of members run an understandable consensus protocol of behavior-based Raft internally to reach an agreement of shards.

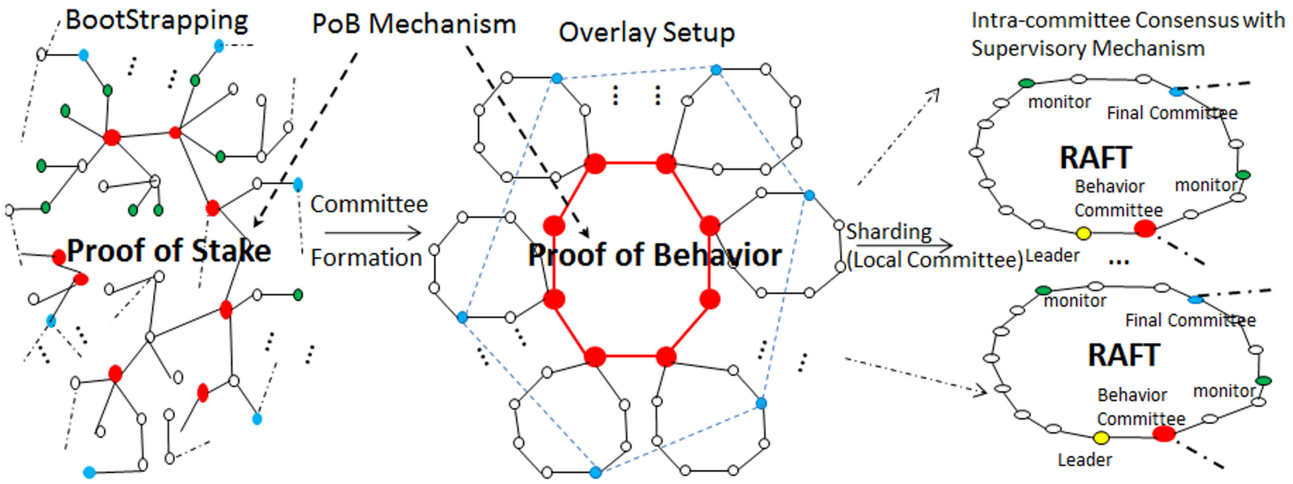


Fig. 1. Beh-Raft-chain architecture overview: (1) At the bootstrapping phase, all nodes are assigned a weight by adopting a PoS mechanism. The level of weights is visualized through different colors. (2) In subsequent epoch, all weights of nodes will be updated based on their behaviors. The higher the weight leading to a higher probability of being a leader/miner, which is the reward for honest nodes. (3) RAFT consensus algorithm is adopted in each local committee. The leader takes full responsibility under multiple monitors' supervision to ensure scalability and the probability of consensus failure in the practical setting.

In subsequent epoch, the protocol is processed in the following four steps: identity establishment and committee formation, intra-committee consensus, overlay setup for committees, committee reconfiguration and weighty updating, as shown in Fig. 1. First, we design a behavior-based method to eliminate the computational cost of PoW while preventing Sybil attacks. The algorithm can divide all nodes into multiple subsets of committees according to their weights; that is, the weight distribution of committees is well-proportioned and similar to that of the network. Then, the c nodes with the highest weights will be chosen and we design a sortition electing algorithm (detailed in 4.3), to enlarge the range of options and ensure that the nodes with higher weights are most likely to be chosen. Second, after the committee formation step, each local committee runs a standard Raft consensus protocol [12] integrating the PoB mechanism and supervisory mechanism to reach an agreement on a shard (or a set of transactions) and add it to its ledger. Third, the consensus results of local committees will be sent to the final committee, which is responsible for checking and piecing these agreed shards together for computing a cryptographic digest. Fourth, the final committee broadcasts only the cryptographic digest and the ID of the relative committee to the whole network to verify. It means that the protocol is to reach an agreement. And the specific transaction data is only stored in the relative ID of the local committee to minimize the storage problem. The last step in the epoch is the update of the weights of all processors by the behavior committee. All local committees will reward or punish their members' weights according to their normal or malicious behaviors and send new weights to the behavior committee. These weights are used in the subsequent epoch(s) as a basis for distinguishing honest processors from malicious ones. The weights are accumulated from previous epochs to ensure that the adversary cannot temporarily disguise themselves to gain high weights during this epoch.

B. Behavior Formulations

To be more clearly described and easier to read, we list the glossary of symbols in Table I and the formal specification of behaviors as follows.

First, we describe the concepts of *Entity* and *Action*. *Entity* (= *{A,B,C……}*) is a set of objects. It can be divided into three classes: *ME, LE, MO*. They are called *Roles*. In the real world, an entity of *ME* is a member of a local committee; An entity of *LE* is a leader of a local committee; An entity of *MO* is a monitor of a local committee.

Action = {receive, send, reply, require, validate, election, success, fail, update, deny, confirm, consensus, report, accept, refuse, timeout}. It consists of all the operations executed by entities in the system.

Definition 1 (Behavior). A behavior of entity A is a binary $<\mathbf{A,tr}>$, where $\mathbf{A}\in$*Entity and tr* is a *Trace*. It expresses a finite action sequence executed by the entity A.

We use three types of Deterministic Finite Automaton (DFA) to identify the malicious behavior of different types of entities.

A DFA is a quintuple $(S,\sum,\delta,S_0,F)$, in which S is a finite set of states, where S_0 is the initial *state*; F is a set of acceptable states, and $\sum$ is the finite set of alphabets. $\delta = s\times\sum\rightarrow S$ is a function of state transition.

We define three DFAs for the 3 *roles ME*, *LE* and *MO*, respectively.

1) The Distinguishing Automaton for the behaviors of an entity in ME (short for ME-A) is defined in Fig. 2. Where, $\sum$ = *Action*, S = {0,1,2,3,4,5}, 0 is the initial state, and F = {1,3,5}.
2) The Distinguishing Automaton for the behaviors of an entity in LE (short for LE-A) is defined in Fig. 3. Where, $\sum$ = *Action*, S = {0,1,2}, 0 is the initial state, and F = {2}.
3) The Distinguishing Automaton for the behaviors of an entity in MO (short for MO-A) is defined in Fig. 4. Where, $\sum$ = *Action*, S = {0,1,2}, 0 is the initial state, and F = {2}.

