Performance Analysis of the Raft Consensus Algorithm for Private Blockchains

Dongyan Huang, Xiaoli Ma, *Fellow, IEEE*, and Shengli Zhang, *Senior Member, IEEE*

***Abstract*—Consensus is one of the key problems in blockchains. There are many articles analyzing the performance of threat models for blockchains. But the network stability seems lack of attention, which in fact affects the blockchain performance. This paper studies the performance of a well adopted consensus algorithm, Raft, in networks with non-negligible packet loss rate. In particular, we propose a simple but accurate analytical model to analyze the distributed network split probability. At a given time, we explicitly present the network split probability as a function of the network size, the packet loss rate, and the election timeout period. To validate our analysis, we implement a Raft simulator and the simulation results coincide with the analytical results. With the proposed model, one can predict the network split time and probability in theory and optimize the parameters in Raft consensus algorithm.**

***Index Terms*—Blockchain, network split probability, private blockchain, Raft consensus algorithm.**

I. Introduction

BLOCKCHAIN technology, which was first coined in Bitcoin [1] by Nakamoto in 2008, has received extensive attentions recently. A blockchain is an encrypted, distributed database/transaction system, where all the peers share information in a decentralized and secure manner. Due to its key characteristics as decentralization, immutability, anonymity, and auditability, blockchain becomes a promising technology for many kinds of assets transfer and point-to-point (P2P) transaction [2]. Recently, blockchain-based applications are springing up, covering numerous fields, including financial services [3]–[5], Internet of Things [6], reputation systems [7], and so on.

Since the essence of blockchain is a distributed system, consensus algorithms play a crucial role in maintaining the safety and efficiency of blockchains. A consensus algorithm for distributed systems is to figure out the coordination among multiple nodes, i.e., how to come to agreement if there are multiple nodes. Several consensus algorithms have been proposed, e.g., proof-of-work (PoW) [1], proof-of-stake (PoS) [8], [9], delegate PoS [22], practical Byzantine fault tolerant (PBFT) [11], Paxos [12], and Raft [13]. Among them, PoW and PoS algorithms have a good support for safety, fault tolerance, and scalability of a blockchain. Thus, PoW and PoS are common choices of public blockchains, in which any one can join the network and there are no trust relationships among the nodes. However, PoW and PoS have slow speed of transaction confirmation, which limits its applications to those requiring high confirmation speed. In a consortium/private blockchain network, all participants are whitelisted and bounded by strict contractual obligations to behave "correctly," and hence more efficient consensus algorithms, such as PBFT and Raft are more appropriate choices. Consortium/private blockchains could be applied into many business applications. For example, Hyperledger [14] is developing business consortium blockchain frameworks. Ethereum has also provided tools for building consortium blockchains [15]. The Raft algorithm is considered as a consensus algorithm for private blockchains [16], which is applied to more ad hoc networks such as the Intranet. Furthermore, several hybrid consensus protocols have been proposed to improve consensus efficiency without undermining scalability. For example, Zilliqa [17] proposed PoW to elect directory service (DS) committee nodes, and then the DS committee has to run PBFT consensus protocol on the transaction block.

Compared with PBFT and Paxos, the Raft algorithm has high efficiency and simplicity and it has been widely adopted in the distributed systems. Raft is a leader-based algorithm, which uses leader election as an essential part for the consensus protocol. Ledger entries in Raft-based system flow in only one direction from the leader to other servers. Paxos and its improved algorithms do not take leader election as an essential part of the consensus protocol, which can balance load well among nodes because any node may commit commands [12], [18]. However, Paxos' architecture requires complex changes to support practical systems. Raft achieves the same safety performance as Paxos and is more convenient in engineering implementation and understanding. Raft consensus algorithm cannot tolerate malicious nodes and can tolerate up to 50% nodes of crash fault. For private blockchains, nodes are verified members. Hence, it is more

Manuscript received June 27, 2018; revised September 30, 2018; accepted January 18, 2019. Date of publication March 12, 2019; date of current version December 31, 2019. This work was supported in part by the Chinese NSF Project under Grant 61771315, in part by the Shenzhen NSF Project under Grant JCYJ20160226192223251, Grant JCYJ20170412104656685, and Grant JCYJ20170302142312688, and in part by the Director Fund of Guangxi Key Laboratory of Wireless Broadband Communication and Signal Processing under Grant GXKL06160111, and in part by QuarkChain Foundation Ltd. This paper was recommended by Associate Editor Y. Yuan. *(Corresponding author: Shengli Zhang.)*

D. Huang and S. Zhang are with the College of Information Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: huangdongyan-gua@163.com; zsl@szu.edu.cn).

X. Ma is with the School of Electrical and Computer and Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: xiaoli@gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2019.2895471

important to solve the crash faults than Byzantine faults for private blockchains.

Network is called split when more than half of the nodes are out of current leader's control. Failure of node and communication interruption caused by packet loss are the main reasons of network split. If the network split occurs, the blockchain network with Raft consensus algorithm would restart a new leader election process. Meanwhile, the blockchain network stops accepting new transactions, i.e., the blockchain network becomes unavailable. Obviously, the consensus efficiency of blockchain is degraded tremendously if network split occurs frequently. The existing works on blockchain mainly focus on designing algorithms or optimizing performance or safety certification, but lack of theoretical analysis of the network split. The impact of the packet loss rate on network split is rarely considered. In fact, the packet loss rate plays an important role on the network split.

In this paper, we concentrate on analyzing the network split probability of the Raft algorithm. We provide a simple model that accounts for protocol details, and allows to compute the performance of distributed networks. The main contributions of this paper are given as follows: 1) a simple analytical model is developed; 2) the network performance in normal conditions is derived; and 3) we explore the parameters' (such as packet loss rate, period of election timeout, and size of network) impact on the network split performance.

II. Review of Raft Algorithm

This section briefly summarizes the Raft algorithm. A more complete and detailed description of Raft refers to [13]. Raft is a consensus algorithm for managing a replicated ledger at every node. At any given time, each node is in one of the three states: 1) leader; 2) follower; or 3) candidate. Raft algorithm divides time into terms with finite duration. Terms are numbered with the consecutive integers. Each term begins with an election, in which one or more candidates attempt to become a leader. If a candidate wins the election, then it serves as a leader for the rest of the term. The state transition is shown in Fig. 1 [19]. All the nodes start from the follower state. If a follower does not hear from the leader for a certain period of time, then it becomes a candidate. The candidate then requests votes from other nodes to become a leader. Other nodes will reply to the vote request. If the candidate gets vote from a majority of the nodes, it will become a leader. This process is called leader election. Specifically, if a follower receives a heartbeat within the minimum election timeout of hearing from a current leader, it does not grant its vote for the candidate. This helps maximizing the duration of a leader to keep working and avoiding frequent disruptions from some isolated/removed nodes.[1]

In normal operation of Raft, there is exactly one leader and all the other nodes are followers. The leader periodically sends out heartbeat to all its followers in order to maintain the authority. All transactions during this term go through the leader. Each transaction is added as an entry in the node's ledger. Specifically, the leader first replicates the received new transaction to the followers. At this time, the entry is still uncommitted and stays in a volatile state. After the leader feedbacks receives from a majority of followers that have written the entry, the leader notifies the followers, that this, the entry is committed. This process is called ledger replication.

