Fault-Tolerant Consensus in Wireless Blockchain System

Yifei Zou, Yufan Li, Dongxiao Yu, Feng Li, Yanwei Zheng(✉), and Yan Zhang

Institute of Intelligent Computing, School of Computer Science and Technology, Shandong University, Qingdao 266237, People's Republic of China
{yfzou,dxyu,fli,zhengyw}@sdu.edu.cn

Abstract. In recent years, the blockchain system on multi-agents has drawn much attention from both of the broad academic research and industries due to its popular applications, such as Bitcoin [15]. Meanwhile, it is also an important part in city/industrial Internet-of-Things and artificial intelligence areas. In this paper, we investigate the fault-tolerant consensus problem in wireless blockchain system. Considering that the multi-agents in reality are inevitable to break down or send wrong messages because of some uncertain errors, in this paper, we propose a fault-tolerant consensus protocol, which can achieve consensus over multi-agents in wireless blockchain network within $O((f+1)\log n)$ time steps with high probability. f is the upper bound of invalid agents, and n is the number of agents in the blockchain system. Rigorous theoretical analysis and extensive simulations are given to show the efficiency of our algorithm.

Keywords: Consensus in multi-agents · Wireless blockchain system · Fault-tolerant · Internet-of-Things

1 Introduction

In the past decades, with an enormous demand on the safety and convenience in distributed systems, it is highly possible for blockchain technology to become ubiquitous in cities and industries. Blockchain technique is envisioned as the core mechanism of publicized cryptocurrencies, and it can be used in a variety of applications, such as the Internet of Things (IoT), which allows direct communication and interaction among multi-agents via the Internet. As the number of multi-agents keeps increasing, traditional IoT applications no longer suit well for the data integrity, security, and robustness. Fortunately, blockchain provides

This work was partially supported by National Key R&D Program of China under Grant 2020YFB1005900, the Blockchain Core Technology Strategic Research Program of Ministry of Education of China (No. 2020KJ010301), and NSFC (No. 61971269, No. 620722278, No. 61832012).

Z. Liu et al. (Eds.): WASA 2021, LNCS 12937, pp. 133–146, 2021.
https://doi.org/10.1007/978-3-030-85928-2_11

a practical solution to overcome the limitations of traditional IoT applications. Briefly, the blockchain technologies consist of P2P network protocol, distributed consensus protocol, cryptography technology, account and storage model. Among all technologies above, the distributed consensus protocol is the cornerstone of blockchain. Agents in the blockchain network exchange their messages and make local decisions by executing distributed consensus protocol, which guarantee the consistency of distributed ledger. And the distributed consensus protocol enables blockchains decentralization indeed. Moreover, the blockchain consensus protocol has great influences on the performance of a blockchain system, including throughput, scalability, and fault tolerance. Therefore, an efficient and effective fault-tolerant consensus protocol is of great significance.

Generally, there are two kinds of consensus protocols widely used in blockchains: the Proof-of-X (PoX) consensus protocols (e.g., Proof-of-Work (PoW), Proof-of-Stake (PoS) [2]), and the Byzantine Fault Tolerant (BFT)-based consensus protocols (e.g., Practical Byzantine Fault Tolerance (PBFT) [4] and HotStuff [21]). Nodes in the blockchain who execute PoX consensus protocols need to prove that they are more competent for appending work than other nodes. Nodes in the blockchain network who execute BFT-based consensus protocols need to exchange the results of verifying a new block before voting on the final decision. However, each of them has its pros and cons. Most of PoX consensus protocols have high scalability but low efficiency, which are more suitable in public blockchain. Whereas, BFT-based consensus protocols have high efficiency but are not scalable, which are mainly used in consortium and private blockchain.

Main Challenges. Considering both the IoT environment and multi-agents constraints, there are four mainly challenges to design an efficient fault-tolerant consensus protocol in blockchain system. Firstly, in order to meet the requirements of low-power multi-agents in the IoT, it is necessary for the consensus protocol to have the characteristics of high throughput, high energy efficiency and insensitive to device mobility. Secondly, multi-agents can access the same wireless channel, so the transmission between nodes can affect each other. One is that the success rate of transmission is uncertain, and the other is that the channel is unstable, i.e., it is difficult to guarantee the long-term effective transmission of information. Both of them make it difficult to achieve a consensus in a wireless environment. Thirdly, the severe state fork problem which occurs when there are multiple valid blocks appearing at the same time is also a challenge for the design of the fault-tolerant consensus protocol. The last challenge is that there are some fault nodes in the wireless blockchain system which may cause the whole system to fail to work or write invalid blocks in the blockchain. As a result, these fault nodes may prevent nodes from achieving a consensus.

