The Design of Hierarchical Consensus Mechanism Based on Service-Zone Sharding

Ji-Young Kwak, Jongchoul Yim, Nam-Seok Ko, and Sun-Me Kim

***Abstract*—The byzantine agreement protocol has a disadvantage that there are many limitations on node scalability, because the performance degradation may occur due to a large amount of traffic. Hence, this classical byzantine agreement algorithm seems to be infeasible to achieve scale-out performance with the same level of security as a Bitcoin. In order to improve the performance degradation due to a large amount of traffic, we propose the hierarchical consensus mechanism based on service-zone sharding rather than how all nodes participate in the consensus process. In the proposed consensus mechanism (*SZHBFT*), a disjoint set of transactions is locally processed by a secure consensus subgroup or globally processed between consensus subgroups. Thus, the transactions that occur related to multiple *Service-Zone Consensus Groups* are updated and maintained in the *Inter-Service Zone Public Ledger*, whereas service transactions occurring locally are processed in parallel by each *Service-Zone Consensus Group* and then distributed into a local *Service-Zone Private Ledger*. The scheme of the proposed *SZHBFT* mechanism provides the hierarchical agreement solution along with distributed multiledger structure for the scalable byzantine resilient agreement by forming secure consensus subgroups in order to minimize the overhead of overall communication messages.**

***Index Terms*—Byzantine agreement, distributed multiledgers, hierarchical consensus.**

I. Introduction

THE future Internet of Things (IoT) system with distributed smart devices, after obtaining and transforming the large volumes of data generated from the devices, will provide the knowledge that is facilitating the development of smart services for improving the service maintenance and the quality of life [2]. Blockchain is able to be integrated with diverse service domains by the potentiality to be leveraged in different aspects of management and technology. Thus, this technology can support a safe micropayments and a convenient deployment of microservices in the trustless environment by creating the blockchain-based IoT ecosystem, such as service and data marketplace. Also, it will improve IoT service interconnection and the access of IoT data in the distributed service platform based on blockchain. The service of a supply chain is one of the best-fit areas for the blockchain ecosystem as it requires the reliable mechanism for multiple participants to cooperate for a business service in a supply chain. This blockchain technology, by its nature, makes it easy for large, disparate groups of service market participants to collaborate and safely transact with each other [3].

The architecture of blockchain ecosystem is organized of a blockchain network model, the participants' information service with programming interfaces, and a collection of smart contracts for service operations. The participants' information service enables them to retrieve or upload business data on the blockchains. Through a collection of smart contracts, the blockchain ecosystem becomes autonomous and credible so that the blockchains can operate without human intervention. Therefore, the blockchain ecosystem provides a decentralized and self-regulating data infrastructure in which all data and transactions are stored [1], [4]. In the blockchain ecosystem, all participants can also observe the entire state of the service market at any time, as opposed to in a proprietary platform. In the IoT system that generates the vast amounts of data in real time, the storage capacity and scalability limitations of blockchain become a big obstacle to its integration with blockchain [5]. Thus, it is the critical issue to be addressed in the current blockchain-based IoT system, for the purpose of constructing a feasible and efficient blockchain service ecosystem.

In the current blockchain system, all nodes store all states, such as account balances and contract codes, and process all transactions. This ensures the stability of security in the form of blockchain with a large number of blocks. However, as the number of processing transactions grows, managing all the processing transactions in all nodes can cause problems in terms of storage capacity as well as throughput. Therefore, research studies are underway on how distributed nodes can have only some ledgers while maintaining existing stability. One of these research directions is to process service transactions and manage the blockchain ledger based on the sharding approach.

Sharding protocols enable to reduce the communication, computation, and storage requirements of each node, since each has been stored at one of the shards after dividing the blockchain into multiple partitions. But, the cross-shard transaction makes the working process for the verification and agreement more challenging. In the sharding technique in which the subset of transactions from the consensus process is executed in parallel, it is also required to perform the agreement process between different shards due to the cross-shard transactions generated

Manuscript received February 8, 2019; revised February 16, 2020 and March 27, 2020; accepted April 28, 2020. Date of publication February 8, 2019; date of current version October 9, 2020. This work was supported by ICT R&D program of MSICT/IITP (2017-0-00045, Hyper-connected Intelligent Infrastructure Technology Development). Review of this manuscript was arranged by Department Editor K.-K. R. Choo. *(Corresponding author: Ji-Young Kwak.)*

The authors are with the Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, South Korea (e-mail: jiyoung@etri.re.kr; hektor@etri.re.kr; nsko@etri.re.kr; kimsunme@etri.re.kr).

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

Digital Object Identifier 10.1109/TEM.2020.2993413

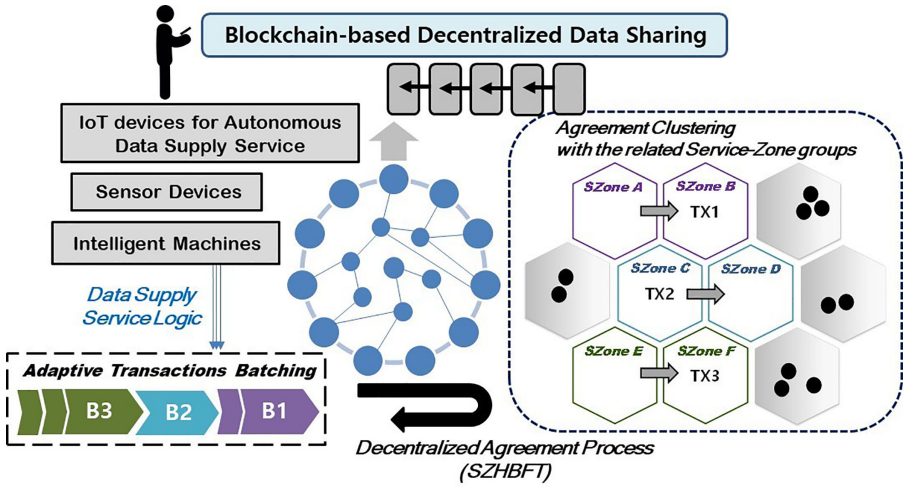


Fig. 1. Concept of decentralized blockchain service with smart things.

randomly into the various services form. Because of processing overhead from the consensus between different shards, the transaction processing performance in the process for the agreement among the distributed nodes generally does not increase linearly with the number of multiple shards. Thus, an efficient mechanism to deal with the cross-shard transactions is crucial in the design of a blockchain sharding protocol. And there are an increasing number of research studies on the design of scalable blockchain using the sharding approach, which is able to reduce the considerable overhead of coordination work on cross-shard transactions to ensure consistency across multiple shards. As the attacker is likely to lead the majority of validator nodes within a single shard to the same side, the mechanism for faster detection and removal of faulty validators is also required. The selection of a malicious primary leader node can lead to a global delay in the overall transaction processing, since he may wait for some time deliberately after receiving an update from the participant nodes before sending the pre-prepare message. However, an optimized agreement protocol with the method of effective view change and the mechanism of group assignment in an unbiased and random manner is capable of avoiding such delay.

In order to improve the throughput of transactions generated from smart IoT devices, we propose the hierarchical consensus mechanism based on service-zone, where a disjoint set of transactions are locally processed by a secure consensus subgroup or globally processed between consensus subgroups in the form of distributed multiblockchains. The key concept of the proposed consensus mechanism (*SZHBFT*) is to obtain a common agreement within the dynamic agreement clustering for consensus subgroups associated with the corresponding block, as described in Fig. 1. The advantage of this concept is to automatically parallelize the available processing power by securely organizing multiple consensus subgroups. And service transactions are managed based on the unit of a service-zone consensus group in the form of distributed multiblockchains. In addition, it supports the structure of efficient distributed service blockchain, which can track the subsets of ledgers related to the service, so that the service function is easily verified.

In this article, the following contributions will be described.

1) We have designed the service-zone-based hierarchical consensus mechanism (*SZHBFT*) along with distributed multiledger structure for scalable byzantine resilient agreement, by forming secure consensus subgroups so as to minimize the overhead of overall communication messages.
2) The presented *SZHBFT* mechanism guarantees the unpredictability of secure consensus group membership by using verifiable random function (VRF) so that each participant is selected as a validator according to the weighting factor based on the proximity with the closest landmark node from an *SZone_Coordinator*.
3) The adaptive batching interval scheme has been designed so that the batching interval of a block is regulated dynamically according to the number of consensus nodes and the transaction occurrence rates via the change strategy of block size considering a limit threshold for the final consensus latency.
4) We have measured on the performance characteristics of the presented *SZHBFT* mechanism under various simulation conditions, such as the number of network nodes, group size, and the fraction of faulty nodes, and then analyzed its effectiveness by using three metrics, including communication complexity, transactions per second (TPS), and consensus latency.

The approach of the proposed *SZHBFT* mechanism can enhance the consensus scalability via the hierarchical byzantine agreement acceleration protocol based on service-zone along with the method to adjust the batching interval of a block adaptively.

II. Background and Related Work

The technology of blockchain is a powerful solution to enable the provision of smart services and the knowledge in the distributed form for the convenient life with the smart IoT devices spread around. Thus, blockchain can enable the creation of

IoT service marketplace along with the autonomy of devices, because it provides a distributed open ledger where devices can query trusted information [5]. The Autonomous Decentralized Peer-to-Peer Telemetry project led by IBM [6] aims to promote the device autonomy, which requires the possibility of processing data at the thing layer by creating smart things with self-inferencing and self-monitoring capabilities [7], [8]. This project leverages the blockchain to build the distributed network of devices and ensure code execution on the distributed systems connected with IoT devices. Autonomous IoT networks powered by blockchain can support an improvement of the fault tolerance and system scalability, but it will issue the challenges to the business models in the form of decentralized architecture. And in the provision of IoT services in the distributed form, the storage capacity and scalability limitations of blockchain make these challenges much greater.

Some recent IoT systems are preparing the integration with blockchain in order to provide a trustworthy service platform based on blockchains for smart contracts. Hyperledger Fabric [9] is a system for operating a permissioned blockchain that targets business services and also setups private infrastructures for the execution of smart contracts. Fabric blockchain enables clients to manage transactions by making use of smart contracts called chaincode, the endorsing peers that validate the submitted transactions, and an ordering service. Ethereum [10] executes its smart contracts on the decentralized Turing-complete virtual machine, called to Ethereum Virtual Machine, for the mass consumption of decentralized services. The creation of these decentralized applications is one of the most important features of Ethereum. Because its scalability limitation occurs from every validating node having to process every transaction, Ethereum plans to improve scalability limitation through the sharding technique.

In order to enable scale-out services while considering the security benefits of a decentralized blockchain along with high TPS performance, the design of a highly scalable blockchain must consider the following requirements [11]: 1) support a large network size equivalent to that of major cryptocurrencies, such as Bitcoin and Ethereum, while achieving a high transaction throughput, and 2) support general workloads and services beyond cryptocurrencies. Current consensus algorithms, for e.g., the Practical Byzantine Fault Tolerance (PBFT) protocol, are only applicable to small-scale systems because of communication complexity [12]. The PBFT protocol solves the problem of replication in the byzantine model, where a number of servers can exhibit arbitrary behavior. And it makes use of an elected leader node to coordinate the protocol based on the flat consensus architecture, which requires acknowledgments from $2f + 1$ out of $3f + 1$ nodes in order to check the behavior of f byzantine nodes. Although it tolerates malicious failures, the PBFT cannot be scaled to a large number of participant nodes because of a large number of protocol exchanges with the $O(N^2)$ cost.