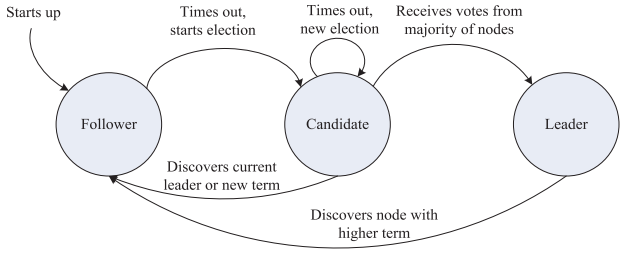


Fig. 1. State transition model for the Raft algorithm.

In a Raft algorithm, there are several timeout settings. One of them controls the election process. The election timeout is the amount of time that a follower needs to wait, to become a candidate. The election time counter is decreased as long as the follower receives no heartbeat. The follower transfers to candidate state when the election time reaches zero. The election time counter resets to a random value when the follower receives a heartbeat from a leader. The random election timers in Raft help to reduce the probability, that several followers transfer to candidates simultaneously.

III. System Model

In this section, we focus on the analysis for network split probability. Consider a distributed network with N nodes and N is odd.[2] The way of message exchanging among nodes is according to the Raft algorithm. To ensure the system efficiency, the interval between heartbeats is much less than the election timeout. The average time between failures of a single node is much larger than the election timeout.

A network split happens if more than half of the nodes are out of the current leader's control. We try to investigate the relationship between the parameters (such as timeout counter, packet loss rate, and network size) and network split probability. First, the process of one follower node transfers to a candidate is modeled as an absorbing Markov chain. Then, the network spilt probability is derived based on the model. Finally, the expected time for a node transferring from the follower to the candidate and the expected number of received heartbeats for one node are derived. Besides, we also discuss the impact of packet loss on the election performance.

In the following analysis, within one term the initial state is defined as after a successful received heartbeat, the follower's election counter is reset. Suppose that there is a leader and $N-1$ followers. We assume that, the communication delay is much less than the heartbeat interval. One heartbeat means one step in the Markov chain.

[1]These nodes will not receive heartbeats, so they will time out and start new elections.

[2]When N is even, the algorithm still works and our claims still hold. Here, for the convenience of analysis, we consider odd N. So that $N-1$ is even.

A. Network Model

Define the packet loss probability as p, and suppose that p is a constant value for a given network. Denote the timeout value for each round of election as E_t, which is initially uniformly chosen from the range $[a, b]$. The interval between two heartbeats is τ. A discrete and integer time scale is adopted. Thus, if a follower fails to receive $K = \lfloor E_t/\tau \rfloor$ heartbeats consecutively, then it assumes there is no viable leader and transitions to candidate state to start an election. Noted that $K \in \{K_1, K_2, \ldots, K_r\}$, and K is the uniformly chosen from the set $\{K_1, K_2, \ldots, K_r\}$, where $K_1 = \lfloor a/\tau \rfloor$ and $K_r = \lfloor b/\tau \rfloor$. In the following analysis, K denotes the maximum number of heartbeats for an election counter to timeout.

Let $g(n)$ be the stochastic process representing the stage status $\{1, 2, \ldots, r\}$ of a given node at time n. Let $b(n)$ be the stochastic process representing the left steps of election time counter for the node at time n. Once the independence between $g(n)$ and $b(n)$ are assumed, we can model it as a two-dimensional process $\{g(n), b(n)\}$.

We Adopt the Short Notation: $P\{i, k_i - 1 | i, k_i\} = P\{g(n+1) = i, b(n+1) = k_i - 1 | g(n) = i, b(n) = k_i\}$. In this Markov chain, the only non-null one-step transition probabilities are

$$\begin{cases} P\{i, k_i - 1 | i, k_i\} = p & (1) \\ P\{i, K_i | j, k_j\} = (1-p)/r & (2) \\ P\{i, 0 | i, 0\} = 1 & (3) \end{cases}$$

where $i, j = 1, 2, \ldots, r$ and $k_i \in \{1, \ldots, K_i\}$. Equation (1) relies on the fact that the follower fails to receive a heartbeat from the current leader and its election time counter is decreased by 1. Equation (2) shows the fact that the follower receives a heartbeat and reset the election time counter. Equation (3) shows that once the election time counter reaches zero, the follower transitions to the candidate state.

Denote $\{i, 0\}$ as the absorbing state. Since $i = 1, 2, \ldots, r$, there are r absorbing states. Denote the other states except state $\{i, 0\}$ in state space of $\{g(n), b(n)\}$ as transient states. There are t transient states, where $t = \sum_{i=1}^{r} K_i$. Let us order the states such that the first t states are transient and the last r states are absorbing. The transition matrix has the following canonical form:

$$\mathbf{P} = \begin{pmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{4}$$

where $\mathbf{Q}$ is a $t \times t$ matrix, $\mathbf{R}$ is a nonzero $t \times r$ matrix, $\mathbf{0}$ is an $r \times t$ zero matrix, and $\mathbf{I}$ is an $r \times r$ identity matrix. Specifically, the entry q_{ij} of $\mathbf{Q}$ is defined as the transition probability from transient state s_i to transient state s_j and the entry r_{mn} of $\mathbf{R}$ is defined as the transition probability from transient state s_m to absorbing state s_n.

When $K_r - K_1 \ll K_1$ or $b - a < \tau$, the election timeout value has only one. Thus, $r = 1$ and $t = K$, and then, the only non-null one-step transition probabilities in (1)–(3) can be simplified as follows:

$$\begin{cases} P\{k-1|k\} = p & (5) \\ P\{K|k\} = 1-p & (6) \\ P\{0|0\} = 1 & (7) \end{cases}$$

where $k \in \{1, \ldots, K\}$. Thus, the transition matrix $\mathbf{P}$ becomes a $(K+1) \times (K+1)$ matrix as

$$\mathbf{P} = \begin{pmatrix} 1-p & p & & \mathbf{0} \\ \vdots & & \ddots & \\ 1-p & \mathbf{0} & & p \\ \mathbf{0} & & & 1 \end{pmatrix}. \tag{8}$$

At the nth step, the transition matrix is $\mathbf{P}^n$ and the entry $p_{ij}^{(n)}$ of the matrix $\mathbf{P}^n$ is the probability of being in the state s_j from the state s_i.

For simplicity, suppose that election timeout counter has a fixed value K in the following analysis. It is straightforward to extend the analytical results to the case in which election timeout is a random value, i.e., $r > 1$.

B. Network Split Probability

Consider a network with one leader and $N-1$ followers. Note that when more than half of N nodes become candidates, the leader will not be qualified and thus the network will split. For a network split probability, we have the following proposition.

