Trigger-Based Automatic Request for Effective Data Recovery of Erasure-Coded Blockchain Storage

So-Hyun Park, So-Yeon Kim, So-Hui Kim, and Il-Gu Lee*

Department of Future Convergence Technology Engineering, Sungshin Women's University
{220227022, 220237014, 220226034, iglee}@sungshin.ac.kr

Abstract

A blockchain is a decentralized peer-to-peer network in which all the nodes store data in copies; thus, the transactions on the network cannot be changed or deleted, ensuring data integrity. However, this can lead to duplicate data storage, which causes high storage overhead. Distributed storage techniques utilizing erasure code (EC) are being studied to solve this problem. Although EC-based blockchain storage increases storage efficiency, encoded chunks distributed across multiple nodes need to be received to restore and get access to the original blocks in EC-based distributed blockchain storage. However, studies on increasing the data transmission efficiency of EC-based blockchain storage have yet to be conducted. In this study, we have proposed a data transmission technique called the trigger-based automatic request (ARQ) technique. This technique increased the throughput efficiency by 8% compared to that under existing data transmission techniques while maintaining the decentralization of the blockchain. Compared to full-node storage, the proposed technique maximized the storage efficiency of EC-based distributed blockchain storage by more than 99.8%, while solving the recovery overhead problem due to data transmission.

Keywords: blockchain, data recovery, distributed storage, erasure code, trigger

1 Introduction

Web 3.0, which has recently attracted attention as the next generation of the World Wide Web, is an intelligent decentralized web system that can generate and obtain data from individuals. This generation provides a semantic web environment that provides customized information for individual situations and contexts [1]. Although the concept of Web 3.0 is still evolving, and implementation technology has not yet matured, decentralized storage, which is enabled by technologies such as blockchain technology, is essential to implement the core goal of Web 3.0 [2]. The metaverse, which is a representative service that utilizes Web 3.0, requires considerable computing and storage resources for storing content [2]. A blockchain, which is a type of ledger, has been used in various storage-based technologies to ensure data integrity and reliability. Owing to the growth of hyperconnected and superconverged systems such as the Internet of Things (IoT), the amount of data stored in systems such as blockchains has increased [3, 4]. As of April 2023, the average Bitcoin and Ethereum block sizes are 472.13 and 908.41 GB, respectively, indicating the colossal storage space required by nodes participating in these systems [5].

Owing to blockchains needing to ensure reliability and transparency, each participating node stores a copy of the entire blockchain, resulting in its storage burden increasing exponentially [3]. With the participation of more nodes, the storage overhead increases linearly, and more blocks

The 7th International Conference on Mobile Internet Security (MobiSec'23), Dec. 19-21, 2023, Okinawa, Japan Article No. W-3

*Corresponding author: Department of Future Convergence Technology Engineering, Sungshin Women's University, seoul, 02844, Republic of Korea, Tel. +82-02-920-7145 are generated in traditional full-node-based blockchain storage systems that replicate all the blocks per node [3]. The requirements associated with large amounts of storage can significantly hamper the operation of blockchains on resource-constrained systems, such as IoTs [6].

Lightweight blockchain storage techniques such as the light node technique [7], which stores only the block header information of the blockchain, or transaction pruning [8], which deletes old transactions and stores only the latest transactions, have been considered. Each technique requires a separate full node that stores all the blocks, which can compromise the decentralization of a blockchain network. While ensuring the decentralization of blockchain networks, methods that reduce blockchain storage requirements need to ensure the decentralization of blockchain networks, while reducing the storage burden of the participating nodes. Distributing blockchain transaction data using coding has emerged as a representative method for reducing storage overheads.

An erasure code (EC), which is a technique used for the distributed storage of blockchain transaction data, can recover the original data even if part of the data is lost. It divides the original data into k pieces and adds m parity symbols ($k > m$) to generate n encoded chunks [2]. When using the Reed Solomon (RS) code, up to m errors can be detected and up to $m/2$ errors can be corrected among n encoded chunks [2].