To tackle the above challenges, we propose a fault-tolerant Proof-of-Communication (FTPoC) consensus protocol in wireless blockchain network in this paper. Specifically, all of miners, i.e., multi-agents of IoT participate in multiple leader competition process via communication. Each time a leader is

elected, this leader will propose a new block, broadcast and record it, and other miners will record the block and count the number of times the block has been received until $f + 1$ the same blocks are recorded. We adopt an efficient leader competition scheme, which makes full use of the signals received in the wireless channel to elect multiple leaders so as to meet the requirements of the multi-agents and environment of IoT and reduce the influence of nodes on each other. We adopt voting mechanism which guarantees that one and only one valid block is written in the blockchain in order to solve the fork problem and the device failure problem.

Our Contributions. In this paper, we present an efficient fault-tolerant consensus protocol for the multi-agents in wireless blockchain system. All multi-agents can achieve a consensus within $O((f + 1) \log n)$ time steps with high probability[1] by executing our protocol when there are at most f fault agents. The main contributions of our work are summarized as follows:

- We consider the fault-tolerant blockchain consensus protocol in multi-agents wireless networks. Therefore, our wireless blockchain consensus protocol is closer to reality and makes the wireless research in blockchain system more completed.
- Under the faulty assumption in the wireless network, we show how to design a scheme based on fault-tolerant PoC in the context of wireless network to achieve a consensus in blockchain through a wireless competition scheme and voting mechanism, which can provide a completely new perspective for the design of wireless blockchain protocol.

2 Related Work

In 1980, the classical consensus in distributed system was proposed by Pease *et al.* in [17], which theoretically supports the blockchain consensus protocol. Their work mainly considers how non-fault nodes achieve a consensus by point-to-point communication when there are fault nodes involved. In 2008, PoW was the first consensus algorithm in blockchain used by Bitcoin [15]. Nowadays, PoW consensus protocol is also one of the most widely used consensus protocols in public blockchain. However, the mining behavior based on PoW also causes a lot of waste of resources [16]. In addition, one of the reasons for the scalability limitation and energy efficiency of PoW consensus protocol is fork problem. In 2012, Peercoin is released by S. King [12], and the concept of PoS was first proposed to solve the limitations of PoW. In the PoS mechanism, the miners no longer consume a lot of energy to calculate hash function, but mortgage their own assets to become candidates and compete for the power to propose blocks. The miners do not consume any resources [3] (called virtual mining), so PoS does not consume energy and hardware equipment like PoW, which avoids

[1] With the probability of $1 - n^{-c}$ for some constant $c > 1$ and w.h.p. for short.

high energy consumption. Although the PoS has its unique advantages, there are still some problems. The system can not guarantee security and fairness well because of centralization risk [10], and weak randomness [11]. In addition, there are other classes of PoX, e.g., Proof-of-Activity [1] and Proof-of-Space [9]. Whereas, they have their own disadvantages, so they are also not suitable for IoT agents. BFT-based consensus protocols are highly efficient and can solve device failure problem. Whereas,because of the communication complexity [18], they are not scalable, which makes them also unsuitable for IoT edge agents. Different from the previous works mentioned above, the consensus problem in our work is considered in the context of wireless network with the faulty assumption, which is more realistic. Besides, the time complexity of our solution on consensus problem is very close to the optimal solution. It is believed that with our work, many problems and applications in reality [5–7,13,14,19,31–34] can get a new solution.

3 Model and Problem Definition

We assume that there is a wireless blockchain network in a two-dimensional Euclidean plane, where each miner can communicate with others by a wireless channel. In each time slot, every miner can listen or transmit but cannot do both. The miners that can work normally are called as the normal miners. In addition, there are some fault miners who may not work or send some wrong messages because of unknown errors, e.g.,calculation error.

