Timely Publication of Transaction Records in a Private Blockchain

Rui Ming
Computer and Information Science Department
University of Massachusetts Dartmouth
Dartmouth, MA 02747, USA
rming@umassd.edu

Haiping Xu
Computer and Information Science Department
University of Massachusetts Dartmouth
Dartmouth, MA 02747, USA
hxu@umassd.edu

***Abstract*—Blockchain technology has been successfully applied in many different application domains. However, in some time-sensitive applications, e.g., a COVID-19 tracking system with case and testing information recorded in blockchains, it is critical to publish information or transactions in blockchains in a timely manner. In this paper, we introduce an approach to timely publishing transaction records in private blockchains, which are small-scale permissioned ones with limited number of transactions broadcast per unit of time. In our approach, when a transaction is broadcast by a peer, it is published immediately in a temporary block after being approved using a consensus mechanism. When a predefined number of temporary blocks have been published, the system will combine them into a permanent one and publish it upon approval using the same consensus mechanism. The experimental results show that our proposed approach is feasible and effective for timely publication of transaction records in a private blockchain.**

Keywords— Private blockchain, timely publication, consensus mechanism, temporary block, permanent block

I. INTRODUCTION

Blockchain is a decentralized and distributed digital ledger technology, which was initially proposed as a key mechanism for a peer-to-peer electronic cash system, called Bitcoin, in 2008 [1]. Due to its unique properties of immutability and decentralization, blockchains have been widely used to record transactions across many computers so that any stored record cannot be altered retroactively without the update of all subsequent blocks. Over the past decade, blockchain has become a groundbreaking technology with applications in many critical domains, such as virtual currencies, real estate title transfers and registration, digital voting, immutable data backup and so on. There are two types of blockchains, namely public blockchain and private blockchain. In a public blockchain network, peers can freely join, read, write, or participate in blockchain activities. In contrast, a private blockchain system places permission restrictions on the peers who can participate in the network and in which transactions. A public blockchain network may have a large number of peers; while the number of peers involved in a private blockchain network could be very limited. In most of the public blockchain systems, e.g., Bitcoin, after a number of transactions have been broadcast by peers, a new block that stores those transactions can be created and validated by the system in just a few minutes. This is because there is a massive amount of transactions generated per unit of time. However, in a private blockchain system, it would take much longer time if multiple transactions need to be included in a single block because new transactions are not generated as often as in a public blockchain system. Many time-sensitive systems, such as a COVID-19 tracking system with case and testing information recorded in blockchains, require that information broadcast by a peer must be published in a new block very quickly. One solution is to allow each block to record only one transaction, which could effectively reduce the time to publish a transaction; however, such an approach is not space efficient, and it also requires much longer time to retrieve and validate transactions in a blockchain that has a rapidly growing size. Thus, there is a pressing need to develop a time and space-efficient mechanism that not only allows efficient usage of space but also ensures timely publication of new transaction records. In this paper, we introduce an approach that facilitates timely publication of a new transaction in a temporary block. Once a predefined number of temporary blocks have been published, they can be combined into a new permanent block, which replaces the temporary ones in the blockchain.

Another challenging issue in our approach is to effectively achieve the necessary agreement on a single recognized blockchain by distributed peers. A consensus mechanism of the blockchain technology is a protocol that allows all peers, who maintain the blockchains, to be synchronized with each other to agree on the legitimacy of transaction records to be added into a blockchain. The Proof-of-Work (PoW) algorithm that has been used in Bitcoin is one example of consensus algorithms, where special peers, called miners, compete against each other to generate new blocks and get rewarded [1]. The disadvantage of PoW is the high energy consumption that is wasted and not applicable anywhere else. Proof-of-Stake (PoS) algorithm was developed as an alternative to the PoW algorithm to overcome the high energy consumption [2]; however, both PoW and PoS do not apply well to a private blockchain system that has a much smaller scale than public blockchains such as Bitcoin. On the other hand, practical Byzantine Fault Tolerance (pBFT) algorithm is a universal solution for distributed systems that tolerate Byzantine faults [3]. In our approach, we introduce a variation of the pBFT algorithm as a consensus mechanism to effectively validate the legitimacy of either a new temporary block or a permanent one in a private blockchain that requires timely publication of transaction records.

978-1-7281-8915-4/20/$31.00 ©2020 IEEE
DOI 10.1109/QRS-C51114.2020.00030

II. Related Work

The scalability of blockchain systems has been a challenging issue since the blockchain technology has been adopted in various application domains. The challenge is mostly due to the significant and growing size of the blockchains. There are some existing approaches to changing or tweaking data structures to deal with the blockchain size problem. For example, the original Bitcoin system used the Simplified Payment Verification (SPV) mechanism to verify transactions without running through the full network nodes. Users can minimize storage space by only storing the block headers of the longest blockchain rather than the entire blockchain [1]. Cryptonite has been among the first implementations of the lightweight mini-blockchain scheme [4]. It has been designed to eliminate the need for a full blockchain by storing important block information locally for each node. This approach can significantly reduce the need for long-term data storage as old transactions are pruned but the account balance value in the hash tree structure can still be maintained. VerSum is an alternative proposed by Hooff et al. that allows light nodes to outsource expensive computation securely over large data structures such as Bitcoin blockchains to multiple servers [5]. It can achieve low server-side overhead for both incremental re-computation and conflict resolution, and easily keep up with Bitcoin's rate of new blocks with new transactions. Different from the above approaches, our method can minimize the storage overhead for published blocks in a private blockchain by combining small-sized blocks, i.e., temporary blocks, into a permanent one.