Since n nodes store n encoded chunks individually, the EC-based distributed storage method can store data that are resistant to m node failures while reducing the storage overhead by $1/k$ compared with conventional full-node-based storage. While studies have researched various methods for applying an EC to blockchain storage, most of them have rarely considered block recovery or the cost of recovering the entire distributed encoded chunks of the original blocks [9, 10, 11, 12, 13]. In [10], multiple encoded chunk copies were redundantly stored to ensure the recovery of the original data and reduce the recovery cost. However, the storage overhead increased because of the redundant encoded chunks. Additionally, optimization algorithms that have been proposed to reduce storage data redundancy have not considered recovery performance [9, 10, 11, 12, 13].

This study has proposed an effective recovery method called the trigger-based automatic request (ARQ) technique for EC-based distributed blockchain storage. This technique can reduce the communication cost of recovering encoded original blocks, even in environments with node failures.

The contributions of this study are as follows.

- The trigger-based ARQ method, which uses trigger signals much smaller than the encoded chunks, reduces the data transmission overhead for recovering the original blocks from EC-based distributed blockchain storage while maintaining the decentralization of the blockchains.
- The proposed trigger-based ARQ method enables stable data recovery while ensuring a low latency and high-throughput performance, even when node failure is frequent.
- Using the trigger-based ARQ scheme with an EC-based distribution technique, blockchains can reduce storage overhead while effectively accessing the original blocks, overcoming the limitation of conventional EC-based distributed storage.

The remainder of this paper is organized as follows. Section 2 examines previous studies on reducing the data recovery costs of EC-based distributed blockchain storage. Section 3 explains the system model of EC-based distributed blockchain storage and existing efficient block recovery methods. Section 4 presents the evaluation of the performance of non-trigger and trigger-based ARQ methods in terms of total data transmission time and throughput. Finally, section 5 concludes the paper.

2 Related Work

Blockchain systems use a replication-based storage method, in which every node stores the same data to ensure storage reliability. An EC is an alternative to reduce the storage burden on nodes, which need to store large amounts of data [14]. This code can achieve high stability and low overhead, leading to high data recovery costs. Research has been conducted to improve the recovery efficiency, mainly suggesting adjustments to the chunk size or physical distance [15, 16] or proposing new coding methods to improve the data recovery performance [17, 18, 19].

Qiu et al. [17] proposed a hybrid EC-fusion method by integrating a RS code and a minimum storage regeneration (MSR) code, and they improved the storage and recovery costs by using the MSR code when decoding was frequently required. The application response time and recovery performance under the hybrid fusion method were better than those under the MSR code, RS code, and local reconstruction code (LRC), another type of coding scheme. Since the hybrid fusion method only conducted a performance evaluation for the failure recovery of a single chunk, it faced difficulties in confirming the performance during multiple chunk failures.

Liu et al. [18] proposed a Z code that achieved both an optimal recovery bandwidth and minimal storage usage. The exclusive OR (XOR) code attribute is deleted and modified to the generalized Z (GZ) code to obtain the maximum separable distance (MSD) attribute. A limitation of this scheme is that the efficiency reduces when multiple nodes fail. Caneleo et al. [19] proposed an XOR-based code to improve the recovery performance of storage systems distributed across various areas. The proposed code can handle failure efficiently because it requires less parity blocks for recovery than RS-like codes do. However, it generates more parity blocks than the RS code does, resulting in storage overhead.

Shan et al. [15] improved the data recovery performance by adjusting the chunk sizes. Furthermore, by using variable chunk sizes, they improved the recovery performance of code regeneration by 1.85 times more than that by the RS code scheme, while maintaining a low level of read time degradation. Regarding regenerating codes, they are EC codes that minimize the data required for recovery and are recovered in chunk units rather than bytes. Song et al. [16] improved the recovery efficiency using a low-cost node selection strategy that selects nodes with physically short distances and low loads. Additionally, they stored the RS parity symbols in the data node, thereby reducing cross-rack traffic during recovery. They experimentally assessed their strategy; the data recovery time was less than 42.41% compared to that under the RS code scheme and less than 36.58% than that under deterministic data distribution (D3). Despite a reduction in the transmission and recovery time between the server racks, the amount of data transmission and storage costs within some racks still increased.

