DQN based Optimization Framework for Secure Sharded Blockchain Systems

Jusik Yun, Yunyeong Goh, and Jong-Moon Chung, *Senior Member, IEEE*

***Abstract*—High levels of scalability and reliability are needed to support massive internet of things (IoT) services. In particular, blockchains can be effectively used to safely manage data from large scale IoT networks. However, current blockchain systems have low transactions per second (TPS) rates and scalability limitations that make them unsuitable. To solve the above issues, this paper proposes a Deep Q Network Shard based Blockchain (DQNSB) scheme that dynamically finds the optimal throughput configuration. In this paper, a novel analysis of sharded blockchain latency and security level characterization is provided. Using the analysis equations, the DQNSB scheme estimates the level of maliciousness and adapts the blockchain parameters to enhance the security level considering the amount of malicious attacks on the consensus process. To achieve this purpose, deep reinforcement learning (DRL) agents are trained to find the optimal system parameters in response to the network status, and adaptively optimizes the system throughput and security level. Simulation results show that the proposed DQNSB scheme provides a much higher TPS than the existing DRL enabled blockchain technology while maintaining a high security level.**

***Index Terms*—Deep reinforcement learning (DRL), scalable blockchain, IoT, optimization, sharding**

I. Introduction

BLOCKCHAINS can provide transparency and data integrity to data systems, which enable various valuable applications to industry, such as, tracing food, identity management, supply chains, value trading, game systems, etc. [1-5]. Blockchains have the potential to provide decentralization and trust to databases and network operations. However, blockchains face inherent problems when used to support a large number of users that are generating significantly large data sizes, such as massive internet of things (IoT) networks. Large user groups result in slower processing speeds due to the complexity in authenticating a block to chain, and scalability issues arise as the ledger size increases and the blockchain size increases, thus making it more difficult to manage the blockchain in a decentralized manner. Recently, Gartner's 2020 top 10 strategic technology trends indicates the need for research on practical blockchains that can overcome the existing low throughput problems [6]. In addition, due to the advent of 5G mobile communications and the explosive growth of IoT devices, a reliable high throughput performance needs to be considered as an essential property of future blockchain architectures to handle huge volumes of massive data transactions from IoT networks. Therefore, improved scalability is an essential feature that future blockchain systems supporting IoT networks have to achieve [7].

The first blockchain, Bitcoin, adopted a Proof of Work (PoW) algorithm, creating a transparent cryptocurrency exchange system without any intermediaries [8]. However, Bitcoin has a low throughput ranging about 3∼4 transactions per second (TPS), and Ethereum also has a low throughput of 14 TPS [9,10]. This low throughput is not suitable for data processing in large scale IoT networks. In case of PoW, a block consensus is achieved by solving a crypto puzzle based on hash computations, which wastes a lot of time, energy, and resources.

Several research efforts have been conducted to improve the throughput of blockchains [11-18]. First, the throughput of blockchains was improved by replacing the existing PoW consensus algorithm with a Proof of Stake (PoS) [11] or Practical Byzantine Fault Tolerance (PBFT) scheme [15], which reduces the heavy computational cost from complex hash computations. Furthermore, more solutions that can improve the scalability have emerged, which can be divided into off-chain and on-chain solutions.

First, off-chain solutions use offline side-chains, reducing the load and redundancy on the main blockchain (e.g., Lightning network, Plasma, Raiden network) [12-14]. However, off-chain solutions are supported by local offline systems that are not fully distributed, which cannot be fully trusted [21]. In a local offline system with limited participating nodes, a malicious node can easily collude with other nodes that are malicious or compromised to have a manipulated block chained to the blockchain [22]. In this system, malicious behavior can easily influence the consensus process and can simultaneously deteriorate the security level and throughput of the blockchain system.