In order to improve the performance and scalability issues of blockchain, several consensus protocols have recently been proposed in the form of hybrid architectures that combine the open-membership nature of Bitcoin with byzantine fault tolerance. Elastico [13] is the first sharding-based consensus protocol to scale by partitioning nodes in the network, where each committee is responsible for processing a subset of transactions. In each consensus epoch, each participating node solves a proof of work (PoW) puzzle in order to use the identity that is created based on the PoW and epoch randomness for guaranteeing the required randomness. A final committee aggregates sets of transactions received from committees into a final block and then runs another round of PBFT to make a global decision. In the Elastico, the performance of block processing is scaled up almost linear to the size of the network, but multishard transactions cannot be processed. ByzCoin [14] is a Bitcoin-like cryptocurrency that leverages a PBFT protocol to get better performance than Bitcoin. ByzCoin incorporates the PBFT along with a collective signing [15] in order to dynamically form a small committee to agree on the next block. The consensus outcome is the collective signature that proves that at least two-thirds of the committee members have attested the microblock. Because each node can verify in $O(1)$ that a microblock has been validated by the committee, the overall number of protocol messages is reduced finally. But the consensus delay of a ByzCoin becomes longer than other consensus protocols because of requiring more computing times caused by the complex operations of its collective signature. Unfortunately, ByzCoin's specification is also incomplete, and the protocol is known to be vulnerable to byzantine faults. Omniledger [16] and Rapidchain [17] are the recently proposed sharding-based public blockchain protocols that try to solve the scalability and security issues of Elastico.

RapidChain is a sharded blockchain protocol that combines the work of Elastico, ByzCoin, and Bitcoin-NG. For the purpose of protecting against the Sybil attack, it relies on the PoW, by requiring that every node that wants to join or stay in the committees has to solve a PoW puzzle. Similar to this method, in the proposed consensus mechanism (*SZHBFT*), PoW is used to determine a share window of candidate validator set with the leader being the latest PoW mining node, when selecting the members of a consensus group based on the service-zone. At the start of each epoch, the reference committee agrees on a reference block consisting of the list of all active nodes as well as their assigned committees, by checking the PoW solutions of all nodes. The cross-shard transaction in RapidChain has largely relied on the intershard routing scheme, which is based on a Kademlia routing algorithm [18]. In other words, the committee members use an intershard routing algorithm to send the transactions to the shards handling the inputs and outputs of the transaction, respectively. To process cross-shard transactions, the leader node of an output shard creates one additional transaction for every different input shard and then sends these transactions to the corresponding input shards for validation. For cross-shard transactions in the RapidChain, one drawback is that it inherently increases the number of transactions to be proceeded for each transaction, since it creates three different transactions to exchange information among multiple shards. Also, the cross-shard transaction largely depends on the routing algorithm, which is a potential bottleneck. RapidChain enhances the efficiency of sharding-based blockchain protocols in a large scale, but the protocol guarantees high efficiency only when leaders in each committee are honest. Unlike other

TABLE I
COMPARISON REGARDING THE SCALABLE BLOCKCHAIN PROTOCOLS

		ByzCoin	**Elastico**	**OmniLedger**	**RapidChain**	***Proposed SZHBFT***
Committee Formation	*Permissionless Scheme*	PoW	PoW	PoW, Randomness generation by using RandHound	PoW, Randomness generated by the VSS-based DRG protocol	PoW, the commit-reveal scheme with using VRF
	Members Allocation	Allocation based on the result of Pow from key blocks to determine the validators set and leader	Allocation based on the least-significant bits for the result of PoW puzzles	Allocation based on the generated Randomness, Leader election by the VRF	Allocation based on the result of PoW puzzles, Randomness generation, identity validation, and shard reconfiguration by the reference committee	Choosing a random subset of adjacent validators from a *SZone_Coordinator* according to per-node weights such as RTT factors
Flexible Transactions Batching		No	No	No	No	Yes, adaptive transactions batching work
Cross-Shards Transaction		No	No	Yes, two-phase lock-then-unlock protocol	Yes, inter-shards routing scheme based on Kademlia algorithm	Yes, hierarchical agreement clustering with the related service-zone groups
Additional Global Blockchain		Yes, key blockchain, micro blockchain	Yes, a global ledger	Yes, identity blockchain	Yes, reference blockchain	Yes, inter-service zone public ledger
Safety	*DoS Resistance*	Yes	Yes	Yes	Yes	Yes
	Fault Tolerance	33%	33%	33%	33%	33%
Performance	*Scalability*	good	excellent	excellent	excellent	excellent
	Communication Complexity	$O(sm^2 + N)$	$O(sm^2 + N)$	$O(sm^2 + N)$	$O(sm^2 + sm \times \log_2 N)$	$O(sm^2 + 2 \times N)$

sharding-based methods that have limited tolerance to Byzantine faults, it can tolerate up to one-third faulty nodes without any trusted setup. Under the condition that there is a proportion of one-third leaders that are malicious in a round, cross-shard transactions may hardly be included in a block.

OmniLedger is a solution for sharding based on the ByzCoinX, which is a ByzCoin-like protocol, instead of the PBFT to efficiently process transactions within shards. The topology of Elastico is hierarchical, whereas OmniLedger makes use of a flat committee topology. Compared with Elastico, OmniLedger uses a bias-resistant distributed randomness generation protocol instead of the PoW to generate a seed for sharding securely. To effectively shard the blockchain and create the committees assigned to multiple shards, OmniLedger employs RandHound [19], a distributed protocol for generating unbiased randomness in an untrusted network. RandHound requires the leader election, which is done with a verifiable random function. Similar to Elastico, OminiLedger also runs a global reconfiguration protocol at each epoch, to allow new participants to join the protocol. In each epoch, it employs the sliding window mechanism that defines the eligible validators to be the nodes mining the latest blocks. Similar to this approach, the proposed *SZHBFT* mechanism utilizes the share window of candidate validators that have performed a PoW mining within a period of time along with the latest PoW mining leader, in order to choose the members of a consensus group based on service-zone. OminiLedger also introduces an efficient cross-shard commit protocol called the AtomiX [16] that can atomically handle transactions that affect multiple shards. Because of unnecessarily complex communication costs, cross-shard transactions in the OminiLedger are handled by using a new two-phase lock-then-unlock protocol, coordinated by a client. In other words, OmniLedger takes a two-phase approach to handle atomic commit, in which each input shard first locks the corresponding input unspent transaction outputs (UTXOs) and issues a "*proof-of-acceptance,*" if the UTXO is valid. And then, the client collects responses from all input committees and then issues an "*unlock to commit*" to the output shard [20]. Therefore, OmniLedger achieves safety for the UTXO model, by relying on the client to coordinate a lock-then-unlock protocol. But this client-driven protocol suffers from indefinite blocking if the client is malicious [11].

Table I provides a brief comparison of the blockchain sharding scheme between existing scalable blockchain systems and the proposed *SZHBFT* after analyzed recent approaches incorporated to improve scalability in terms of secure subgroups formation, cross-shard transactions handling, and performance improvement. In contrast to other solutions, OmniLedger takes care to not compromise security or permissionless decentralization while addressing similar requirements to be considered in terms of two design directions in the proposed *SZHBFT* mechanism as follows: 1) choosing statistically the candidates of validators via the permissionless algorithms supporting the sybil attack resistance, such as PoW or PoS; and 2) handling cross-shard transactions correctly. Since the sharding-based permissionless blockchain in the Byzantine setting is challenging, only a few protocols exist aiming to resolve this problem along with the issue of handling cross-shard transactions.

III. BYZANTINE AGREEMENT ACCELERATION MECHANISM BASED ON HIERARCHICAL STRUCTURE

We present a design for the service-zone-based hierarchical agreement mechanism that can speed up the processing of service transactions and save storage resources by using the

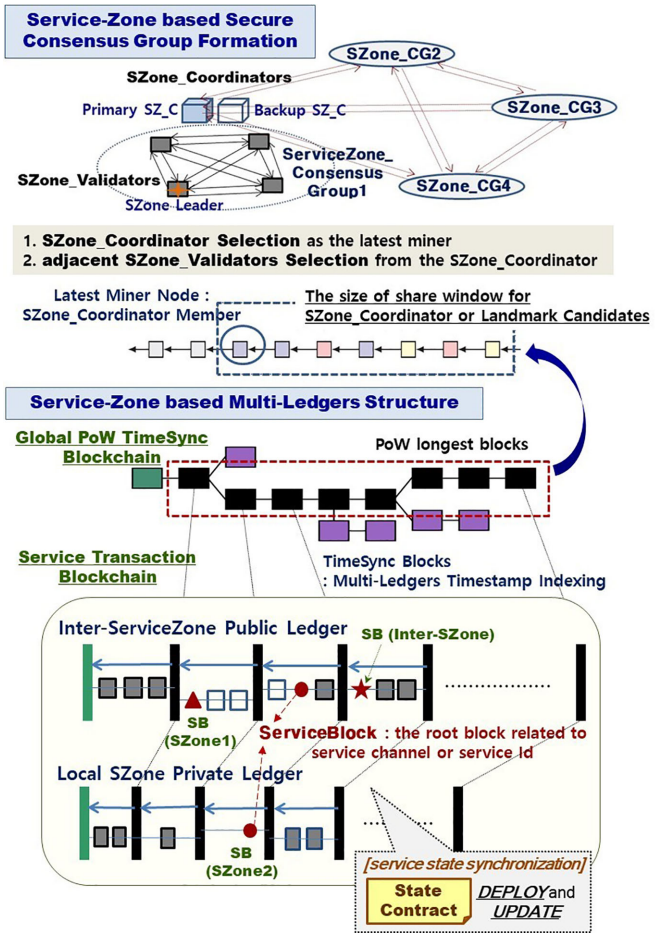


Fig. 2. Service-zone-based consensus group formation with multiledger structure.

structure of distributed multiledgers for managing the related service transactions on multiple service-zone consensus groups. A key design goal of this article is to achieve scalability in terms of high throughput and low latency by making use of the sharding strategy with the two-layer hierarchical consensus structure. The effective solution to reach a safe agreement among distributed nodes using the sharding strategy is must handle the open issues, such as the formation of a random consensus subgroup and the processing of cross-shard transactions related to multiple consensus subgroups. Actually, cross-shard communication is a very difficult problem along with the dynamic configuration of consensus subgroups in the research of blockchain sharding.

The proposed consensus mechanism (*SZHBFT*) is divided into two phases: 1) the formation of a secure consensus group, which employs PoW as well as VRF [21] scheme, and 2) the process of hierarchical consensus to perform the byzantine resilient agreement in the unit of a service-zone along with multiledger structure. This mechanism is performed based on a two-layer structure shown in Fig. 2 along with multiple *Service-Zone Consensus Group*s, containing validators in another layer associated with their service-zone coordinators that are elected among the candidate PoW mining nodes. The election process of an *Inter-SZone Leader* among the *SZone_Coordinators* in the multiple *Service-Zone Consensus Group*s is repeated with

TABLE II
CRYPTOGRAPHIC WEIGHTING RANDOMNESS FUNCTION BASED ON THE VRF

- **sk$_i$, pk$_i$** : the private key and public key of node i
- w_i: the distance factor of node i to measure closeness with the corresponding *SZone_Coordinator*
- **TH** : the threshold value for limited boundary or distance factor to direct a selection range -> adjusting the threshold ratio according to ***the expected number of members to be selected as the SZone_Validators***

▪ **Cryptographic_RandomHash**(sk$_i$, seed, w_i, W)
- VRF(sk$_i$, seed) => {hash, proof}
IF *hash* / $2^{hashlength}$ < *TH* * w_i / *W*
Then return {hash, proof}

▪ **Cryptographic_VerifyHash**(pk$_i$, hash, proof, seed, wi, W)
IF *hash* / $2^{hashlength}$ < *TH* * w_i / *W*
Then return VerifyVRF(pk$_i$, hash, proof, seed)

random period terms. Therefore, to prevent attacks from malicious nodes, the node that receives a large number of responses to the voting message of leader election for each period term is randomly selected as an *Inter-SZone Leader*. As illustrated in Fig. 2, a random *Service-Zone Consensus Group* is dynamically constructed by selecting the *SZone_Validator* members physically close to the *SZone_Coordinator*, so that validation and consensus works for local service transactions are separated from each other and processed fast.