Few studies have been conducted from the perspective of network communication, which causes a direct load on recovery. Research is required to reduce overhead from the perspective of data transmission and improve the recovery performance for original blocks in EC-based distributed storage systems.

3 System Model

3.1 Erasure Code-based Distributed Blockchain Storage

A RS code can detect up to m errors by adding m parity symbols to k original data and correcting up to $m/2$ errors. When k blocks are stacked, m parities are added to encode the (n, k) RS code and generate $n (n > k)$ encoded chunks. The storage overhead of each node is reduced to $1/k$, because every n node stores only 1 out of n blocks one by one, whereas the full node-based blockchain nodes store all n blocks.

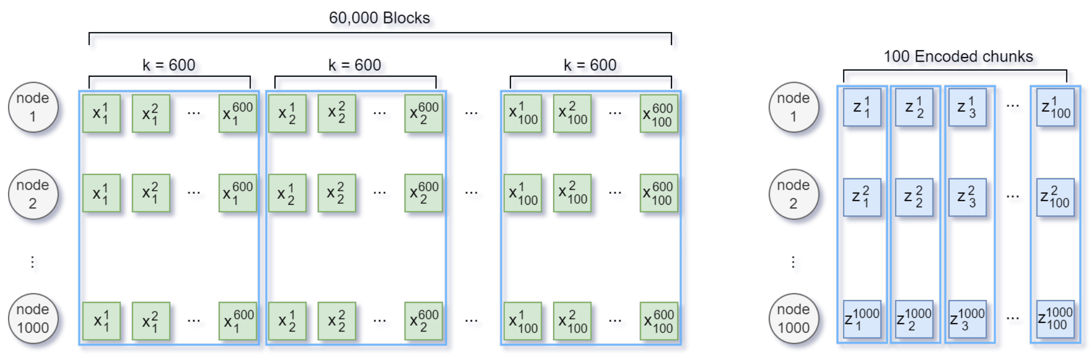

Figure 1: Full node-based blockchain storage and EC-based distributed blockchain storage.

In an encoding round in which k blocks are stacked and encoded by applying an (n, k) RS code, x_r^i $(1 \leq i \leq k)$ refers to the ith block of the rth encoding round, and z_r^j $(1 \leq j \leq n)$ refers to the jth encoded chunk of the rth encoding round. As shown in Figure 1, 60,000 blocks were distributed and stored by applying the $(1000, 600)$ RS code 100 times to a blockchain network of 1,000 nodes. All 1,000 nodes in the existing full-node based blockchain stored 60,000 pieces of data equally. When 600 blocks from x_1^1 to x_1^{600} were stacked, 400 parity symbols were added to apply the $(1000, 600)$ RS code to generate 1,000 encoded chunks from z_1^1 to z_1^{1000}. The encoded chunks z_1^1 to z_1^{1000} were distributed and stored individually in 1,000 nodes. After repeating RS encoding 100 times, the 60,000 blocks were reduced to 100 encoded chunks for a single node, reducing the storage space burden to 1/600.

3.2 Effective Block Recovery

To recover original EC-based distributed blocks, at least k encoded chunks need to be received. Since each node stores a single encoded chunk, the fastest method is to recover the original block by receiving $k - 1$ chunks from the other nodes. However, if a node cannot deliver the data correctly, the encoded chunks are retransmitted until the destination node receives the data correctly. However, as more errors occur under this method, the data transmission time required to recover the original block increases, and the throughput and energy efficiency are degraded.

Encoded chunks require a long transmission time owing to their large data sizes. An original block can be recovered faster and more effectively by using a trigger signal smaller than the encoded chunk. The link status can be recognized first through the trigger signal, and then, the encoded chunk can be transmitted with a high throughput by ensuring the $k - 1$ normal nodes necessary for recovery.

