IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 12, 2019, accepted November 25, 2019, date of publication December 4, 2019,
date of current version December 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957496

Reducing the Bandwidth of Block Propagation in Bitcoin Network With Erasure Coding

MING JIN, XIAOJIAO CHEN, AND SIAN-JHENG LIN, (Member, IEEE)
CAS Key Laboratory of Electromagnetic Space Information, School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China

Corresponding author: Sian-Jheng Lin (sjlin@ustc.edu.cn)

This work was supported in part by the Hundred Talents Program of Chinese Academy of Sciences, and in part by the Natural Science Foundation of Anhui Province under Grant BJ2100330001.

ABSTRACT With the popularity of Bitcoin, there is a greater demand for the scalability of the Bitcoin blockchain, which is susceptible to the efficiency of block propagation. In the Bitcoin blockchain, efficient block propagation approach can reduce the computing power and the risk of forks. Meanwhile, larger blocks help to improve the throughput of transactions. Thus, the block propagation is a major issue of the scalability of the Bitcoin network. This paper introduces a method to reduce the required bandwidth of block propagation with erasure coding. To begin with, the network nodes are classified into several clusters. When a node wants to propagate a block, the node does not need to propagate the whole information of the block. Instead, the node can only transmit the transaction IDs and the coded information to each cluster. The simulation shows that the proposed method can significantly ease the network traffic among these clusters.

INDEX TERMS Blockchain, block propagation, clustering evaluation.

I. INTRODUCTION

In recent years, blockchain has shown its wide range of applications, such as the digital currency and the smart contract. However, there is a major issue in the scale of the blockchain. In particular, Bitcoin [1] and Ethereum [2] demonstrate that the transaction speed is not sufficient to handle large-scale transactions.

If the miners cannot obtain the latest block header promptly, they will waste the computing power. Thus, the performance of propagating blocks is a core metric of the scalability of the Bitcoin network, and the transmission delay is generally considered as one of the major bottlenecks. A larger block usually requires more time for propagation, and the block is more likely to be a orphan block.

Though the Bitcoin network allows each node A to propagate the entire information of a block L, the mechanism have to consume a lot of bandwidth. It is known that each node maintains a memory pool containing the received transactions that have not yet been packaged into blocks. As a result, most transactions in L may also be stored in the memory pool of each node. Based on this observation, a number of techniques are proposed to reduce the required bandwidth of propagating a block in the Bitcoin network [3]–[5].

These techniques heavily rely on the assumption that all transactions in L are also in the memory pool of each node. However, this assumption is not always true in practical applications. The report [6] indicated that it has the probability 53% that all transactions in L are also in the memory pool (except for the coinbase); it has the probability 17% that there is a transaction in L but not in the memory pool; it has the probability 9% that there are two transactions in L but not in the memory pool. When there are some transactions in L that are not in the memory pool of a node B, the node A have to transmit B these missing transactions indicated by the transaction IDs [3], [4].

In this paper, we focus on the bandwidth of the block propagation when the receiver misses a few transactions in L. Precisely, we propose an erasure-coding based protocol to reduce the bandwidth of transmitting missing transactions. An example to transfer missing transactions is shown in Figure 1, where node A has a block L containing three transactions $\{a, b, c\}$, and the memory pools of other nodes $\{B, C, D\}$ miss a transaction, respectively. In the conventional approach in Figure 1(a), node A needs to know the IDs of missing transactions in each node, and then respectively sends these nodes the missing transactions. Instead, in the proposed approach

The associate editor coordinating the review of this manuscript and approving it for publication was Haruna Chiroma.

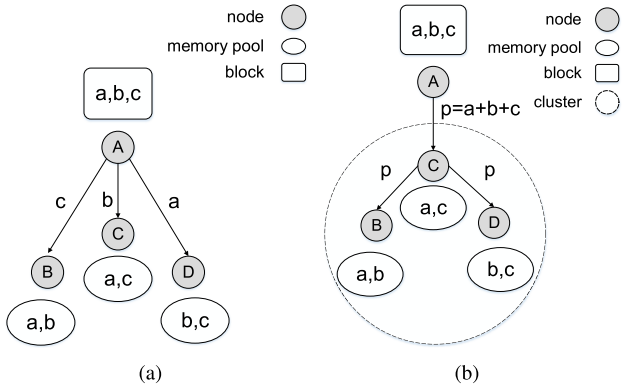

FIGURE 1. An example of the proposed protocol.

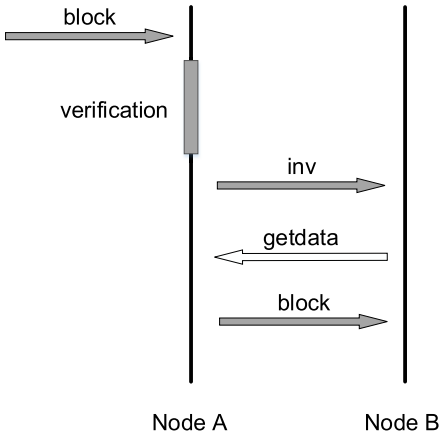

FIGURE 2. Conventional block propagation.

in Figure 1(b), node A transmits other nodes $\{B, C, D\}$ the checksum $p = a + b + c$. Then the receiving nodes can obtain the missing transactions by using p and other existing transactions. This allows that 1). node A does not need to know the IDs of missing transactions in other nodes, and 2). nodes A can save the bandwidth by only sending p to other nodes.