TABLE I
GLOSSARY OF SYMBOLS

Symbol	*Expression*
Entity	A set of objects in system, denoted as *{A, B, C, ……}.*
Action	A set operations executed by entities in the system, denoted as *{receive, send, ……}.*
Trace	A sequence on the set *Action.* For example, *receive→reply→confirm.*
Behavior	<A,*tr*>, where *A ∈Entity* and *tr* is a Trace.
ME, LE, MO	Subsets of *Entity*， called as Roles in committees.

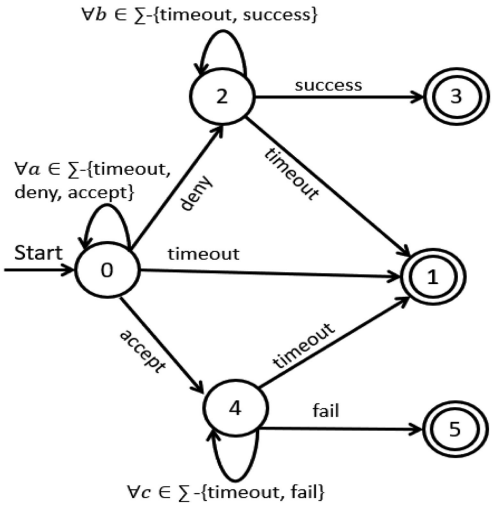


Fig. 2. The state graph of ME-A.

Criterion 1 (Malicious Behavior of Member). Given that <A,tr> is a behavior of an entity A. If A∈ME and *tr* is accepted by the DFA ME-A, then <A,tr> is considered a malicious behavior.

According to Criterion 1 and the DFA ME-A**,** malicious behaviors for the member entity consists of the following three cases:

1) ME-A reaches the acceptable state 1, meaning that the entity A keeps *silent* when one should respond to.
2) ME-A reaches the acceptable state 3, meaning that the entity A *denies* the message that should be accepted.
3) ME-A reaches the acceptable state 5, meaning that the entity A *accepts* the message that should be denied.

Criterion 2 (Malicious Behavior of Leader). Given that <A,tr> is a behavior of an entity A. If *A*∈LE and *tr* is accepted by the DFA LE-A, then <A,tr> is considered a malicious behavior.

According to Criterion 2 and the DFA LE-A, if LE-A reaches the acceptable state 2, the leader entity A was successfully reported by monitors. It must execute some sorts of malicious behavior.

Criterion 3 (Malicious Behavior of Monitor). Given that <A,tr> is a behavior of an entity A. If A∈MO and *tr* are accepted by the DFA MO-A, <A,tr> is considered a malicious behavior.

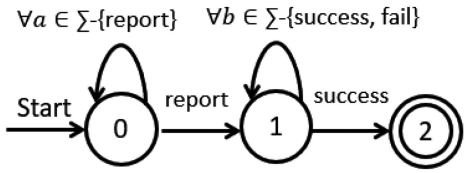


Fig. 3. The state graph of LE-A.

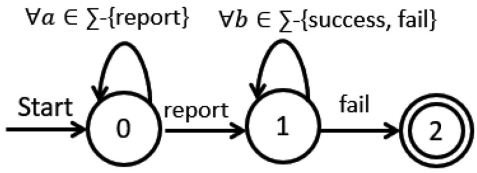


Fig. 4. The state graph of MO-A.

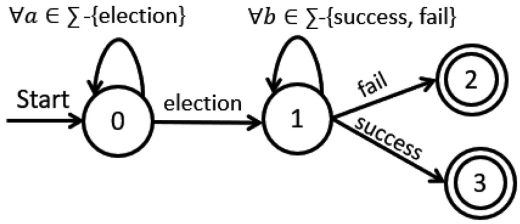


Fig. 5 The state graph of EL-A.

According to Criterion 3 and the DFA MO-A**, i**f MO-A reaches the acceptable state 2, the monitor entity A starts a malicious behavior *report* of leadership-node and the report is not acceptable by most of members. Thus, the behavior of monitor A is a malicious behavior.

Moreover, we provide the distinguishing method of double mechanism defined in Definition 3 for entities as shown in Fig. 5.

Where, $\sum = Action$, S = {0,1,2,3}, 0 is the initial state, and F = {2,3}.

Definition 2 (Leadership-node). If an entity is selected as a local leader/monitor or a member of the final/behavior committee, the entity is considered a leadership-node. They are dynamically adjusted at different epochs.

According to the DFA EL-A**,** it consists of the following two cases:

1) EL-A reaches the acceptable state 2, meaning that the entity A fails in an election.
2) EL-A reaches the acceptable state 3, meaning that the entity A succeed in being elected as a *leadership*-node.

Definition 3 (Double Mechanism). Double Mechanism is an adaptive adjustment of a penalty or a reward to stimulate the leadership-node.

Criterion 4 (Proof of Behavior Mechanism). If a behavior of a member entity A is accepted by the state 2 of DFA EL-A, we perform the normal behavior *mechanism* based on its behavior.

Criterion 5 (Joining of Leadership-node). For a member entity A, if its behavior is accepted by *the* state 3 of DFA EL-A, we perform the double mechanism based on its behavior.

Algorithm 1 Allocation Committee algorithm

```
input: all nodes with weights: node(seed, weight), the orderly
sequence of nodes based on their weigths: sortNodes
// This step divides the sorted nodes into n groups
subsets = GetSubset(sortNodes)
for subset in subsets
    for peer in subset
        peer.value = seed % weight
    end for
// Sort the calculated weights
    allotArray = SortWeight(subset)
    AllotAccordWeight(subset)
end for
Output:the set of committees with uniform distribution
```

Certainly, for all leadership-node, we *perform* the double mechanism based on their behaviors.

C. Identity Establishment and Committee Formation

In the following section, we will explain each step of identity establishment and committee formation in detail. First, each processor locally chooses its own identity of the form (IP, PK), which are IP address and public key respectively for authenticated communication later. To limit the number of Sybil identities created by malicious processors and ensure that the majority of processors would be honest, we adopt the behavior-based Proof-of-Stake mechanism of Bitcoin, which is called PoB for short, to accept these chosen identities in the network. Different from the PoS mechanism, our PoB mechanism first weighs all nodes based on their accounts at the committee formation phase, and then updates weight at the nodes based on their behaviors in the following epochs. And our PoB mechanism can avoid the disadvantage that the priority is controlled by a few people by setting an adjustment parameter. In detail, as shown in Fig. 1, we weigh nodes based on money in their accounts just as Proof-of-Stake (PoS) initially. During any epoch after that, all nodes are weighted based on their behaviors, which is different from PoS. The weights will be adjusted according to their behaviors. The first c nodes with high weights are appointed as the members of the behavior committee. This is under the assumption that most of the money belongs to honest nodes, which can ensure that the behavior-based Proof-of-Stake mechanism is safe.