In order to get closer to reality, SINR model [22–25,29] is adopted in our model to depict the message reception in this paper. For a miner v, let $Signal(v)$ denote the strength of the signal received by v, and $SINR(u, v)$ denote the SINR ratio of miner v for the transmission from miner u. In this model, we have

$$Signal(v) = \sum_{w \in S} P_w \cdot d(w, v)^{-\alpha} + \mathcal{N},$$

$$SINR(u, v) = \frac{P_u \cdot d(u, v)^{-\alpha}}{\sum_{w \in S \setminus \{u\}} P_w \cdot d(w, v)^{-\alpha} + \mathcal{N}}.$$

In the above two equations, P_w denotes the transmission power of miner w, $d(w, v)$ is the Euclidean distance between miner w and miner v, S denotes the set of miners who transmit in the current slot, $\mathcal{N}$ is the background noise, $\alpha \in (2, 6]$ is the path-loss exponent. In this model, a message sent by miner u is successfully decoded by miner v if and only if $SINR(u, v) \geq \beta$, where threshold $\beta > 1$ depends on hardware. We use N to denote a close upper bound of background noise $\mathcal{N}$. They are so close that for any miner the accumulation of background noise and signals from transmissions is larger than N, when there are transmissions in network. Therefore, it indicates that for any listening miner v, as long as the set S is not empty, the strength of the received signal $Signal(v)$ is larger than N. In addition, each miner is equipped with physical carrier sensing, which is part of the IEEE 802.11 standard in the media access control (MAC)

layer. The miners adopt synchronous wake-up mode, that is, each miner will wake up at the beginning of the same round. [26–28,30,35–37] are the works who have the similar SINR assumptions with our model in this paper.

Problem Definition. We assume that there are n miners in a 2-dimensional Euclidean plane, all of which wake up synchronously at the beginning and can transmit to each other via the multiple access wireless channel. In addition, we assume that n is sufficiently large and miners only have a rough estimation on n. Among the n miners, there are normal miners and faulty miners, where the number of faulty miners is not larger than f, f is a known number and smaller than $\lceil \frac{n}{2} \rceil$. Normal miners can work normally. However, faulty miners may be out of work (broken down) or send wrong messages (e.g., broadcast an invalid block) because of unknown errors such as the calculation errors, which prevents the normal miners from achieving a consensus. In each interval containing sufficient large slots, the group of miners need to achieve a consensus on a valid block including network transactions that have occurred in the current interval in their local blockchain.

4 FTPoC Consensus Protocol

4.1 Framework of Consensus Protocol

The consensus process in our work is divided into three phases: leader election, block proposal and validation, and chain update. **Leader Election (LE) Phase:** in each phase, one miner will be elected as the leader from the group of miners in network to schedule the following two phases. **Block Proposal and Validation (BPV) Phase:** each time when a leader is elected, it will propose a new block and record it, and other miners will record the block and count the number of times the new block is received. Then, the miners who are not leaders will return to the previous phase until $f+1$ the same blocks are recorded. **Chain Updation (CU) Phase:** all normal miners will write the new block which has been recorded $f+1$ times into the local blockchain.

In each time of leader election phase, a miner will be elected as the leader w.h.p. After repeating this phase at most $2f+1$ and at least $f+1$ times, at most $2f+1$ (at least $f+1$) miners will be elected as leaders, in which $f+1$ leaders are normal miners and will propose the same blocks. In BPV phase, each leader will propose a block and record it, and other miners will record the block and count the number of times the block is received. The miners who are not leaders will return to the previous phase until $f+1$ the same blocks are recorded. In the final phase, all miners will write the new block which has been proposed $f+1$ times in the block.

4.2 Goal of Consensus

Generally, in a distributed system, the consensus is a state that all nodes in system agree on the same data values, which satisfies the following three requirements: termination, validity and agreement [8].

Termination: for each normal miner in blockchain system, a valid transaction/block is written into its local blockchain in finite time. **Validity:** If all normal miners approve the same valid transaction/block, then the transaction/block should be written in their local blockchain. **Agreement:** each valid transaction/block should be written by each normal miners to its local blockchain. For each normal miner, the written block in its local blockchain should be assigned the same sequence number.