The proposed protocol is based on the following insights. First, the nodes in the Bitcoin network are categorized into a number of clusters. In each cluster, there is a leader to assist the block propagation in the cluster. To propagate a block, the node A first sends transaction IDs to receiver nodes. If these nodes aware missing transactions, they reply the number of missing transactions. Then the node A calculates and sends the parity information of all transactions in the block with erasure coding. After receiving the parity information, the receiver nodes rebuild the block by decoding the missing transactions. The contributions of the paper are enumerated below.

1) Based on erasure coding and clustering, a block propagation protocol is proposed to optimize the bandwidth of the block propagation.
2) The simulation shows that the proposed method has significant bandwidth savings. Precisely, our method can reduce the bandwidth by 40% to 60% compared to existing state of the art method [4].
3) Some issues derived from the proposed method are discussed. Such as a variant of our protocol, which are combined with other block propagation protocols.

The rest of the paper is organized as follows. Section II is the background. Section III describes the proposed protocol, and Section IV gives the simulations. Some issues about the proposed method is discussed in Section V, and Section VI concludes this work.

II. BACKGROUND

A. REED-SOLOMON CODE

Reed-Solomon (RS) code [7] is a class of maximum distance separable (MDS) codes widely used in communication and storage systems. The (n, k) RS code over a Galois field GF(2^m) encodes k message symbols $(m_0, m_1, \ldots, m_{k-1})$ into a n-symbol codeword, where each message/codeword symbol consists of m bits, and $k < n < 2^m$. In particular, RS code possesses an important property that the k message symbols can be completely reconstructed by using any k out of n codeword symbols. If the code is systematic, the first k symbols are the the message symbols and the latter $n - k$ symbols are the parity symbols.

The elements of $GF(2^m)$ are denoted as $\{\alpha_i\}_{i=0}^{2^m-1}$. In encoding, we choose an information polynomial $m(x)$ of degree less than k by using the message symbols. For example, the systematic version can choose a polynomial such that

$$m_i = m(\alpha_i),$$

for $i = 0, 1, \ldots, k-1$. Based on the Lagrange polynomial [8], we obtain

$$m(x) = \sum_{i=0}^{k-1} m_i \prod_{j \neq i} \frac{x - \alpha_j}{\alpha_i - \alpha_j}.$$

Then the codeword is defined as the evaluations of $m(x)$ at n distinct points $\{\alpha_i\}_{i=0}^{n-1}$

$$(m(\alpha_0), m(\alpha_1), \ldots, m(\alpha_{n-1})).$$

The RS codes can tolerance the error-and-erasure channel, and this paper only uses the erasure decoding. In erasure decoding, the decoder receives k symbols $\{y_r = m(r)\}_{r \in R}$ and $|R| = k$, then $m(x)$ can be uniquely determined

$$m(x) = \sum_{r \in R} y_r \prod_{s \in R \setminus \{r\}} \frac{x - s}{r - s}.$$

Then the message symbols are $m_i = m(\alpha_i)$, for $i = 0, 1, \ldots, k-1$.

B. BLOCK PROPAGATION IN BITCOIN AND ETHEREUM

The block propagation mechanism is used in the consensus protocol of blockchain. In the Bitcoin network, the propagation mechanism is based on the advertisement protocol [9], that is shown in Figure 2. When node A receives a block L, A will send the message *inv* to its neighbor nodes once L passes the verification. The *inv* is a message type containing

the hash of L. When the node B receives *inv*, B will check whether L is in the local blockchain. If no, B will send A the message *getdata*. After receiving *getdata*, A delivers the block L to B. This mechanism announces the availability of blocks via sending the message *inv*. This can significantly save the propagation bandwidth, as opposed to directly transferring the entire block.

In contrast to the block propagation of Bitcoin, the Ethereum network uses the push/advertisement hybrid propagation mechanism [10]. Assuming that the node A connects to n neighbor nodes. When node A receives a block L, A sends NewBlockMsg to $\sqrt{n}$ out of n neighbor nodes, where NewBlockMsg is a message type that includes the whole information of L. Further, A also sends NewBlockHashesMsg to the remaining $n - \sqrt{n}$ neighbor nodes, where NewBlockHashesMsg is a message type that includes the hash of L.

C. COMPACT BLOCKS

In the block propagation, most transactions in the block L can be found in the memory pool of the receiver B. Based on this fact, it is possible to further reduce the bandwidth required in the block propagation. That is, the sender A only sends B the block header of L and some necessary information. Then B shall reconstruct the whole block L by utilizing the transactions in the memory pool. However, there is an issue that the block L is associated with which transactions in the memory pool. The Compact Blocks [4] uses transaction IDs to identify these corresponding transactions in the memory pool. The outline of Compact Blocks is as follows.