There is also previous work on using consensus algorithms for peers to reach agreement on the state of certain data stored on distributed nodes in a blockchain system. The most well-known consensus method is the PoW algorithm used in Bitcoin [1]. In this method, different nodes in a blockchain try to solve a cryptographic hash function, SHA-256 in particular, which generates a unique and fixed size 256-bit hash value. The process of creating new blocks by first solving this hash problem is called mining, and the users performing this task are called miners. To counteract the great deal of energy waste in the PoW algorithm, another consensus algorithm PoS was proposed [2]. The core idea of PoS is to attribute the mining power based on the number of coins held by a miner. Therefore, a PoS miner is allowed to mine only a limited number of blocks that reflects the miner's ownership stake. Proof-of-Authority (PoA) is a new consensus method that provides high performance and fault tolerance [6]. In PoA, a peer must first pass a preliminary authentication, and is allowed to create a new block only if it has proven its authority for the task. The most related consensus algorithm to the one we used in this paper is pBFT, which could be suitable for a private blockchain [3]. The pBFT mechanism can work in asynchronous environments with improved response time. In our approach, we have revised the pBFT mechanism by utilizing a majority vote for approval of a new temporary or permanent block after being verified.

Additional related work focuses on revising blockchain architecture to improve blockchain performance. Traditional blockchain uses a single chain to record transactions in a block. Since blocks in a single chain structure cannot be generated concurrently, the throughput of publishing new blocks in a blockchain is limited. To deal with this issue, DAG-based blockchain architectures are proposed, which enable concurrent block generation and allow multiple vertices to connect to a previous vertex in a directed acyclic graph [7]. A cryptocurrency for the Internet-of-Things (IoT), called IOTA, was among the first approaches to implementing the "blockless blockchain," which uses a network of nodes to speed up the validation process [8]. It uses a structure called tangle, which is a DAG for storing transactions. In an early effort, Nicol and Xu developed a DAG-based blockchainless approach for trusted public construction bidding with cryptographic guarantees to enforce fairness in the bidding process [9]. Instead of mining blocks of transactions as in a traditional blockchain, a DAG links a transaction containing a list of its parents, documents and transaction signatures, to other transactions via a less complex validation process. A recently proposed consensus protocol called SPECTRE can be applied to a DAG-based blockchain structure to create and publish blocks of transactions by extending local DAGs [10]. In SPECTRE, peers can perform blockchain activities without having to know other peers' synchronization status. Similar, PHANTOM also applies blockDAG to achieve faster block generation and higher transaction throughput [11]. Moreover, PHANTOM proposes a greedy algorithm to order transactions embedded in blockDAG and supports smart contract. Our approach differs from the above approaches by following the traditional blockchain architecture; however, our approach is flexible and efficient, which allows timely publication of temporary blocks and simplification of a blockchain by combining temporary blocks into a permanent one.

III. The Blockchain Structure

The block time is defined as the time required to create a new block in a blockchain [12]. In most of the blockchain systems, such as Bitcoin and Ethereum, an expected block time can be set to require the systems to wait for a fixed amount of time before a new block can be generated and published [1, 13]. However, in many critical and time-sensitive situations, users would request their transactions to be published in blockchains immediately rather than have to wait for a fixed amount of time for the transactions to be published. For example, in a blockchain-based COVID-19 real-time updating and tracking system, when infection cases are broadcast, such information needs to be available immediately to all those who had contact with the infected persons for emergency treatment. To avoid waiting for a fixed amount of time for the cases to be published, we define two types of blocks, namely the temporary block and the permanent block. A temporary block contains only one transaction, which can be generated in a timely manner and published efficiently after being verified. On the other hand, a permanent block is a traditional block that cannot be altered retroactively without the alteration of all subsequent blocks. A permanent block, generated by combining a predefined number of temporary blocks in a blockchain, can record multiple transactions and can effectively reduce the size of the blockchain by eliminating the temporary ones.

A. The Blockchain Structure with Temporary Blocks

We first define a set of *super peers* as a group of trusted peers who are responsible for creating new blocks and participating in the consensus process. There is also a voted super peer called the *primary* super peer or *PSP*, who serves as a leader for the consensus process. On the other hand, a regular peer is one who

has the access to the latest blockchain and can broadcast a new transaction to super peers for processing; however, it is not allowed to create new blocks or get involved in the consensus process. When a new transaction is generated, it can be announced by the peer who generates the transaction in a request message broadcast to all super peers. When the *PSP* receives the request message, it creates a new temporary block containing the transaction information and broadcast it to all super peers. In addition to the transaction information, the new block also contains the hash value of the last permanent block in the blockchain, the digital signature of the transaction signed by the peer who broadcast the message as well as other related information. When the super peers receive the newly created temporary block from the primary one, they use a consensus algorithm to verify and agree on the validity of the new block. If the new temporary block is verified to be correct, it will be published in the blockchain and broadcast to all peers. Then the transaction information recorded in the temporary block can be readily accessed by all peers. Fig. 1 shows how a new temporary block TB_i is attached to a blockchain.

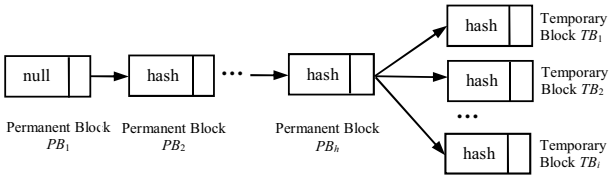

Fig. 1. The blockchain structure with temporary blocks.

For future transactions broadcast by peers, the system repeats the above operations until there is a predefined number Φ of temporary blocks published. At this time, *PSP* is responsible for creating a new permanent block to include all transactions recorded in the Φ temporary blocks. Once a new permanent block is created and broadcast to all super peers, it will be verified and approved by super peers using the same consensus algorithm that has been applied to temporary blocks. If the new permanent block is approved, the temporary blocks will be removed, and the new permanent block will be added to the blockchain by linking it to the last permanent block. This change is then broadcast to all peers in the blockchain system for blockchain updating.