Next, on-chain solutions use TPS scaling methods by adjusting the blockchain parameters (e.g., block size, interval, block producer, etc.) or structural improvement. Delegated Proof of Stake (DPoS) [18] and Sharding [16-17] are examples of on-chain solutions. DPoS (e.g., EoS) shortens the time it takes to reach a consensus by limiting consensus participants and reducing the message complexity [18]. However, DPoS limits the number of nodes that participate in the consensus, which can be seen as a centralization-based control approach applied to blockchains. Such centralization characteristics in a blockchain result in a higher risk of vulnerability, because

This research was supported by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2020-2018-0-01799) supervised by the Institute for Information and communications Technology Planning and Evaluation (IITP).

The authors are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea (e-mail: awp212@yonsei.ac.kr; rhdbsdud@yonsei.ac.kr; jmc@yonsei.ac.kr).

if fewer nodes participate in the consensus process, malicious users can more easily form a majority of the consensus and collude to validate a fabricated block [19]. Blockchain sharding (e.g., Zilliqa) has the feature of parallelizing transaction processing, which maximizes the throughput in proportion to the number of shards [16-17]. In the case of sharding, the blockchain TPS can be dramatically increased by increasing the number of shards, but the security level may be weakened due to a single shard takeover, where a compromised malicious node takes control of the consensus initiative [20]. On the other hand, reducing the number of shards does not compromise the security level much, but the scalability gain will also be reduced.

According to the blockchain trilemma, any blockchain system can only have at most two of the following three features: decentralization, security, and scalability. These three features have a trade-off relationship with each other, where maximizing one feature can drastically degrade the other feature. In other words, practical solutions that pursue scalability may need to consider the performance degradation in the security level or decentralization. Therefore, it is essential to find the optimal scalability point suitable for the current network situation without compromising other performance values (e.g., security, decentralization) as much as possible.

To solve the above problem, this paper proposes a scalable shard based blockchain system to optimize the throughput while maintaining a high security level. The advantages of using sharding are as follows: 1) The sharding protocol used in the proposed Deep Q network shard based blockchain (DQNSB) model is based on [16] to perform decentralization of shard clustering, where the sharding technique does not delegate mining tasks to other nodes (e.g., DPoS) or use any offline process (e.g., off-chain solutions). 2) Sharding results in a TPS and security trade-off based on the number of shards. The security level can be actively changed by adjusting the number of shards. In addition to the sharding method, the proposed scheme adopts the DRL approach to optimize the performance of blockchains to satisfy large scale and dynamic IoT operations. In particular, the trust concept is integrated into the consensus process based on the proposed DRL framework, which estimates the network's malicious probability by monitoring the consensus result of each epoch. Based on the network trust, the number of safe shards are computed and adaptive control is used to maintain optimal throughput conditions. The results show that the proposed DQNSB scheme shows an improved performance compared to the optimized DRL based blockchain (DRLB) scheme of [23]. The main contributions of this paper are described as follows:

1) The latency of a shard based blockchain system was mathmatically analyzed. In addition, the range of the number of shards required for secure consensus under a malicious validator is derived (proved) in the lemmas. Additionally, a trust model is proposed to compute the inconsistency of block consensus. Based on the calculated inconsistency, the ratio of malicious nodes is estimated, and the security bound for the current consensus status is calculated.

2) The proposed DQNSB scheme combines sharding technology with DRL to acheive an improved performance in throughput while maintaining a high security level. The mathematically obtained latency and security bound values are used in the estimations to form constraints on the DRL process, allowing TPS performance optimization considering the current network status.

Through the malicious attack analysis conducted in previous research [24], the proposed DQNSB scheme can robustly sustain the security performance under malicious consensus conditions. The proposed DQNSB scheme uses the security bound obtained through the trust computation of the network, where the number of shards is selected in a dynamic fashion.

The simulation results show that the proposed DQNSB scheme can provide improved scalability and achieves a higher level of security compared to existing DRL schemes under both honest and malicious network environments. In addition, the proposed scheme shows robustness in dynamic network states in which the malicious rate may be continuously changing. This feature of the proposed scheme makes it more suitable for blockchain-enabled IoT systems that require more stringent levels of QoS support.