1) Node A sends node B the message *inv* containing the hash of L.
2) Node B replies node A the message *getdata*(*CMPCT*) to ask for the unknown block.
3) Node A sends B a compact block that contains the 80-byte block header, all the 6-byte short transaction IDs in the block, and some additional transactions that A expects that B does not have yet.
4) Node B tries to reconstruct the block. If B is still missing some transactions, a message *getblocktxn* containing the IDs of missing transactions is sent back.
5) Node A sends these missing transactions.
6) Node B reconstructs the new block and validates it completely.

D. RELATED WORKS

As described in Section II-C, the receiver B has most transactions in the block L, and Compact Blocks utilizes transaction IDs to identify the transactions in the memory pool of B. There are some other approaches. For example, Xthin [3] uses a Bloom filter [11] and 64 bit transaction hash to ensure smaller thinblock size. Graphene [5] also relies on highly synchronized memory pools among participating nodes. It effectively resolves the problem of set reconciliation in the p2p network by coupling a Bloom filter with an IBLT [12]. Xthinner [13], a new block propagation protocol, leverages the benefits of lexical transaction ordering rule (LTOR). For the ordered transactions in a block, senders can identify the block by using one or two bytes per transaction in most cases.

In addition, the clustering approaches can reduce the transaction propagation delay in Bitcoin network. In [14], Bitcoin Clustering Based Super Node (BCBSN) protocol is proposed. A set of geographically diverse clusters is generated, and each cluster has a cluster leader that is responsible for maintaining the cluster. Each node connects to one cluster leader and each cluster leader connects to other cluster leaders. This can reduce the unnecessary hops that the transaction passes through. Similarly, Location Based Clustering (LBC) protocol is proposed to increase the locality of connectivity in the Bitcoin network [15]. The communication link cost measured by the distance between nodes is significantly reduced using LBC. In [16], a proximity-aware extension to the current Bitcoin protocol, named Bitcoin Clustering Based Ping Time protocol (BCBPT) is introduced. By grouping Bitcoin nodes based on ping latencies between nodes, BCBPT is more effective at reducing the transaction propagation delay compared to BCBSN and LBC.

Some methods reduce the block propagation delay by choosing the selection of neighbor nodes carefully. The method [17] makes the message propagate in time by selecting the closest neighbor of a node, where the distance between two nodes is measured by its transmission latency. In [18], each node evaluates other nodes by the speed of block delivery, and the nodes with good scores are acted as its neighbors.

Some studies consider the synchronization of the memory pools among Bitcoin nodes. In [19], a prioritized data synchronization protocol, called FalafelSync, is proposed. This protocol periodically synchronizes the transactions that are most likely to be included in upcoming blocks, and this achieves superior performance than Compact Blocks and Graphene.

Some works use coding schemes for scalability improvements in the blockchain systems. Velocity [20] is a block propagation approach based on fountain codes [21]. By supporting increased block sizes without sacrificing consensus guarantees, Velocity provides significant increases in transaction throughput. In [22], a polynomially coded sharding (PolyShard) scheme is introduced to achieve the scaling efficiency and the security simultaneously. PolyShard allows each node to store and compute a coded shard generated by linearly mixing uncoded shards. In addition, the work [23] proposes a novel network-coded Practical Byzantine Fault Tolerant (PBFT) consensus protocol. The consensus protocol improves the scalability by reducing the maximum required bandwidth between nodes.

III. PROPOSED PROTOCOL

As addressed in Section II-C, Compact Blocks relies on a fact that most transactions in the block L can be found in the memory pool of the receiver B in the block propagation. However, as stated in Section I, the probability that the memory pool

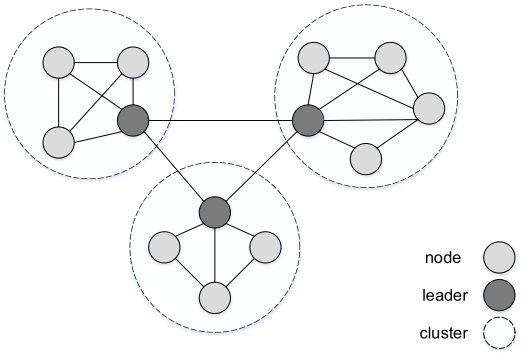

FIGURE 3. Example for a network topology after clustering.