We define height h of a blockchain as the number of permanent blocks in the blockchain. As in a traditional blockchain, each permanent block PB_i, where $1 < i \leq h$, includes the hash value of its prior permanent block PB_{i-1}; therefore, any changes to PB_{i-1} can be easily detected by checking the hash value of PB_{i-1} included in PB_i. On the other hand, since all temporary blocks are linked to the last permanent block PB_h as shown in Fig. 1, they all include the hash value of PB_h.

B. Temporary Block Structure and Generation

As a temporary block can contain only one transaction, it can be published in a blockchain immediately upon approval without the need to wait for new transactions to be broadcast. As shown in Fig. 2, a temporary block TB_i, contains four major components, namely the block header, the data, the hash value of TB_i, denoted as $ha(TB_i)$, and a list of digital signatures $ds[TB_i]_v$, where v is a super peer in the consensus group, and each $ds[TB_i]_v$ is the digital signature of super peer v for temporary block TB_i. In the block header, $ha(PB_h)$ is the hash value of the last permanent block PB_h, h is the height of the blockchain without counting temporary blocks, t is the timestamp when the transaction was made, and p is the identification number of the peer who broadcasts the transaction. In the data portion, TX is a transaction with its required information to be recorded in the blockchain and $ds[t, p, TX]_p$ is the digital signature of peer p for transaction TX. The hash value of the temporary block TB_i, denoted as $ha(TB_i)$, is defined as in (1).

$$ha(TB_i)=H(ha(PB_h), h, t, p, TX, ds[t, p, TX]_p) \quad (1)$$

where H is a hash function. The list of $ds[TB_i]_v$ are digital signatures signed by super peers during the consensus process, which is used to check the validity of the temporary block TB_i. Note that in the blockchain, we use digital signature to guarantee that the contents of a message have not been altered in transit [14]. Each peer has a pair of keys. One key is called the *private* key, which is only known to the peer itself; while the other key is called the *public* key, which is known to the public. In the digital signature process, peer $p1$ uses its private key to encrypt the hash value of message m into a digital signature $ds[m]_{P1}$, and then sends $ds[m]_{P1}$ along with message m to peer $p2$. Peer $p2$ can decrypt $ds[m]_{p1}$ using $p1$'s public key to verify the authenticity of message m, i.e., the hash value of message m, decrypted using $p1$'s public key, exactly matches with the hash value of the original message m.

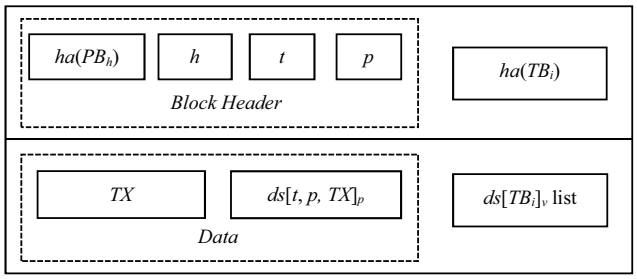

Fig. 2. The structure of temporary block TB_i.

When peer p in the blockchain system generates a transaction TX, it broadcasts it to super peers as a request message <*TB-request*, $[t, p, TX]$, $ds[t, p, TX]_p$> with all required information defined in a temporary block except the height of the blockchain, the hash value of the temporary block to be created, and the list of digital signatures of super peers for the temporary block. When *PSP* receives the request message for creating a temporary block to publish the transaction, it verifies the message for its hash value and the digital signature. If the message is valid, *PSP* generates a temporary block (as shown in Fig. 2) based on the information provided in peer p's request message, and initiates a consensus process by broadcasting the new temporary block with its own digital signature $ds[TB_i]_{primary}$ to all other super peers for approval. Algorithm 1 shows the procedure how *PSP* creates a temporary block when receiving a *TB-request* message. Note that in Algorithm 1, *PSP* needs to check if transaction TX sent by peer p has already been published in the blockchain. If TX has been recorded in an existing temporary block, a *null* value is returned; otherwise, *PSP* uses the sender's public key to verify the authenticity of the request

message. If the message has been altered, a *null* value is returned; otherwise, *PSP* generates all components including the hash value $ha(TB_{k+1})$ and its digital signature $ds[TB_{k+1}]_{primary}$ of the new temporary block TB_{k+1}, and returns the newly constructed block temporary block TB_{k+1}.

Algorithm 1: Generation of a Temporary Block

Input: Request message <*TB-request*, [*t*, *p*, *TX*], $ds[t, p, TX]_p$>
Output: A temporary block TB_{k+1} signed by *PSP*

1. Let *h* be the height of the current blockchain
2. Let *k* be the number of temporary blocks linked to PB_h
3. Let TB_{k+1} be the temporary block to be created
4. **if** $k == \Phi$ // the predefined number of temporary blocks
5. **return** *null* // a permanent block shall be generated first
6. Compute the hash value of the last permanent block $ha(PB_h)$
7. Extract *t*, *p*, *TX* from the *TB*-request message
8. **for** $i = 1$ **to** *k*
9. **if** [*t*, *p*, *TX*] matches with TB_i
10. **return** *null*
11. Use the public key of peer *p* to verify message [*t*, *p*, *TX*]
12. **if** message [*t*, *p*, *TX*] has been altered
13. **return** *null*
14. **else**
15. Calculate the hash value of temporary block $ha(TB_{k+1})$
16. Add $ha(PB_h)$, *h*, *t*, *p*, *TX* and $ds[t, p, TX]_p$ into TB_{k+1}
17. Calculate hash value for TB_{k+1}, i.e., $ha(TB_{k+1})$ as in (1), and add it into TB_{k+1} structure
18. Calculate the digital signature $ds[TB_{k+1}]_{primary}$, and add it into the list of digital signatures of super peers for TB_{k+1}.
19. **return** TB_{k+1}

C. Temporary Block Structure and Generation

Different from a temporary block, a permanent block is one that cannot be removed from a blockchain. Similar to the structure of a temporary block, a permanent block also consists of four major components. As shown in Fig. 3, if the height of a blockchain is *h*, a new permanent block PB_{h+1} contains the block header, the data, the hash value of PB_{h+1}, denoted as $ha(PB_{h+1})$, and a list of digital signatures $ds[PB_{h+1}]_v$, where *v* is a super peer in the consensus group, and each $ds[PB_{h+1}]_v$ is the digital signature of super peer *v* for permanent block PB_{h+1}.