A. Secure Consensus Group Formation

The proposed *SZHBFT* mechanism makes usage of VRF as the algorithm for choosing a random subset of validator nodes according to per-node weights; in that, if given a set of weights w_i, the probability that node i is selected is proportional to $w_i/W (W = \sum_i w_i)$. The *SZone_Coordinator* determined by the latest PoW mining operation sends a creation request for the service-zone consensus group to adjacent mining nodes. A packet of the creation request includes the account of a deployed status contract with the information of random landmark nodes set. And then, the information of round trip time (RTT) from a participating node is measured along with PoW-based managed nodes within the set of random landmark nodes. Also, it is applied to the weight factors of the VRF described in Table II and Fig. 3, in order to select adjacent *SZone_Validators*.

The process for selecting the next validators of a consensus subgroup is dynamically performed using the algorithmic randomness of VRF based on the seed input value. This seed input is calculated from both the information of the last generated block with a creation timestamp and the RTT factors measured with random landmark nodes. Every participant node can independently determine if they are chosen to be on the member of a corresponding consensus group by using the commit-reveal approach described in Fig. 3. This approach forces each participant to submit their secret input value with the corresponding proof after computing the VRF hashing of their private key and then to provide into the form of public information on the blockchain. By performing the VRF hashing described in Table II, a specific seed value combined with the executor's private key will produce a resulting pseudorandom value with the corresponding proof. These values of resulting random hash and proof enable any nodes that know the corresponding public key to check the

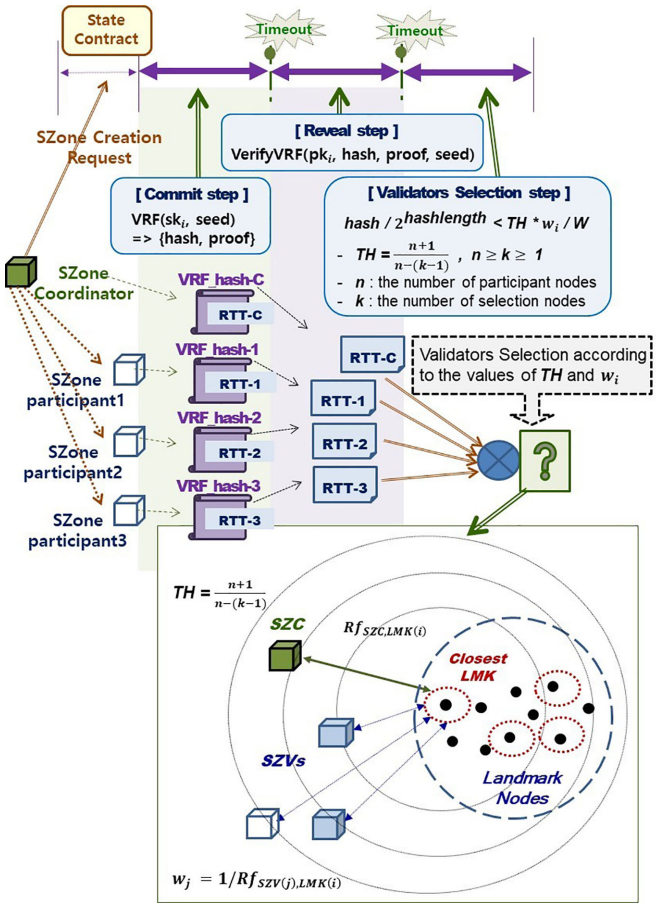


Fig. 3. Validators selection through the commit-reveal process.

correctness of a resulting hash and also to prove the membership of the service-zone consensus group for this node to all other members.

In Figs. 3 and 4, the process of selecting validator members for secure consensus group formation can be divided into three steps, including commit, reveal, and validators selection steps. In the commit step, each participant runs an instance of the VRF with the seed input value including the RTT factors (RTT_i) measured from random landmark nodes. And then, he submits the resulting random hash set for determining if he is eligible to become a validator

$$\mathrm{RTT}_i = \left[\, Rf_{i,LMK(1)} \,,\, Rf_{i,LMK(2)} \,\cdots\, Rf_{i,LMK(j)} \,\right]$$

where

i, j	the number of participant nodes and landmark nodes;
RTT_i	RTT factors set of node *i*
$Rf_{i,LMK(j)}$	the RTT of node *i* measured from landmark node *j*.

As shown in Figs. 3 and 4, the main flows of the commit-reveal process are that all participants

1) measure their own RTT factors from random landmark nodes and then commit their secret RTT factors by submitting the result of performing a VRF hashing with a seed value, including the measured secret RTT factors;
2) reveal their secret RTT factors of participant nodes within a specified time period, and then the secret RTT factors of an *SZone_Coordinator* is provided at the last order;
3) find the closest landmark node from an *SZone_Coordinator* for the purpose of selecting proximity participant node from the closest landmark node by referencing all secret RTT factors afterward

$$\text{closest LMK}\,(c) = \mathrm{argmin}_c \left\{\, Rf_{i,\mathrm{LMK}(c)} \right\}$$

where closest $\mathrm{LMK}(c)$ the landmark node *c* with the shortest RTT measured from an SZone_Coordinator (node *i*)

$$\mathrm{diff}_{i,\mathrm{LMK}(j)} = Rf_{i,\mathrm{LMK}(j)} - Rf_{\mathrm{LMK}(j),i}$$

$\mathrm{diff}_{i,\mathrm{LMK}(j)}$ the difference value between the RTT of node *i* from landmark node *j* and the RTT of landmark node *j* from node *i*.

In the commit-reveal process, comparison of the differences between the RTT of node *i* measured from landmark node *j* and the RTT of landmark node *j* measured to the reverse direction is made, because a dishonest participant can modify the output of the measured RTT factor that is beneficial for him. Although the output becomes unpredictable from the committed secret RTT factor and unmodifiable after the commit step, the participant to lastly reveal the secret RTT factors will be able to know the resulting final output beforehand and refuse to reveal his own secret value. To prevent this attack, each participant has to submit his measured RTT factors before seeing other RTT factors within a specified time period, and then the secret RTT factors of an *SZone_Coordinator* will be provided at the last order in the reveal step

$$W = \sum\nolimits_i 1 \,/\, Rf_{i,\text{closest LMK}(c)}$$

$$w_i = 1 \,/\, Rf_{i,\text{closest LMK}(c)}$$

where

$Rf_{i,\text{ closest LMK}(c)}$	the RTT of node *i* from the closest landmark node *c;*
W	a total sum for the RTT factors set of participant nodes from closest $\mathrm{LMK}(c)$;
w_i:	the weight of node *i.*

In the validator selection step shown in Figs. 3 and 4, all participants make usage of the VRF hashing described in Table II to determine their own eligibility. In Table II and Fig. 3, the threshold value (TH) is set to adjust the expected number of participants to be selected as validators. If given a random input, the VRF hashing produces a pseudorandom hashing output, which is uniformly distributed between 0 and $2^{\mathrm{hashlength}} - 1$. In order to check whether he is eligible or not, each participant *i* compares that whether the hash resulting from the VRF is smaller than the TH, which is proportional to w_i/W. Thus, the

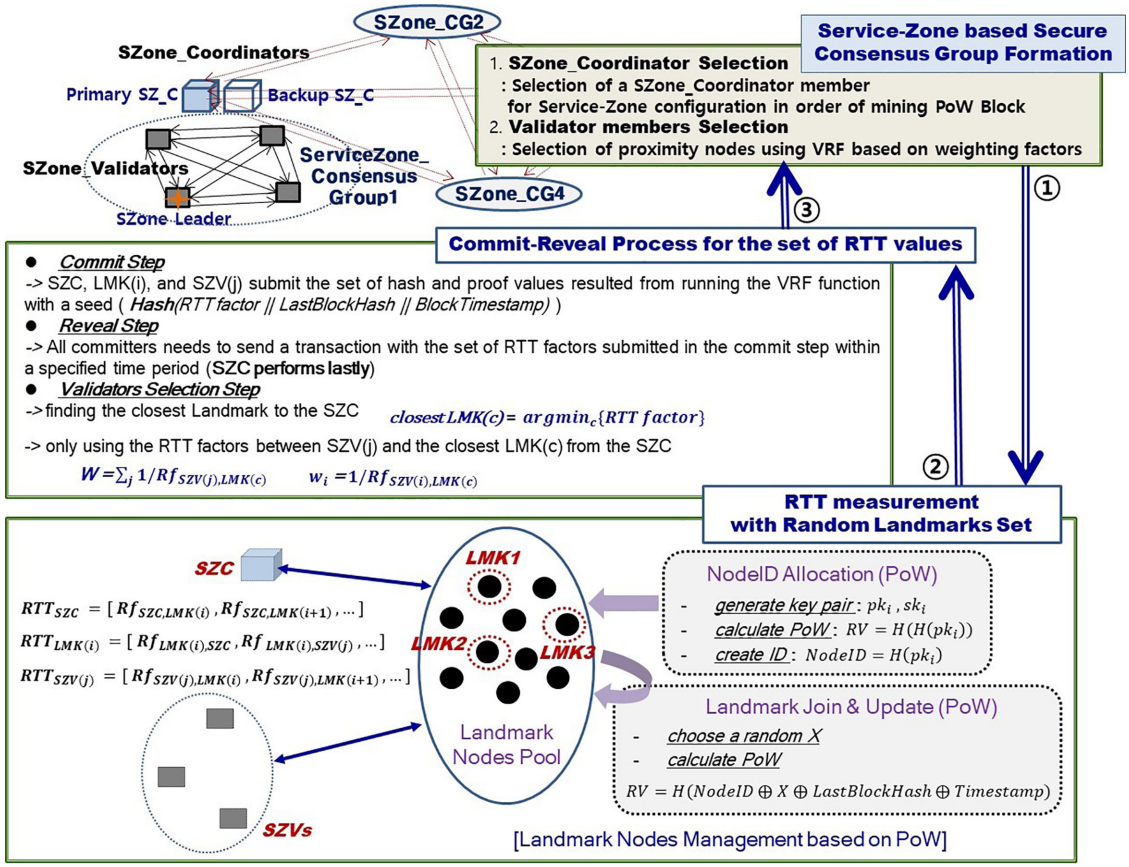


Fig. 4. Service-zone-based secure consensus group formation.

participant i is selected with probability $p = \mathrm{TH}/W$ based on the weight w_i, where W is the total sum of distance-based weight values $(1 \,/\, Rf_{i,\ \text{closest LMK}(c)})$ for the RTT factor set of participant nodes from the closest landmark node c (closest LMK(c)). The probability that k_i validators out of w_i participants are selected is the form of binomial distribution with probability $p = \mathrm{TH}/W, \mathrm{B\_Pr}(k_i; w_i,\ p) = \binom{w_i}{k_i} p^{k_i}(1-p)^{w_i - k_i}$, where $\sum_{k_i=0}^{w_i} \mathrm{B\_Pr}(k_i; w_i,\ p) = 1$. All other participants can verify if participant i is a valid *SZone_Validator* by estimating from the publicly known information published from participant i.