And other processors' identities will be allocated by the behavior committee according to their weights while ensuring that each local committee has members with high and low weights uniformly distributed. The detail is shown in algorithm 1. Once the identities are allocated then the corresponding committees are determined.

D. Overlay Setup for Committees

We borrow from ELASTICO's overlay setup for committees instead of flooding broadcasts to reduce the number of broadcast messages. We design a special behavior committee of size c to serve as a set of "indices" and a weight ledger of all nodes. More specifically, the behavior committee is simply a committee of the c nodes with the highest weights. But the difference is that each leader of the local committee contacts several members of the behavior committee and gets the final list of their committee peers' links. If the list of members obtained is different, then contact all members of the behavior committee and reserve the majority of nodes and abandon the minority of nodes while reporting the corrupt nodes for punishments. This is a supervisory mechanism in the behavior committee to ensure security. In the end, the local members send the final list of all identities in their committees to the behavior committee for setting up point-to-point links. The Raft mechanism can greatly reduce the number of communication messages.

To prevent collusion attacks, each behavior member belongs to a different local committee. To ensure that the members of the behavior committee are honest, we employ the original PoS mechanism to obtain the weight of nodes for the first time and employ the behavior-based incentive mechanism to obtain the weight of nodes in the subsequence epoch.

E. Intra-Committee Consensus With Supervisory Mechanism

Next, we discuss how to efficiently implement transaction verification checks in a hypothetical cryptocurrency built on top of Beh-Raft-Chain. Typical Raft consensus algorithms make progress when any majority of their processors are available without considering the malicious ones; Raft implements consensus by first electing a distinguished leader, then giving the leader complete responsibility for checking the transactions. And data flows in a simple fashion from the leader to other processors. Thus, we just need to make sure that the leader is honest and most processors are available.

In each local committee of Beh-Raft-Chain, there are three roles: one leader, m monitors (one in the behavior committee, one in the final committee), and other processors. The unavailable nodes are considered malicious nodes. All processors run a standard Raft consensus protocol [12] integrating the PoB mechanism and supervisory mechanism to reach an agreement on a shard (or a set of transactions). The difference between us and [12] is that the leader performs the consensus under the supervision of the m monitors and is elected based on their behaviors to prevent a malicious leader in our case. We elect the leader based on their weights to ensure the honesty of the leader because increased weights also mean a higher degree of honesty in normal operation. In this way, our Beh-Raft-Chain can run on real-world scenes which include both honest and malicious processors. Because Beh-Raft-Chain can ensure that the elected leader is honest and the followers, even malicious ones, should perform the normal operation or at most keep silent under the leadership of the honest leader. The protocol can perform securely and continuously when there are less than c/2 malicious members in the local committee.

Besides, we set the supervisory mechanism to ensure the failure of consensus and verify it in the experiment. Each committee has several monitors that can supervise whether the behaviors of the leader were acting honestly or not. The reason is that the supervising node can receive all the messages of the

leader's to judge the leader's behavior. If the leader node is found to be dishonest, a report will be launched by a monitor; then, all supervisory nodes will run a consensus and reach an agreement. The leader will be removed and re-elected when the report is accepted. The supervisory mechanism stimulates monitors by offering a double reward/punishment. Yet, it is a challenging problem to set an appropriate number of supervisory nodes to balance the security of consensus and the number of interaction messages. Detailed security theoretical analysis is shown in Section IV-A and experimental results are shown in Section V.

F. Committee Reconfiguration and Storage

Reconfiguration allows new nodes to join the existing committees and regenerate only a fraction of the committees' members against the corrupt attack. A new challenge called *churn* will be introduced by partitioning the nodes into committees for scalability. To be specific, the adversary can control corrupt nodes to leave and rejoin the system strategically for taking over one or more committees. Moreover, the adversary can actively rejoin while its weight is lower than a new one. Thus, the weight is not further down when it is equal to zero in order to prevent churn in our solution. Then, we improve the Cuckoo rule [18], [19] based on weight to re-organize only a subset of committee members (*such as 10%*) during the reconfiguration event at the beginning of each epoch. We design an algorithm against the corrupt attack by using the Cuckoo rule [18], [19] with re-organizing only a subset of committee members during the reconfiguration event. And we improve the Commensal cuckoo rule [18] by considering the weight to distinguish honest new nodes from malicious ones.

Definition 4 (Weight-based cuckoo rule). When a new node wants to join the system, pick a random $x \in [0, 1)$, and get its weight w_i over the highest weight w_{max} in the system so as to obtain a priority coefficient $y = w_i/w_{max}$. If the group containing x has a node at x with weighty degree $y' < y$ or the group containing x has received at least k -1 secondary joins since its last primary join, we place the node at x and move (cuckoo) kg'/g random nodes in the group to random locations in [0, 1). g is the average group size and g' is the group's current size. Otherwise, we start over with a new random $x' \in [0,1)$.

As defined in [18], [19], we call the new node's join a primary join and the sub-sequent joins of the cuckoo nodes secondary joins. A node is called new only when it joined the system for the first time. Interestingly, there is one modification based on commensal Cuckoo rule [18], which can force repeated joins to join distinct groups by vetting repeated join attempts. First, we pick a random x to assure the randomness in the re-join process. Then, we add a condition on it: the new node with a sufficient weight has a higher privilege to join. The condition can distinguish whether joins come from a malicious node or a real new node because the malicious joined nodes do not have sufficient money in their accounts to gain higher weight or the adversary must invest enough money to obtain the joining privilege. In such a way, the condition can accelerate the joining speed of the real new ones without affecting the random distribution of malicious nodes. Besides, we design the condition to improve the problem of the commensal Cuckoo rule [18]: if too few secondary joins occurred, the commensal Cuckoo rule would risk deadlocking because no groups would accept a join.

To significantly improve the storage and communication overhead, the newcomer only has the permission to download the transactions within the sharding ledger, in order to verify future transactions and adopt the routing design of the Kademlia, which is inspired by the Rapidchain [16]. In Kademlia [33], each node stores information about all nodes within a logarithmic distance. We keep a routing table with log n records pointing to log n different committees at the distance of 2^i ($0 \leq i \leq \log n - 1$) in every committee. The communication between the committees can enable node discovery and message routing in log n steps. We omitted the detail of the Kademlia routing protocol due to the space limit. Interested readers may find more details in [33]. Under the routing mechanism Kademlia, we maintain disjoint ledgers according to the committee ID. Each committee only stores the specific transaction data of its committee ID to minimize the storage problem.