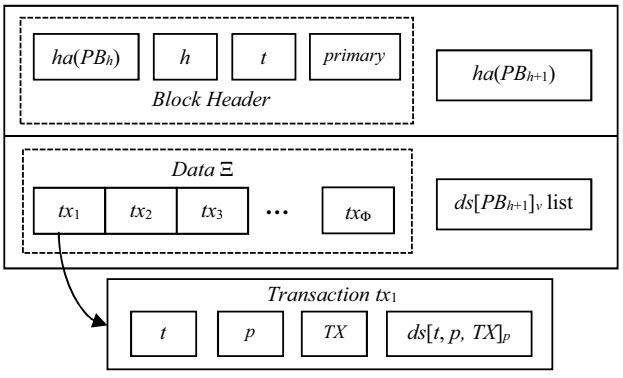

Fig. 3. The structure of permanent block PB_{h+1}.

In the block header, $ha(PB_h)$ is the hash value of the last permanent block, *t* is the timestamp when the permanent block was created, and *primary* is the identification number of *PSP* who broadcasts the permanent block for verification. The data portion Ξ consists of a list of Φ transactions tx_1, tx_2, … tx_Φ, each of which records the transaction information published in a temporary block. For example, the information recorded in the first transaction tx_1 of Ξ is extracted by *PSP* from temporary block TB_1. This is illustrated in a box labeled as "Transaction tx_1" that includes the timestamp *t* when the transaction was made, the identification number *p* of the peer who broadcasts the transaction, transaction *TX*, and peer *p*'s digital signature $ds[t, p, TX]_p$. The hash value of the permanent block PB_{h+1}, denoted as $ha(PB_{h+1})$, is defined as in (2).

$$ha(PB_{h+1})=H(ha(PB_h), h, t, primary, \Xi) \quad (2)$$

where *H* is a hash function and Ξ is a list of Φ transactions. The list of $ds[PB_{h+1}]_v$ are digital signatures signed by super peers during the consensus process, which is used to check the validity of the new permanent block PB_{h+1}.

To make a blockchain space and time efficient, we need to combine the temporary blocks into a permanent one when there have been Φ temporary blocks published in the blockchain. Algorithm 2 shows the process how *PSP* generates a permanent block PB_{h+1} by combining a list of temporary blocks.

Algorithm 2: Generation of a permanent block by combining a list of temporary blocks

Input: A list of temporary blocks <TB_1, TB_2, …, TB_Φ>
Output: A permanent block PB_{h+1} signed by *PSP*

1. Let *h* be the height of the current blockchain
2. Let Φ be the number of temporary blocks linked to PB_h
3. Let PB_{h+1} be the permanent block to be created
4. Compute the hash value $ha(PB_h)$ of the last permanent block
5. Set current timestamp *t* and identification number *primary*
6. Add [$ha(PB_h)$, *h*, *t*, *primary*] into PB_{h+1} as its block header
7. **for** $i = 1$ **to** Φ
8. extract [*t*, *p*, *TX*] and $ds[t, p, TX]_p$ from TB_i, and add them into data Ξ of PB_{h+1} structure as transaction record tx_i
9. Calculate hash value for PB_{h+1}, i.e., $ha(PB_{h+1})$ as in (2), and add it into PB_{h+1} structure
10. Calculate the digital signature $ds[PB_{h+1}]_{primary}$, and add it into the list of digital signatures of super peers for PB_{h+1}
11. **return** PB_{h+1}

In Algorithm 2, *PSP* first calculates the hash value $ha(PB_h)$ of the last permanent block PB_h. It then sets *t* to the current timestamp and *primary* to its identification number. The four items [$ha(PB_h)$, *h*, *t*, *primary*] are added into PB_{h+1} as its block header. Next, *PSP* extracts the transaction information from each temporary block and adds them into PB_{h+1} as data Ξ of the block. The hash value $ha(PB_{h+1})$, the third component of PB_{h+1}, is calculated according to (2). Finally, *PSP* signs PB_{h+1}, adds its digital signature $ds[PB_{h+1}]_{primary}$ into the $ds[PB_{h+1}]_v$ list, and returns the newly constructed permanent block PB_{h+1}. Once PB_{h+1} has been generated by *PSP*, it sends it to all other super peers to start the consensus process. If PB_{h+1} is approved by a major vote from the super peers, *PSP* deletes all temporary blocks in the blockchain and adds the new permanent block PB_{h+1} right after PB_h. The changes are then broadcast to all peers including regular peers for blockchain updating.

IV. A Variation of the pBFT Consensus Mechanism

We revised the pBFT consensus algorithm [3] to make it a suitable consensus mechanism for the private blockchain system we introduced in this paper. Some major revisions of the algorithm include a combination of the "Prepare" and "Commit" phases of the original pBFT algorithm into a "Verification & Approval" phase, called V&A. We further revised the "Reply" phase of the original pBFT algorithm into a majority "Voting" phase. The revised pBFT consensus algorithm can be used to approve either a temporary block or a permanent one, announced by *PSP*, for timely publication in a private blockchain. The only differences for the two usages are the ways to verify a new block. For a temporary block, the transaction recorded in the temporary block needs to be verified; while in the case of a permanent block, a list of recorded transactions must be verified to ensure they have been published in temporary blocks. In the following, we use the approval process of a permanent block as an example to show how the revised pBFT consensus algorithm works.