The remaining parts of this paper are organized as follows. Section 2 describes the related work, and section 3 describes the secure shard-based blockchain system model. A performance parameter analysis of shard-based blockchains is given in section 4, and section 5 describes network trust computation methods based on consensus history. Finally, section 6 compares the performance of the proposed DQNSB scheme compared to the DRLB scheme of [23].

II. Related Works

The performance optimization framework of scalable blockchains for IoT systems have three areas of related work: IoT blockchains, sharding, and DRL technology applied to blockchains.

A. IoT Blockchains

Recently, with the development of blockchain technology, much research is being conducted to develop new methods to apply blockchains to IoT systems. A blockchain is a cryptography-based data management system that combines peer-to-peer (P2P) networking, distributed ledger, and consensus technology. Due to the structural features of blockchains, attempts to integrate blockchains into various IoT fields are increasing. For example, the authors of [25] propose an energy blockchain system based on the consortium blockchain, enabling safe energy trading in industrial IoT (IIoT) environments. The authors of [26] propose a blockchain based anonymous reputation management system between consumer-retailer to preserve the reviewers' identity and maintain system transparency. The author of [27] also indicates that integration of IIoT and blockchain technology can enhance the efficiency and security of various industrial sectors, bringing new business opportunities.

B. Sharding

Blockchain sharding is a representative technique that can make blockchains more scalable and increase the throughput by parallelizing the verification process of a blockchain. In Ethereum 2.0, which has not been fully disclosed yet, the concept of sharding has been introduced as a roadmap for scalable blockchains. ELASTICO and Zilliqa also have been proposed as more scalable blockchains where the TPS increases linearly corresponding to the number of shards through transaction sharding techniques [16,17].

C. DRL Technology Applied to Blockchains

Reinforcement learning is a well-known approach to solve dynamic control problems [28]. DRL is a type of dynamic programming that combines traditional reinforcement learning with deep neural networks (DNN), which has been used in games, such as Atari [29]. A promising DRL technology is DQN, which uses a neural network as a function to find the best action whenever a particular state comes in. By incorporating neural nets into reinforcement learning, DRL technologies can efficiently approximate even more complex and diverse space states. Recently, performance optimization schemes with reinforcement learning have recently been proposed in the blockchain-enabled IIoT field. The authors of [23] first proposed a DQN-based blockchain scheme to a large scale IoT network. The scheme optimized the blockchain throughput under various constraints by adjusting the blockchain parameters, such as, time interval, consensus algorithms, block size, etc. However, [23] does not deal with malicious attacks from nodes in the blockchain network. The authors of [30] proposed a blockchain enabled data collection and sharing scheme for IIoT, ensuring high levels of security under malicious attacks. However, [30] focuses on the energy efficiency and fairness of data collection but does not consider throughput scalability. The authors of [31-32] proposed blockchain optimization frameworks in various IoT environment, and performed mathematical analysis and optimization on throughput and latency. However, both [31] and [32] do not consider dynamic malicious attacks secnarios.

III. System Model

In this section, the structure of a sharded blockchain supporting an IoT network is introduced, where the overall system structure is illustrated in Fig. 1.

A. Sharded Blockchain Supporting IoT Networks

In Fig. 1, transactions are generated in various application sectors of the IoT network (e.g., smart factory, smart home, smart vehicle, robotics, monitoring systems, etc.), where the data may be shared or processed in different domains. The generated transaction data (e.g., monitoring data, GPS information of smart vehicles, home information, etc.) observed in different domains can share the database through a blockchain infrastructure. The blockchain network receives transactions from the IoT network and records them in distributed ledgers to ensure reliable data management.