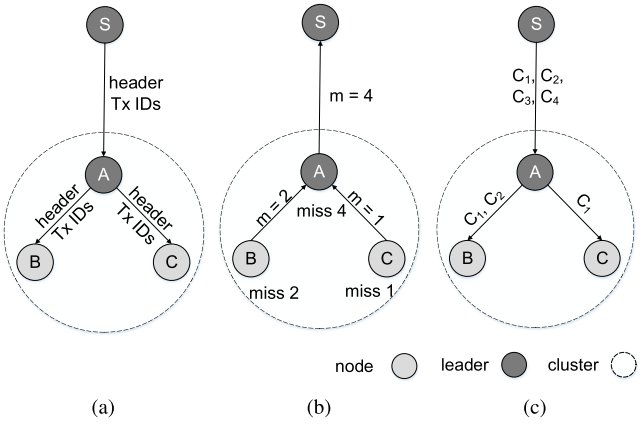

FIGURE 4. An example of block propagation.

misses at least one transactions in L is 47% [6]. The proposed protocol aims to reduce the bandwidth of propagating these missing transactions. The proposed protocol consists of two stages. The first stage clusters the network nodes into several non-overlapping groups, and each group has a leader node. Figure 3 shows an example that the 13 nodes are clustered into three non-overlapping groups. In the second stage, the sender node broadcasts a block to other nodes. Figure 4 gives an example that leader node S wants to sends a block L to a cluster. First, Figure 4(a) shows that S sends A the block header and the transaction IDs. Then A broadcasts the information to other nodes in the cluster. Second, Figure 4(b) shows that each node checks the number of missing transactions m, and sends m to its leader A of the cluster. Then A sends S the maximum of these ms. In this case, the value is 4. Third, Figure 4(c) shows that S sends A the parity part $\{C_i\}_{i=1}^{4}$ of $(k+4, k)$ MDS codes. The MDS code uses all transactions of L as the input message. Then A respectively sends the parity part to each node by the value m.

The following describes the details. Section III-A introduces Packet-level Reed-Solomon codes used to encode transactions, and Section III-B presents the clustering stage. After grouping the nodes in the blockchain network, the block transmission can be divided into two categories, termed the intra-cluster block propagation and the inter-cluster block propagation. These are respectively described in Section III-C and Section III-D.

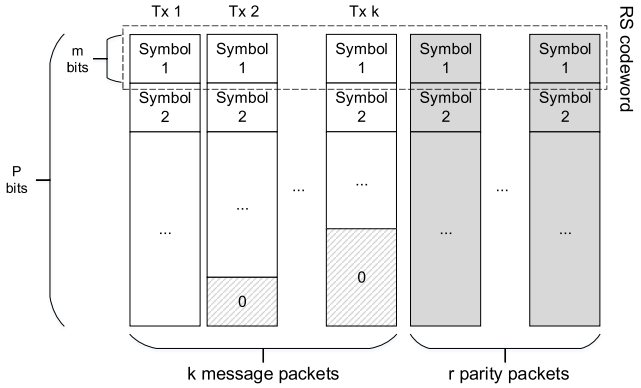

FIGURE 5. The PRS codes used in the proposed protocol.

A. PACKET-LEVEL REED-SOLOMON CODES

The proposed protocol makes use of an erasure-correcting code to encode the transmitted packets. This subsection introduces the Packet-level Reed-Solomon (PRS) code [24]. A (n, k) PRS code over a Galois field GF(2^m) consists of k message packets and $r = n - k$ parity packets, and $2^m > n$. As shown in Figure 5, a (n, k) PRS code can be viewed as a $(P/m) \times n$ matrix over GF(2^m), where each column is a P-bit packet, and each row is a codeword of (n, k) RS codes. The sender transmits these n packets. After receiving any k of these n packets, the receiver can reconstruct all message packets by the RS decoder.

Figure 5 gives the details of the code used in the proposed protocol, where each message packet is a transaction of a block L. As a block has around a couple thousand transactions [25], we can choose $m = 16$ to meet the requirement $2^m > n$. In addition, as the length of transactions are variant (In general, the size of a transaction is between 500 and 800 Bytes [19]), it is necessary to align the size of transactions. Thus, the value P is defined as the largest transaction in a block (In Figure 5, the largest one is $Tx1$). Other transactions smaller than P bits are padding with zeros.

B. CLUSTERING STAGE

In the clustering stage, the nodes in the network are divided into a number of clusters, and each cluster has a leader. A number of clustering algorithms are proposed, and most of them can be used in the proposed scheme. In the following, we introduce an applicable approach [14] in the proposed scheme. As shown in Figure 3, the first step is to choose some nodes to act as the leaders via a leader selection scheme. In the second step, each leader connects to the closest nodes to form a cluster.

For security reasons, the leader selection is based on a reputation protocol. The protocol is implemented by referring the weight, that is a positive real number, of each node. The weight is calculated by the number of Bitcoins obtained by the node, as well as the length of online time for the node. The node keeping the largest weight in an area will be chosen as a leader. The scheme ensures that malicious nodes are difficult to impersonate leaders. Further, to incentive

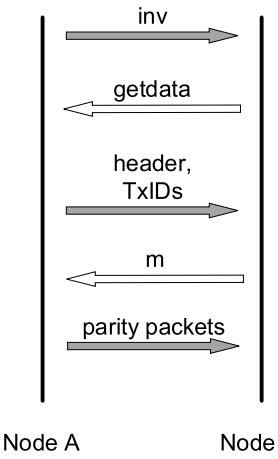

FIGURE 6. Proposed protocol for sending a block from node *A* to node *B*.