Proposition 1: For a network with N nodes, the transition matrix is given in (8). Then, the probability of a network split before the nth step is given by

$$p_n = 1 - \sum_{m=0}^{\lfloor \frac{N}{2} \rfloor} \binom{N-1}{m} \left(p_{1(K+1)}^{(n)}\right)^m \left(1 - p_{1(K+1)}^{(n)}\right)^{N-1-m} \tag{9}$$

where $p_{1(K+1)}^{(n)}$ is the $(1, K+1)$th entry of the matrix $\mathbf{P}^n$.

Proof: The entry $p_{i(K+1)}^{(n)}$ of the matrix $\mathbf{P}^n$ is the probability of being absorbed before the nth step, when the chain is started from state s_i. Thus, $p_{1(K+1)}^{(n)}$ is the probability of the follower starts from the initial state and transits to candidate state before the nth step.

Denote Y_n being the number of nodes which transit to candidate state before the nth step given that all nodes start from the initial state. Thus, $P\{Y_n = m\}$ is the probability that m nodes become candidates before the nth step. Suppose that all nodes are independent. Therefore, Y_n is a binomial distribution random variable with the form

$$P\{Y_n = m\} = \binom{N-1}{m} \left(p_{1(K+1)}^{(n)}\right)^m \left(1 - p_{1(K+1)}^{(n)}\right)^{N-1-m}. \tag{10}$$

Therefore, $P\{Y_n \geq \lfloor (N/2) \rfloor + 1\}$ is the probability that more than half of followers become candidates before the nth step. We have

$$p_n = P\left\{Y_n \geq \left\lfloor \frac{N}{2} \right\rfloor + 1\right\} = 1 - \sum_{m=0}^{\lfloor \frac{N}{2} \rfloor} P\{Y_n = m\}. \tag{11}$$

Thus, Proposition 1 is proved. ■

Based on Proposition 1, we derive the following properties of the network split probability.

Property 1: The transition probability at the nth step $p_{1(K+1)}^{(n)}$ has the following form:

$$p_{1(K+1)}^{(n)} = \begin{cases} 0, & \text{if } n < K \\ p^K, & \text{if } n = K \\ p_{1(K+1)}^{(n-1)} + \left(1 - p_{1(K+1)}^{(n-K-1)}\right)(1-p)p^K, & \text{if } n > K. \end{cases} \tag{12}$$

Proof: According to the transition matrix $\mathbf{P}$ shown in (8), $p_{1(K+1)}^{(n)}$ has the following forms when $n < K$ and $n = K$.

If $n < K$, then a follower cannot transit to the candidate state. Therefore

$$p_{1(K+1)}^{(n)} = 0. \tag{13}$$

If $n = K$, by calculating $\mathbf{P}^n$, we obtain

$$p_{1(K+1)}^{(n)} = p^K. \tag{14}$$

In the following analysis, we derive the form of $p_{1(K+1)}^{(n)}$ when $n > K$. According to $\mathbf{P}$ shown in (8), we have

$$p_{1(K+1)}^{(n)} = p_{1(K+1)}^{(n-1)} + p \cdot p_{1K}^{(n-1)} \tag{15}$$

$$p_{1i}^{(n)} = p \cdot p_{1(i-1)}^{(n-1)} \qquad \forall i \neq 1 \text{ and } i \neq K+1 \tag{16}$$

and

$$p_{11}^{(n)} = (1-p) \cdot \sum_{i=1}^{K} p_{1i}^{(n-1)}. \tag{17}$$

From (16), we obtain

$$p_{1K}^{(n)} = p^{K-1} \cdot p_{11}^{(n-K+1)}. \tag{18}$$

Since the row sums of transition matrix $\mathbf{P}$ are equal to one, we obtain

$$\sum_{i=1}^{K} p_{1i}^{(n-1)} = 1 - p_{1(K+1)}^{(n-1)}. \tag{19}$$

By combining (19) with (17), we have

$$p_{11}^{(n)} = (1-p)\left(1 - p_{1(K+1)}^{(n-1)}\right). \tag{20}$$

By combining (18) and (20) to (15), we obtain

$$p_{1(K+1)}^{(n)} = p_{1(K+1)}^{(n-1)} + \left(1 - p_{1(K+1)}^{(n-K-1)}\right)(1-p)p^K. \tag{21}$$

Thus, Property 1 is proved. ∎

Property 2: When the network size $N \to \infty$ and $p_{1(K+1)}^{(n)} \to 0$, the number of nodes transferring to candidate state before the nth step Y_n is approximated by the Poisson distribution random variable, and $Y_n \sim \mathcal{P}((N-1)p_{1(K+1)}^{(n)})$. The probability of the network split before the nth step is given by

$$p_n = 1 - \sum_{m=0}^{\lfloor \frac{N}{2} \rfloor} e^{(N-1)p_{1(K+1)}^{(n)}} \frac{\left((N-1)p_{1(K+1)}^{(n)}\right)^m}{m!}. \tag{22}$$

C. Average Number of Replies

Since Y_n is a binomial distribution random variable, given the network size N, the expected value of the number of candidates at the nth step is

$$N_C^{(n)} = (N-1)p_{1(K+1)}^{(n)}. \tag{23}$$

The expected value of the number of followers at the nth step is

$$N_f^{(n)} = (N-1)\left(1 - p_{1(K+1)}^{(n)}\right). \tag{24}$$

Thus, the average number of replies collected by the leader in the nth step is

$$E\left(N_{\text{reply}}\right) = N_f^{(n)}. \tag{25}$$

D. Expected Number of Received Heartbeats for Follower

Proposition 2: Given that the follower starts from the initial state, then the expected number of received heartbeats before it transfers to candidate state is n_{11}, where n_{11} is the first entry of matrix $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$.

Proof: According to the theorem of absorbing Markov chain in [20], the entry of matrix $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$ is the expected number of times the chain is in state s_j, given that it starts in state s_i. Detailed proof is given as follows.

Since

$$\mathbf{P} = \begin{pmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

we have

$$\mathbf{P}^n = \begin{pmatrix} \mathbf{Q}^n & \left(\mathbf{Q}^{n-1} + \mathbf{Q}^{n-2} + \cdots + \mathbf{Q} + \mathbf{I}\right)\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}. \tag{26}$$

Note that

$$(\mathbf{I} - \mathbf{Q})\left(\mathbf{I} + \mathbf{Q} + \mathbf{Q}^2 + \cdots + \mathbf{Q}^{n-1}\right) = \mathbf{I} - \mathbf{Q}^n. \tag{27}$$

In the Appendix, we prove that the absolute values of the eigenvalues of $\mathbf{Q}$ are all strictly less 1. Thus, $\mathbf{I} - \mathbf{Q}$ is invertible. Define $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$. Multiplying both sides of (27) by $\mathbf{N}$ gives

$$\mathbf{I} + \mathbf{Q} + \mathbf{Q}^2 + \cdots + \mathbf{Q}^{n-1} = \mathbf{N}\left(\mathbf{I} - \mathbf{Q}^n\right).$$

Thus, we have

$$\mathbf{P}^n = \begin{pmatrix} \mathbf{Q}^n & (\mathbf{I} - \mathbf{Q})^{-1}(\mathbf{I} - \mathbf{Q}^n)\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}. \tag{28}$$

In the Appendix, we also prove that when n goes to infinity, $\mathbf{Q}^n$ goes to $\mathbf{0}$. Therefore, when n goes to infinity

$$\mathbf{N} = \mathbf{I} + \mathbf{Q} + \mathbf{Q}^2 + \cdots \tag{29}$$

and thus

$$n_{ij} = q_{ij}^{(0)} + q_{ij}^{(1)} + q_{ij}^{(2)} + \cdots \tag{30}$$

where $q_{ij}^{(k)}$ is defined as the (i,j)th entry of $\mathbf{Q}^k$.