B. Byzantine Fault Tolerant (BFT) Agreement Acceleration Mechanism

From the characteristics of a global public ledger, including all transactions, there are significant issues, such as scalability limit, privacy problem, and insufficient storage resource. For the purpose of resolving these issues, the proposed *SZHBFT* mechanism works in conjunction with the structure of service-zone-based multiledgers, which consists of an *Inter-Service Zone Public Ledger*, and the *Local SZone Private Ledgers* distributed in multiple service areas. Here, the *Inter-Service Zone Public Ledger* manages service transactions related to a plurality of service-zone consensus groups occurring on the multiple channels. In this structure of service-zone-based multiledgers, to synchronize the checkpoints of distributed ledgers at regular intervals, every time the block data (*Time Sync Block*) of the *Global PoW Time Sync Blockchain* is added, these block data are also updated on the distributed *Local SZone Private Ledgers* and *Inter-Service Zone Public Ledger*. And then, the mining node of a corresponding block is set into the candidate of *SZone_Coordinator* or *Landmark*. In addition, because service-zone-based multiledgers are the structure of multilayers starting from the *Service Block*, including the service id corresponding to a service channel, it is easy and fast to verify the data related to the corresponding service.

After service-zone-based consensus groups are dynamically organized in the hierarchical approach for the agreement of requested transactions, the mechanism of hierarchical byzantine agreement acceleration based on the Exponential Information Gathering (EIG) tree proceeds according to the algorithms shown in Fig. 5 and Table III.

When an arbitrary service request occurs, the *Inter-SZone Leader* determines the *Service-Zone Consensus Group* that is processing the smart contract or account state related to the corresponding service transaction. When block creation via the batch processing of received service transaction requests is completed, an *Inter-SZone Leader* demands to perform both the *Input SZone Agreement* and the *Inter-SZone Agreement* by sending an *Inter-SZonePREPARE* request to the *SZone_Coordinator*s associated with input and output service-zone groups.

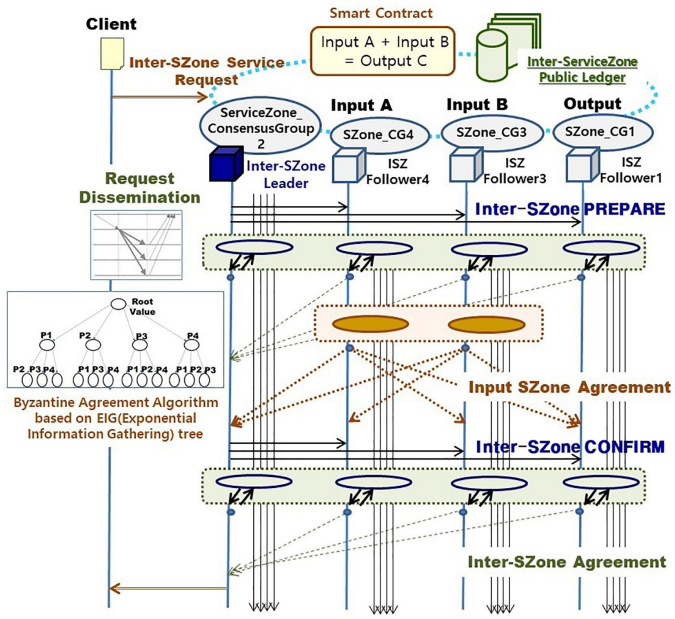


Fig. 5. Hierarchical byzantine agreement acceleration protocol.

By the message of *Inter-SZonePREPARE* request, only the *SZone_Validator* members belonging to the input service-zone groups perform the algorithm of PBFT with a root value on the EIG tree, and concurrently the *SZone_Validator*s in all related service-zone groups process the block validation of *Inter-SZonePREPARE* message. If the results validating the block of *Inter-SZonePREPARE* message is agreed from the $(3f + 1)$ or more than *SZone_Validator* members, the agreed block is finally updated into the *Inter-Service Zone Public Ledger*. But when that result of block validation is received from less than $(3f + 1)$ *SZone_Validator* members, the validation confirmation of the corresponding block is also requested by transferring the *Inter-SZoneCONFIRM* message with history signatures. Hence, the final consensus is decided depending on whether the *Input SZone Agreement* is agreed or the validation result of *Inter-SZoneCONFIRM* request. When the validation confirmation of the corresponding block included in the *Inter-SZoneCONFIRM* message is agreed from the $(2f + 1)$ or more than *SZone_Validator* members, the agreed block is also added into the corresponding ledger. When performing the PBFT algorithm for *Input SZone Agreement*, each *SZone_Validator* member determines a root value of the EIG tree into final consensus, after deriving the result of byzantine agreement in the forms of an EIG tree. The process of this byzantine agreement in the forms of an EIG tree proceeds in the following two steps: 1) the messaging step to fill the received messages in the top-down order, and 2) the step of byzantine agreement to evaluate a plurality values of received messages in the bottom-up sequence.

If the result of final consensus is not agreed (*INTER_SZ_REJECTED*), the proceeding agreement is retried or a new block agreement is performed with the indirect notification of doubtful validators, by resending the *Inter-SZonePREPARE* message including the list of unquestionable *SZone_Validators*. Thus, in the view of the next consensus, the reconfiguration into new *SZone_Validator*s quorum is executed excluding the suspected *SZone_Validator*, in case there are many doubtful validators. In the proposed *SZHBFT* mechanism, an agreement for the joining of new *SZone_Validator* proceeds at the same time during the consensus process of transactions data, so that all future transaction requests can be completed even in presence of faulty or unresponsive *SZone_Validator* members. This coupling of reconfiguration agreement with the consensus of transaction requests can reduce the overhead incurred by the reconfiguration of *SZone_Validator* quorum and also is capable of the faster detection and removal of faulty *SZone_Validator*s.

TABLE III
HIERARCHICAL BA ACCELERATION MECHANISM

- *Smart Contract S : {input A, B} => output C*
- *ServiceZone_ConsensusGroups related to input A, B : SZCG(A), SZCG(B)*
- *Members of ServiceZone_ConsensusGroup K : SZCG(K).m(0) - SZone_Leader , SZCG(K).m(1)…, SZCG(K).m(l) - SZone_Validators*
- *EIG tree of Node i : EIG_T[i]*

BEGIN:

---$r_i \leftarrow 0$, $v_i \leftarrow 0$, $ts_i \leftarrow 0$, result ← false, f ← the number of byzantine nodes
--- cond ← consensus status, rpy_num ← the number of receiving replies

While(result $\neq$ true) {
  $r_i \leftarrow r_i + 1$; if($r_i > f + 1$) break;
  if(*cond* == **_AGREED* || **_REJECTED*) result ← true;

In the status *cond*;
  - *INPUT_SZ_AGREED, INTER_SZ_AGREED*:
    attach the $block_c$ to *Inter-ServiceZone Public Ledger*;
  - *INTER_SZ_REJECTED*: resend a ***Inter-SZonePREPARE*** including ***the list of unquestionable replier nodes*** for the **reconfiguration agreement**;

In the role *Inter-SZone Leader*;
  IF(receive the requests from clients during specific time) {
    perform the batching process [$block_c$];
    repeat at input *SZCG(A), SZCG(B),* and output *SZCG(C)*;
      - multicast ***Inter-SZonePREPARE***(r_i, v_i, ts_i, $block_c$);
  } ELSE IF(receive *the replies of Inter-SZonePREPARE*) {
    case[*rpy_num*< 2*f*+1]: *cond* ← *INTER_SZ_REJECTED*;
    case[*rpy_num*≥ 3*f*+1]: *cond* ← *INTER_SZ_AGREED*;
    default:
      repeat at input *SZCG(A), SZCG(B),* output *SZCG(C)* and *SZCG(ISZL)*;
        - multicast ***Inter-SZoneCONFIRM***(r_i, v_i, ts_i, $block_c$);
        - *cond* ← *INTER_SZ_PREPARED*;
      IF((replies for the *Input SZone Agreement* ≥ 2f+1)&&
        (*cond*==*INTER_SZ_PREPARED*)) { cond ← *INPUT_SZ_AGREED*; }
  } ELSE IF(receive *the replies of Inter-SZoneCONFIRM*) {
    case[*rpy_num* < 2f + 1]: *cond* ← *INTER_SZ_REJECTED*;
    default: *cond* ← *INTER_SZ_AGREED*; }

In the role *SZone_Leader & SZone_Validators*;
  IF(receive ***Inter-SZonePREPARE***) {
    At *SZone_Validators of* **output** *SZCG(C)*;
      - reply with a signature after validating $block_c$;
    At input *SZCG(A), SZCG(B);*
      - perform *Func* ***EIG_algorithm_for_BA***(SZCG(cg).m(i), EIG_T[i]);
      - reply with including the decision of *Input SZone Agreement*;
  } ELSE IF(receive ***Inter-SZoneCONFIRM***) {
    At Every *SZone_Validators*;
      - reply after confirming $block_c$ with the history signatures; }
}

END

▪ *Func* **EIG_algorithm_for_BA**(Node i, EIG_T[i])
  : Byzantine Agreement Algorithm using **EIG tree(f+1 rounds)** for **n nodes and up to f faulty nodes**
  --- *Exchanged Information Filling Stage* : Decorate ***val(x)*** in order to **top-down** of EIG tree per each round, where x is the node label to receive incoming information from any node x.
  --- *Byzantine Agreement Decision Stage* : Redecorate the tree **bottom-up**, defining ***newval(x)*** at the end of f+1 rounds.
    • Leaf Node : **newval(x) = val(x)**
    • Non-leaf Node : **newval(x) = {strict majority value of children nodes in the tree}**
  Node i applies a final decision with newval(root) in the EIG_T[i].

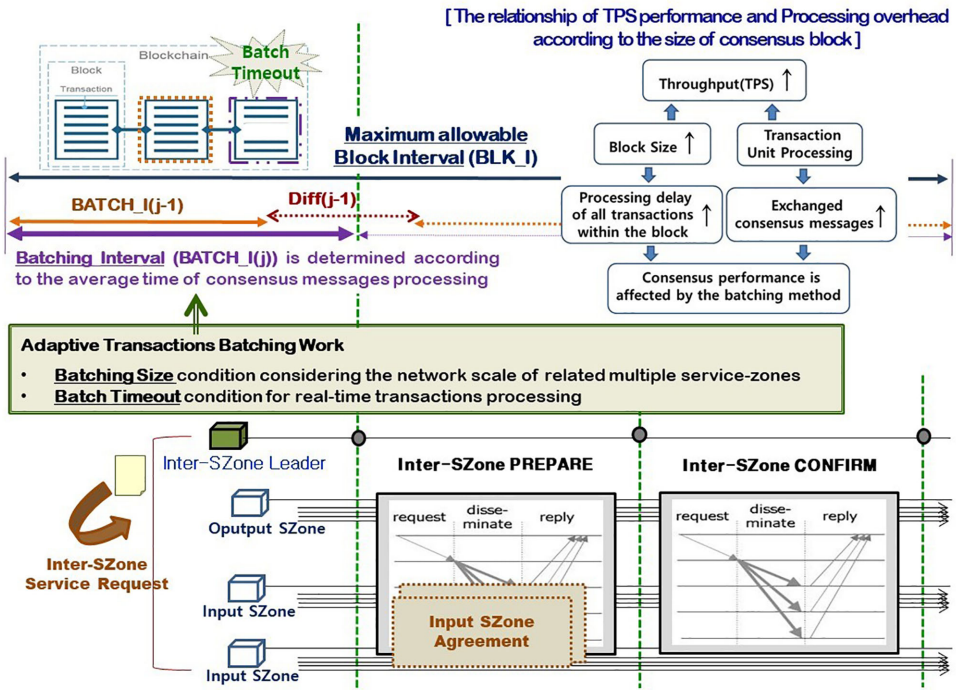


Fig. 6. Concept of adaptive transactions batching work.