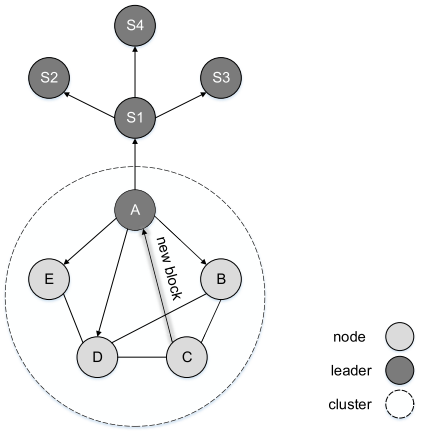

FIGURE 7. An example of intra-cluster block propagation.

information propagation in Bitcoin network, the article [26] introduces an incentive mechanism for leader nodes. The mechanism will reward a leader when it behaves honestly to propagate a valid transaction.

For fairness, leaders should be updated over time. When a node S wants to act as a leader, S will send the node ID and the weight to its neighbors. In addition, this message will also be forwarded to other nodes in the network. When a node K receives this message, this node accepts the invitation if the node S is closer and it has larger weight than the original leader of K.

C. INTRA-CLUSTER PROPAGATION STAGE

To begin with, we consider the block transmission between two adjacent nodes. The following presents that node A wants to send a block L to node B. Figure 6 shows the protocol visually.

1) Node A sends node B the message *inv* containing the hash of L.
2) Node B replies node A the message *getdata* to ask for the unknown block.
3) Node A sends B the header of L and the message *TxIDs*, which contains all the short transaction IDs of the transactions in L. Notably, these short transaction IDs in *TxIDs* possesses the same order of the transactions in block L.
4) Node B starts to find out the missing transactions. If no any missing transactions, B can reconstruct the block L, and the transmission is completed. Otherwise, B sends A the number of missing transactions m, then go to the next step.
5) Node A encodes all the k transactions of L with $(k+m, k)$ PRS codes, and sends B the parity packets $\{C_i\}_{i=1}^{m}$.
6) Node B decodes the missing transactions by performing $(k+m, k)$ PRS decoding on the obtained transactions in Step 4) and $\{C_i\}_{i=1}^{m}$. After that, the new block can be reconstructed.

Notably, in Step 3), the 6-byte transaction ID is used to prevent the Denial-of-Service attacks [4]. In Step 4), node A only need to know the number of missing transactions m. In contrast, Step 4) of Compact Blocks (Section II-C) requires the message *getblocktxn* to identify all missing transactions. Further, as stated in [4], the number of missing transactions in a warmed up node is rarely more than 6, and thus we can consider that m is small in most cases.

Figure 7 illustrates that using the protocol Figure 6 in the intra-cluster block propagation. When node C generates a block L, then C sends the leader A the block L. Then A broadcasts L to other nodes in the cluster and other leader nodes.

D. INTER-CLUSTER PROPAGATION STAGE

An example of the inter-cluster block propagation introduced in the first paragraph of Section III (see Figure 4). The following gives the details of the protocol.

1) Leader S sends leader A the message *inv* containing the hash of L.
2) Leader A replies leader S the message *getdata* to ask for the unknown block.
3) Leader S sends A the header of L and the message *TxIDs*. Then A broadcasts the information to other nodes in the cluster.
4) In the cluster, each node N_i checks the number of missing transactions m_i, and sends m_i to the leader A. Then A sends S the maximum $m' = \max\{m_i\}$.
5) Leader S encodes all the k transactions of L with $(k+m', k)$ PRS codes, and sends A the parity packets $\{C_i\}_{i=1}^{m'}$.
6) Leader A respectively sends each node N_i a portion of parity packets $\{C_i\}_{i=1}^{m_i}$. Then N_i decodes the missing transactions by performing $(k+m', k)$ PRS decoding. After that, the new block can be reconstructed.

Note that, in Step 2), the message *getdata* sent by leader A means that no other nodes in the cluster possess this block. Thus, A does not need to ask other nodes in this cluster for this block.

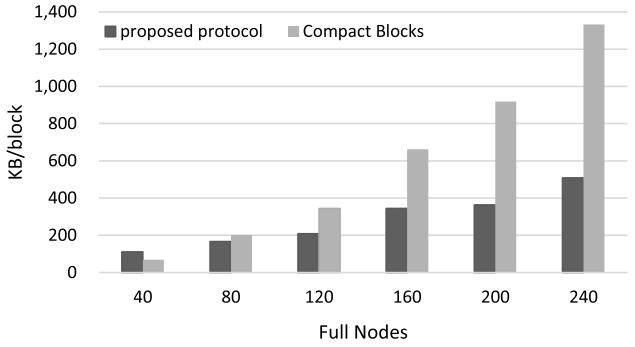

FIGURE 8. The bandwidth for broadcasting a block over the entire network.