Let Θ be the total number of super peers including *PSP*, who are involved in the consensus process. Fig. 4 shows an example of the consensus group with 5 super peers, i.e., $\Theta = 5$, where *PSP* is the primary one, *SP*1, *SP*2 and *SP*3 are super peers that are all up and running but *SP*4 is currently down and not available. The consensus algorithm can be divided into four phases, which are the *Announcement* phase, the *V&A* phase, the *Voting* phase, and the *Notification* phase. In the *Announcement* phase, *PSP* broadcasts a message that contains a newly generated permanent block PB_{h+1} signed by *PSP*. Upon receiving PB_{h+1}, super peers *SP*1, *SP*2 and *SP*3 start to verify the block to make sure all information included in PB_{h+1} is correct. This would include the verification of each transaction record tx_i, where $1 \le i \le \Phi$, has been published in temporary block TB_i. Note that the verification steps (denoted as shaded bars in Fig. 4) can be performed by the super peers concurrently, but would take different amount of time depending on the performance of the machines on which the super peers are running.

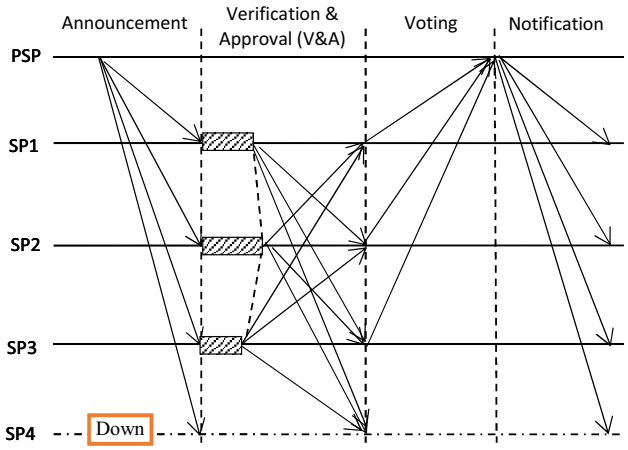

Fig. 4. A variation of the pBFT consensus mechanism.

Once a super peer SP_k completes its verification step and if there is no error in PB_{h+1}, SP_k signs PB_{h+1} and add its digital signature $ds[PB_{h+1}]_k$ to the list of digital signatures of super peers for PB_{h+1}. The new permanent block PB_{h+1} with SP_k's digital signature is broadcast by SP_k to all super peers except *PSP*. Then SP_k waits for signed PB_{h+1} from other super peers before it approves or rejects PB_{h+1}. Super peer SP_k approves PB_{h+1} only when the following conditions are satisfied:

1. PB_{h+1} contains digital signature of *PSP*
2. SP_k has verified PB_{h+1} and added its digital signature
3. SP_k has received ($\lfloor\Theta/2\rfloor$-1) messages from other super peers with signed PB_{h+1}

This would require a total number of $\lfloor\Theta/2\rfloor$+1 digital signatures for SP_k to approve PB_{h+1}. If the new block is approved by SP_k, SP_k enters the "Voting" phase and sends its approval vote to *PSP*. When *PSP* receives at least $\lfloor\Theta/2\rfloor$ approval votes from other super peers, the total number of votes ($\lfloor\Theta/2\rfloor$+1) including its own vote represents a major vote; thus PB_{h+1} is officially approved. In this case, *PSP* replaces all temporary blocks by PB_{h+1} in the blockchain, enters the "Notification" phase, and notifies all super peers and regular peers to update their blockchains. Algorithm 3 summarizes the process how super peer SP_k approves a new permanent block PB_{h+1}.

Algorithm 3: Verification and approval of a new permanent block by super peer SP_k

Input: A new permanent block PB_{h+1} announced by *PSP*
Output: *approval* or *rejection*

1. Let Φ be the number of transactions recorded in PB_{h+1}
2. **if** $ds[PB_{h+1}]_{primary}$ is not valid, **return** *rejection*
3. **for** $i = 1$ **to** Φ
4. **if** transaction record tx_i has *not* been published in TB_i
5. **return** *rejection*
6. **else if** tx_i contains errors
7. **return** *rejection*
8. Calculate the digital signature $ds[PB_{h+1}]_k$, and add it into the list of digital signatures of super peers for PB_{h+1}
9. Broadcast PB_{h+1} with digital signatures $ds[PB_{h+1}]_{primary}$ and $ds[PB_{h+1}]_k$ to all other super peers except *PSP*
10. **while** (*not* timeout)
11. **if** received signed PB_{h+1} from at least ($\lfloor\Theta/2\rfloor$-1) super peers
12. **return** *approval*
13. **else return** *rejection*

Note that in Algorithm 3, if super peer SP_k approves the new permanent block, it will send an *approval* vote to *PSP* along with its digital signature $ds[PB_{h+1}]_k$; otherwise, a *rejection* vote will be sent to *PSP* without the need for attaching any digital signature. Upon receiving at least $\lfloor\Theta/2\rfloor$ approval votes from other super peers, *PSP* is now responsible for adding all digital sigatures into the list of digital signatures of super peers for PB_{h+1} before publishing PB_{h+1} in the blockchain.

V. Case Study

To illustrate the feasibility and the effectiveness of our approach, we implemented a prototype COVID-19 tracking system with case and testing information recorded in a private blockchain. In the system, the results of every test case are broadcast by a peer hospital and published in a blockchain. As such, the published case information can be shared by local hospitals and reliably maintained in the blockchain with no risks of being forged or manipulated by attackers.