Let $X^{(k)}$ be a random variable, and

$$X^{(k)} = \begin{cases} 1 & \text{if the chain is in state } s_j \text{ after } k \text{ steps} \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

According to $\mathbf{P}^k$, we have

$$P\Big(X^{(k)}=1\Big)=p_{ij}^{(k)}=q_{ij}^{(k)},\qquad i,j=1,\ldots,K \tag{32}$$

and

$$P\Big(X^{(k)}=0\Big)=1-q_{ij}^{(k)}. \tag{33}$$

These equations hold when $k=0$ since $\mathbf{Q}^0=\mathbf{I}$. Therefore, $E(X^{(k)})=q_{ij}^{(k)}$.

The expected number of times the chain is in state s_j in the first n steps, given that it starts from state s_i

$$E\Big(X^{(0)}+X^{(1)}+\cdots+X^{(n)}\Big)=q_{ij}^{(0)}+q_{ij}^{(1)}+\cdots+q_{ij}^{(n)}. \tag{34}$$

Letting n goes to infinity, we have

$$E\Big(X^{(0)}+X^{(1)}+\cdots\Big)=q_{ij}^{(0)}+q_{ij}^{(1)}+\cdots \tag{35}$$

By comparing (30) with (35), we obtain that the entry of matrix $\mathbf{N}$ is the expected number of times the chain is in state s_j, given that it starts from state s_i. Denoting s_1 as the initial state, n_{11} is the expected number of received heartbeats before a follower transfers to candidate state when it starts from the initial state. ∎

E. Time to Transition to Candidate

Proposition 3: Suppose that a follower starts from the initial state, the expected time for this follower to transition to candidate state is given as

$$t_c=\sum_{j=1}^{t}n_{1j} \tag{36}$$

where n_{1j} is the $(1,j)$th entry of matrix $\mathbf{N}$.

Proof: According to Proposition 2, the entry n_{ij} of $\mathbf{N}$ gives the expected number of times that the follower is in the transient state s_j if it is started from the transient state s_i. Therefore, the sum of the entries in the ith row of $\mathbf{N}$ is the expected times in any of the transient states for a given starting state s_i, i.e., the expected time required before the follower transfers to the candidate state. Denoting s_1 as the initial state, we obtain the proposition. ∎

Property 3: The average interval of the received heartbeats for a follower in one term is given by

$$t_{in}=\frac{\sum_{j=1}^{t}n_{1j}}{n_{11}}. \tag{37}$$

F. Time to Election New Leader

When the leader crashes or network split happens, the node that times out first will get its own vote plus votes from the other available nodes. If the candidate gets votes from a majority of the nodes, it will become a leader. If the first candidate's vote request fails to reach a majority of the nodes before they time out due to packet loss or network delays, split votes will happen. When split votes happen, no candidate wins the election. Then, all nodes have to wait for another period of election timeout before starting a new election.

In [21, Sec. IX], the analysis of split vote rate with fixed latency was presented. Unlike [21], we investigate the impacts of packet loss on the election performance in this part. Specifically, we try to analyze the upper bound of time that a successful leader election will take.

Denote the time to detect the failed leader and re-elect a new leader as T_e. We have $T_e=E_t\times N_e$, where E_t is the timeout value uniformly chosen from the range $[a,b]$, and N_e are the number of elections needed to detect the failed leader and re-elect a new leader. The upper bound of the time to detect the failed leader or restart next election is the maximum election timeout value.

In the normal condition, it takes one round of election to re-elect a new leader. When split votes happen, more elections are needed to elect a new leader. Let s be the number of available nodes. Suppose that the average time between failures of a single node is much larger than election timeout so that s is the constant during the period of election. Denote N_v as the number of votes obtained by the first candidate in one round of election. The probability of the first candidate's success in one round of election is $P(N_v>\lfloor(N/2)\rfloor)$, is given by

$$P\left(N_v>\left\lfloor\frac{N}{2}\right\rfloor\right)=\sum_{k=\lfloor\frac{N}{2}\rfloor}^{s-1}\binom{s-1}{k}(1-p)^{2k}\times\left(1-(1-p)^2\right)^{s-1-k}. \tag{38}$$

It is highly unlikely that the other nodes those time out later win the election because less available votes left. Suppose that the node that times out later are unable to collect majority votes. Thus, $P(N_v>\lfloor(N/2)\rfloor)$ is approximately equal to the probability of success in one round of election. It is should be pointed out the $P(N_v>\lfloor(N/2)\rfloor)$ is smaller than the probability of success in one round of election. Therefore, the expected value of N_e is give as

$$E(N_e)<\sum_{n=1}^{\infty}n\left(1-P\left(N_v>\left\lfloor\frac{N}{2}\right\rfloor\right)\right)^{n-1}P\left(N_v>\left\lfloor\frac{N}{2}\right\rfloor\right)=\frac{1}{P\left(N_v>\left\lfloor\frac{N}{2}\right\rfloor\right)}. \tag{39}$$

Therefore, we have

$$E(T_e)<\frac{E_{t_{\max}}}{P\left(N_v>\left\lfloor\frac{N}{2}\right\rfloor\right)} \tag{40}$$

where $E_{t_{\max}}$ is the maximum timeout value.

IV. Simulation Results

In this section, we first validate the efficiency of the analysis model, and then, we investigate the impacts of the parameters (such as packet loss rate, election timeout period, and network size) on availability and network split probability in more details.

To validate the model, we have compared its results with those obtained with the discrete event simulator according to [13]. In simulations, each message was assigned a latency chosen randomly from the uniform range of [0.5, 10] ms, and the interval between two heartbeats $\tau=50$ ms. Furthermore,

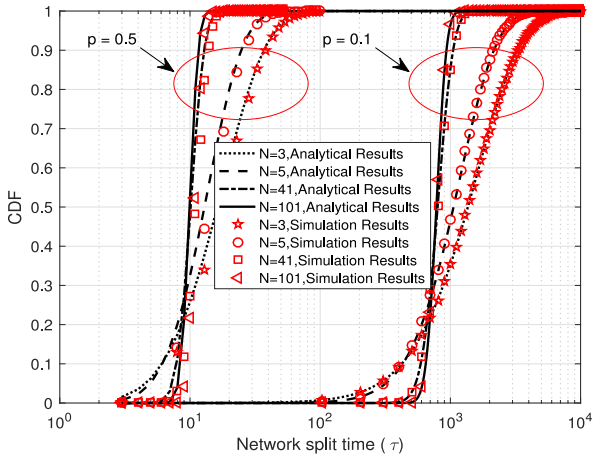


Fig. 2. CDF of network split time, $K = 3$.