At the end of each epoch, the leaders of the local committee reward/punish their members' behavior via increasing/decreasing their weights and sending the agreed values list to the behavior committee. Then, the behavior committee updates these values for the next epoch.

IV. Security and Performance Analysis

A. Security Analysis

In this section, we provide our security analysis of BehRaft-Chain. We also discuss how malicious adversaries gain no significant advantage. We begin by clarifying the definition of behavior-based incentive mechanism.

Definition 5 (Behavior-based Incentive Mechanism). In the process of information exchange, honest nodes responding properly will get a bonus as a reward while malicious nodes will receive a minus as a punishment. The typical behaviors of changing weights are defined according to the criterions 1-3 in Section III.B.

As defined in definition 3 and criterions 4-5, we design the double mechanisms to stimulate honest behavior and neutralize malicious attacks for reducing the impact of malicious behaviors. If the node is a leadership-node, the corresponding reward or penalty will be adjusted adaptively. When a new node joins the system, its weight is zero when its account balance is zero. If the node maintains normal behavior, its weight will increase and have more opportunities to be selected as a leadership-node. Otherwise, if the node violates protocol rules, its weight continuously decreases until it reaches zero. When the weight is lower than the threshold, the node will be judged as a malicious node and will be ineligible to become a leadership candidate node.

Next, we present the detail theoretical analysis of how to reduce the impact of malicious behaviors and ensure the success rate of consensus.

Hypothesis 1. The number of honest nodes is more than 2/3 in the whole blockchain. The network is partially synchronous with the maximum delay of Δ.

Lemma 1 (Good Majority). In the first epoch, as assumed in [10], the possibility of the first c identities being controlled by the malicious adversary is $c/4$ at most. At any epoch after that, we elect the leadership based on their behaviors; honest members will be selected with a higher probability.

Proof. Let c be the size of committees. Based on hypothesis 1 and algorithm 1, we can ensure that a committee has $b<c/2$ malicious nodes and they perform any malicious activity has a probability $p\in[0,1]$. Let $p = 1/2$, then these $c/4$ nodes have been decremented with a minus. The positive possibility of the honest nodes being selected as honest nodes is $C_c^{c/2}$ at least while the negative possibility of malicious nodes is $C_c^{c/4}$ at most. With several interaction processes, more and more malicious nodes will be minus, thus the positive possibility will go higher and higher. ■

Let x_i be the random variable, f be the rate of malicious nodes when the system reaches stability (As proven above $f = 1/2^*1/2 = 1/4$). If the ith node is an honest node, x_i is 1. X presents the number of honest nodes in the committee, which follow a binomial distribution $X = \sum_{i=1}^{c} X_i$. Thus, the security is

$$\Pr\left[X < \frac{3c}{4}\right] = \sum_{i=0}^{\frac{3c}{4}} \Pr[X = i] = \sum_{i=0}^{\frac{3c}{4}} \binom{c}{i} f^{c-i}(1-f)^i$$

Obviously, the probability is lower as the size of the committee c is larger, and according to our experimental results, when c is equal to 600, the probability is lower than 10^{-7}.

According to Lemma 1, in the absence of any other assumptions, the maximum number of malicious identities that can be created before honest processors find all c identities is c/4. Furthermore, we choose the members of the behavior committee based on the Proof-of-stake which will greatly reduce the ratio of malicious nodes. The reason is that most of the money belongs to the honest. At any epoch after that, we can choose the leadership-node based on their weights, which can reduce the possibility of malicious nodes being chosen. The reason is that the malicious node not obeying the network rules must be punished by reducing their weights and the honest node following the rules must be awarded by giving a bonus to their weights. Thus, the gap between the honest and malicious nodes will get wider and wider. So, the possibility of malicious nodes being elected as a leadership-node is getting smaller and smaller.

However, as a member of the following node, the malicious processor can only withhold their identities and responds with nothing. In our case, honest members always share the same list of honest identities.

Next, we show how a committee (including the final committee and other committees) decides a single value correctly.

Lemma 2 (Local Consensus). In every epoch with hypothesis 1 and a good majority, the honest members agree on a unique set X_i with at least c/2 + 1 signatures, with high probability.

TABLE II
COMPLEXITIES OF PREVIOUS SHARDING-BASED BLOCKCHAIN PROTOCOLS

Protocol	Resiliency	ID Generation	Bootstrap	Consensus	Storage per Node
Beh-Raft-Chain	$f<n/3$	O(nlogn +n)	O(nlogn +n)	O(c/b+ c*log*n)	O(c·|B|/ n)
Elastico	$f<n/4$	O(n2)	Ω(n2)	O(c^2/b+n)	O(|B|)
OmniLedger	$f<n/4$	O(n^2)	Ω(n^2)	Ω(c^2/b+n)	O(c·|B|/ n)
RapidChain	$f<n/3$	O(n^2)	O($n\sqrt{n}$)	O(c^2/b+c*log*n)	O(c·|B|/ n)

Proof. The leader and monitors are honest nodes, we can assume the probability of malicious nodes performing a malicious activity is $p = 1/2$, and there are $c/4$ nodes responding with nothing or fake messages at most. Thus, the local consensus will certainly be reached with $3c/4$ signatures. ■

Besides, we design a supervisory mechanism to guarantee the success rate of consensus by setting several monitoring nodes as described above. Apparently, there is no problem when the leadership node is honest. Next, we will give a theoretical analysis of the failure of consensus if there is a problem with the leadership node. Let Y be the total number of honest leaders and monitoring nodes, m' be the total number of leader and monitoring nodes, and the consensus fails when the number of signatures is less than m'/2. Thus, the security is

$$\Pr\left[Y < \frac{m'}{2}\right] = \sum_{i=0}^{\frac{m'}{2}} \Pr[Y = i] = \sum_{i=0}^{\frac{m'}{2}} \frac{\binom{[cf]}{m'-i}\binom{c-[cf]}{i}}{\binom{c}{m'}}$$

where f represents the percentage of malicious nodes when the system reaches stability.

For example, if the committee size c is 600 and m' is c/10, the probability of failure is 10^{-6}. In addition, the final or behavior consensus protocol we borrowed from reference [8] will be reached with high probability as proven in [8].

In subsequent steps, we elect the leaders/monitors of local committees and the members of final or behavior committees based on our behavior-based incentive mechanism, which can guarantee that honest nodes occupy important roles. Since the number of malicious nodes is less than $c/2$, the consensus is successfully reached and the impact of malicious nodes is minimal and almost negligible.