C. Adaptive Transaction Batching Work

The factors of the most significant challenges in the design of scalable consensus protocol are to increase their transaction processing performance and to minimize latency in terms of the finality decision of block consensus along with transactions validation and confirmation. Thus, an appropriate choice for the average size of a block and the block generation frequency is crucial in order to maximize the transaction throughput with low latency. However, a block interval in existing consensus algorithms is not adjustable and is usually periodic. On the other hand, when a blockchain protocol is applied to the domain of IoT service, the block interval should be as short as possible so that the transaction can be confirmed immediately. Hence, the two parameters of block interval and block size need to be tuned to optimize the tradeoff between the transaction throughput and a consensus latency at a given security level. The tuning of these parameters is also subject to the network size and network topology, such as the number of consensus nodes and the number of service-zones and service-zone members.

In the process of batching for service transactions, the *Inter-SZone Leader* performs the batching interval adjustment described in Fig. 6 and Table IV according to the number of consensus nodes and the transactions occurrence rates, so that ordered transactions can be created into the consensus block based on a batching timeout and the maximum size of a block. If the batching interval is smaller, then the transaction is processed faster, whereas the consensus overhead for a block with the small size will be caused by the round-trip delay from the exchange of frequent consensus messages. For realizing dynamic adaptation of the batching algorithm, in the proposed *SZHBFT* mechanism, the *Inter-SZone Leader* continually estimates and coordinates the workloads of service transactions for batching transactions according to the time interval of transactions occurrence. Also, the batching timeout is estimated based on the average time of agreement processing for service transaction requests, which is associated with the

TABLE IV
ADAPTIVE BATCHING INTERVAL SCHEME FOR EFFICIENT BLOCK PROPAGATION

- N: the number of consensus nodes
- n_{sz_i}: the number of sz_i(***service-zone group i***) members
- f_{sz_i}: the number of fault tolerant nodes in sz_i
- BS_{max}, $BS(j)$: the maximum block size, block size on the round j
- TS_{avg} : the average transaction size
- TX_{max}: the maximum number of transactions carried within a block
- $X\%$ ***block consensus delay***: the time taken for $X\%$ of the nodes to receive an agreed block
- $R_{tr}(t)$: the average rate of transaction occurrence on the time t
- $\lambda_b(r)$: the average delay of block propagation on the round r
- $d_{bp}(r)$: the average response delay of a block prepare request on the round r
- α : the weighting factor of EWMA(Exponentially Weighted Moving Average) statistics ($0 < \alpha < 1$)

▪ $X\%\ effective throughput = BS_{max} / (X\%\ block consensus\ delay)$

▪ $BS_{max} / (X\%\ effective throughput) < BLK\_I$ ⋯ Block Interval

▪ $TX_{max} = BS_{max} / TS_{avg}$

▪ $\lambda_b(r) = \alpha \times d_{bp}(r-1) + (1-\alpha) \times \lambda_b(r-2)$

▪ **Batching_Interval**(TX_{max}, $R_{tr}(t)$, N, $max\ n_{sz_i}$, f_{sz_i}, $\lambda_b(r)$)

BATCH_I >= $\lambda_b(r) \times [\, n_{sz_i} \times (2f_{sz_i}+1) + 2N \,]$
or BATCH_I =< $TX_{max} \times R_{tr}(t)$
Return minimum **BATCH_I**

▪ ***Adjust a Batching Interval***

Diff(j-1) = [**BLK_I - BATCH_I**(j-1) *]*
- [the average time of block(j-1) consensus processing]

BATCH_I(j) = minimum **BATCH_I(j)** + *Diff(j-1)* $\times \frac{BS(j)}{BS(j-1)}$

number of consensus nodes and the number of service-zone and service-zone members. When the batching timeout expires or the number of buffering transaction requests fits within a batching size, the *Inter-SZone Leader* assigns a sequence number to the block made through this batching mechanism and then runs the agreement process for the corresponding block.

We consider that the *Inter-SZone Leader* is an honest node due to the Randomness property of Secure Grouping from Theorem 1 presented in Section V. The maximum allowable block interval is defined as the time it takes for a block to be agreed upon and stored in the ledger after one more transaction is included in the block. Thus, this block interval is made up of the average time of block propagation during the consensus process with the delay time of block generation. As described in Fig. 6, when the maximum allowable block interval is fixed, the interval of adaptive transaction batching work is adjusted according to the average time of processing consensus messages to determine how often to generate a block, which is a set of *m* transactions. Consequently, because the size of the consensus block is related to the transaction processing performance, the batching interval has to be regulated with the change strategy of block size considering the limit of consensus latency. The idea of the algorithm described in Fig. 6 and Table IV is to adjust the parameter of a batching interval dynamically to find a balance between high transaction processing performance and low service latency.

As represented in Table IV, the throughput of a blockchain consensus protocol is obtained by dividing the number of transactions in a block or the block size by the interval of block generation. In Table IV, the key metric "*X% effective throughput*" is defined as which blocks propagate within an average block interval period to the percentage of nodes [22]. In order to consider an optimized tradeoff between higher throughput and lower latency, the two most straightforward conditions to set the batching interval (*minimum BATCH_I*) are the interval reduction until a minimum block interval and the interval increase within the generation of a block with maximum size. Due to above-mentioned conditions, the algorithm of adaptive batching interval described in Table IV needs to setup the lower bound and upper bound of a batching interval for satisfying the minimum latency limit from the minimal consensus processing overhead and the maximum throughput limit from a maximal block size.

If applied the way to reduce a block interval, the interval of transaction batching is possible to shorten until a minimum block propagation time. Thus, in Table IV, there is the minimal lower threshold such as $\mathrm{BATCH\_I} >= \lambda_b(r) \times [n_{sz_i} \times (2f_{sz_i} + 1) + 2N]$ expressed from the condition of minimum latency limit. Here, $\lambda_b(\mathrm{r})$ represents the average delay of block propagation, which utilizes an exponentially weighted moving average (EWMA), so as to calculate a cumulative weighted delay for the average delay of majority responses $(d_{bp}(\mathrm{r}))$ regarding to the request of block prepare (*Inter-SZone PREPARE*). Also, the minimal lower threshold such as $\mathrm{BATCH\_}I >= \lambda_b(r) \times [n_{sz_i} \times (2f_{sz_i} + 1) + 2N]$ can be induced from the minimum block propagation time, by referencing the consensus messages complexity of $O(szm^2 + 2 \times szg \times szm)$ required when conducting the process of agreement clustered with the related multiple service-zones (N: the number of total nodes, szg: the number of service-zones, szm: the number of service-zone members). But the minimal interval reduction may also be problematic because it increases the exchanged consensus messages in the form of a block of transaction unit.

If used the method raising the number of transactions in one block, as the size of a block increases, the block propagation time between nodes also increases. Hence, if the interval of transaction batching increases until filling a maximum block size, it increases the interval of block generation, and also the situation that exceeds the maximum allowable block interval can occur. In an algorithm for the strategy of adaptive batching interval described in Table IV and Fig. 6, if a batching interval (*BATCH_I(j)*) is decided into an input value, then an output value is computed into the time that it takes for one more transaction included in a block to be agreed upon and stored on the ledger. In order to fit the corresponding output value inside a maximum allowable block interval, a next input value (*BATCH_I(j + 1)*) is adjusted dynamically according to the calculated output value previously. After finishing the agreement on the ordered transaction requests within a batching block, the *Inter-SZone Leader* sends the related responses for the processed transaction requests to the requesting clients.

IV. Experimental Evaluation

In this article, we have used a BFTSim simulator [23] to analyze the performance of the proposed consensus mechanism (*SZHBFT*) on a network topology in the NS-2 simulation environment. The performance of hierarchical byzantine agreement acceleration mechanism for processing the cross-shard transactions over interservice zones has been evaluated by the experiments with the 200-nodes running on the simulation network based on the NS-2. In these setup environments, we have measured on the performance characteristics of the proposed *SZHBFT* mechanism under various simulation conditions, such as the number of network nodes, group size, and the fraction of faulty nodes. And then, we have analyzed its effectiveness using three metrics: the communication complexity, the TPS throughput, and a consensus latency.

First, we compare the proposed *SZHBFT* with a PBFT protocol in two sets of experiments, varying the group size and the fault ratio of group nodes and increasing total consensus nodes. Basically, the traffic occurred from the exchanged communication messages is the significant factor of consensus performance. Hence, reducing the traffic of exchanged messages during the consensus process is required to improve the performance and scalability. For these reasons, the block size, which is the maximum number of transactions included within a block, is the important factor to be considered related to the overhead of exchanged messages traffic. If the size of a block increases greatly, the propagation time of the corresponding block between nodes also increases. The consensus for a block in the form of a transaction unit rapidly raises the communication overhead due to the great increase in exchanged consensus messages.

Therefore, we have also conducted the experiment for checking the correlation between the propagation delay of a block in the consensus network and the block size.

We assume that there are total N consensus nods containing at most F malicious nodes in the simulation network environments. In the proposed *SZHBFT* mechanism, the process for agreement clustering is conducted among the related service-zones, which are configured into the number of service-zone groups of szg with the size of szm members by dividing total N nodes (N: the number of total nodes, szg: the number of service-zones, and szm: the number of service-zone members). The message complexity exchanged in the PBFT can reach $O(N^2)$ according to the consensus process. If a client sends N messages to total N consensus nodes for delivering a transaction, he receives N messages in the reply stage after performing the agreement process via the exchange of consensus protocol messages, such as following: the $(N-1)$ messages transmitted by the primary node in the pre-prepare stage, the $((N-1)\times(N-1))$ messages in the prepare stage transmitted by replica nodes, and the $(N\times N)$ messages delivered from replica nodes in the commit stage. Because of requiring the exchange of $(2\times N^2+N)$ messages via above-mentioned five consensus communication steps, the PBFT will impose a large traffic overhead. On the other hand, according to the mechanism of hierarchical agreement clustering with the related service-zones, the presented *SZHBFT* protocol must exchange the number of consensus communication messages of $((2\times szm^2+szm)+2\times szg\times szm)$ between total N nodes via the consensus process. However, in the algorithm of the proposed *SZHBFT*, the communication overhead may vary depending on the ratio of the number of input service-zones and output service-zones.

Fig. 7 shows the comparison results for the traffic of consensus communication messages in the proposed *SZHBFT* with that in the PBFT according to the number of total consensus nodes (N) under the various sizes of a service-zone group (szm). In Fig. 7, we define the *SZM* $\times$ *SZG_messages traffic* into the traffic of consensus communication messages required in the presented *SZHBFT* mechanism on the multiple service-zones configured into szg subgroups having szm members. If conducting the hierarchical agreement clustering with the related service-zone groups, as one can see in the graph shown in Fig. 7, the traffic of consensus communication messages substantially decreases more than that in the PBFT according to the impact of the size of a service-zone group (szm) on the traffic. Hence, the selection of an optimal size for a service-zone group (szm) is the issue to be considered for the transaction processing performance.