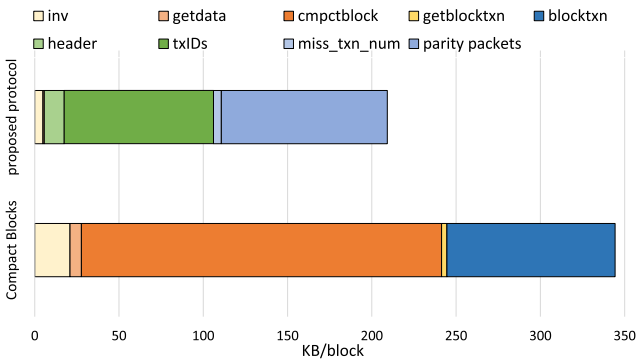

FIGURE 9. Amount of traffic for each message type for 120 nodes.

IV. SIMULATION

The proposed protocol is evaluated via a bitcoin network simulator [27], which is a discrete event-based simulator implemented by Python. In this simulator, the behaviors of nodes, as well as the interactions via the P2P network, can be simulated and recorded, and thus we can extract the relevant information for the simulation. The library Simpy [28] in the simulator is applied to manage the event handing. In addition, we also implement some functions on the simulator to meet the usage of the simulation.

A. CONFIGURATION

In each simulation, 40, 80, 120, 160, 200 or 240 full nodes are generated. Further, we randomly choose 6 of these nodes as miners. In the P2P network, each node has A outgoing connections and B incoming connections, where $1 \leq A \leq 8$ and $1 \leq B \leq 20$ are chosen in random. For comparisons, two block propagation protocols, termed Compact Blocks and the proposed protocol with cluster, are implemented. Each experiment simulated the blockchain activities 3 hours. In the simulation, a block is generated every M minutes, for $M = 2, 4, 6, 8, 10$. In the simulation, the connection latency in a cluster is always less than that between any two distinct leaders. For the leader selection, the miners are appointed as leaders, and the nodes producing the latest block are likely to be the leaders. The number of leaders is 5% of the number of nodes.

B. RESULT

In both protocols, we evaluated the total bandwidth to broadcast a block over the entire network. For each configuration, we record the average value by running the simulation five times. The result is shown in Figure 8. Clearly, the bandwidth is positively correlated with the number of nodes in the network. For a small network, Compact Blocks requires less bandwidth. However, for a larger network, the proposed protocol requires lower bandwidth. For the case 120 nodes in Figure 8, we show the amount of traffic for each message type in Figure 9. It can be seen that the amount of the massages *inv* and *getdata* are significantly reduced.

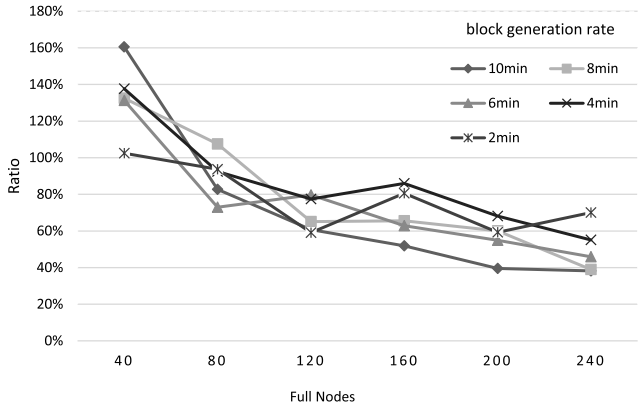

FIGURE 10. The bandwidth saving ratios between the proposed protocol and Compact Blocks.

Let

$$Bandwidth\_Saving\_Ratio = \frac{Bandwidth\_for\_Ours}{Bandwidth\_for\_CB}.$$

Figure 10 shows the bandwidth saving ratios for various number of nodes and various block generation rates. One can see that the proposed scheme performs better when the block generation ratio is 10 min. In particular, the bandwidth saving ratio is significant in a larger network.

The number of missing transactions in memory pools is also a key factor for the block propagation. Notably, the number of missing transactions in a memory pool is determined by

$$Sync.\_Ratio = \frac{\sharp Transactions\_Received\_By\_A\_Node}{\sharp All\_Transactions\_In\_Network}.$$

Figure11 shows the bandwidth saving ratios, for the synchronization ratios 70%, 75%, . . . , 100%. In this simulation, w adjust the transaction verification time by modifying the transaction verification rate. Then the transaction propagation will be influenced so that synchronization of memory pools changes. Figure11 shows that the proposed scheme performs better traffic savings, when the synchronization ratio is lower.

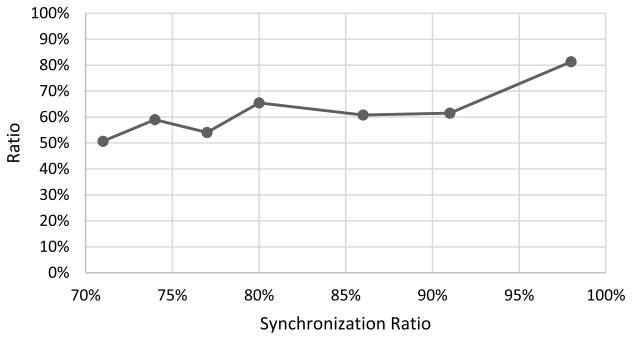

FIGURE 11. The bandwidth saving ratios between the proposed protocol and Compact Blocks of 120 nodes, for various synchronization ratios.