B. Performance Analysis

Compared to previous works, we make a theoretical analysis of the performance of Beh-Raft-Chain and summarize them in Table II.

Complexity of Consensus. We calculate the complexity of consensus per transaction in general cases. We refer to the symbol definition in [16], let t denote a transaction that belongs to a committee C with size c. The total number of nodes is n. First,

Algorithm 2 Sortition algorithm

input: sk, seed, τ, role, weight, α,num
1: <hash,π>←$\mathrm{VRF_{sk}}$(seed$\mid\mid$role)
2: if (hash/2^{hashlen}<f(weight* $\alpha^{\mathrm{num}}$$_*\tau$))
3: return <hash, π, weight* α^{num} >;

Algorithm **3** Verify algorithm

input: pk, seed, τ, role, weight, α,num
1: if ¬$\mathrm{VerifyVRF_{pk}}$(hash, π, weight* α^{num}, seed$\mid\mid$role) then return 0;
2: if (hash/2^{hashlen}<f(weight* $\alpha^{\mathrm{num}}$$_*\tau$))
3: return 1;

the user sends *t* to the leader and a constant number of monitors (e.g., *m*). This imposes a communication and computational overhead of $(m+1) = \mathrm{O}(1)$. Next, the leader of C drives an intra-committee consensus Raft, which requires $m^*c/b = \mathrm{O}(c/b)$ communication to be amortized on the block of size *b* due to batching. In the end, the final committee broadcasting the results to all nodes requires O(c*log*n) communication when employing the Kademlia routing mechanism. Thus, the total cost of consensus iteration is O(*c*/*b*+ c*log*n).

Complexity of ID Generation. During ID Generation, each node sorting based on weights requires O(*n*log*n*) communication. After sorting, broadcasting the result incur an O(*n*) overhead. Thus, the total complexity is O(*nlogn*+*n*).

Complexity of Bootstrap Protocol. In the bootstrapping step, the sorting algorithm requires O*(nlogn*) communication. Then, each node will be allocated to a local committee incurring O(*n*) overhead. Thus, the total complexity is O(*nlogn*+*n*).

Complexity of Storage. Let $\mid \mathrm{B} \mid$ denote the size of the blockchain. We divide the ledger among *n/c* shards, thus each node stores $\mathrm{O}(c\cdot \mid \mathrm{B} \mid /n)$ units of data. Note that the cost of weighty storage is asymptotically negligible relative to the size of the ledger that each node has to store.

C. Sustainability Analysis

One challenge of our weight-based election mechanism introducing is that a few given users may be chosen more than once on account of their high weights which may incur a case of the leadership only being controlled by few nodes in a situation known as *monopoly* attack. To solve this attack, we design an active parameter $\boldsymbol{\alpha}$ *(i.e., 0.9)* to adaptively adjust the possibility of being elected to incentivize more nodes being elected. Parameter *num* indicates the number of times a user has been chosen. Meanwhile, we set a threshold $\rho(0<\rho<1)$ to determine the minimum node weight that is capable of being selected to guarantee honesty. Namely, a node with weight w_i can be a candidate of the leadership if and only if its weight w_i is bigger than the threshold ρ. For example, given a set of nodes with weights $w_i > \rho$ and the total weight of them $\mathrm{W} = \sum_i w_i * \alpha^{num}$, the probability that user *i* is selected is equal to $\mathrm{p_i} = w_i{*}\alpha^{num}$ */W*. In the initial, the value of *num* is 0.

We use VRFs to implement cryptographic sortation as shown in algorithm 2 on the basis of reference [20]. Aiming at implement cryptographic sortation, we use for reference [17] the algorithmic framework of sortition and introduce the *role* parameter for distinguishing the different roles that a user may be selected for; such as the local committee leader or the member of the behavior/final committees at a certain step. We specify a threshold τ that determines the expected number of users selected for that role. The *f* function is a monotone increasing function that maps weights to the interval (0,1). *f* can guarantee that higher weights gain higher probabilities of selection. $\boldsymbol{\alpha}^{num}$ is the adjustment parameter according to the number of times(*num*) a user has been chosen in the previous phase. The adjustment parameter is used to adjust the probability of being chosen and expand the options beyond simply choosing the node with the highest weights. Informally, $\mathrm{VRF_{sk}}(x)$ can returns two values of a hash and a proof for any input string *x*. The hash, a hashlen-bit-long value, is indistinguishable from random values to those adversaries who do not know the value of *sk* since it is uniquely determined by *sk* and *x*. The proof π makes anyone with knowing *pk* and without having to know *sk* for checking whether the hash indeed corresponds to *x* or not. AS shown in algorithm 2 and *3*, we require that the $\mathrm{VRFs}(pk_i, sk_i)$,which is a public/private key pair, provides these properties for security even that *pk* and *sk* are chosen by an attacker. Once the specified number of nodes has been elected, the round terminates. This mechanism enables the protocol to withstand deviations such as malicious nodes and enables us to show that honest behavior is in an approximate equilibrium under reasonable assumptions regarding the costs of running the protocol.

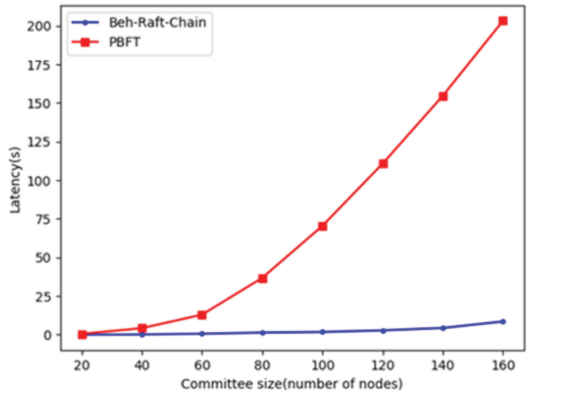
V. Evaluation

In this section, we implement Beh-Raft-Chain in Python and empirically evaluate the scalability of Beh-Raft-Chain and previous solutions. The goals of our evaluation are threefold. We first measure the system efficiency by latency and consensus cost as the committee size increases. The second goal is to guarantee the success rate of consensus under different monitors to find the suitable number of monitors. The third goal is to measure the scalability of Beh-Raft-Chain by system performance and communication cost over different network sizes. The experimental environment was set on a Windows 10 computer equipped with an eight-i7 Core CPU and 8 GB RAM. In our experiments, we consider the worst case of that c/4 members are unavailable as described in Lemma 1. All experiments take the average of the results over 10 iterations.