A. Experimental Setup

The computers running in the private blockchain system are connected with each other as a peer-to-peer (P2P) network. The system automatically broadcasts the most recent version of the blockchain over the network, and each peer can update its local copy of the blockchain from the P2P network. As the prototype blockchain-based COVID-19 tracking system was developed as a private blockchain system, we limit the total number of peers to no more than 300, and the total number of reported cases to no more than 1,000 per day. In our experiments, we set the number of super peers from 10 to 150. The number of regular peers will *not* affect our experimental results because only super peers are involved in the consensus process. When a peer hospital, either a regular peer or a super one, broadcasts a case (a transaction), *PSP* creates a temporary block and starts the consensus process. If the temporary block is approved by a majority vote, it will be published in the blockchain in a timely manner. When there is a predefined number Φ of published temporary blocks, *PSP* combines them into a permanent one. Upon approval by a major vote for the the permanent block, *PSP* removes all temporary blocks from the blockchain and replace them by the new permanent block. In the following experiments, we consider different scenarios when Φ is 10, 20 and 30. The P2P system is running on a network with 10 machines, each of which runs Windows 10, configured with 16GB of RAM and a 512G hard drive. To conduct the experiments, each machine runs no more than 30 peers.

To simulate a real P2P network, we assume the network latency for communications between peers is randomly distributed over range [10, 300] milliseconds (ms). The time for checking a digital signature signed by a peer follows the normal distribution $N(\mu_1, \sigma_1^2)$, where $\mu_1 = 500$ms and $\sigma_1 = 150$. Verifying a transaction recorded in a new permanent block requires reading a temporary block from a hard disk. We assume the reading time follows the normal distribution $N(\mu_2, \sigma_2^2)$, where $\mu_2 = 2000$ms and $\sigma_2 = 500$. In the following sections, we present the experimental results for consensus latency, consensus failure rate and publication rates of temporary and permanent blocks.

B. Consensus Latency

In this experiment, we show how the number of super peers may affect the consensus latency. We assume all super peers are up and running. We record the latency time for consensus process, which is the period between the time when *PSP* starts the consensus process and the time when a new block is published. We conducted experiments for three cases, namely a temporary block with 1 transaction only, a permanent block with 10 transactions, and a permanent block with 20 transactions. Fig. 5 shows the consensus latency using our revised pBFT algorithm. From the figure, we can see that in all cases, with more super peers, the consensus latency increases. This is because for a temporary block, the consensus latency is mainly due to the verification time used by super peers to verify digital signatures signed by other super peers. If the total number of super peers is Θ, each super peer needs to verify at least $\lfloor \Theta/2 \rfloor$ digital signatures including the one signed by *PSP*. This explains why the consensus latency for a temporary block increases with more super peers. For a permanent block, the consensus latency consists of two major parts: the verification time for digital signatures signed by other super peers, and the reading time for published temporary blocks from a hard disk, which is needed for verifying if the transactions included in the new permanent block are valid. The more transactions that need to be verified, the longer consensus latency. This explains when a permanent block contains 20 transactions, it would take longer time to complete the consensus process than in the case when a permanent block contains only 10 transactions. It also explains why a temporary block can be published most efficiently among the three cases.

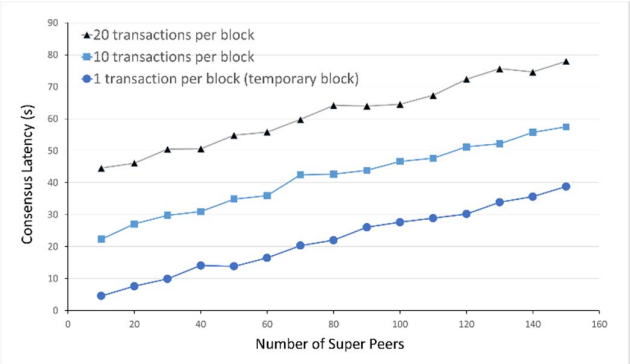

Fig. 5. Consensus latency using the revised pBFT algorithm.

C. Consensus Failure Rate

After *PSP* initiates the consensus process for a new block, at least $\lfloor \Theta/2 \rfloor$ super peers must vote for approval; otherwise, it is considered a consensus failure. However, a super peer may not be able respond with an approval vote due to various reasons including network latency and device corruption. In this experiment, we study the impact of the number of super peers on the consensus failure rate when each super peer (except *PSP*, who is considered a reliable one) has certain chances of no response at 40%, 50% and 60%. The number of super peers range from 10 to 150. For each case, we run the consensus process for 1000 times and calculate the average consensus failure rate. Fig. 6 shows the experimental results.

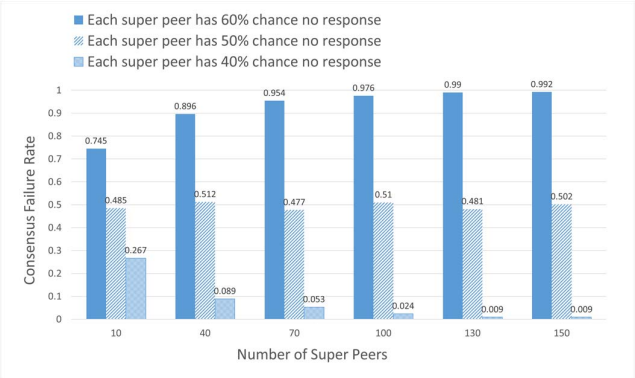

Fig. 6. Consensus failure rate over the number of super peers.