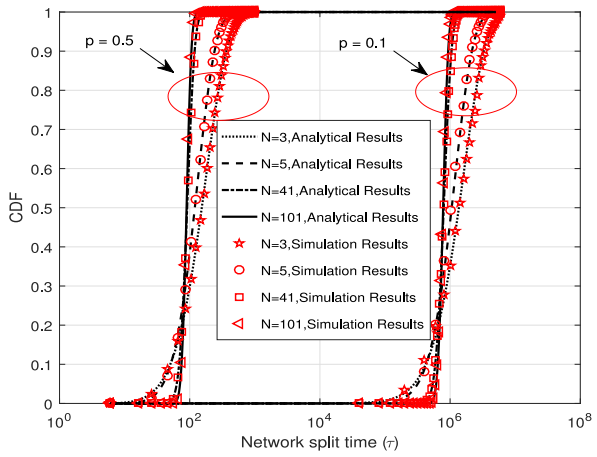


Fig. 3. CDF of network split time, $K = 6$.

CPU time should be short relative to network latency, and the speed of writing to disk does not play a significant role anyhow. To derive the statistical results, we run simulated program 10 000 times and then record the network split time of each run. Each run follows the following steps.

Initialization: Initialize simulation time $t = 0$; initialize the list of the followers; initialize the number of candidates to be zero.

While (the number of candidates is less than half of nodes) then do the following.

Step 1: For each follower i, determine whether the follower receives a heartbeat according to the given packet loss rate. If the follower receives a heartbeat, then reset the election time counter. If the follower fails to receive a heartbeat from the current leader and its election time counter is decreased by 1. If the election time counter reaches zero, then denote follower i as a candidate and delete follower i from the list of followers.

Step 2: Count up the number of candidates. If the number of candidates is larger than half of the nodes, then record time t as network split time. Otherwise, update $t = t + \tau$, go back to step 1.

Figs. 2 and 3 plot the cumulative distribution function (CDF) of the number of heartbeats for a network to split.

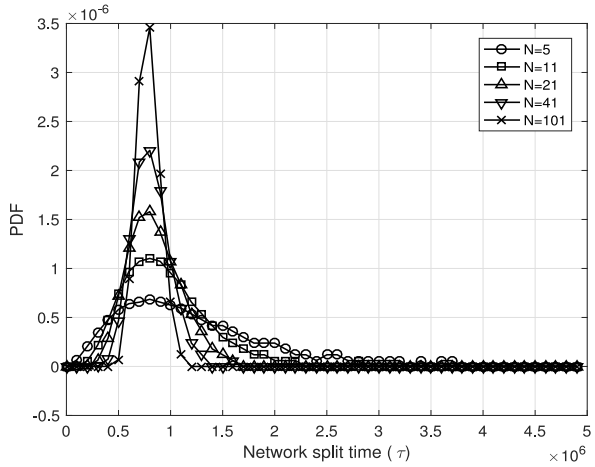


Fig. 4. PDF of network split time $K = 6$ and $p = 0.1$.

Each CDF summarizes 10 000 simulated trials. The analytical results are calculated based on (9). Given the value of election timeout K, network size N, and packet loss rate p, Figs. 2 and 3 show that analytical results match well with the simulation results. Therefore, one can detect when the network is abnormal by comparing with the reference value given by the analytical model.

From Figs. 2 and 3, we observe that: 1) network split probability highly depends on the packet loss rate p and the election timeout value K. As one expected, as p decreases or K increases, for the same size of network, the probability of network split before the nth step p_n decreases and 2) as the network size increases, the CDF curves become steeper. Given p and K, when n is small, p_n decreases with when N increases. This can be observed from p_n in (9). To further understand the second point, we show the PDF of network split time. Fig. 4 shows the PDF of network split time for different network sizes. We observe that large network has smaller split probability than the one in small network at the beginning of running time. When running time increases over certain point, the network split probability increases with the size of network. Because the follower's probability of transition to candidate is small at the beginning of running time, the probability that more than half of all nodes become candidates is lower for larger networks. When the follower's probability of transition to candidate gets greater with the running time, the network split probability increases with the network size.

The analytical model given in Section III is convenient to determine the probability of network split time. Based on the proposed model, we exploit the impacts of the parameters (such as packet loss rate, election timeout period, and network size) on availability and network split performance in more details. Figs. 5 and 6 describe the expectation and variance of network split time for different network sizes, respectively. The results show that the variance of network split time for large network is smaller than that for small network. However, the expectation of network split time for larger network is very close to that for small network. Therefore, a larger network has better stability in terms of network split time.

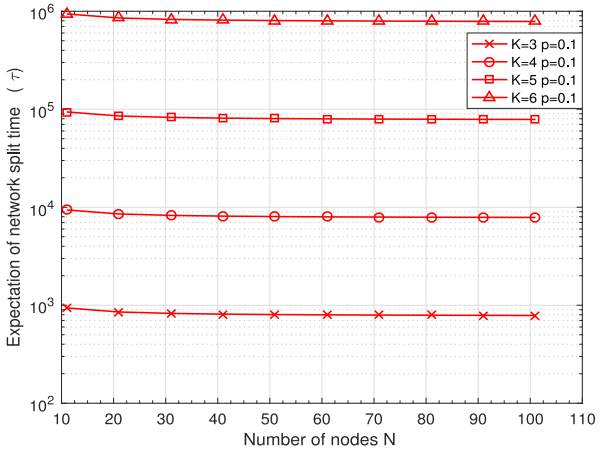


Fig. 5. Expectation of network split time given different network sizes.

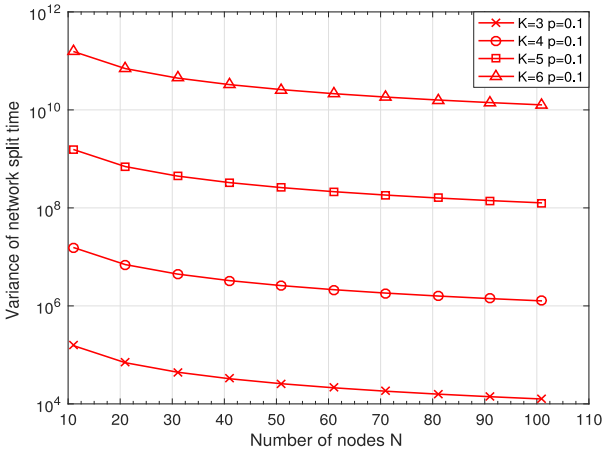


Fig. 6. Variance of network split time given different network sizes.