In addition to the above basic goals of the consensus protocol design in blockchain network, the efficiency, and security of consensus protocols should be considered comprehensively in our application context. Thus, in Sect. 5, in addition to analyzing the correctness of our fault-tolerant consensus protocol, we also analyze the efficiency and security of it.

4.3 Detail Description for FTPoC Consensus

We divide the execution time into rounds, and each round contains three slots. There are three functions in the protocol: Leader Election ( ), Block Proposal and Validation ( ), and Chain Update ( ). In each round, each miner will execute the above three functions successively. We give the pseudo code in Algorithm 1.

Algorithm 1: FTPoC Consensus Protocol for each miner v

Initialization: $state_v = \mathbb{C}, count_v = 0, C_v = 0, i = 1$;

In each round, each miner v does:

1 Slot 1: Leader Election ( );
2 Slot 2: Block Proposal and Validation ( );
3 Slot 3: Chain Update ( );

We use the following $2f + 3$ states to represent the state of the miners in execution: $\mathbb{C}$ is the candidate state, indicating that the miners are in the leader competition; $\mathbb{S}$ is the silence state, indicating that the miners have given up the leader election; and $\mathbb{L}_i (i = 1, 2..., 2f + 1)$ are the leader states, indicating that the miner succeeded in leader competition and be elected as the i-th leaders.

Initially, each miner is in state $\mathbb{C}$, when it wakes up. Then, it will execute Algorithm 1 in each round. Each miner v is initially in state $\mathbb{C}$ after waking up. Then, it begins to execute the three functions given in Algorithm 1 in each of the following rounds. In each round, all miners execute Algorithm 2 in slot 1.

Algorithm 2: Leader Election ( ) for each miner v

```
Slot 1:
1  if state_v == ℂ then
2      Transmit a message M_v with constant probability p_v or listen
       otherwise;
3      if Received signal is larger than N then
4          state_v = 𝕊;
5      else
6          count_v ++;
7          if count_v > k * log n then
8              state_v = 𝕃_i;
```

Each miner in state $\mathbb{C}$ transmits with a constant probability. Then, if a miner v listens and the signal it receives has its strength larger than N, miner v becomes in state $\mathbb{S}$, i.e., miner v gives up the leader competition. If miner v keeps in state $\mathbb{C}$ in at least $k * \log n$ rounds, where k is a sufficiently large constant, miner v becomes in state $\mathbb{L}_i$, which indicates that miner v becomes the i-th elected leader. By the action in slot 1, there are multiple miners (at most $2f+1$ and at least $f+1$) being elected as the leaders w.h.p, of which $f+1$ leaders are normal miners, and we will give the proof in the next section.

In slot 2 of each round, each miner execute the function Block Proposal and Validation() which is shown in Algorithm 3. Each time a leader is elected, this leader will propose a new block B_v and disseminate it to all other miners in the wireless blockchain network, and then record the block. If the new block B_v is received by the other miners, they will record the block and count the number of times the block is received. Then, to participate the leader election again, the miners in state $\mathbb{S}$ will move to state $\mathbb{C}$. If a leader is a faulty miner, it may propose an invalid block, but each leader who is a normal miner will propose valid and the same blocks. In particular, this is to make sure that only one valid block is proposed in each time of the consensus process.

In slot 3, each miner execute the function Chain Update() which is given in Algorithm 4. If the new block has been recorded $f+1$ times by each normal miner in the previous phase, the new block B_v will be written into the local blockchain by each normal miner in the blockchain network.

5 Protocol Analysis

In this section, we first analyze and prove the correctness of the FTPoC consensus protocol, and then analyze the efficiency and security of it.