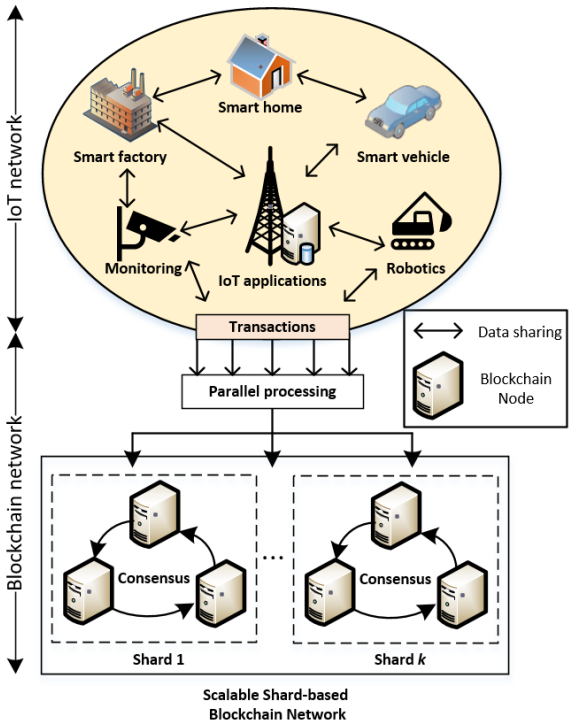

Fig. 1: Sharded blockchain system supporting an IoT network.

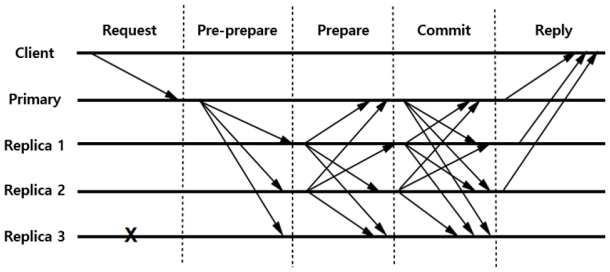

Fig. 2: PBFT consensus algorithm.

To efficiently handle significantly large amounts of data from the IoT network, a shard blockchain network is used to process the numerous transactions in a parallel manner. The shard based blockchain will use the following steps to complete a transaction request. First, blockchain validators are clustered into different shard groups, where different shards independently create blocks and verify the integrity of the blocks through intra shard consensus within the shards. Second, the blocks created in each shard are merged and verified again by a final consensus, and the new block is chained to the blockchain.

B. Sharded Blockchain Architecture

To explain the detailed structure of the sharded blockchain, a description of the PBFT consensus algorithm, the mechanism for clustering shards, and the consensus model assumptions are provided.

1) PBFT Consensus Algorithm

PBFT is a distributed consensus algorithm widely used in many blockchain systems, such as, EOS, Hyperledger, Zilliqa [15]. PBFT used in blockchains consists of the following three

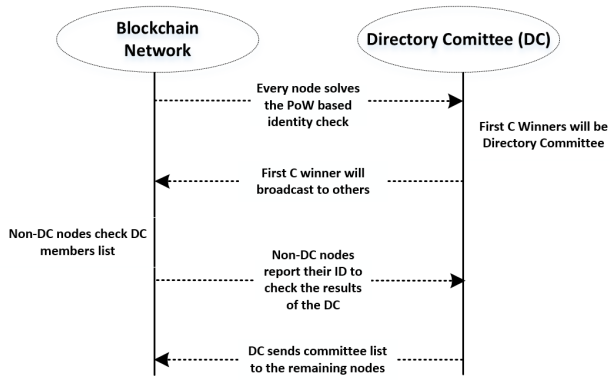

Fig. 3: Shard clustering mechanism.