All nodes need to receive encoded chunks from other nodes to recover original blocks; however, there is no protocol for controlling the order or flow of transmission in a decentralized network. All the distributed nodes in a blockchain are source nodes (SNs) and destination nodes (DNs) that simultaneously transmit and receive data, respectively. In real-world distributed networks, multiple nodes simultaneously receive data while requesting them, as shown in Figure 2 (a), resulting in congestion and many potential collisions. There are many other considerations for transmitting data at a low latency while occupying limited channel resources fairly without

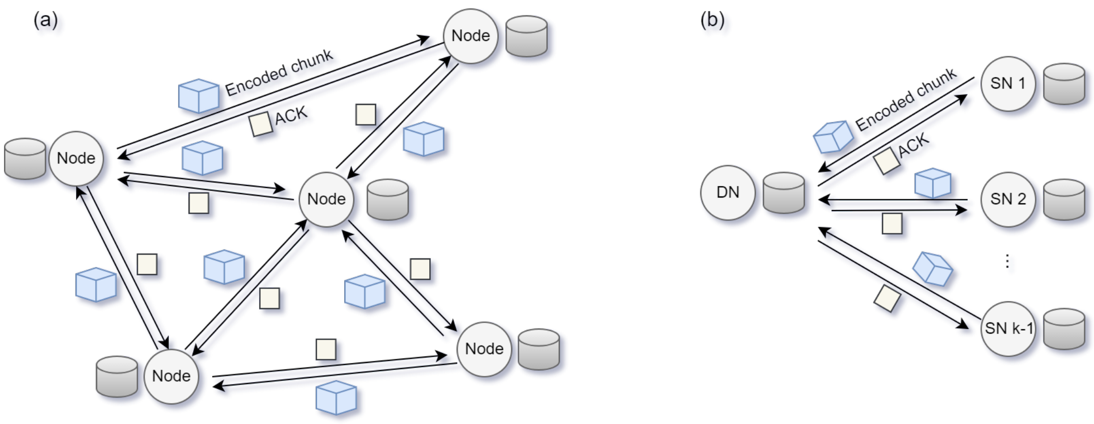

Figure 2: (a) Data transmission of a decentralized peer-to-peer (P2P) blockchain network and (b) encoded chunk transmission between multiple source nodes (SNs) and a single destination node (DN).

conflicts. For example, in a network where data transmission occurs through distributed competition, time duration for channel access should be allocated to all the nodes [20]. In this study, we propose a method for receiving data with a low delay when one node gets the time duration for data reception, as shown in Figure 2 (b). Through channel occupancy competition, a DN receives an encoded chunk from the remaining $k-1$ SNs. If all the $k-1$ SNs send encoded chunks without failure, the original block can be recovered without retransmission. However, if the encoded chunk sent by SN2 is incorrectly transmitted, and the DN does not send an ACK, SN2 needs to send that chunk again, which results in the original block recovery process being delayed.

Various errors can occur during blockchain data transmission. For example, if the data stored on a node are lost, a malicious Byzantine node can send incorrect data, or the received signals can be degraded through network interference, collision, or noise during transmission. In this study, all data transmission failures, such as nodes operating abnormally or leaving the blockchain network, were assumed to be errors caused by *node failure*. Additionally, even if a failed node did not transmit data correctly, it was assumed that it could become a normal node through several retransmission attempts and transmit the data correctly.

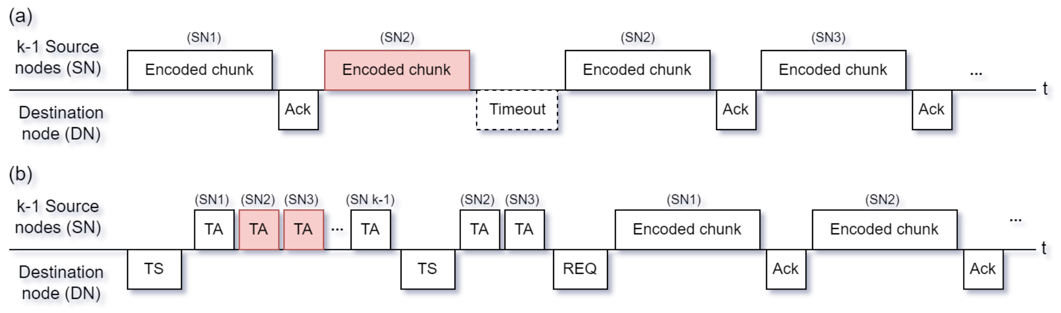

Figure 3: Data transmission process of (a) non-trigger ARQ and (b) trigger-based ARQ.