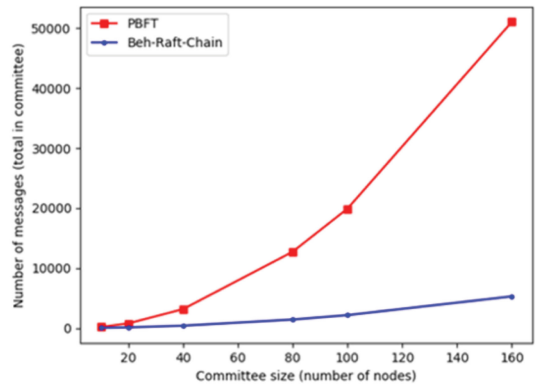
A. The Accuracy of the Consensus Algorithm

This paper compared the operation efficiency of PBFT and Raft consensus mechanisms. We compared the number of message exchanges of the consensus algorithm and the time required to reach an agreement. The scale of our experiment is from 20 to 160, and the step size is 20. The average of all experimental results is taken from 10 runs for the average. We generate the network bandwidth by reference [31], [32] to simulate variable delays.

The experimental results are shown in Fig. 6. As depicted in Fig. 6(a), the delay of our consensus algorithm is almost a



(a) Latency to reach consensus with different committee sizes



(b) Cost in consensus

Fig. 6. The accuracy performance of consensus algorithms.

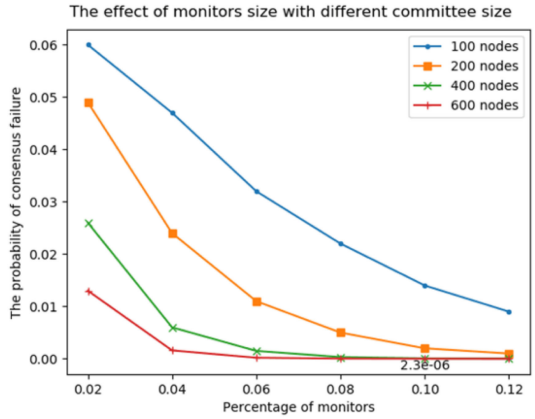


Fig. 7. The change of fail rate over the number of monitors.

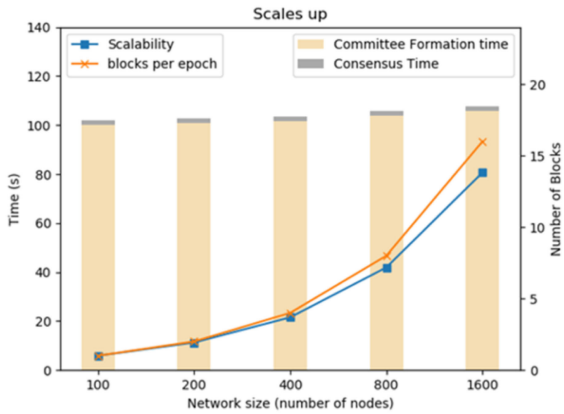


Fig. 8. The performance of system scale with network size.

straight line. The delay is calculated in terms of the end time of consensus minus the start time. It is less than 2 seconds when the number of nodes is 100, and about 8 seconds even when the number of nodes is 160. In contrast, the curve of PBFT delay experiences a dramatic surge from 25 seconds to over 3 minutes when the number of nodes increased from 60 to 140. In fact, the delay of PBFT consensus will further increases in practice given the existence of Byzantine nodes. A similar conclusion can be observed in terms of message traffic as shown in Fig. 6(b). This phenomenon can be explained by the difference in the communication complexity of the PBFT and our consensus algorithm, which are $O(n^2)$ and $O(n)$, respectively.

B. The Security of the Consensus Algorithm

The security of consensus is measured by the probability of consensus failure as the number of monitors varies. Simulation results in Fig. 7 shows that the rate of consensus failures decreases as the number of the monitoring nodes and the size of the committee increase. The failure rate is about 0.014 in the case of the committee size is 100 with 10% rate of the monitoring nodes, while the failure rate is about 10e-6 while the committee size is 600 with 10% rate of the monitoring nodes. Since these results have validated our theoretical analysis in Section IV-A, we set 10% of monitoring nodes as the default rate in later experiments. Note that, all of the above do not consider the influence of weights. Thus, the failure rate will be much less than the rate in the actual situation. The reason is that the leadership of the committee with higher weights is impossible to perform malicious behavior.

C. The Scalability of the Blockchain

In this experiment, we expanded the network scale from 100 to 1600 when each district committee has a size of 100. We counted the time spent on the formation and consensus in each epoch, and took the average of the results over 10 iterations.

As shown in Fig. 8, our approach scales up the block throughput at an almost linear rate of growth relative to the size of the network. Different from ELASTICO [10], whose establish time increases as the number of nodes increases because of the PoW mechanism, the time consensus and establishment of Beh-Raft-Chain almost maintain a constant value. The reason is that each partition is an independent parallel processing transaction and we eliminate the PoW mechanism. So, the total delay time of the system remains nearly constant even while the node size increases, which means that our method makes a much more scalable system.

D. The Efficiency of Blockchain

The efficiency of blockchain, mainly indicated by the number of communication messages sent and received per node,

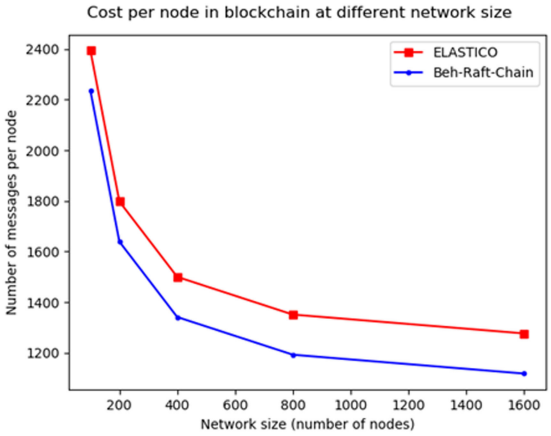


Fig. 9. The cost of communication over different network size.

was summarized in Fig. 9. We show the average number of messages per node under different system scales. Note that the amount of communication decreases as the system size increases. This is because a lot of focus message nodes (mainly of behavior committees and final committees) traffic has been split. Experiments have shown that our methods communicate less because we use the Raft consensus algorithms in the local committees with lower complexity.

VI. Related Work

The scalability of consensus algorithms in open blockchains is an active problem. There have been several solution proposals from both academia and industry. We summarize these existing consensus protocols of blockchain into the following two aspects.