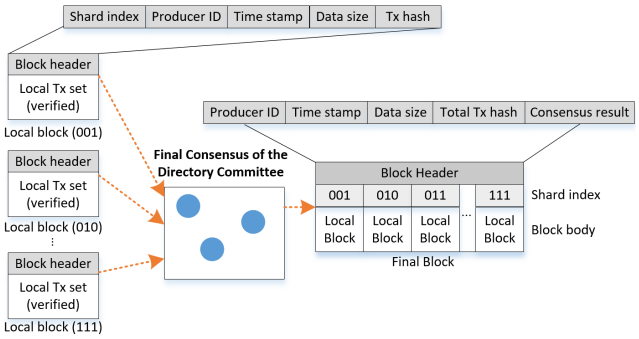

Fig. 4: Structure of a local and final block.

phases, as shown in Fig. 2, which are ***Pre-prepare***, ***Prepare***, and ***Commit***.

In the pre-prepare phase, the primary node collects transactions from the transaction pool, creates a block, and broadcasts the block to the other nodes, where these nodes are called replicas in [15]. Next, in the prepare step, all replicas (except the primary) compare whether the blocks received from the primary node are identical. This procedure prevents selective forwarding of the primary node. If the replicas receive the same message from the primary, the commit step validates the message. In each procedure of PBFT, all procedures are successfully processed if more than two-thirds of the votes from the participating nodes consent. Therefore, the ratio of faulty nodes allowed in a PBFT agreement must not exceed one third of the total number of nodes participating in the consensus process, which is the PBFT consensus bound. PBFT generates a large amount of messages to reach a consensus, which can take a long time, where the time consumption will increase exponentially as the number of participating nodes increase.

2) Shard Clustering & Two Step Consensus Architecture

All blockchain validators should be clustered into different shard groups. For this technique, a shard clustering architecture is adopted to allocate a shard number to every validator in a decentralized way (i.e., without a central controlling entity) [16]. After the shard distribution, each transaction is evenly distributed to the shards and processed through a two-step consensus process, consisting of an intra shard consensus and a final consensus stage.

After finishing the intra shard consensus, the final consensus for validation of each local block is performed. All consensus processes are conducted using PBFT while a simple PoW is used first only in the shard configuration stage. If there is no faultiness in the local block, all local blocks are merged and distributed to all blockchain nodes. Details of each process are described as follows.

2-1) Identity Setup and Shard Clustering

Identity setup is based on [16], the first shard-based blockchain system. Nodes participating in the network form their ID by solving a simple PoW as

$$ID = H(EpochRand \parallel IP \parallel PK \parallel Nonce) < D \quad (1)$$

where *EpochRand* represents a random seed for the PoW generated, whenever a shard is reconstructed. Public key (*PK*), *IP*, and *Nonce* are additionally used to compute the node *ID*. *H* is the hash operator, *D* is the difficulty level of the PoW algorithm. The shard number assignment is determined by the last *l*bits of the ID. For example, if *l* is 3, the last 3 bits represent the shard number, enabling up to 2^3 shard configurations. Each node knows its own shard number after the ID computation, and nodes of the same shard should set up a point-to-point connection, as shown in Fig. 3. To share shard information, a Directory Committee (DC) is first selected (which consists of *C* nodes), and then all non-DC nodes need to report their ID information to the DC. The shard-based blockchain consists of *K* shards and a single DC group. Therefore, there are a total *K*+1 groups among a total of *N* validators. The DC is formed with *C* nodes that solves the PoW first (i.e., $\lfloor \frac{N}{k+1} \rfloor$) in order to keep the number of members of all the other *K*+1 groups similar [16]. The non-DC nodes (which solved the PoW) transmit their ID information (i.e., *PK*, *EpochRand*, *nonce*, and *ID*) to the DC. After the DC member collects IDs from all other nodes, the list is broadcasted throughout the network.