Fig. 8 exhibits the transaction throughput and the average consensus delay measured from the process of agreement among multiple nodes with the increase in total consensus nodes (N). In Fig. 8, each of *SZM* $\times$ *SZG_TPS throughput* and *SZM* $\times$ *SZG_consensus delay* represents the transaction processing performance and average consensus latency that is measured from the proposed *SZHBFT* mechanism over the multiple service-zones configured into szg sub-groups having szm members. From the result graph shown in Fig. 7, the number of consensus communication messages among the participant nodes

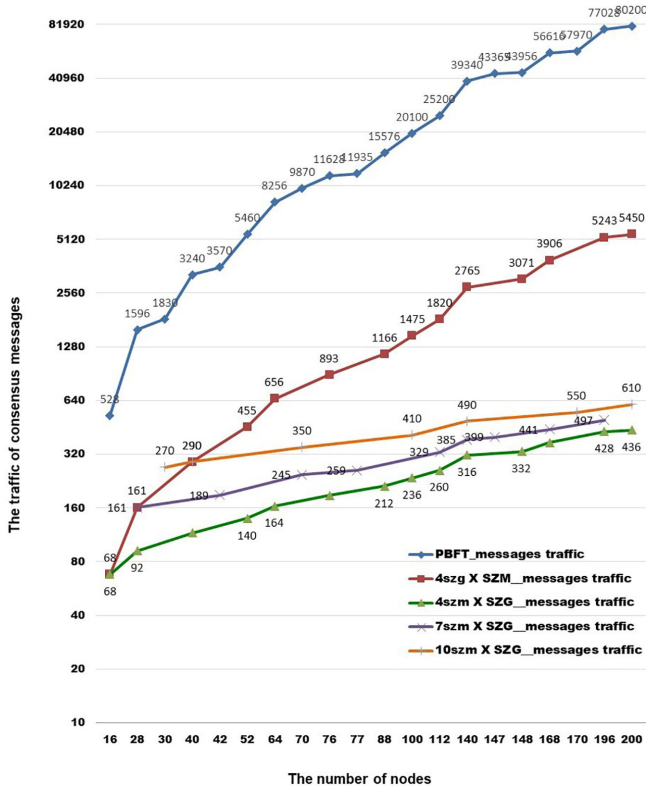


Fig. 7. Traffic of consensus messages according to the number of nodes.

in the *SZHBFT* $(szm^2+2\times szg\times szm)$ has been minimized than those in the PBFT (N^2). Therefore, as shown in the comparison results of *SZM* $\times$ *SZG_consensus delay* with *PBFT_consensus delay* within the graphs of Fig. 8, the overall latency overhead of the proposed *SZHBFT* protocol has been reduced than that in the PBFT accordingly. Regardless of the size of a service-zone group (*szm*), the result graph shown in Fig. 8 demonstrates that the proposed *SZHBFT* mechanism has higher performance than the PBFT in terms of a consensus delay and the TPS throughput due to the lower overhead of communications messages.

We have also evaluated the proposed *SZHBFT* mechanism to analyze the effect of the size of a service-zone group (*szm*) on the communication overhead in the process for hierarchical agreement clustering based on multiple service-zones. First, for our examination regarding the size impact of a service-zone group (*szm*) from the result graph shown in Fig. 8, we have analyzed the graph item of *4szg* $\times$ *SZM_consensus delay* along with comparing that of *4szm* $\times$ *SZG_consensus delay* under the condition of various sizes for a service-zone group (*szm*) from the same total number of consensus nodes. As known from the comparison results of *4szg* $\times$ *SZM_consensus delay* with *4szm* $\times$ *SZG_consensus delay* within the graphs in Fig. 8, the consensus latency in the graph item of *4szg* $\times$ *SZM_consensus delay* is much greater than the latency value displayed in the result graph of *4szm* $\times$ *SZG_consensus delay*, as the size of a service-zone group (*szm*) increases. This is similar to the comparison results for the traffic of consensus communication messages in the graph items of *4szg* $\times$ *SZM_messages traffic*

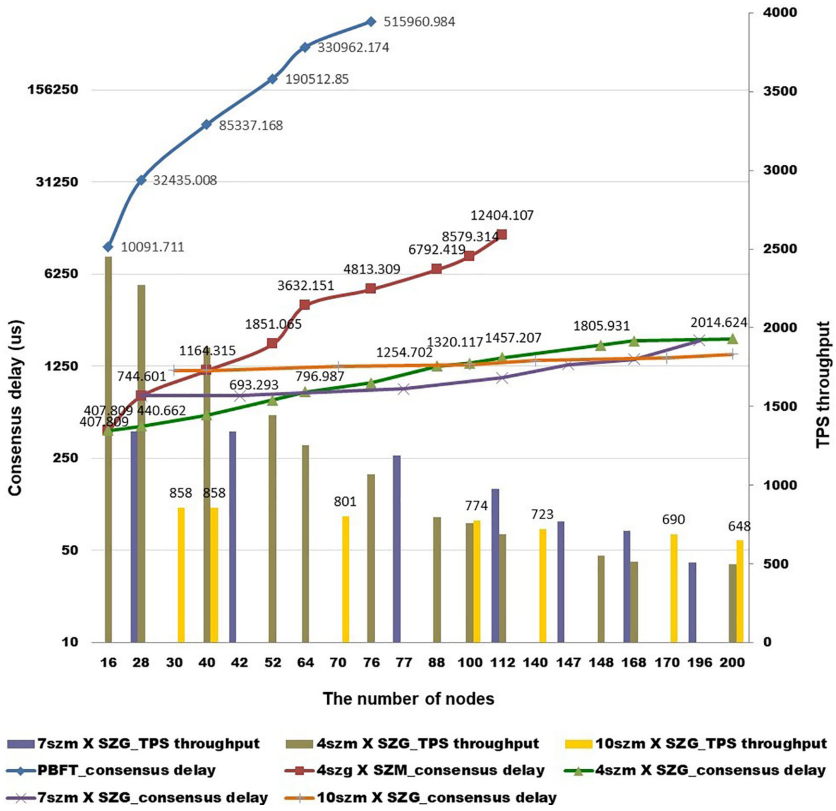


Fig. 8. Consensus delay and TPS throughput according to the number of nodes.

and *4szm* × *SZG_messages traffic* in Fig. 7. From this point of view, the size of a service-zone group (*szm*) has more impacts on the average consensus delay than the number of service-zone groups (*szg*).

In the proposed *SZHBFT* mechanism, we examine the relationship between an average consensus delay and both factors, the number of total consensus nodes N and the size of a service-zone group *szm*. We have compared the results of consensus delay in the graph items of the *4szm* × *SZG_consensus delay*, *7szm* × *SZG_consensus delay*, and *10szm* × *SZG_consensus delay*, when the size of a service-zone group (*szm*) for each item is 4, 7, and 10, from the result graph shown in Fig. 8. When analyzed the above-mentioned comparison results from the graph of Fig. 8, as the number of total consensus nodes N increases, the *10szm* × *SZG_consensus delay* has the graph form with slightly more flat slope than a little steep slope on the graph item of the *4szm* × *SZG_consensus delay*. This is because an increase rate for the number of additional formed groups in the graph item of *4szm* × *SZG_consensus delay* is much higher than those in the graph item of *10szm* × *SZG_consensus delay,* as the number of total consensus nodes increases. In case of the graph item of *10szm* × *SZG_consensus delay* in Fig. 8, since the number of total formed service-zone groups is much smaller than those in the graph item of *4szm* × *SZG_consensus delay* , the consensus delay mostly increases from the influence due to the complexity of consensus communication messages by the size of a service-zone group ($szm = 10$). Similarly, as the number of total consensus nodes N increases, only the graph

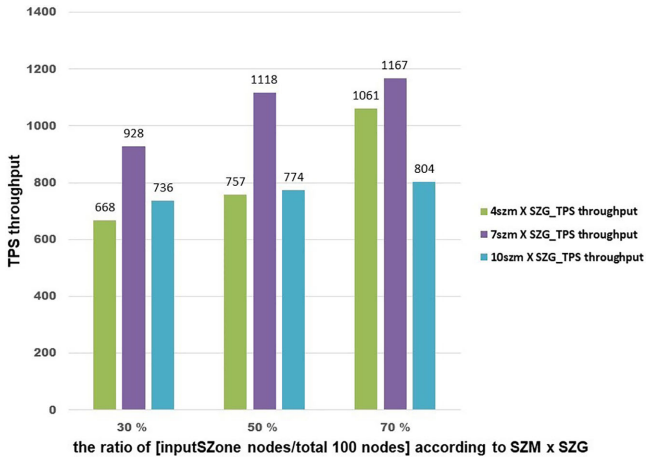


Fig. 9. TPS throughput according to the ratio of input service-zone nodes.

item of *10szm* × *SZG_ TPS throughput* has been reduced by a slight change rate, if comparing the results with the graph items of *SZM* × *SZG_TPS throughput* in Fig. 8. Therefore, when the size of a service-zone group (*szm*) is fixed to a constant value, the communication overhead also remains confined to some average value in terms of a consensus delay and the TPS throughput.

Fig. 9 displays the results of TPS throughput according to the ratio of input service-zone nodes among the total 100 nodes under the condition of different group sizes. As shown in Fig. 9,

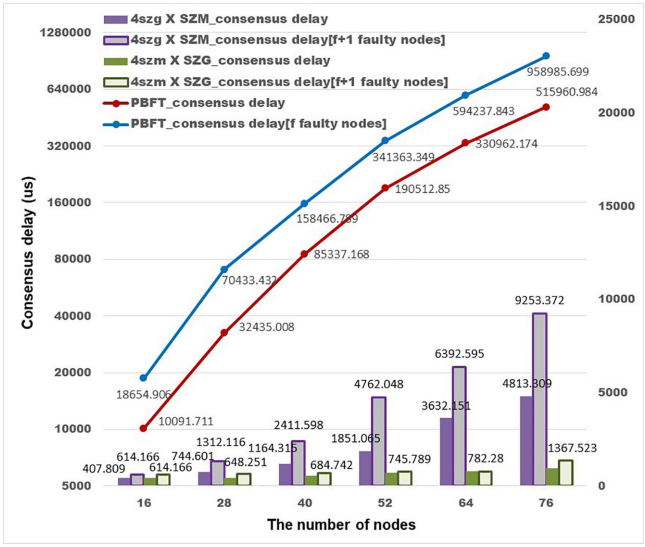


Fig. 10. Consensus delay according to the fraction of faulty nodes.

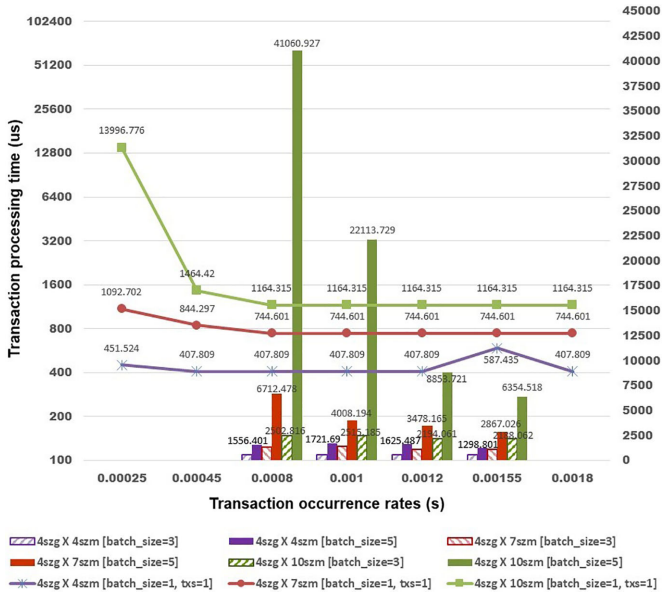


Fig. 11. Transaction processing time according to the batching size and transactions occurrence rates.

overall, the higher the ratio of input service-zone nodes, the higher the performance of TPS throughput, regardless of the size of a service-zone group (*szm*). Also, the size of a service-zone group (*szm*) has affected the performance of TPS throughput along with the ratio of input service-zone nodes, if examining the result graph in Fig. 9. If analyzing the TPS throughput under the same condition having the number of total nodes of 100, the comparison of the results in Fig. 9 demonstrates that when the size of a service-zone group (*szm*) is seven, the maximum TPS throughput is able to be achieved.