Algorithm 3: Block Proposal and Validation ( ) for each miner v

Slot 2:

if $state_v == \mathbb{L}_i$ **then**
  Broadcast B_v;
  Record B_v;
  if *v has recorded the same B_v $f+1$ times* **then**
    $C_v = 1$;
if $state_v == \mathbb{L}_p(p = 1, 2, ..., i-1)$ *or* $state_v == \mathbb{S}$ **then**
  Listen ;
  if *receive* B_v **then**
    Record B_v;
    if *v has recorded the same B_v $f+1$ times* **then**
      $C_v = 1$;
  if $state_v == \mathbb{S}$ *and* $C_v == 0$ **then**
    $state_v = \mathbb{C}$;
    $count_v = 0$;
  $i++$;

Algorithm 4: Chain Update () for each miner v

Slot 3:

if $C_v == 1$ **then**
  Use the B_v to update the local Blockchain;
$state_v = \mathbb{C}, i = 1, count_v = 0, count_v^a = 0 (a = 1, 2, ..., f+1), C_v = 0$;

Theorem 1. *It takes at most $O((f+1)\log n)$ rounds to make all normal miners in blockchain network achieve consensus w.h.p.*

Our proof of Theorem 1 is unfolded in the following three lemmas: Lemma 1, Lemma 3 and Lemma 4. By the three lemmas, we prove that our protocol satisfies termination, validity and agreement respectively.

Lemma 1. *It takes at most $O((f+1)\log n)$ rounds for all miners to terminate the consensus process, w.h.p.*

It is obvious that after $f+1$ normal leaders are elected out, an extra round is enough for miners to make a consensus on the block. Therefore, the termination of our consensus protocol mainly depends on the leader election process, which is analyzed and proved by the following Lemma 2.

Lemma 2. *In each $O(\log n)$ rounds, a leader will be elected out by the function Leader Election ( ) in our algorithm w.h.p.*

Proof. Since all of the miners wake up synchronously and become the candidates for the leader election when they wake up, each time a miner is elected as a leader, the remaining miners in state $\mathbb{S}$ will change to state $\mathbb{C}$ at the same time, and then participate in the next leader election. Then, We take into account the reduction of the candidates in each leader election. According to our algorithm, we just need to prove that after $k * \log n$ rounds, there is only one candidate left in each leader election w.h.p, and it will become the leader.

Let set A denote the collection of the miners who are the candidates for a leader election. In the first leader election, all the miners participate in the leader competition, i.e. $|A| = n$. After each leader election, only those who are not elected will participate in the next leader election, i.e. $|A|$ will decrease by 1. After at least $f + 1$ and at most $2f + 1$ times leader election, all normal miners will record a valid block $f + 1$ times, and then achieve consensus. As for the analysis for candidates' reduction in slot 1, we divided it into following two cases: (1) $|A| = 1$; (2) $|A| > 1$. Obviously, if $|A| = 1$, the miner in set S will become leader. For another case $|A| > 1$, we can see that there are $|A| * p$ miners transmit and $|A| * (1 - p)$ miners listen in expectation in each round. Thus, in expectation, there will be $\min\{1, |A| * p\} * (1 - p) * |A|$ nodes giving up the leader election in each round. By taking a Chernoff bound, we can prove that within $O(\log n)$ rounds, there will be only one leader left in set A and becomes the leader. A detailed and specific proof for the process of Chernoff bound can be found in [20].

According to Lemma 2, one leader will be elected out within $O(\log n)$ rounds in each leader election process w.h.p. Moreover, after at least $f + 1$ and at most $2f + 1$ times leader election, $f + 1$ normal leaders will be elected out. Thus, within $O((f + 1) \log n)$ rounds all miners will get a consensus. Combining the two Lemmas above, we prove that our consensus protocol satisfies the termination requirement and has an efficient time complexity.

Lemma 3. *The protocol satisfies the validity requirement.*

Proof. We divide our analysis into two cases as follows: (1) B_v is invalid, i.e., the leader v who generated B_v is a faulty miner. There are at most f faulty miners in the wireless blockchain network, and only the leader who is a faulty miner may propose and broadcast an invalid block. Therefore, B_v can not be recorded by $f + 1$ times. (2) B_v is valid, i.e., the leader v who generates B_v is a normal miner. Note that any normal leader will propose and broadcast the same valid block, and the invalid blocks from faulty miners can not be recorded by $f + 1$ times. According to Algorithm 3, there must be $f + 1$ normal miners being elected as leaders, and they all propose the same B_v, which indicates that B_v will be recorded by any miner u by $f + 1$ times. Thus, for any u, it will set the value of C_u equal to 1.

Because of the execution of Algorithm 3, if and only if the new block B_v is valid, it will make value $C_u = 1$ for any miner u. Then in the last slot, all of the miners would update their local blockchain with the new block B_v in the current round.