2-2) Intra Shard Consensus

After the shard configuration, each node receives its own transaction pool. The transaction division is decided by account-based sharding, which means that the transaction allocation is assigned to the shard that matches the last *l*bit hash value of the sender account. This is to prevent the double spending of malicious users generating transactions that exceed their balance and assign them to different shards. Each shard processes transactions independently, creating local blocks and performs a local PBFT consensus process. The structure of the local block is shown in Fig. 4. The block header of a local block contains the block producer's ID, shard index, timestamp, data size, and local transaction hash. After the local block is determined by the consensus, it is retransmitted back to the DC for a final consensus.

2-3) Final Consensus

In the final consensus process, the DC node receives each local block from each shard. By the intra shard consensus process, all DC nodes receive a set of local blocks from each shard. The primary node of the DC generates the final block by merging all local blocks in ascending order of their shard number, and then proceeds to the final consensus based on

PBFT. The final block includes a block producer (primary in final consensus), time stamp, data size, Tx hash, and consensus trust. Consensus trust is a value calculated from the difference of consensus opinions among nodes that occurred during the PBFT of each shard, and is used to bound the action space in the DRL process to satisfy the security constraints. The DC broadcasts the final block throughout the network and repeats the consensus procedure.

3) System Model Assumption

In the proposed DQNSB system, it is assumed that there are *N* validators and *k* shards. After shard clustering, a node is denoted as $n_{i,j}$, which represents the jth node in the ith shard. In each shard, the block producer is designated as the primary to proceed the PBFT consensus. Several additional assumptions are made in the consensus process.

- In order to limit the unresponsiveness waiting time of the message exchange, a timeout (ζ) bound of the maximum waiting time is configured. In addition, message tasks are processed in a round-robin fashion [33].
- After the shard clustering process has completed, the block consensus is performed during a predefined period R, which is set to 1 epoch. The shard is reshuffled every epoch to prevent a validator from participating in a particular shard for a long period of time.
- The block consensus process involves a message exchange and verification process. For the message exchange process, the data transmission rate $R_{i,j}(t)$ is applied based on a finite-state Markov channel model, as in [19], where $R_{i,j}(t)$ denotes the data transmission rate between validators i and j, which is quantized into H levels with $\boldsymbol{\mathcal{R}} = \{\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_H\}$. The state transition probability matrix is given as $[p_{\mathcal{R}}(t)]_{H\times H}$, where $p_{\mathcal{R}}(t)$=Pr$[R_{i,j}(t+1) = \mathcal{R}_b \mid R_{i,j}(t) = \mathcal{R}_a]$, $(\mathcal{R}_a, \mathcal{R}_b \in \boldsymbol{\mathcal{R}})$.

The message verification process includes verifying signatures and message authentication code (MAC) operations (in order to generate and verify the MAC), using θ and α CPU cycles, respectively [33].

IV. Performance Analysis

The scalability performance of a blockchain is an indicator that can be used determine if a sufficient level of TPS can be achieved even as the network scale grows. On the other hand, increasing the TPS by increasing the number of shards K can decrease the security level of the shard system due to the possibility of a single shard takeover event [20]. The time latency incurred during the message exchange and verification procedures during the consensus process also needs to be considered. These performance parameters have a trade-off relation with each other. This section presents an analysis of the performance parameters of shard based blockchain systems.

A. Scalability (Throughput)

Blockchain TPS refers to the number of transactions that the blockchain system can process per second. During the consensus process to verify a new block, the block producer generates a local block with a maximum B (bytes) size, for each block interval period T^I. If the average transaction size is b, the block header size is B_H and the number of shards is k, then the maximum TPS of the blockchain system can be obtained from (2).

$$\mathcal{T}(B, T^I) = \frac{k\lfloor (B - B_H)/b \rfloor}{T^I} \tag{2}$$

Based on (2), increasing the block size or number of shards can increase the TPS performance.