From the viewpoint of analyzing results presented in Figs. 8 and 9, because the larger the number of consensus nodes, the greater the performance of TPS throughput, and the shorter the consensus delay if compared with the PBFT, the scalability of the proposed *SZHBFT* mechanism is overall achieved based on the relation between the number of total nodes and the group size.

In the algorithm of the proposed *SZHBFT* mechanism, even though there are more than F faulty nodes, the process for the agreement clustering based on multiple service-zone groups can be conducted and also reach to the final consensus conclusion, according to the ratio of consensus nodes within input and output service-zones. As displayed in the result graph of Fig. 10, in the experiment where there are more than F faulty nodes, the PBFT cannot finalize the agreement process due to the infinite consensus time as the number of nodes increases. But the proposed *SZHBFT* mechanism handles transactions normally along with an increased consensus delay.

We consider an impact of the size of a block on the performance of the hierarchical BFT agreement clustering mechanism over the related multiple service-zones. There is the correlation between the propagation delay in the network and the size of a block, which is able to vary until the maximum number of transactions to be included within a block. Therefore, this size is a significant parameter that controls the TPS throughput attained by the proposed *SZHBFT* mechanism. Fig. 11 plots the transaction processing time according to the size of a service-zone group, under the condition of various sizes for the transactions batching formed from the different time intervals of transactions occurrence. When the processing latency for one transaction according to the increase in transaction occurrence rates becomes close to the saturation point (the case of the transaction occurrence rates of 0.00025 s in Fig. 11), the processing latency increases at high speed significantly. This is related to the propagation delay of communication messages depending on the number of network nodes participating in the agreement process for one batching block. Overall, at the high rates of transaction occurrence, the average transaction processing latency for a batching block including multiple transactions is much lower than the processing latency for one transaction in terms of processing multiple transactions, as shown in Fig. 11.

However, due to the propagation delay of communication messages depending on the size of a service-zone group and the number of consensus nodes, the average occurrence rates of transactions traffic that can be handled becomes much lower, as the size of agreement group is larger or the number of participating nodes is greater (if referencing the graph items of $4szg \times 4szm\ [\ batch\_size = 5]$, $4szg \times 7szm\ [\ batch\_size = 5]$, and $4szg \times 10szm\ [batch\_size = 5]$). At the high rates of transaction occurrence, the batching block can be made faster rather than waiting for a batching timeout due to the limited maximum size of a block packet according to the network bandwidths. Thus, the final consensus latency decreases due to time savings in the batching operations when the rates of transaction occurrence increases above the average occurrence rates that can fill up to the maximum size of a block. In order to achieve high TPS throughput and low transaction processing latency, in the adaptive batching scheme, the larger block is used at the high rates of transaction occurrence, and the smaller block is used for fast transaction processing at the low rates of transaction occurrence.

Overall experiments show that it is possible to improve a consensus delay and the TPS throughput of the proposed *SZHBFT* mechanism than those in the PBFT, and also its performance does not deteriorate dangerously in terms of security-related metrics.

V. System Model and Security

The activities of malicious nodes are arbitrary, and they may also falsify information or simply stop working by joining the network at any time. Thus, we assume the byzantine failure model where the malicious node cannot break cryptographic techniques, such as collision-resistant hash, encryption, and signature. The goal of the proposed agreement mechanism (*SZHBFT*) is to provide high throughput and low latency based on multiple consensus subgroups composed dynamically. Hence, by the consensus mechanism based on service-zone groups, each consensus subgroup concurrently processes the transactions assigned to it, and also these transactions are committed securely both within a consensus subgroup and across consensus subgroups.

Consensus nodes based on multiple service-zone groups can establish a common consensus, which satisfies correctness conditions when the following constraints are met [24].

1) *Consistency:* All correct nodes in the network must reach a common agreement value, and all decisions are final.
2) *Validity:* If the source node is correct, then all correct nodes in the network should agree on the initial value sent by the source node.
3) *Termination:* All correct nodes reach a decision.

These correctness conditions are defined as the safety and liveness properties that a distributed consensus algorithm should satisfy. The safety property is violated by the consistency and validity conditions if any two processes decide on different values. And a termination condition should be specified by the liveness property. For a system to continue executing correctly until it terminates, all nodes must eventually reach the same conclusion.

When improving a BFT protocol only, there is a limit on the scalability of consensus protocol due to the quadratic communication cost of $O(N^2)$. In the proposed *SZHBFT*, the network and the blockchain states are partitioned into smaller consensus subgroups based on service-zone, where each consensus subgroup is small enough to run a BFT protocol. But the process of assigning nodes to consensus subgroups, called the secure shard formation, must be done randomly to ensure security by tolerating a certain number of byzantine failures. Each shard needs to obtain an agreement localized within the composed consensus subgroup based on service-zone, which makes the consensus procedure more efficient.

System Model: Assume that there exists the blockchain system of *N* participating nodes with a fraction *s* of the network, which is controlled by the byzantine adversarial node. The nodes are partitioned into *k* consensus subgroups based on service-zone, and then each consists of $n \ll N$ nodes while tolerating at most $f < n$ byzantine nodes. Let $tx_{i,j}$ represent the *j*th transaction in the *i*th block. The set of transactions *X* is divided into *k* disjoint subsets $X_i = \{tx_{i,j}\}$, for every j $\in \{1 \ldots |X_i|\}$. We consider a system consisting of *N* validating nodes and *C* clients, which communicate by passing a message, m, digitally signed by a validating node *i* as $\langle m \rangle_{\sigma_i}$. Here, we employ digital signatures and make use of a cryptographic hash function to compute message digests. Each consensus subgroup based on service-zone (SZ_i) has at least $3 \times (f_i) + 1$ validating nodes, where f_i is the maximum number of nodes $(f_i \leq \frac{n-1}{3})$, which may be faulty within a service-zone (SZ_i). A client demands the work of operation (*query/update*) to the system by sending a message to one or more validating nodes. Thus, each validating node produces a batching block, including a sequence of operations, as its output that reflects the order in which he executes client operations. After the validation and execution of a corresponding operation, he sends a reply containing the result of the operation to the client. The transaction tx_c issued by a client *c* is considered to be completed by the *c* if *c* receives at least $(f+1)$ valid $\langle REPLY,\ s, v, c, r, t, i \rangle_{\sigma_i}$ messages from different replica i, where s is the sequence number, v is the view number, c is the client, t is the timestamp, and r is the result of execution.

We will prove that the proposed *SZHBFT* satisfies the correctness criteria, such as randomness and safety.

Definition 1 (Randomness Property of Secure Grouping): If more than a fraction THR of the weighted participants for a secure grouping selected at random is honest, then all honest participants agree on the validation result of the same block. Also, all participants will execute the VRF hashing to determine if they are selected as validators with their own eligibility based on the proximity from the closest random landmark node.

Definition 2 (Safety Property of Consensus): A system is safe if the delivered block is committed by the agreement of at least $(2f + 1)$ replicas after a correct primary leader proposes the block of ordered transactions with block number b. And then, in the safety blockchain system, any block that has been committed earlier will have smaller block number $(b' < b)$ within the chain. Also, the order of transactions within the block will remain identical in all correct replicas.

We now provide the proof of correctness and analysis of the proposed *SZHBFT*.

Lemma 1 (Randomness Property of VRF): Given a random input, the verifiable random function produces a pseudorandom output along with a proof for the random output, which is considered to be uniformly distributed between 0 and $2^{l_{Rf}} - 1$, where l_{Rf} is the length of the random output

$$Rf_{(\cdot)}(\cdot) : \{0,\ 1\}^{l} \rightarrow \{0,1\}^{l_{Rf}}.$$

Proof: Using a VRF provides interesting properties, such as uniqueness and provability. By the proof π of VRF, every participants that know the public key can verify whether y was generated correctly from x or not. Hence, if $(y,\ \pi) = Rf_{(sk)}(x)$, then $Vf_{(pk)}(x, y,\ \pi) = 1$. Because of the uniqueness property, no values $(pk, x,\ y_1, y_2, \pi_1, \pi_2)$ can satisfy $Vf_{(pk)}(x,\ y_1,\ \pi_1) = Vf_{(pk)}(x, y_2, \pi_2)$ when $y_1 \neq y_2$. Since the hash of VRF is an 1-bit-long value that is uniquely determined from any random input x and the private key, it is indistinguishable from random for any participant that does not know the private key.

Lemma 2 (Unpredictability of Secure Grouping Membership): Until the set of secret RTT factors measured from the closest random landmark node is revealed publicly, a malicious participant without the secret key is not able to modify anything about the final outcome or determine whether the participant p_i is eligible or not, except with a negligible probability.

Proof: From Lemma 1, we can see that a malicious participant can only guess the correct weighting factors, which are the secret RTT factors measured from the closest random landmark node, with at most $1/2^{l_{Rf}}$ probability. In the commit-reveal process, comparison of the differences between the RTT of node *i* measured from landmark node *j* and the RTT of landmark node *j* measured to the reverse direction is made, so as to prevent the modification of the RTT output that is beneficial for a dishonest participant. Also, after the secret RTT factors of participants are revealed within a specified time period, the secret RTT factors of an *SZone_Coordinator* is provided at the last order publicly. With a proof for the priority of the chosen participant, each participant is selected as a validator according to the priority, which is the weighting factor based on the proximity from the closest landmark node from an *SZone_Coordinator*. Hence, the unpredictability of secure grouping membership is ensured.

With properly chosen TH, we can control the probability that exactly *k* out of *n* participants are selected as the validators, who are eligible within the service-zone-based consensus group. By the cryptographic weighting randomness function, the mechanism of secure grouping membership can be modeled as the random sampling problem with two possible independent outcomes, honest node and byzantine node

$$\Pr[X = x] = p^x(1-p)^{1-x}, \;\; x = 0,\ 1.$$

Due to the cryptographic weighting randomness property in the membership decision mechanism, the probability that a participant is eligible for the service-zone-based consensus group is $p = \frac{\mathrm{TH}}{W}$, where $W (W = \sum_i w_i)$ is the total sum for the RTT factors set of participant nodes measured from the closest random landmark node. Thus, the probability that exactly *k* out of n participants are selected is $\Pr[X = k] = B\_\Pr(k; n,\ p)$, and when $k = \mathrm{mode}(X) = \lfloor (n+1)p \rfloor,\ Pr[X = k]$ will be maximum: $\frac{k-1}{n+1} < p \leq \frac{k}{n+1}$. Therefore, the TH that maximizes the probability for selecting exactly *k* out of *n* participants should satisfy the following condition:

$$\frac{k-1}{n+1}W < \mathrm{TH} \leq \frac{k}{n+1}W.$$

Theorem 1 (Randomness Property of Secure Grouping): The unpredictability of secure grouping membership is ensured due to the randomness property of VRF via the commit-reveal process.

Proof: According to Lemma 1 and Lemma 2, we can see directly that a service-zone-based consensus group can securely be composed of the random method.

A malicious node can interrupt the communication between correct nodes until an agreement group is established with less than two-thirds correct members, at which point safety may be violated. In the asynchronous distributed system where network nodes may fail to deliver messages or deliver them out of order, the proposed *SZHBFT* mechanism achieves the safety property if each consensus group has no more than *f* byzantine nodes. In addition, at least $(f + 1)$ rounds of messages exchange are required to achieve the byzantine agreement that agrees on a specific common sequence of actions, despite some *f* malicious nodes within the group [14].

Lemma 3 (Liveness and Majority Properties of Consensus): The block with a specific block number is committed by the validation from at least $(2f + 1)$ correct replicas with a primary leader, and the proposed SZHBFT mechanism satisfies the liveness property when a primary leader is correct.