Lemma 4. *The protocol satisfies the agreement requirement.*

Claim. In each consensus process, all normal miners in the blockchain network will update their local blockchain with the same block B_v.

Proof. According to Claim 1, we can draw two conclusions: (1) If the new block B_v is valid, B_v will be recorded by $f+1$ times, and then all normal miners will update the local blockchain with B_v; (2) On the contrary, if the new block B_v is invalid, B_v can not be recorded by $f+1$ times, which indicates that normal miners will not update the local blockchain with B_v. Thus, for each normal miner the latest block of its local blockchain is the same.

Claim. For any pair of normal miners u and v in the blockchain network, B_v^i and B_u^i are the same, where B_u^i is the i'th block in local blockchain of u and B_v^i is the i'th block in local blockchain of v.

Proof. If both of the normal miners u and v are leaders. We assume that the i-th valid block B_v^i is generated by v and the i-th valid block B_u^i is generated by u. Because any leader who is a normal miner will propose the same block, B_v^i is the same as B_u^i. From the proof of Claim 1, they will use the same new block to update their local blockchain simultaneously.

If only one of the normal miners u and v is a leader. Without loss of generality, we assume that u is the leader, and the i-th valid block B_v^i is generated by it. From the proof of Claim 1, both v and u will use the block B_v^i to update their local blockchain simultaneously.

If neither u nor v is the leader. Assuming that leader w who is a normal miner propose the i-th valid block B_w^i. From the proof of Claim 1, they will use the block B_w^i to update their local blockchain. Thus, in any case, both B_v^i and B_u^i are the same.

5.1 Discussion for Efficiency and Security

High Throughput and Energy Efficiency. Theorem 1 indicates that the FTPoC consensus protocol has the time complexity of $O((f+1)\log n)$. In our protocol, each round is divided into 3 slots, and each slot is the minimum time to transmit a packet. Therefore, our protocol has a high throughput. In addition, In the best case, only $f+1$ leaders need to be elected, and even in the worst case, only at most $2f+1$ leaders need to be elected. It indicates that only at most $2f+1$ miners need to propose and broadcast the new block, which avoids the participation of all miners and the sending of a large number of messages. Therefore, it is energy-saving and resource-friendly for the multi-agents in IoT.

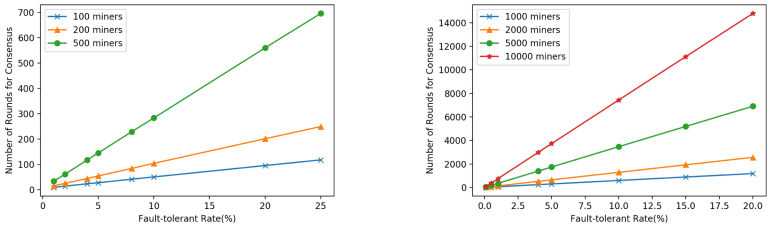
Fairness. Because of setting $p_v = p$, each v in slot 1 transmits $\mathcal{M}_v$ with a same constant probability p , which guarantees the fairness of each miner in State $\mathbb{C}$ to compete for leadership. The randomness of leaders can also be guaranteed by the fairness of the leader election, which can guarantee the security of the protocol to a certain extent. It helps prevent the blockchain network from denial-of-service attacks by adversaries who know in advance which leaders will be elected.

State Fork Problem Avoided. In wireless blockchain network, due to the transmission delay and contention in the shared channel, it is difficult to solve the state fork problem. Although there are multiple leaders in our protocol, each leader who is a normal miner propose the same valid block, and only the valid block can be recorded by $f + 1$ times. Therefore, the protocol can ensure that in each time of the consensus process only one valid block is written into the local blockchain of each normal miner eventually, thus avoiding the state fork problem. Meanwhile, it makes our protocol more secure and could prevent the blockchain network from fork attacks by adversary as well.

6 Simulation Result

Parameter Setting. We set that n miners are randomly and uniformly distributed in a wireless blockchain network, of which f are set as faulty miners and the rest are set as normal miners. The background noise upper bound N is normalized as 1.0. The transmission probability p_v is set as 0.2.