B. Latency (block finality time)

Latency is the time it takes for a transaction to enter the blockchain system and finally be processed and become irreversible. Transactions entering the network are automatically assigned to the shard via the sender's last $l$bit and verified through the intra-shard consensus and final consensus process. The transaction process is composed of two steps: *block interval* and *total consensus time* [23]. The total latency ($T_{latency}$) for a transaction can be obtained from

$$T_{latency} = T^I + T^k_{con} \tag{3}$$

where T^I and k are respectively the block interval and number of shards, and T^k_{con} is the total consensus time when there are K shards in the blockchain. In addition, the proposed shard-based blockchain consensus structure applied in the DQNSB scheme is shown in Fig. 5. The total consensus time consists of the following factors,

$$T^k_{con} = T^k_{intra} + T^k_{final} \tag{4}$$

where T^k_{intra} and T^k_{final} respectively represents the *intra shard consensus* and *final consensus* duration. Note that the consensus delay consists of *message propagation* and *message verification delay* (i.e., the time required to verify a signature and MAC operation). Therefore, the consensus delays can be computed as

$$T^k_{intra} = T^k_{in\_prop} + T^k_{in\_val} \tag{5}$$

$$T^k_{final} = T^k_{f\_prop} + T^k_{f\_val} \tag{6}$$

where $T^k_{in\_prop}$ and $T^k_{in\_val}$ are respectively the *message propagation* and *validation delay* of the intra shard consensus process based on a blockchain with k shards, while $T^k_{f\_prop}$ and $T^k_{f\_val}$ are the *propagation* and *validation delay* for the final consensus process. The propagation and validation delay of the message are considered in the following latency analysis.

1) Intra-Shard Consensus Delay (T^k_{intra})

At the start of the initial consensus in each shard, the primary node creates $\mathcal{M}$ (batch size) blocks and broadcasts the replicas to the other nodes [38]. The primary performs one MAC verification for each request while each replica has one MAC, and signature verification is processed for every block request.

a) Pre-prepare

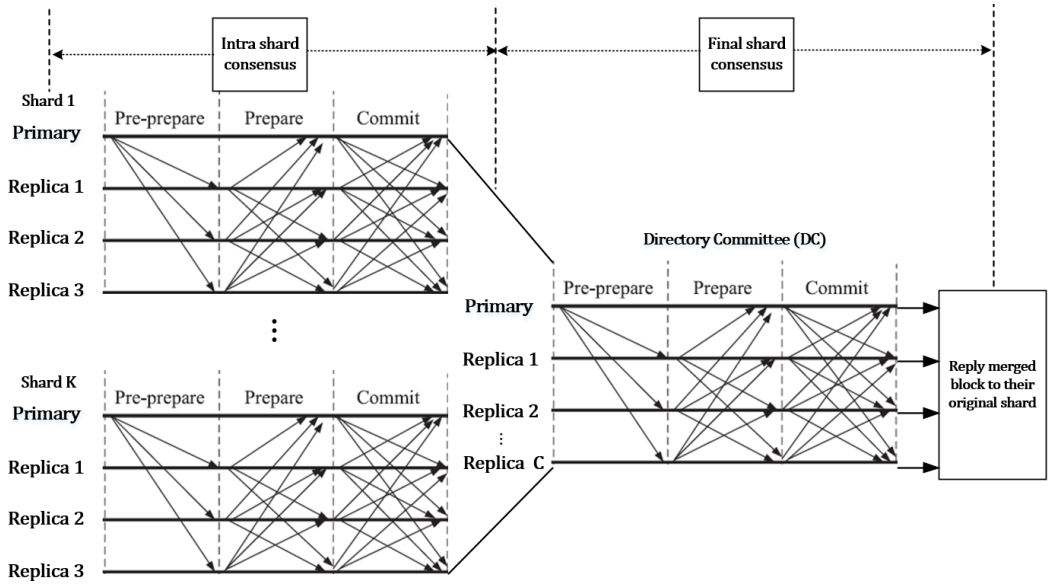

Fig. 5: Consensus structure of shard based blockchain.