Figure 3 (a) and (b) compare the operating methods of the non-trigger ARQ and trigger-based ARQ techniques, respectively. The non-trigger ARQ receives encoded chunks sequentially from $k-1$ SNs. If SN2 does not correctly transmit the data, and an ACK is not received from the DN, as shown in Figure 3 (a), the SN2 sends the encoded chunk again and waits for the ACK. If SN2 receives an ACK, SN3 sends the encoded chunk. All nodes can continuously detect whether a channel is in use or transmit data sequentially by computing the link reservation time. As the number of node failures increases, more encoded chunks are retransmitted, resulting in an increase in the total data transmission time and deterioration in the throughput, which represents the actual number of encoded chunks transmitted within unit time.

The trigger-based ARQ technique shown in Figure 3 (b) can reduce the overhead caused by data retransmission, which is a disadvantage of the non-trigger ARQ technique, by retransmitting only the trigger acknowledgement (TA), using the trigger signal (TS) proposed in this study. The DN selects $k-1$ SNs to receive data and includes these SNs in the TS group address field for broadcasting. The SNs that receive the TS send the TA sequentially. If the TA does not arrive at the DN correctly, the DN sends the TS back to the SN to receive the TA. If more than $k-1$ normal TAs are received, the DN sends an REQ to the $k-1$ among the normal SNs to request data and finally starts to receive the encoded chunks sequentially. Since the normal link operation of each node was confirmed through the trigger-based ARQ, the recovery time could be shortened by reducing the number of encoded chunk retransmissions owing to node failure.

The trigger-based ARQ technique reduced the data transmission time needed for recovering EC-based distributed blockchain storage. In an environment with node failures, this technique can improve the overall throughput of data transmission compared to the non-trigger ARQ technique, which performs extensive data retransmission. If the link conditions at the time of triggering remain the same, even an encoded chunk can be transmitted quickly with a low error probability. In general, the data for link congestion control are small because they do not contain much information and are transmitted at low data rates for accurate transmissions even in situations with potentially high network errors. However, because the data containing actual information are large, they are transmitted at fast data rates to improve throughput, resulting in the signal-to-noise ratio of the reception getting impaired. Therefore, when the trigger-based ARQ is applied for transmitting a long encoded chunk, network interference, reception signal degradation due to loss, and data loss may occur due to noise.

Nevertheless, to recover the original data block of the distributed blockchain, it is essential to guarantee at least k normal nodes that can stably transmit encoded chunks; the trigger-based ARQ technique, which identifies normal nodes faster than the non-trigger ARQ technique does, shortens the data recovery process.

4 Evaluation and Analysis

We compared the throughput of data transmission under the non-trigger and trigger-based ARQ methods when distributed encoded chunks were being transmitted from each SN to the DN to recover the original blockchain data. Additionally, the recovery cost and processing performance under the proposed trigger-based ARQ data transmission method were compared with those under different blockchain storage methods: full node-based storage, redundant storage based on the repetition code (RC), and EC-based distributed storage. The trigger-based ARQ technique could overcome the limitations in EC-based distributed storage by simultaneously reducing storage and recovery overheads.

Parameter	Value
Encoded chunk	1 KB
Ack, Trigger Ack (TA)	14 Bytes
Trigger Signal (TS)	20 Bytes
Data rate	5 Mbps

Table 1: Performance evaluation parameters

4.1 Experimental Environment

To evaluate the performance of the trigger-based ARQ, 60,000 original blocks were RS-encoded and stored in a blockchain network of 1,000 nodes, as shown in Figure 1, and the data transmission time and throughput when one DN received data from $k-1$ SNs to recover the original blocks were compared. Table 1 lists the encoded chunk size, ACK size, trigger signal size, and data transmission speed in the experimental environment. The size of the encoded chunk was set to 1 KB, and those of the ACK and trigger signals were set to 14 and 20 bytes, respectively. The trigger signal and encoded chunk were assumed to be transmitted at the same data transmission rate.