Proof: Based on Lemma 2, since the unpredictability property of a VRF hashing is applied into the method to assign validators securely, the proposed *SZHBFT* mechanism can guarantee that each selection is completely random, leading to the ratio of adversarial nodes to be at most $p = \frac{f}{n}$, where $n \geq 3f + 1, f \leq \frac{n-1}{3}$ in the BFT consensus. When it is guaranteed that upon receipt of $2f + 1$ *RESPONSE* messages from different replicas, the primary leader node along with all correct replicas also forward a reply message $\langle \mathrm{REPLY},\ s, v, c, r, t, i \rangle_{\sigma_i}$ to each client and the clients will mark the corresponding transaction as the completed status. Also, when there is at least one correct replica that has committed both $B_{s'}$ and B_s, the $B_{s'}$ has a block number s' that is smaller than the block number s of the B_s committed later. In order to guarantee the safety consensus, we are interested in the probability that the system picks less than f byzantine nodes as the validators of consensus groups.

First, we can compute the probability of a faulty consensus group using the hypergeometric distribution, when randomly selecting a validator of n nodes without replacement from a population of *N* nodes containing at most F malicious nodes. And, we are able to bound the probability of a faulty consensus group to be negligible by carefully configuring the size of a consensus group, according to the following equation [11]:

$$\Pr\ [X \geq f] = \sum_{x=f}^{n} \frac{\binom{F}{x}\binom{N-F}{n-x}}{\binom{N}{n}}.$$

The security of validator assignment mechanism in the multiple consensus groups based on service-zone is modeled as a random sampling problem by using the cumulative binomial distribution function that represents the probability of safety consensus guarantee

$$\Pr\left[X \geq \frac{2N}{3}\right] = \sum_{sz=0}^{ISZ_g} \sum_{\lfloor 2N/3IOSZ_g \rfloor}^{\lceil N/IOSZ_g \rceil} \binom{N/IOSZ_g}{k} p^{\left(\frac{N}{IOSZ_g}\right)-k}(1-p)^k.$$

This equation calculates the probability of majority agreement that at most $F = sN$ nodes are controlled by the byzantine nodes within a fraction s of the network in the multiple consensus groups of *N* nodes sampled from an infinite pool of nodes, where the probability of each node being malicious is $p = \frac{F}{N}$, and *X* indicates the random variable corresponding to the number

of honest nodes in the sampled consensus groups (ISZ_g: the number of input groups, $IOSZ_g$: the number of input and output groups).

Theorem 2 (Safety Property of Consensus): The proposed SZHBFT mechanism is safe.

Proof: Based on Theorem 1 and Lemma 3, we can see directly that the proposed *SZHBFT* mechanism satisfies the majority agreement property in the multiple consensus groups composed in the random method, due to the unpredictability of secure grouping membership.

VI. Discussion and Conclusion

Blockchain ecosystem can enhance significant economical and operational value to many services in different industrial domains. The value can be improved in terms of autonomous operational procedures, such as reduction of operations and administration costs, reduction of operations time delays, and reduction of human errors. In addition, the blockchain feature, such as digital identity for humans, organizations, and properties, can enable many new service models via the support for the agreement and security in the form of a distributed method without the need for an authoritative body. Leveraging both the added value and the capability of establishing new business service models will lead to enhancing the competitive capabilities and flexibility of many industries [25]. Thus, the blockchain ecosystem has the ability to revolutionize the paradigm of IoT services with a trusted and auditable open-sharing platform, where any exchanged information is reliable and traceable. The convergence of blockchain and IoT brings a number of benefits, such as [26], [27]

1) decentralization and scalability;
2) digital identity;
3) autonomous operation;
4) security and reliability;
5) secure code deployment.

Currently, although the development of blockchain has made great progress, the application of blockchain into IoT ecosystem is far from the ideal scale, in spite of IoT and blockchain convergence benefits. There are still many research issues that have to be addressed in order to seamlessly use these two technologies together by considering the benefits of IoT and blockchain convergence. Although the blockchain ecosystem converged with IoT brings many opportunities in upgrading the industrial automation, it can face a lot of challenges mainly in five aspects as follows:

1) scalability;
2) security;
3) anonymity and data privacy;
4) resource utilization;
5) smart contract [5].

The scalability of blockchains limits the wide usage of blockchains in the large-scale IoT domain. Many blockchain systems are suffering from poor transactions throughput. Thus, most of the blockchain systems may not be suitable for the IoT services with a large volume of transactions. The transaction throughput is affected by the relevant factors with transaction propagation communication, consensus mechanism, and transaction verification. In the process of transaction verification and consensus, it is necessary to consume a lot of time and power, so the speed of transaction processing is limited. In order to address this issue of blockchain scalability, the proposed consensus mechanism has been designed according to the research direction which processes service transactions based on the sharding approach along with distributed multiledgers structure.

The scheme of the proposed *SZHBFT* mechanism provides the hierarchical agreement solution along with distributed multiledger structure for scalable byzantine resilient agreement by forming secure consensus subgroups in order to minimize the overhead of overall communication messages. The key idea is to automatically parallelize the available processing power by securely organizing multiple consensus subgroups that process a disjoint set of transactions, record agreed transactions, keep track of their validity, and manage the state of objects. The proposed *SZHBFT* protocol can improve the performance for processing service transactions by the hierarchical agreement scheme that is performed in parallel in the form of distributed multiblockchains along with the *Local SZone Private Ledger*s managed on the unit of a service-zone consensus group. In addition, it supports the structure of efficient distributed service blockchain that can track the subsets of ledgers related to the corresponding service, so that the service function can be easily verified. The proposed *SZHBFT* mechanism requires the minimum number of rounds for messages exchange while tolerating the maximum number of allowable faulty nodes in order to reach a common agreement within the related multiple consensus subgroups. And, in the presented *SZHBFT*, the method of reconfiguration agreement integrated with the consensus for processing transactions does not increase the overhead of protocol messages and is also capable of the faster detection and removal of faulty validators.

References

[1] I. Ko, D. Chambers, and E. Barrett, "Unsupervised learning with hierarchical feature selection for DDoS mitigation within the ISP domain," *ETRI J.*, vol. 41, no. 5, pp. 574–584, 2019.

[2] W. Wang *et al.*, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[3] M. WU, K. Wang, X. Cai, S. Guo, M. Guo, and C. Rong, "A comprehensive survey of blockchain: From theory to IoT applications and beyond," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8114–8154, Oct. 2019.

[4] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A survey," *IEEE Int. Things J.*, vol. 6, no. 5, pp. 8076–8094, Oct. 2019.

[5] A. Reyna, C. Martín, J. Chen, E. Soler, M. Díaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 88, pp. 173–190, 2018.

[6] P. Veena, S. Panikkar, S. Nair, and P. Brody, "Empowering the edge-practical insights on a decentralized internet of things," IBM Institute for Business Value, vol. 17, 2015.

[7] M. Samaniego and R. Deters, "Internet of smart things-IoST: Using blockchain and CLIPS to make things autonomous," in *Proc. IEEE Int. Conf. Cogn. Comput.*, Honolulu, HI, USA, Jun. 2017, pp. 9–16.

[8] J. Y. Kwak, S. T. Kim, K. H. Lee, and S. Yang, "Service-oriented networking platform on smart devices," *IET Commun.*, vol. 9, no. 3, pp. 429–439, 2015.

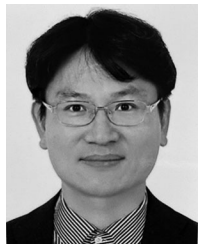
[9] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th ACM SIGOPS Eur. Conf. Comput. Syst.*, Porto, Portugal, 2018, pp. 1–15.
[10] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, 2019. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf
[11] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 123–140.
[12] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd USENIX Symp. Oper. Syst. Des. Implementation*, Feb. 1999, pp. 1–14.
[13] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016. pp. 17–30.
[14] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 279–296.
[15] E. Syta *et al.*, "Keeping authorities "honest or bust" with decentralized witness cosigning," in *Proc. IEEE Symp. Secur. Privacy.*, 2016, pp. 526–545.
[16] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 583–598.
[17] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 931–948.
[18] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *International Workshop on Peer-to-Peer Systems*. Berlin, Germany: Springer, 2002, pp. 53–65.
[19] E. Syta *et al.*, "Scalable bias-resistant distributed randomness," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 444–460.
[20] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "SoK: Sharding on blockchain," in *Proc. 1st ACM Conf. Adv. Financial Technol.*, 2019, pp. 41–61.
[21] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proc. 40th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 1999, pp. 120–130.
[22] K. Croman *et al.*, "On scaling decentralized blockchains," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2016, pp. 106–125.
[23] A. Singh, T. Das, P. Maniatis, P. Druschel, and T. Roscoe, "BFT protocols under fire," in *Proc. 5th USENIX Symp. Netw. Syst. Des. Implementation*, 2008, pp. 189–204.
[24] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.
[25] J. Al-Jaroodi, and N. Mohamed, "Blockchain in industries: A survey," *IEEE Access*, vol. 7, pp. 36500–36515, 2019.
[26] M. Maroufi, R. Abdolee, and B. M. Tazekand, "On the convergence of blockchain and Internet of Things (IoT) technologies," 2019, *arXiv:1904.01936*.
[27] I. Mistry, S. Tanwar, S. Tyagi, and N. Kumar, "Blockchain for 5G-enabled IoT for industrial automation: A systematic review, solutions, and challenges," *Mech. Syst. Signal Process.*, vol. 135, 2020, Art. no. 106382.

Ji-Young Kwak received the B.S. and M.S. degrees in computer engineering from Chonnam National University, Gwangju, South Korea, in 1999 and 2001, respectively.

In 2001, she joined the Electronics and Telecommunications Research Institute, Daejeon, South Korea, where she is currently a Research Engineer with the Network Research Division. Her research interests include blockchain, intelligent network, software-defined network, cloud computing, ad-hoc group communication, multimedia middleware, Internet of Things, and quality of service management.

Jongchoul Yim received the B.S. and M.S. degrees in computer science from the University of Seoul, Seoul, South Korea, in 1997 and 2000, respectively. He is currently working toward the Ph.D. degree in computer network and security with the Chungnam National University, Daejeon, South Korea.

He has been a Research Engineer with Electronics and Telecommunications Research Institute, Daejeon, South Korea, since 2000. He has carried out several projects, including developments of softswitch system and open application programming interface (API) gateway for telecommunication systems. He is currently developing the blockchain for network operators. His research interests include pervasive computing, distributed systems, mobile network, future internet, and blockchain.



Nam-Seok Ko received the B.S. degree in computer engineering from Chonbuk National University, Jeonju, South Korea, in 1998, and the M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2000 and 2015, respectively, both in information and communications engineering.

In 2000, he joined Electronics and Telecommunication Research Institute (ETRI), Daejeon, South Korea, and has participated in various projects, including developments of ATM switching systems and flow-based routers. He is currently the Director who is in charge of Data-Centric Networking Research Section, ETRI. His research interests include future network architecture, information-centric networking, software-defined networks, 5G/6G mobile core network, edge computing, in-network computing, and blockchain.

Sun-Me Kim received the B.S. degree in computer science from Chungnam National University, Daejeon, South Korea, in 1991, and the M.S. degree in computer science from the Pohang University of Science and Technology, Pohang, South Korea, in 1993.

Since 1993, she has been with Electronics and Telecommunications Research Institute, Daejeon, South Korea, where she is currently a Project Leader of hyperconnected Intelligent Infrastructure technology development. Her research interests include packet–circuit–optical converged switching systems, multilayer transport network control and management in packet–optical converged networks, and internet data center (IDC) network control.