The first category is consensus protocols based on the committee. Based on PBFT [11], PeerCensus [25] is the first consensus protocol inside a committee resulting in strong consistency guarantees, while how a committee is formed and we can ensure an honest majority is not mentioned. ByzCoin [26] proposes using a multi-signature protocol inside a committee to reduce the communication complexity of PBFT. However, they do not remove trailing unstabilized blocks from the blockchain for committee election and consistency can be broken when the nodes do not agree on the committee. Unfortunately, both of them are vulnerable to the same type of selfish mining attack [27]. That is, they cannot tolerate 1/3 corruption due to degraded chain quality. Hybrid Consensus [28] studied the feasibilities of achieving responsiveness in permissionless consensus. They propose a Byzantine consensus protocol that allows transactions confirmed after a warmup period and assumes that the adversary can only corrupt honest nodes without stickiness. Algorand [17] proposed a new Byzantine Agreement protocol (called BA⋆) to reach a consensus for oncoming transactions with a delay of no more than one minute. They weighed users based on the money in their account and ran consensus protocol by a committee chosen based on the users' weights in a private and non-interactive way. Algorand mitigated targeted attacks on the committee members by requiring committee members to speak just once and electing new committee members in each step of BA⋆. However, the randomness used in each VRF invocation can be biased by the adversary, which resulted in a biased coin problem. To solve the issue, Dfinity [29] proposed a new VRF protocol with unbiasable and verifiable property based on a unique-deterministic and non-interactive threshold signature scheme.

The other category is consensus protocols based on sharding, which partition the whole nodes into multiple sharding and allow them dealing with incoming transactions in parallel for increasing its transaction processing power. We only focus on these researchers' work on handling sharding in the Bitcoin transaction model. ELASTICO [10] is the first consensus protocol based on sharding to improve the throughput and latency of open blockchain and its performance has upgraded several orders of magnitude. However, as analyzed in [16], it requires extra time to solve enough PoWs to fill up the trust problem between all the committees and the sizes of committees can heavily influence the security of the protocol. Moreover, ELASTICO requires a trusted setting for generating unbiased randomness without affects from any adversary. Besides, ELASTICO runs a Byzantine fault-tolerant protocol which can only tolerate up to a quarter of parties being faulty even with a high probability of failure. Thus, OmniLedger was proposed by *Kokoris-Kogias et al. [15]* recently. To fix some of the issues of ELASTICO, OmniLedger is designed with concurrency to generate identities and assign participants to committees, in which both the whole system and each shard separately validate transactions in parallel. Each epoch generates a random number with bias-resistant property using a VRF. However, OmniLedger is difficult to resist highly adaptive attacks. For example, a malicious user can leverage the atomic cross-shard protocol to lock any transaction to launch a denial-of-service (DoS) attack. Furthermore, *Zamani et al.* [16] proposed a Byzantine-resilient public blockchain protocol, called RapidChain, to improve upon the scalability and security limitations of previous works. They scale the throughput of the system proportionally to the number of committees by partitioning nodes into committees and enabling parallelization of the consensus work. They also reduce storage by maintaining disjoint ledgers and building upon the Cuckoo rule [18], [19] to provably protect against a slowly-adaptive Byzantine adversary which is unlike the basic consensus protocol. More recently, *Huang et al.* [30] proposed a RepChain with double chain architecture which enables a reputation chain with moderate generation speed to support a high throughput transaction chain. However, this method would incur other problems: high storage and message costs.

VII. Conclusion

In this study, we focused our attention on the scalability of sharding-based blockchain protocols and developed a Beh-Raft-Chain solution to reduce the traffic complexity while enhancing the transaction rates and the amount of fault-toleration for complex network applications. Compared with existing blockchain protocols as discussed in Section IV-B, Beh-Raft-Chain will get more applications in 5G networks,

health care industries, and IIoTs. In detail, we replaced the classic PBFT consensus protocol with the other classic Raft consensus protocol based on the PoB mechanism, which can incentivize honest behaviors and neutralize malicious attacks. We also devised a well-proportioned allocation algorithm to assign a local committee to each node while guaranteeing that each committee has the same distribution of weights. Then, we designed a supervisory mechanism to guarantee the security of local consensus. Under our solution, we set a self-adaptive parameter to satisfy sustainability requirements by incentivizing honest behavior by enlarging the range of candidate nodes being chosen to resist monopoly attacks. Through extensive experiments, we measured the scalability and efficiency of Beh-Raft-Chain by comparing it with previous approaches in [10], [15], [16]. The results demonstrated that our solution preserves a higher level of scalability than the existing sharding-based protocols for different utility requirements.

ACKNOWLEDGMENT

The authors would like to thank the editors and the anonymous reviewers for their helpful comments and suggestions for this paper.