Under the non-trigger ARQ technique, the SN transmits an encoded chunk to the DN, which responds via an ACK. If a node failure occurs and an ACK is not received, the SN retransmits an encoded chunk, attempting it up to eight times. If data transmission fails even after this, a new request for data is sent to another SN to initiate the ARQ-based data transmission. The trigger-based ARQ technique is a transmission technique in which the DN sends a trigger signal to the SN, receives a trigger ACK from the SN to check the link status in advance, and then receives encoded chunks from $k-1$ SNs. The trigger ACK is retransmitted up to eight times, and if the retransmission fails, the DN sends a trigger signal to another SN to ensure the reliability of the $k-1$ SNs and then receives an encoded chunk.

4.2 Throughput Evaluation

The throughputs when transmitting an entire encoded chunk were compared based on the node failure rate. The throughput was the total transmission time compared to the number of bits of the encoded chunks that needed to be transmitted. Data are not retransmitted in the absence of node failures, resulting in time only being taken to receive data sequentially from $k-1$ nodes. However, as the number of node failures increases, the data transmission time increases proportionally owing to data retransmission. Figure 4 shows the average throughput according to the node failure rate. The node failure rate was 0 – 20% of the total 1,000 nodes; this refers to the case in which the node failed to transmit data. The transmission time for the encoded chunk to recover a total of 60,000 original blocks when using the non-trigger and trigger-based ARQ techniques as the node failure rate increased was simulated, and the throughput deteriorates as a results.

When the node failure rate was 0%, TS and TA acted as overheads; the transmission time under the non-trigger ARQ transmission method, which transmitted only encoded chunks and ACKs, was less than that under the trigger-based ARQ method. The same was observed at a node failure rate of 2%. However, with the increase in the node failure rate, this time increased by more than 10% than that when there were no node failures, and the delay increased proportionally with this increase in the node failure rate.

However, under the trigger-based ARQ technique, the transmission delay due to retransmission increased by only approximately 0.3% when the node failure rate increased beyond 2%,

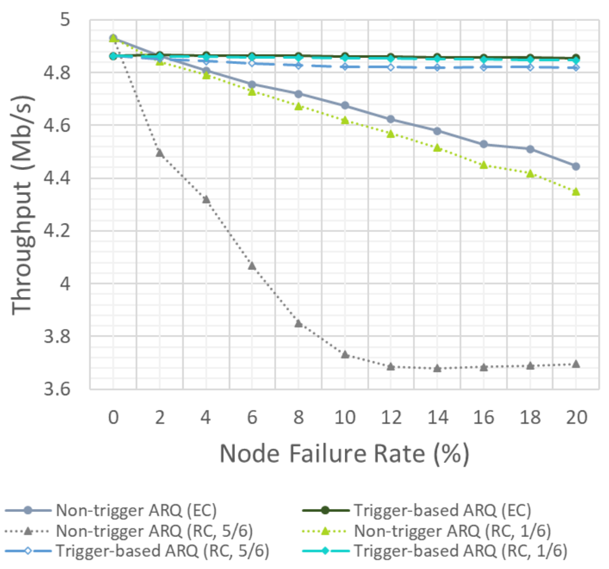

Figure 4: Throughputs under various node failure rates.

allowing encoded chunks to be transmitted with a low delay, even during frequent node failures. With the increase in the retransmission of the encoded chunks TS and TA, the throughput deteriorated. At 5 Mbps data transmission, as shown in Table 1, the non-trigger ARQ technique guaranteed a throughput of 4.93 Mbps when there was no node failure, which dropped to 4.44 Mbps when the node failure rate reached 20%. On the other hand, when using the trigger-based ARQ technique, the throughput was 4.86 Mbps when there was no node failure, which was slightly lower than that under the non-trigger ARQ technique, but the data transmission rate was 4.85 Mbps even when the node failure rate was 20%.