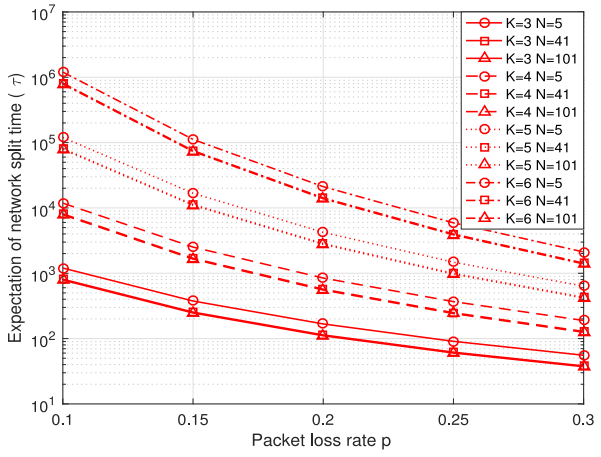


Fig. 7. Expectation of network split time given different packet loss rates.

Figs. 7 and 8 show the expectation and variance of network split time in different packet loss rates, respectively. As Fig. 7 shows, given the packet loss rate $p = 0.1$ and $N = 5$, the expectation of split time is about 1000 and 10 000 when $K = 3$ and $K = 4$, respectively. This result means that the network's stable time is prolonged ten times by adding one more heartbeat. Given the packet loss rate $p = 0.3$ and $N = 5$, the

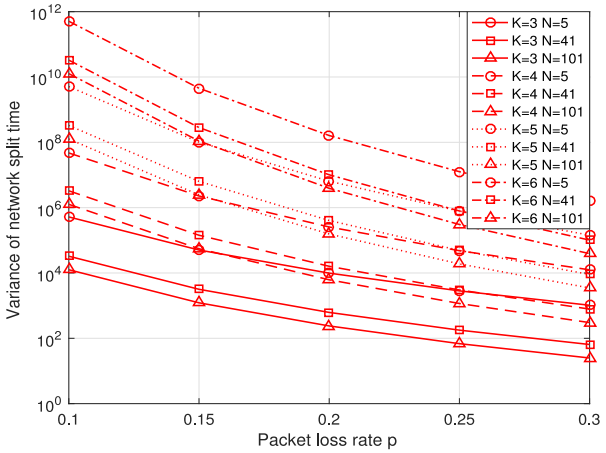


Fig. 8. Variance of network split time given different packet loss rates.

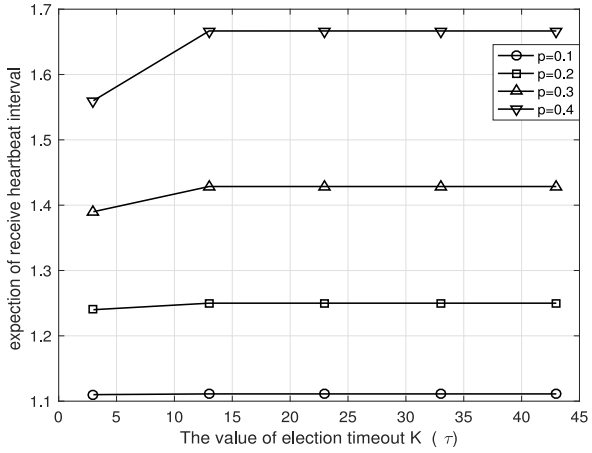


Fig. 9. Average interval to receive a heartbeat for a follower.

expectation of split time is about 50 and 110 when $K = 3$ and $K = 4$, respectively. The expectation of network split time is prolonged one time by adding one more heartbeat. Increasing election timeout is helpful to lower the network split probability caused by packet loss, especially under smaller packet loss rate.

Fig. 9 plots the average interval of receiving one heartbeat for one follower per term. The results show that packet loss rate has significant impact on receiver heartbeat interval. On the other hand, election timeout period has insignificant impact on receiver heartbeat interval. Especially, the interval of receiving one heartbeat increases with election timeout period but goes to a constant.

However, prolonging the election timeouts would result in longer time to detect a leader failure and re-elect the new leader in failure cases rises. Consider that there are $N-1$ nodes available (since the current leader is failed). Figs. 10 and 11 show the probability of the number of elections given different packet loss rates when network size $N = 3$ and $N = 101$, respectively.

The results show that 80% elections complete in the first round of election given $p = 0.1$ and $N = 3$, and nearly 100% elections complete in the first round of election given $p = 0.1$ and $N = 101$. Given $p = 0.5$, 25% elections complete in the

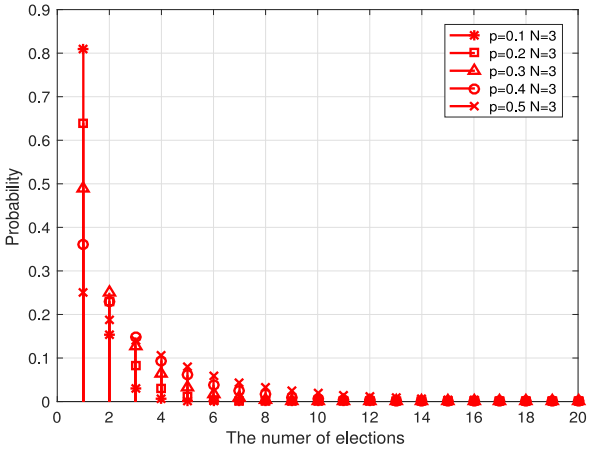


Fig. 10. Probability of the number of elections, $N = 3$.

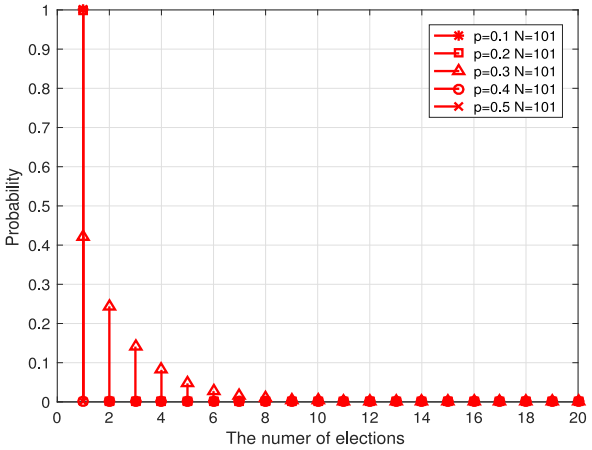


Fig. 11. Probability of the number of elections, $N = 101$.

first round of election when $N = 3$ and nearly 0% elections complete in the first round of election when $N = 101$.

Fig. 12 plots the expectation of the number of elections in different packet loss rates. As Fig. 12 shows, when $p > 0.3$, the expected value of N_e of larger networks increases more rapidly than that of smaller networks. The results mean that: 1) when p is small (i.e., $p < 0.2$), the election performance of larger networks does not so highly depends on the packet loss rate p as that of smaller networks. The average N_e of larger networks is smaller than that of smaller networks and 2) as p increases, the election performance is degraded tremendously, especially in the lager network. The availability of larger networks is much poorer than that of the smaller network under larger packet loss rate.