Protocol Performance. The efficiency and fault-tolerant rate of our protocol used to achieve the consensus in the wireless blockchain network is given in Fig. 1(a) and Fig. 1(b) in which the x-axes and y-axes represent the fault-tolerant rate (i.e., proportion of faulty miners among all miners) and rounds used to achieve consensus respectively.



(a) (b)

Fig. 1. Rounds used for achieving consensus

From Fig. 1(a) and Fig. 1(b), we can see that when the total number of miners is fixed, the higher the proportion of faulty miners in the network, the more rounds it takes to achieve consensus. Figure 1(a) shows the protocol can guarantee both high efficiency and high fault-tolerant rate when the scale of the network is small (e.g., $n = 100, 200, 500$). Even in the case that fault-tolerant rate is more than 20%, which means that proportion of faulty miners among all miners is more than 20%, our protocol can achieve consensus within 800 rounds. Figure 1(b) shows when the scale of the network is large (e.g., $n = 1000, 2000, 5000, 10000$), if we want to ensure high fault-tolerant rate, we need to spend more rounds to achieve consensus.

7 Conclusion

In this paper, for the first time, we consider the fault-tolerant consensus protocol for multi-agents in wireless blockchain system, which is closer to reality compared with the previous consensus protocols in blockchain system, and has an efficient performance on time complexity. Specifically, we propose a fault-tolerant Proof-of-Communication consensus protocol with the time complexity of $O((f+1)\log n)$, which has high energy efficiency and its security guaranteed. In addition, our protocol can provide some new perspectives for the higher-level protocol designs on multi-agents in the blockchain system. Also, byzantine behaviors is a common phenomenon for multi-agents in the blockchain system, and is far more complex. Thus the research in wireless blockchain system to against byzantine will become our direction in the future.

References

1. Bentov, I., Lee, C., Mizrahi, A., Rosenfeld, M.: Proof of activity: extending bitcoin's proof of work via proof of stake [extended abstract]y. SIGMETRICS Perform. Eval. Rev. **42**(3), 34–37 (2014)
2. Bitcoinwiki: Proof of stake (2014). https://en.bitcoin.it/wiki/Proof_of_Stake
3. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: SoK: research perspectives and challenges for bitcoin and cryptocurrencies. In: IEEE Symposium on Security and Privacy (2015)
4. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: OSDI (1999)
5. Chan, W., Chin, F.Y.L., Ye, D., Zhang, G., Zhang, Y.: On-line scheduling of parallel jobs on two machines. J. Discrete Algorithms **6**(1), 3–10 (2008)
6. Chan, W., Zhang, Y., Fung, S.P.Y., Ye, D., Zhu, H.: Efficient algorithms for finding longest common increasing subsequence. J. Comb. Optim. **13**(3), 277–288 (2007)
7. Chin, F.Y.L., et al.: Competitive algorithms for unbounded one-way trading. Theor. Comput. Sci. **607**(1), 35–48 (2015)
8. Dwork, C., Lynch, N.A., Stockmeyer, L.J.: Consensus in the presence of partial synchrony (preliminary version). In: PODC (1984)
9. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of space. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 585–605. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_29
10. Gazi, P., Kiayias, A., Russell, A.: Stake-bleeding attacks on proof-of-stake blockchains. In: CVCBT (2018)
11. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
12. King, S., Nadal, S.: PPcoin: peer-to-peer crypto-currency with proof-of-stake (2012). https://peercoin.net/assets/paper/peercoin-paper.pdf
13. Li, F., Luo, J., Shi, G., He, Y.: ART: Adaptive fRequency-Temporal co-existing of ZigBee and WiFi. IEEE Trans. Mobile Comput. **16**(3), 662–674 (2017)
14. Li, F., Yu, D., Yang, H., Yu, J., Karl, H., Cheng, X.: Multi-Armed-Bandit-based spectrum scheduling algorithms in wireless networks: a survey. IEEE Wirel. Commun. **27**(1), 24–30 (2020)

15. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf
16. O'Dwyer, K.J., Malone, D.: Bitcoin mining and its energy footprint. In: ISSC/CIICT (2014)
17. Pease, M.C., Shostak, R.E., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**(2), 228–234 (1980)
18. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In: Camenisch, J., Kesdoğan, D. (eds.) iNetSec 2015. LNCS, vol. 9591, pp. 112–125. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39028-4_9
19. Yu, D., Duan, X., Li, F., Liang, Y., Yang, H., Yu, J.: Distributed scheduling algorithm for optimizing age of information in wireless networks. In: IPCCC (2020)
20. Xu, Q., Zou, Y., Yu, D., Xu, M., Shen, S., Li, F.: Consensus in wireless blockchain system. WASA **1**, 568–579 (2020)
21. Yin, M., Malkhi, D., Reiter, M.K., Golan-Gueta, G., Abraham, I.: HotStuff: BFT consensus with linearity and responsiveness. In: PODC (2019)
22. Yu, D., Ning, L., Zou, Y., Yu, J., Cheng, X., Lau, F.C.M.: Distributed spanner construction with physical interference: constant stretch and linear sparseness. IEEE/ACM Trans. Netw. **25**(4), 2138–2151 (2017)
23. Yu, D., Wang, Y., Halldórsson, M.M., Tonoyan, T.: Dynamic adaptation in wireless networks under comprehensive interference via carrier sense. In: IPDPS (2017)
24. Yu, D., Zhang, Y., Huang, Y., Jin, H., Yu, J., Hua, Q.: Exact implementation of abstract MAC layer via carrier sensing. In: INFOCOM (2018)
25. Yu, D., et al.: Stable local broadcast in multihop wireless networks under SINR. IEEE/ACM Trans. Netw. **26**(3), 1278–1291 (2018)
26. Yu, D., Zou, Y., Wang, Y., Yu, J., Cheng, X., Lau, F.C.M.: Implementing the abstract MAC layer via inductive coloring under the Rayleigh-fading model. IEEE Trans. Wirel. Commun. https://doi.org/10.1109/TWC.2021.3072236
27. Yu, D., et al.: Competitive age of information in dynamic IoT networks. IEEE Internet Things J. (2020). https://doi.org/10.1109/JIOT.2020.3038595
28. Yu, D., et al.: Implementing the abstract MAC layer in dynamic networks. IEEE Trans. Mob. Comput. **20**(5), 1832–1845 (2021)
29. Yu, D., et al.: Distributed dominating set and connected dominating set construction under the dynamic SINR model. In: IPDPS (2019)
30. ?ibitemch112704 Yu, D., Zou, Y., Zhang, Y., Sheng, H., Lv, W., Cheng, X.: An exact implementation of the abstract MAC layer via carrier sensing in dynamic networks. IEEE/ACM Trans. Netw. (2021) https://doi.org/10.1109/TNET.2021.3057890
31. Zheng, X., Cai, Z.: Privacy-preserved data sharing towards multiple parties in industrial IoTs. IEEE J. Sel. Areas Commun. **38**(5), 968–979 (2020)
32. Zhu, S., Cai, Z., Hu, H., Li, Y., Li, W.: zkCrowd: a hybrid blockchain-based crowdsourcing platform. IEEE Trans. Ind. Inform. **16**(6), 4196–4205 (2020)
33. Zhang, Y., Chen, J., Chin, F.Y.L., Han, X., Ting, H.-F., Tsin, Y.H.: Improved online algorithms for 1-space bounded 2-dimensional bin packing. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010. LNCS, vol. 6507, pp. 242–253. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17514-5_21
34. Zhu, S., Li, W., Li, H., Tian, L., Luo, G., Cai, Z.: Coin hopping attack in blockchain-based IoT. IEEE Internet Things J. **6**(3), 4614–4626 (2019)
35. Zou, Y., Xu, M., Sheng, H., Xing, X., Xu, Y., Zhang, Y.: Crowd density computation and diffusion via Internet of Things. IEEE Internet Things J. **7**(9), 8111–8121 (2020)

36. Zou, Y., et al.: Fast distributed backbone construction despite strong adversarial jamming. In: INFOCOM (2019)
37. Zou, Y., Yu, D., Yu, J., Zhang, Y., Dressler, F., Cheng, X.: Distributed Byzantine-Resilient multiple-message dissemination in wireless networks. In: IEEE/ACM Trans. Netw. (2021). https://doi.org/10.1109/TNET.2021.3069324