The throughput efficiency of the data transmission was measured using the actual throughput and compared to the transmission data rate when 60,000 blocks were recovered under each transmission technique. Under the non-trigger ARQ technique, the throughput efficiency fell from 98.6 to 89.6% when the node failure rate was 20%, while under the trigger-based ARQ technique, the throughput efficiency changed from 97.2 to 97.3% at the same node failure rate. The throughput efficiency under the trigger-based ARQ scheme was 8% higher than that under the non-trigger ARQ technique. When one DN received an encoded chunk from $k-1$ SNs, an increased throughput efficiency was observed.

When DNs receive encoded chunks in real-world scenarios, many factors such as in-channel interference, collision, and transmission delay due to the contention for channel access delay the recovery time. Therefore, the non-trigger ARQ technique requires more data transmission time for block recovery. The trigger-based ARQ technique can reduce the original data recovery time by enabling highly reliable and efficient data transmission, even in crowded networks.

4.3 Storage Efficiency Analysis

In full-node-based blockchain storage, each node stores a copy, resulting in data redundancy and consequently, a vast storage load. However, because all the nodes have blocks, there is no need

Storage Scheme	Storage Efficiency	Throughput (Mb/s) at a Node Failure Rate of 20%	
		Non-trigger ARQ	Trigger-based ARQ
Full node-based storage	1	-	-
RC-based distributed storage (code rate of 5/6)	1.2	3.69	4.81
RC-based distributed storage (code rate of 1/6)	6	4.34	4.84
(1000, 600) RS-based distributed storage	600	4.44	4.85

Table 2: Storage efficiencies and data transmission throughputs under various storage schemes

to recover the data to access them. Table 2 lists the storage efficiencies under various storage schemes and compares the data transmission throughput under each ARQ scheme at a node failure rate of 20%. The storage efficiency per node is the ratio of the actual stored data size to the total data. To increase the storage efficiency, the RC scheme can be applied instead of storing the entire block; storing only a portion of the data in copies can save storage space and restore the original block faster than EC-based storage can.

The EC-based storage method encodes 600 blocks into 1,000 encoded chunks and divides 1,000 nodes individually, whereas the RC-based storage method stores only 600 blocks partially according to the code rate. For example, if the code rate is 1/6, only 100 of the 600 blocks are stored in copies, and the remaining 500 are transmitted from the other nodes to recover the original block. Although the EC-based distribution method and the RC-based one with a code rate of 1/6 had similar throughputs under the trigger-based ARQ technique, the EC-based method was more efficient regarding storage efficiency and data transmission performance, considering that it could save 99% more storage space than the RC-based method could.

5 Conclusion

An EC-based distributed storage technique has been proposed to reduce the storage overhead in full-node blockchain storage. The throughput performance of the trigger-based ARQ scheme for EC-based distributed blockchain storage was 8% higher than that of the non-trigger ARQ data transmission technique, resulting in the former saving more than 99.8% of storage space compared to full-node storage and 99% of RC-based storage. The manner in which all the distributed nodes efficiently exchange encoded chunks to recover the original block has yet to be considered. Efficient data transmission techniques that consider mitigating network congestion and collisions between multiple distributed nodes will be studied in future work.

6 Acknowledgments

This work was partly supported by a Korea Institute for Advancement of Technology (KIAT) grant funded by the Korean Government (MOTIE) (P0008703, The Competency Development Program for Industry Specialists), and the MSIT under the ICAN (ICT Challenge and Advanced Network of HRD) program (No. IITP-2022-RS-2022-00156310) supervised by the Institute of Information Communication Technology Planning and Evaluation (IITP)

References

[1] Abdul Ghaffar Khan, Amjad Hussain Zahid, Muzammil Hussain, M Farooq, Usama Riaz, and Talha Mahboob Alam. A journey of web and blockchain towards the industry 4.0: An overview. In

2019 International Conference on Innovative Computing (ICIC), pages 1–7, 2019.

[2] Yijing Lin, Zhipeng Gao, Hongyang Du, Dusit Niyato, Jiawen Kang, Ruilong Deng, and Xuemin Sherman Shen. A unified blockchain-semantic framework for wireless edge intelligence enabled web 3.0. *IEEE Wireless Communications*, pages 1–9, 2023.