V. DISCUSSION

In this section, we discusses a number of issues derived from the proposed method.

A. OVERHEADS WITH USING OTHER PROTOCOLS

In Bitcoin network, the order of transactions in a block should be specified. If we change the order of transactions, the hashing of the block will be changed, and this leads that the block cannot pass the verification. Thus, in Step 3) of the proposed scheme (Section III-C), node A shall inform node B the order of these transactions in the block L. In this proposed protocol, node A sends node B the message *TxIDs*, that contains all the transaction IDs in correct order. Thus, the new block can be recovered correctly.

In contrast, if we use Graphene [5] to transmit the block, node A shall send B extra $n\log_2(n)$ bits to notice the ordering of transactions, where n is the number of transactions in L. In addition, Xthinner [13] requires that all transactions shall follow the rule LTOR. This causes that there is only one valid ordering for a set of transactions in a block. This limits the flexibility of the protocol in bitcoin blockchain.

B. TRANSMISSION PROTOCOL WITH BLOOM FILTER

LTOR is a rule of sorting the transactions in a block [13]. Precisely, LTOR is the requirement that transactions are sorted numerically (or alphabetically in hex) by transaction IDs. With LTOR, Xthinner [13] introduces a method to further compress the transaction IDs. However, Xthinner can only be applied on Bitcoin Cash (BCH), and BTC network is unable to adopt the default version of Xthinner, as Bitcoin Core (BTC) does not follow LTOR.

To solve this issue, the author [13] presents another version of Xthinner without LTOR. In this case, each sender node should reorder the transactions in a block by LTOR, and the receiver should revert the transaction ordering after receiving the block. Thus, the encoder should also send the ordering information to the receiver. In this way, the new Xthinner can be performed on BTC. The above idea can also be applied to the proposed protocol. The steps are as follows.

1) Node A sends node B the message *inv* containing the hash of L.
2) Node B replies node A the message *getdata* to ask for the unknown block.
3) Node A creates a Bloom filter F seeded with the transactions of L, then sends F, the block header, the ordering information and the number of transactions k.
4) Node B tests each transaction in the memory pool with F to know whether the transaction is in L. If the number of matched transactions is k, B can reconstruct the block L. Otherwise, B sends back a Bloom filter F' seeded with the matched transactions.
5) Node A tests each transaction in L with F' to know whether the transaction is missing for B. Then A sends B the missing transactions.
6) Node B can reconstruct the new block with the ordering information.

The following gives an example to show the bandwidths of transmitting a block with several schemes. Assuming that a block contains $n = 2000$ transactions, and a memory pool contains $p = 6000$ transactions. In this case, Xthin [3] requires $2000 \times 8B = 16KB$ to transmit the hashes. In addition, Xthin creates a Bloom filter with the false positive rate (FPR) $f = 0.0005$. Notably, a Bloom filter with p items inserted and a FPR of f is well known to be $\frac{-p\times\ln(f)}{\ln^2(2)\times 8} = 11.87KB$. Then the total required bandwidth is $16KB + 11.87KB = 27.87KB$. Further, Compact Blocks requires $2000 \times 6B = 12KB$ to transmit transaction IDs.

In contrast, in the propose protocol, the two Bloom filters F and F' requires $\frac{-n\times\ln(f)}{\ln^2(2)\times 8} \times 2 = 7.91KB$. Additionally, the ordering information requires $n\log_2(n)bits = 2.74KB$. In summary, the proposed protocol requires a total of $7.91KB + 2.74KB = 10.65KB$, which is more efficient than Xthin and Compact Blocks.