V. Discussion

A. Extended Model

This paper tries to model and analyze the network split process, where one follower node transfers to a candidate is modeled as an absorbing Markov chain and the packet loss probability p for all the nodes is a constant value, for the sake of simplicity. From a realistic point view, this paper can be extended in the following ways.

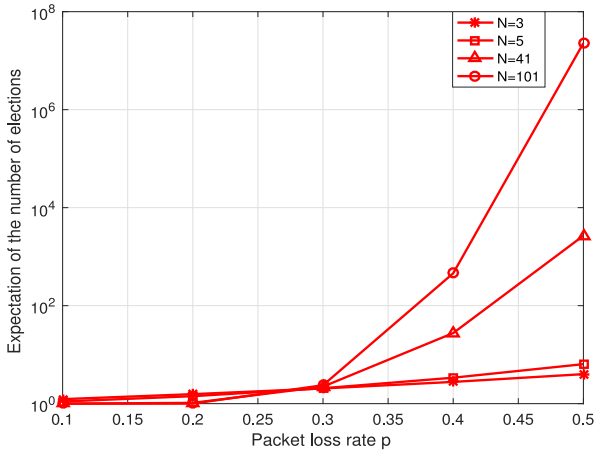


Fig. 12. Expectation of the number of elections.

In a network, the packet loss probability of each node may be different, and some node may crash. For example, a leader might be more prone to failures or packet loss because of an enhanced workload, congested uplinks, etc. To address those issues, we can extend the proposed model to support biased per-node packet loss probability. For follower s, define the packet loss probability of P2P link between leader and follower s as p_s. Thus, the only non-null one-step transition probabilities of Markov chain are

$$\begin{cases} P\{i, k_i - 1 | i, k_i\} = p_s & (41) \\ P\{i, K_i | j, k_j\} = (1 - p_s)/r & (42) \\ P\{i, 0 | i, 0\} = 1 & (43) \end{cases}$$

where $i, j = 1, 2, \ldots, r$ and $k_i \in \{1, \ldots, K_i\}$. Equation (41) relies on the fact that follower s fails to receive a heartbeat from the current leader and its election time counter is decreased by 1. Equation (42) shows the fact that follower s receives a heartbeat and reset the election time counter. Equation (43) shows that once the election time counter reaches zero, follower s transitions to the candidate state.

For node crashing, take the leader failure as an example. Let X be a random variable, where

$$X = \begin{cases} 1 & \text{if the leader fails} \\ 0 & \text{otherwise.} \end{cases} \quad (44)$$

Then, the packet loss probability of the P2P link between the leader and follower s can be modified as

$$p_s = \begin{cases} p_{s_0} & X = 0 \\ 1 & X = 1 \end{cases} \quad (45)$$

where p_{s_0} is the packet loss probability of the P2P link between the leader and follower s in the normal condition. When the current leader crashes, p_s is equal to 1. Thus, the follower transits to the candidate state within K steps according to the extended model in (41)–(43), where K is the maximum number of heartbeats for an election counter to timeout.

The detailed analysis of network split time would be more complex based on the extended model. We would like to work on it for our future work.

B. System Availability and Consensus Efficiency

This paper focuses on the network splitting time, which is only one specific metric of the blockchain network. Several general metrics, such as the system availability and the consensus efficiency, are more interested.

The system availability can be given as follows:

$$\eta = \frac{T_{\text{normal}}}{T_{\text{normal}} + T_e} \tag{46}$$

where T_{normal} is the period of normal operation (i.e., more than half of nodes are under the control of the current leader) and T_e is the period including detection a failed leader and re-election a new leader.

To evaluate the system availability, we have to analyze T_{normal} and T_e. Since the failure of the current leader and communication interruption caused by packet loss are the main reasons of unavailability of the network, T_{normal} not only depends on the time between failures of a single node but also the stable time that the network experiences before network split caused by packet loss. Typical time between failures of a single node is several months or more [13], and the average stable time is about 16 h ($p = 0.1$, election timeout is 300 ms, the interval between heartbeats is 50 ms). Suppose there is no failure occurs, T_{normal} can be derived by the network split probability that is presented in Section III. However, the analysis of T_{normal} would be more complexly when considering the failure of leader.

It should be pointed out that the speed of the transaction confirmation varies with the number of available nodes[3] within a period of normal operation, i.e., when the number of available nodes is large, the leader is likely to receive the feedbacks from a majority of followers more quickly.

Hence, the consensus efficiency which can be defined as the average speed of transaction confirmation is a more important metric of the blockchain network. Based on the proposed model, we can obtain the average number of replies collected by the leader in the given time (see Section III-C), which is helpful to analyze the speed of transaction confirmation since the confirmation on transactions requires the agreements from a majority of nodes in a Raft consensus algorithm.

We would like to work on the analysis about the system availability and the consensus efficiency in our future work.

VI. Conclusion

Consensus is the basis for blockchains. Raft is a well adopted consensus algorithm for private blockchains. In this paper, an analytical model for Raft consensus algorithm is proposed. The analytical model given in this paper is very convenient to obtain the network split probability and several performance metrics, which provides guidance on how to determinate the parameters. Furthermore, the analytical model is able to monitored the condition of network and detect the abnormal condition by providing reference value of network performance. The simulation results match the analytical results well. Using the proposed model, we have shown the parameters' (such as packet loss rate, election timeout, and network size) impacts on availability. Increasing election timeout is helpful to lower the network split probability caused by packet loss. However, the availability of larger networks is much poorer than that of the smaller network under larger packet loss rate since it takes much longer time to elect a leader in larger network. We also observe that larger network has smaller split probability than the one in small network at the beginning of running time and more focused splitting time.

[3]Here, the available node means the node that is under the control of current leader.

Appendix

1) The absolute values of eigenvalue of $\mathbf{Q}$ are all strictly less than 1.

Based on the Gershgorin circle theorem [22], for $K \times K$ matrix $\mathbf{Q}$, all eigenvalue satisfy

$$|a_k - q_{ii}| \le \sum_{j \ne i} |q_{ij}|, \quad i, k = 1, 2, \ldots, K \tag{47}$$

where q_{ij} is the (i, j)th element of $\mathbf{Q}$.

Due to $\sum_{j=1}^{K} q_{ij} \le 1$ and $q_{ij} \ge 0$

$$\begin{aligned} -\sum_{j \ne i} q_{ij} \le a_k - q_{ii} \le \sum_{j \ne i} q_{ij} \\ -\sum_{j=i}^{K} q_{ij} + 2q_{ii} \le a_k \le \sum_{j=i}^{K} q_{ij} \qquad \forall i, k = 1, 2, \ldots, K. \end{aligned} \tag{48}$$

Since $\mathbf{R}$ in (12) is not zero matrix, there exists one row of $\mathbf{Q}$ such that $\sum_{j=1}^{K} q_{ij} < 1$. Thus

$$-1 < a_k < 1, \quad k = 1, 2, \ldots, K. \tag{49}$$

That means $|a_k| < 1$.