REFERENCES

[1] A. W. Peters *et al.*, "Blockchain technology in health care: A primer for surgeons," *Bull. Amer. College Surgeons*, vol. 12, pp. 1–5, 2017.
[2] P. Angin *et al.*, "A blockchain-based decentralized security architecture for IoT," in *Proc. Int. Conf. Internet Things*, Cham, Switzerland, 2018, pp. 3–18.
[3] H. Sun, S. Hua, E. Zhou, B. Pi, J. Sun, and K. Yamashita, "Using ethereum blockchain in internet of things: A solution for electric vehicle battery refueling," in *Proc. Int. Conf. Blockchain*, Cham, Switzerland, 2018, pp. 3–17.
[4] L. Zhou, L. Wang, Y. Sun, and P. Lv, "Beekeeper: A blockchain-based iot system with secure storage and homomorphic computation," *IEEE Access*, vol. 6, pp. 43472–43488, 2018.
[5] O. Novo, "Blockchain meets IoT: an architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.
[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[7] Blockchain stats, "Bitcoin statistics," 2019, Accessed: Aug. 28, 2019. [Online]. Available: https://www.blockchain.com/en/stats
[8] M. Ruta *et al.*, "Semantic blockchain to improve scalability in the internet of things," *Open J. Internet Things.*, vol. 3, no. 1, pp. 46–61, 2017.
[9] J. Yang, Z. Lu, and J. Wu. "Smart-toy-edge-computing-oriented data exchange based on blockchain," *J. Syst. Architecture*, vol. 87, pp. 36–48, 2018.
[10] L. Luu *et al.*, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 17–30.
[11] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Des. Implementation*, 1999, pp. 173–186.
[12] D. Ongaro, J. K. Ousterhout. "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 305–319.
[13] J. R.. Douceur, "The sybil attack," in *Proc. 1st Int. Workshop Peer-to-Peer Syst.*, 2002, pp. 51–260.
[14] J. Newsome, E. Shi, D. Song, and A. Perrig. "The sybil attack in sensor networks: Analysis & defenses," in *Proc. 3rd Int. Symp. Inf. Process. Sensor Netw.*, New York, NY, USA, 2004. pp. 259–268.
[15] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 583–598.
[16] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 931–948.
[17] Y. Gilad *et al.*, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 51–68.
[18] S. Sen and M. J. Freedman, "Commensal cuckoo: Secure group partitioning for large-scale services," *ACM SIGOPS Operating Syst. Rev.*, vol. 46, no. 1, pp. 33–39, 2012.
[19] B. Awerbuch and C. Scheideler, "Towards a scalable and robust DHT," *Theory Comput. Syst.*, vol. 45, no. 2, pp. 234–260, 2009.
[20] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proc. IEEE 40th Annu. Symp. Found. Comput. Sci.*, 1999, pp. 120–130.
[21] X. Ji *et al.*, "Overview of 5G security technology," *Sci. China Inf. Sci.*, vol. 61, no. 8, pp. 081301:1–081301:25, 2018.
[22] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial internet of things," *J. Softw. Eng. Appl.*, vol. 9, no. 10, pp. 533–546, 2016.
[23] M. Andoni *et al.*, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renewable Sustain. Energy Rev.*, vol. 100, pp. 143–174, 2019.
[24] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[25] C. Decker, J. Seidel, and R. Wattenhofer, "Bitcoin meets strong consistency," in *Proc. 17th Int. Conf. Distributed Comput. Netw.*, 2016, pp. 13:1–13:10.
[26] E. K. Kogias *et al.*, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th {USENIX} Secur. Symp.*, 2016, pp. 279–296.
[27] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, 2018.
[28] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," in *Proc. 31st Int. Symp. Distributed Comput. (DISC 2017), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik*, 2017, vol. 91, pp. 39:1–39:16.
[29] T. Hanke, M. Movahedi, and D. Williams, "DFINity technology overview series, consensus system," 2018.
[30] C. Huang *et al.*, "RepChain: A reputation based secure, fast and high incentive blockchain system via sharding," 2019.
[31] P Megyesi *et al.*, "Available bandwidth measurement in software defined networks, " in *Proc. 31st Annu. ACM Symp. Appl. Comput.*, 2016, pp. 651–657.
[32] A. Botta, W. De Donato, A. Pescapé, and G. Ventre, "Discovering topologies at router level: Part II," in *Proc. IEEE Global Telecommun. Conf.*, 2007, pp. 2696–2701.
[33] P. Maymounkov, D. Mazieres, and D. Kademlia, "A peer-to-peer information system based on the xor metric," in *Proc. Int. Workshop Peer-to-Peer Syst.*, Berlin, Heidelberg, 2002, pp. 53–65.

Li-e Wang received the master's degree in software engineering from Hunan University, Changsha, China, in 2007. She is currently working toward the Ph.D. degree with the College of Computer Science and Information Engineering, Guangxi Normal University, Guilin, China, where she is also an Associate Professor. Her research interests mainly include data privacy, computer networking, Healthcare, and distributed system security. She has authored more than 20 refereed papers in these areas. She has served as a reviewer for several high impact research journals and ACM/IEEE flagship conferences.

Yan Bai received the Ph.D. degree in Electrical and Computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2003. She is an Associate Professor with the School of Engineering and Technology, University of Washington Tacoma, USA. Her research interests include computer networking, multimedia communications, cybersecurity, eHealth, Internet of Things, blockchain, cloud and edge computing. She has authored more than 70 refereed papers in these areas. She has served as a General Chair/Program Chair/ Technical Program Committee Member for over 50 IEEE conferences and workshops, and as a Reviewer for a wide range of high impact research journals and ACM/IEEE flagship conferences.

Quan Jiang received the bachelor's degree in measurement and control technology and instrument from Chongqing Technology and Business University, Chongqing, China. He is currently working toward the master's degree with the College of Computer Science and Information Engineering, Guangxi Normal University, Guilin, China. His research interests include data privacy, Internet of Things, and distributed system security.

Victor C. M. Leung (Fellow, IEEE) is a Distinguished Professor of Computer Science and Software Engineering with Shenzhen University, Shenzhen, China. He is also an Emeritus Professor of Electrical and Computer Engineering and the Director of the Laboratory for Wireless Networks and Mobile Systems with the University of British Columbia (UBC), Vancouver, BC, Canada. He has coauthored more than 1300 journal/conference papers and book chapters. His research is in the broad areas of wireless networks and mobile systems. He is serving on the editorial boards of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE ACCESS, IEEE NETWORK, and several other journals. He was the recipient of the IEEE Vancouver Section Centennial Award, 2011 UBC Killam Research Prize, 2017 Canadian Award for Telecommunications Research, and 2018 IEEE TCGCC Distinguished Technical Achievement Recognition Award. He coauthored papers that won the 2017 IEEE ComSoc Fred W. Ellersick Prize, 2017 IEEE Systems Journal Best Paper Award, 2018 IEEE CSIM Best Journal Paper Award, and 2019 IEEE TCGCC Best Journal Paper Award. He is a Fellow of the Royal Society of Canada, Canadian Academy of Engineering, and Engineering Institute of Canada. He is named in the current Clarivate Analytics list of "Highly Cited Researchers".

Wei Cai (Member, IEEE) received the B.Eng. degree in software engineering from Xiamen University, Xiamen, China, in 2008, the M.S. degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2011, and the Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, BC, Canada, in 2016. From 2016 to 2018, he was a Postdoctoral Research Fellow with UBC. He joined the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, in August 2018, where he is currently an Assistant Professor. His recent research interests include cloud computing, blockchain systems, crowd intelligence, and human-cloud interaction. He is serving as an associate editor of IEEE Transactions on Cloud Computing. He was the recipient of the 2015 Chinese Government Award for the Outstanding Self-Financed Students Abroad, UBC Doctoral Four-Year-Fellowship from 2011 to 2015, and the Brain Korea 21 Scholarship. He was also the recipient of the best paper awards from CloudCom2014, SmartComp2014, and CloudComp2013.

Xianxian Li is a Professor with the College of Computer Science and Information Engineering, Guangxi Normal University, Guilin, China. He has authored more than 60 refereed papers in these areas. His research interests include data security, Internet of Things, and software theory. He has served as a Program Co-Chair/Technical Program Committee Member for several IEEE conferences and workshops.