VI. CONCLUSION

The block propagation is vital for ensuring the scalability of the Bitcoin blockchain. Currently, the propagation mechanisms rely on the synchronized memory pools among participating nodes. However, when the transactions in a new block are not all in the memory pool of each node, the missing transactions need to be transmitted, and this severely affects the efficiency of these propagation mechanisms. In this paper, we introduce a method to reduce the bandwidths of block propagation based on erasure coding and clustering. The simulation shows that the proposed method has better network traffic savings than existing block relay protocols. Some related issues have also been discussed. There are some unsolved issues as follows. First, this paper does not analysis the security of the proposed method, and this is one of the possible future research issue. Second, the transaction propagation mechanism for the erasure-coding aid protocol is another issue.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.
[3] P. Tschipper. (2016). *BUIP010: Xtreme Thinblocks*. Accessed: Jan. 1, 2016. [Online]. Available: https://bitco.in/forum/threads/buip010-passed-xtremethinblocks,
[4] M. Corallo. (2018). *BIP 152: Compact Block Relay*. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki
[5] A. P. Ozisik, G. Andresen, G. Bissias, A. Houmansadr, and B. Levine, "Graphene: A new protocol for block propagation using set reconciliation," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Cham, Switzerland: Springer, 2017, pp. 420–428.
[6] A. Clifford, P. R. Rizun, A. Suisani, A. Stone, and P. Tschipper. (Jun. 2016). *Xthin-On-Chain-Scaling*. [Online]. Available: https://medium.com/@peter_r/towards-massive-on-chain-scaling-block-propagation-results-with-xthin/-3512f3382276#.g50cw43hq
[7] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960.
[8] D. A. Quadling, "Lagrange's interpolation formula," *Math. Gazette*, vol. 50, pp. 372–375, Dec. 1966.
[9] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," in *Proc. IEEE P2P*, Sep. 2013, pp. 1–10.
[10] K. Wüst and A. Gervais, "Ethereum eclipse attacks," ETH Zürich, Zürich, Switzerland, Tech. Rep., 2016.
[11] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
[12] M. T. Goodrich and M. Mitzenmacher, "Invertible Bloom lookup tables," in *Proc. 49th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2011, pp. 792–799.
[13] J. Toomim. (Sep. 2018). *Benefits of LTOR in Block Entropy Encoding*. [Online]. Available: https://medium.com/@j_73307/benefits-of-ltor-in-block-entropy-encoding-or-8d5b77cc2ab0
[14] M. Fadhil, G. Owenson, and M. Adda, "A Bitcoin model for evaluation of clustering to improve propagation delay in Bitcoin network," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE), IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC), 15th Int. Symp. Distrib. Comput. Appl. Bus. Eng. (DCABES)*, Aug. 2016, pp. 468–475.
[15] M. Fadhil, G. Owenson, and M. Adda, "Locality based approach to improve propagation delay on the Bitcoin peer-to-peer network," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 556–559.
[16] M. Fadhil, G. Owenson, and M. Adda, "Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2411–2416.
[17] W. Bi, H. Yang, and M. Zheng, "An accelerated method for message propagation in blockchain networks," 2018, *arXiv:1809.00455*. [Online]. Available: https://arxiv.org/abs/1809.00455
[18] Y. Aoki and K. Shudo, "Proximity neighbor selection in blockchain networks," 2019, *arXiv:1906.00719*. [Online]. Available: https://arxiv.org/abs/1906.00719
[19] N. Younis, M. A. Imtiaz, D. Starobinski, and A. Trachtenberg, "Improving Bitcoin's resilience to churn," 2018, *arXiv:1803.06559*. [Online]. Available: https://arxiv.org/abs/1803.06559
[20] N. Chawla, H. W. Behrens, D. Tapp, D. Boscovic, and K. S. Candan, "Velocity: Scalability improvements in block propagation through rateless erasure coding," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 447–454.
[21] D. J. C. MacKay, "Fountain codes," *IEE Proc. Commun.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.
[22] S. Li, M. Yu, S. Avestimehr, S. Kannan, and P. Viswanath, "PolyShard: Coded sharding achieves linearly scaling efficiency and security simultaneously," 2018, *arXiv:1809.10361*. [Online]. Available: https://arxiv.org/abs/1809.10361
[23] B. Choi, J.-Y. Sohn, D.-J. Han, and J. Moon, "Scalable network-coded PBFT consensus algorithm," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 857–861.
[24] H. Che, Y. Bai, and C. Feng, "Packet-level Reed–Solomon codes over burst-loss channels," in *Proc. IEEE 14th Int. Conf. Commun. Technol.*, Nov. 2012, pp. 180–185.
[25] (Feb. 2018). *Average Number of Transactions Per Block*. [Online]. Available: https://blockchain.info/charts/n-transactions-per-block?timespan=2years
[26] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, "On Bitcoin and red balloons," in *Proc. 13th ACM Conf. Electron. Commerce*, 2012, pp. 56–73.
[27] V. M. Afonso, "Design and implementation of a Bitcoin simulator," Universitat Oberta de Catalunya, Barcelona, Spain, Tech. Rep., 2013.
[28] SimPy Team. (Oct. 2015). *Simpy*. [Online]. Available: http://simpy.readthedocs.org/en/latest/index.html

MING JIN received the B.E. degree in information security from the China University of Geosciences (CUG), Wuhan, China, in 2018. He is currently pursuing the M.Sc. degree with the University of Science and Technology of China (USTC), Hefei, China. His research interest focuses on the scalability of bitcoin's blockchain.

XIAOJIAO CHEN received the B.E. degree in communication engineering from the Nanjing University of Science and Technology (NJUST), Nanjing, China, in 2017. She is currently pursuing the M.Sc. degree with the University of Science and Technology of China (USTC), Hefei, China. Her research interest focuses on blockchain.

SIAN-JHENG LIN (M'16) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2004, 2006, and 2010, respectively. He was a part-time Lecturer with Yuanpei University, from 2007 to 2008, and with Hsuan Chuang University, from 2008 to 2010. From 2010 to 2014, he was a Postdoctoral Researcher with the Research Center for Information Technology Innovation, Academia Sinica. From 2014 to 2016, he was also a Postdoctoral Researcher with the Electrical Engineering Department, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia. He is currently a Project Researcher with the School of Information Science and Technology, University of Science and Technology of China (USTC), Hefei, China. His research focuses on the algorithms for MDS codes and its applications to storage systems.