2) Proof of $\lim_{n \to \infty} \mathbf{Q}^n = \mathbf{0}$. Note that $\mathbf{Q} = \mathbf{U}\Lambda\mathbf{U}^{\mathrm{H}}$, where $\mathbf{U}^{\mathrm{H}}\mathbf{U} = \mathbf{I}$ and

$$\Lambda = \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_t \end{bmatrix}. \tag{50}$$

Since $|a_i| < 1$, we have $\lim_{n \to \infty} \Lambda^n = \mathbf{0}$.

Furthermore, since $\mathbf{Q}^n = \mathbf{U}\Lambda^n\mathbf{U}^{\mathrm{H}}$, $\therefore \lim_{n \to \infty} \mathbf{Q}^n = \mathbf{0}$.

References

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] Y. Yuan and F.-Y. Wang, "Blockchain and cryptocurrencies: Model, techniques, and applications," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 9, pp. 1421–1428, Sep. 2018.

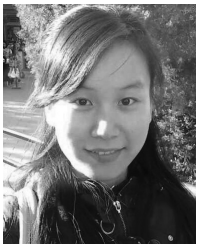
[3] G. W. Peters, E. Panayi, and A. Chapelle. (2015). *Trends in Crypto-Currencies and Blockchain Technologies: A Monetary Theory and Regulation Perspective*. [Online]. Available: http://dx.doi.org/10.2139/ssrn.2646618

[4] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, 2016, pp. 839–858.

[5] B. W. Akins, J. L. Chapman, and J. M. Gordon. (2013). *A Whole New World: Income Tax Considerations of the Bitcoin Economy*. [Online]. Available: https://ssrn.com/abstract=2394738

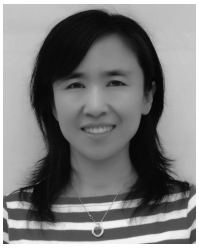
[6] Y. Zhang and J. Wen, "An IoT electric business model based on the protocol of bitcoin," in *Proc. 18th Int. Conf. Intell. Next Gener. Netw. (ICIN)*, Paris, France, 2015, pp. 184–191.

[7] M. Sharples and J. Domingue, "The blockchain and kudos: A distributed system for educational record, reputation and reward," in *Proc. 11th Eur. Conf. Technol. Enhanced Learn. (EC-TEL)*, Lyon, France, 2015, pp. 490–496.
[8] S. King and S. Nadal. (Aug. 2012). *PPCoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf
[9] Nxtwiki. (2015). *Whitepaper:Nxt*. [Online]. Available: http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt
[10] *Delegated Proof-of-Stake (DPOS)*. (2014). [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/
[11] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. Symp. Oper. Syst. Design Implement.*, New Orleans, LA, USA, 1999, pp. 173–186.
[12] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998.
[13] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, Philadelphia, PA, USA, Oct. 2013, pp. 305–320.
[14] *Hyperledger Project*. (2015). [Online]. Available: https://www.hyperledger.org/
[15] *Consortium Chain Development*. (2018). [Online]. Available: https://github.com/ethereum/wiki/wiki/Consortium-Chain-Development
[16] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Banff, AB, Canada, 2017, pp. 2567–2572.
[17] The ZILLIQA Team. (2017). *The ZILLIQA Technical Whitepaper*. [Online]. Available: https://docs.zilliqa.com/whitepaper.pdf
[18] I. Moraru, D. G. Andersen, and M. Kaminsky, "There is more consensus in egalitarian parliaments," in *Proc. ACM Symp. Oper. Syst. Principles (SOSP)*, Farmington, PA, USA, 2013, pp. 358–372.
[19] H. Howard, M. Schwarzkopf, A. Madhavapeddy, and J. Crowcroft, "Raft refloated: Do we have consensus?" *Oper. Syst. Rev.*, vol. 49, no. 1, pp. 12–21, 2015.
[20] *Markov Chains*. (2017). [Online]. Available: http://www.academia.edu/7549558/Chapter-11-Markov-Chains
[21] D. Ongaro, "Consensus: Bridging theory and practice," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2014.
[22] *Gershgorin Circle Theorem*. (2018). [Online]. Available: https://en.wikipedia.org/wiki/Gershgorin-circle-theorem



Dongyan Huang received the B.S. degree in electronics and information engineering and the M.S. degree in communication and information systems from the Guilin University of Electronic Technology, Guilin, China, in 2006 and 2009, respectively, and the Ph.D. degree in communication and information systems from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015.

After that, she joined the School of Information and Communication, Guilin University of Electronic Technology. She is currently a Post-Doctoral Fellow with Shenzhen University, Shenzhen, China. Her current research interests include performance analysis for blockchains, consensus algorithm for blockchains, and machine learning.



Xiaoli Ma (F'16) received the B.S. degree in automatic control from Tsinghua University, Beijing, China, in 1998, the M.S. degree in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in 2000, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 2003.

She joined the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL, USA, where she served as an Assistant Professor in 2005. Since 2006, she has been with the School of Electrical and Computer Engineering, Georgia Tech, Atlanta, GA, USA, where she is currently a Professor. Her current research interests include wireless networking and communications, including network performance analysis, transceiver designs for wireless time- and frequency-selective channels, channel estimation and equalization algorithms, carrier frequency synchronization for OFDM systems, Internet of Things, and performance analysis for blockchains.

Dr. Ma was a recipient of the Lockheed Martin Aeronautics Company Dean's Award for Teaching Excellence by the College of Engineering in 2009 and the Outstanding Junior Faculty Award by the School of Electrical and Computer Engineering, Georgia Tech in 2010. She served as a Senior Area Editor for IEEE SIGNAL PROCESSING LETTERS from 2014 to 2017 and *Digital Signal Processing* (Elsevier) from 2012 to 2016, and an Associate Editor for IEEE SIGNAL PROCESSING LETTERS from 2007 to 2009 and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2008 to 2013.

Shengli Zhang (SM'17) received the B.Eng. degree in electronic engineering and the M.Eng. degree in communication and information engineering from the University of Science and Technology of China, Hefei, China, in 2002 and 2005, respectively, and the Ph.D. degree in information engineering from the Chinese University of Hong Kong, Hong Kong, in 2008.

He joined the Communication Engineering Department, Shenzhen University, Shenzhen, China, where he is currently a Full Professor. From 2014 to 2015, he was a Visiting Associate Professor with Stanford University, Stanford, CA, USA. He is the pioneer of physical-layer network coding. He has published over 20 IEEE top journal papers and ACM top conference papers, including the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON COMMUNICATIONS, and ACM Mobicom. His current research interests include physical layer network coding, interference cancelation, cooperative wireless networks, and blockchain.

Dr. Zhang severed as an Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE WIRELESS COMMUNICATIONS LETTERS, and *IET Communications*. He has also severed as a TPC member in several IEEE conferences.