From the figure, we can see that with more super peers, the consensus failure rate increases when each super peer has 60% chance of no response. Since our revised pBFT algorithm adopts a majority voting scheme, when the number of super

peers is low (e.g., 10), the results show that there would be chances for *PSP* to receive at least 5 approval votes to declare a successful consensus process. However, when the number of super peers goes up (e.g., 150), it becomes nearly impossible for *PSP* to receive at least 75 approval votes for a successful consensus process. This might sound strange as one would expect the failure rate remains constant regardless of the number of super peers, but our experiment results do follow the mathematical calculation of the successful rate using (3), where θ is the total number of super peers and p is the chance of no response from each super peer except the primary one.

$$f(\theta) = \sum_{k=\lfloor \theta/2 \rfloor}^{\theta-1} [C_k^{\theta-1}(1-p)^k p^{\theta-k-1}] \quad (3)$$

On the other hand, the experimental results show that with more super peers, the consensus failure rate decreases when each super peer has 40% chance of no response. When the number of super peers goes up (e.g., 150), it becomes almost constantly possible for *PSP* to receive at least 75 approval votes for a successful consensus process. Finally, the results show that the consensus failure rate is around 0.5 when each super peer has 50% chance of no response. This implies the number of super peers have no significant impact on the consensus failure rate in this case. Based on the experiment results, we can conclude that when the chance of no response from the super peers is reasonable low (e.g., below 40%), with enough number of super peers, the revised pBFT algorithm can always perform reliably for successful consensus of a new block.

In more realistic scenarios, suppose each super peer (except the primary one) has 10% chance of no response; however, when a super peer responds, it may reject a new block due to various reasons, e.g., errors found in the new block and being dishonest. Such situations will inevitably affect the successful rate of the consensus process. In the following experiment, we assume each super peer has 10% chance of no response, and when it responds, it will have 30%, 40% or 50% chance of rejection. Fig. 7 shows the consensus failure rate over the number of super peers in the three different scenarios.

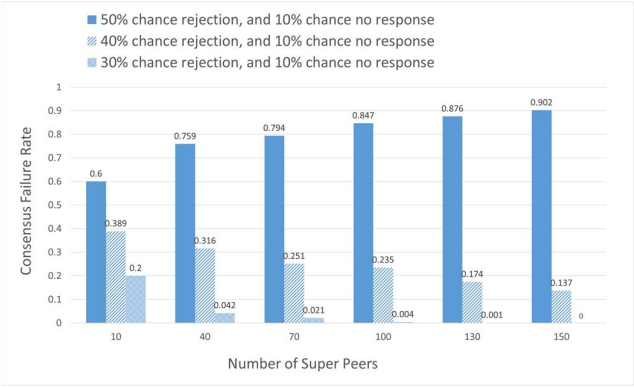

Fig. 7. Consensus failure rate with chances of rejection.

The experiment results match those of our previous experiment. When the chance of rejection is 50%, the probability of approving a new block by a super peer with 10% chance of no response would be 45%; therefore, with more super peers, the consensus failure rate increases. In the other two scenarios, the probabilities of approving a new block by a super peer are all above 50%; thus, with more super peers, the consensus failure rate decreases. With the above experimental results, we can further conclude that when the chance of no response and the chance of rejection from the super peers are reasonable low (e.g., 10% and 30%, respectively), with enough number of super peers, the revised pBFT algorithm can always perform reliably for successful consensus of a new block.

D. Publication Rate of Temporary and Permanent Blocks

We now demonstrate how temporary blocks and permanent blocks can be published timely in a private blockchain. Note that with private blockchain, we do not expect a huge volume of cases reported per day. In the following experiments, we set the number of super peers (i.e., trusted hospitals) to 80. We assume there are about 1,000 cases per day to be published in the blockchain. Table I shows the number of cases generated every 4 hours during a day, with more cases reported during the daytime.

TABLE I. NUMBERS OF CASES GENERATED DURING A DAY

Hours	0-4	4-8	8-12	12-16	16-20	20-24
No. of Cases	100	200	200	300	100	100

We test our blockchain system in three different scenarios, where a permanent block contains 10, 20 and 30 records from temporary blocks, respectively. Fig. 8 shows the number of permenant blocks published every 4 hours during a day.

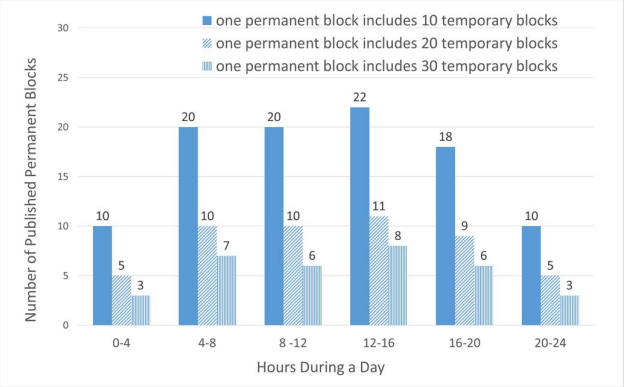

Fig. 8. Number of permanent blocks published during a day.

From Fig. 8, we can see that almost all reported cases can be published timely except for the 12-16 time period, where about 60-80 cases have to be delayed for publication in the following time period 16-20. This is not a major problem because in our blockchain-based COVID-19 case recording and tracking system, cases reported earlier will be processed and published earlier. Therefore, the delay was actually within a couple of hours rather than 4 hours. In all three scenarios with different numbers of records in a block, the numbers of cases that can be published during a time period are very close.

Figure 9 shows the number of temporary blocks published every 4 hours during a day. Similarly, we can see that all cases can be published timely except for the 12-16 time period, where 72 out of 300 cases generated during that period have to be delayed for publication in the following time period 16-20. For

the same reason as described for permanent blocks, this is also not a major issue as a couple of hours delay for publishing temproary blocks is generally acceptable in our blockchain-based COVID-19 case recording and tracking system.