[3] Zhengyi Du, Xiongtao Pang, and Haifeng Qian. Partitionchain: A scalable and reliable data storage strategy for permissioned blockchain. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):4124–4136, 2023.

[4] Sun-Woo Yun, Eun-Young Lee, and Il-Gu Lee. Selective layered blockchain framework for privacy-preserving data management in low-latency mobile networks. *Journal of Internet Technology*, 24(4):881–891, 2023.

[5] Blockchain website. https://www.blockchain.com/explorer. [Online; accessed: 21-Sep-2023].

[6] Hong-Ning Dai, Zibin Zheng, and Yan Zhang. Blockchain for internet of things: A survey. *IEEE Internet of Things Journal*, 6(5):8076–8094, 2019.

[7] Elizabeth Reilly, Matthew Maloney, Michael Siegel, and Gregory Falco. An iot integrity-first communication protocol via an ethereum blockchain light client. In *2019 IEEE/ACM 1st International Workshop on Software Engineering Research Practices for the Internet of Things (SERP4IoT)*, pages 53–56, 2019.

[8] Wai Kok Chan, Ji-Jian Chin, and Vik Tor Goh. Simple and scalable blockchain with privacy. *Journal of Information Security and Applications*, 58:102700, 2021.

[9] Chunlin Li, Jing Zhang, Xianmin Yang, and Luo Youlong. Lightweight blockchain consensus mechanism and storage optimization for resource-constrained iot devices. *Inf. Process. Manage.*, 58(4), jul 2021.

[10] Xiaodong Qi, Zhao Zhang, Cheqing Jin, and Aoying Zhou. Bft-store: Storage partition for permissioned blockchain via erasure coding. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1926–1929. IEEE, 2020.

[11] Ruiran Wang, Laurent Njilla, and Shucheng Yu. A-c: An ndn-based blockchain network with erasure coding. In *2023 International Conference on Computing, Networking and Communications (ICNC)*, pages 591–595, 2023.

[12] Doriane Perard, Jérôme Lacan, Yann Bachy, and Jonathan Detchart. Erasure code-based low storage blockchain node. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1622–1627, 2018.

[13] Yibin Xu and Yangyu Huang. Segment blockchain: A size reduced storage mechanism for blockchain. *IEEE Access*, 8:17434–17441, 2020.

[14] Yifei Xiao, Shijie Zhou, and Linpeng Zhong. Erasure coding-oriented data update for cloud storage: A survey. *IEEE Access*, 8:227982–227998, 2020.

[15] Yingdi Shan, Kang Chen, Tuoyu Gong, Lidong Zhou, Tai Zhou, and Yongwei Wu. Geometric partitioning: Explore the boundary of optimal erasure code repair. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, SOSP '21, page 457–471, New York, NY, USA, 2021. Association for Computing Machinery.

[16] Ying Song, Tiantong Mu, and Bo Wang. Hv-snsp: A low-overhead data recovery method based on cross-checking. *IEEE Access*, 11:5737–5745, 2023.

[17] Han Qiu, Chentao Wu, Jie Li, Minyi Guo, Tong Liu, Xubin He, Yuanyuan Dong, and Yafei Zhao. Ec-fusion: An efficient hybrid erasure coding framework to improve both application and recovery performance in cloud storage systems. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 191–201, 2020.

[18] Qing Liu, Dan Feng, Hong Jiang, Yuchong Hu, and Tianfeng Jiao. Systematic erasure codes with optimal repair bandwidth and storage. *ACM Trans. Storage*, 13(3), sep 2017.

[19] Pablo Ignacio Serrano Caneleo, Lakshmi J Mohan, Udaya Parampalli, and Aaron Harwood. On

improving recovery performance in erasure code based geo-diverse storage clusters. In *2016 12th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 123–129, 2016.

[20] Summera Nosheen and Jamil Y. Khan. An adaptive qos based video packet transmission technique for ieee802.11ac wlan. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5, 2019.