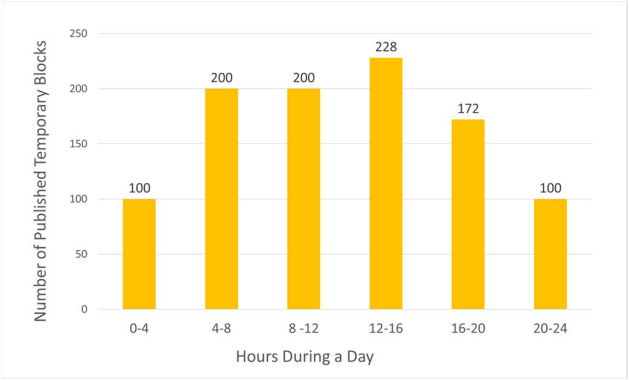

Fig. 9. Number of temporary blocks published during a day.

It is worth noting that when the number of cases has been signficantly reduced, e.g., 20 cases per day as shown in Table II, those cases cannot be published in a timely manner without utilizing temporary blocks as in our approach. For example, when a permanent block contains 20 cases, a case reported during a day may have to be delayed for 24 hours before it can be published. However, using our approach, a reported case can always be published immedidately using a temporary block.

TABLE II. REDUCED NUMBER OF CASES GENERATED DURING A DAY

Hours	0-4	4-8	8-12	12-16	16-20	20-24
No. of Cases	2	4	4	6	2	2

VI. CONCLUSIONS AND FUTURE WORK

Blockchain technology has been successfully applied in many different domains; however, there is very little work on timely publication of transaction records in a private blockchain system. In this paper, we introduce an approach that facilitates timely publication of new transaction records using temporary blocks, which can be combined and published in a permanent block later when there is a predefined number of published temporary blocks. This approach differs from traditional blockchain-based systems where new transaction records are published either in fixed time cycles or in permanent blocks only. Our approach not only allows publishing time-senstive transaction records in a timely manner using temproary blocks, but also reduces the blockchain storage space and potentially supports efficient transaction queries. In addition, we have introduced a variation of the pBFT consensus mechanism that requires a majority vote for approval of either a new temporary block or a permanent one to be added into a private blockchain.

In future work, we plan to improve the block merging mechanism to allow for dynamic determination of a reasonable number of transaction records in a permanent block. Our current approach assumes a primary super peer is always reliable for accomplishing its tasks; however, in a real scenario, a primary super peer may also fail for unexpected reasons. To address this issue, we will further improve our revised pBFT consensus mechanism and show how to efficiently and effectively replace a primary super peer in real time when it is found to be disqualified for its tasks due to various reasons such as being not reliable or having been found dishonest. One related challenge in addressing this issue is to effectively monitor the reliality of super peer software components as in previous work [15]. Finally, we plan to implement a fully functioning private blockchain system for timely publication of transaction records that can be applied in various critical and time-sensitive application domains.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009. Retrieved on March 5 2019 from https://bitcoin.org/bitcoin.pdf

[2] S. King and S. Nadal, "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake," August 19, 2012. Retrieved on March 5 2020 from https://decred.org/research/king2012.pdf

[3] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," In *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI'99)*, New Orleans, LA, USA, February 22-25, 1999, pp. 173-186.

[4] J. D. Bruce, "The Mini-Blockchain Scheme," Cryptonite, Mini-blockchain Project, Revision 3, March 2017. Retrieved on January 18, 2020 from http://cryptonite.info/files/mbc-scheme-rev3.pdf

[5] J. Hooff, M. F. Kaashoek, and N. Zeldovich, "Versum: Verifiable Computations Over Large Public Logs," In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS 2014)*, Scottsdale, Arizona, USA, November 3-7, 2014 2014. Available: https://people.csail.mit.edu/nickolai/papers/vandenhooff-versum.pdf

[6] S. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain," In *Proceedings of the Italian Conference on Cyber Security (ITA-SEC18)*, Milan, February 6-9, 2018, pp. 1-11.

[7] H. Pervez, M. Muneeb, M. Irfan, and I. Haq, "A Comparative Analysis of DAG-Based Blockchain Architectures," In *Proceedings of the 12th International Conference on Open Source Systems and Technologies (ICOSST)*, December 19-21, 2018, Lahore, Pakistan, pp. 27-34.

[8] S. Popov and Q. Lu, "IOTA: Feeless and Free," *IEEE Blockchain Technical Briefs*, IOTA Foundation, January 2019. Retrieved on June 1, 2020 from https://blockchain.ieee.org/technicalbriefs/january-2019/iota-feeless-and-free

[9] N. Nicol and H. Xu, "A Blockchainless Approach for Trusted Public Construction Bidding," *Computer Science Technical Report*, Computer and Information Science Department, University of Massachusetts Dartmouth, December 2018.

[10] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol," *IACR Cryptology ePrint Archive*, 2016, pp. 1159.

[11] Y. Sompolinsky and A. Zohar, "PHANTOM: A Scalable BlockDAG Protocol," *IACR Cryptology ePrint Archive*, 2018, pp. 104.

[12] D. Fullmer and A. Morse, "Analysis of Difficulty Control in Bitcoin and Proof-of-Work Blockchains," In *Proceedings of the 57th IEEE Conference on Decision and Control (CDC)*, Miami Beach, FL, USA, December 17-19, 2018, pp. 5988-5992.

[13] V. Buterin, "Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform," *Online Resource for the Ethereum Community*, Ethereum for Enterprise, 2013. Retrieved on June 20, 2020 from https://ethereum.org/whitepaper/

[14] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, 1978, pp. 120-126.

[15] J. Rahme and H. Xu, "Dependable and Reliable Cloud-Based Systems Using Multiple Software Spare Components," In *Proceedings of the 14th IEEE International Conference on Advanced and Trusted Computing (ATC 2017)*, August 4-8, 2017, San Francisco, CA, USA, pp. 1462-1469.