The primary node of the ith shard collects a batch of $\mathcal{M}$ requests (including $\mathcal{M}$ signatures) and sends a single pre-prepare message to the nodes. Then, the primary node processes N_i-1 MAC generations and each node processes a single MAC operation to verify the blocks.

b) Prepare

In the prepare phase, each node exchanges messages with all other nodes to determine if the pre-prepare message from the primary is valid. At this time, each node generates N_i-1 MACs and validates N_i-2 MACs.

c) Commit

In the commit phase, all nodes exchange messages for block verification. The primary and node processes N_i-1 messages each during message transmission and reception, respectively.

At the end of the commit phase, the primary and replica node reply their intra shard consensus to the DC for the final consensus. At this time, the primary and replica node create C MACs for each request, which are the number of DC members. In summary, the primary node performs a total of $\mathcal{M}$ signature checks and $\mathcal{M}(1+C)+4(N_i-1)$ MAC operations. The replica also processes $\mathcal{M}$ signatures and $C\mathcal{M}+4(N_i-1)$ MAC operations. Then, the processing time of the primary of the ith intra shard consensus is given as $T^i_{in\_primary}$=$\frac{\mathcal{M}\theta+[\mathcal{M}(1+C)+4(N_i-1)]\alpha}{c_{i,p}}$, where $c_{i,p}$ represents the computational speed of the primary node of the ith shard. The processing time of the replica of the ith intra shard consensus requires a processing time of $T^i_{in\_replica}=\frac{\mathcal{M}\theta+[C\mathcal{M}+4(N_i-1)]\alpha)}{c_{i,r}}$, where $c_{i,r}$ is the computational speed of the replica node in the ith shard. Note that the intra shard consensus is processed in parallel, and the latency is determined by the shard that has the largest delay. In addition, the validation process of the primary and replicas are performed in parallel. Therefore, the validation time of each request message in the intra shard consensus ($T^k_{in\_val}$) can be expressed as follows.

$$T^k_{in\_val}=\frac{1}{\mathcal{M}}\max_{i=1,\cdots,k}(T^i_{in\_replica},T^i_{in\_primary}) \tag{7}$$

Next, the message propagation delay is the time it takes for a message to reach the destination node during the consensus steps (which include pre-prepare, prepare, and commit). Note that a timeout ζ is set in each consensus step to prevent unresponsive nodes from delaying the consensus process too excessively. Replicas that do not respond within the timeout ζ are considered to have a *reject* opinion of the corresponding consensus step. Then, the intra-propagation delay of the request for each intra shard consensus step is computed as follows

$$\begin{aligned}T^k_{in\_prop}&=\frac{1}{\mathcal{M}}(T^k_{in\_preprepare}+T^k_{in\_prepare}+T^k_{in\_commit})\\&=\frac{1}{\mathcal{M}}\max_{i=1,\cdots,k}(\min\left\{\max_{j\neq p}\frac{\mathcal{M}B}{R_{n_{i,p},n_{i,j}}},\zeta\right\}\\&\quad+\min\left\{\max_{j\neq l}\frac{\mathcal{M}B}{R_{n_{i,j},n_{i,l}}},\zeta\right\}\\&\quad+\min\left\{\max_{j\neq l}\frac{\mathcal{M}B}{R_{n_{i,j},n_{i,l}}},\zeta\right\})\end{aligned} \tag{8}$$

where $n_{i,p}$ and $n_{i,j}$ represents the node index of the primary and jth replica of shard i, respectively.

2) Final Consensus Delay (T^k_{final})

The blocks agreed in the k intra shard consensus are delivered to the DC group for final consensus. The DC group validates $k\mathcal{M}$ signatures and validates the $k\mathcal{M}$ MAC of the blocks received from each shard. The DC nodes (that have C members) perform the PBFT consensus again and then return the merged block to all other nodes. Then, the validation time of the primary and replicas in the DC group can be expressed as

$$\begin{aligned}T^k_{f\_primary}&=\frac{k\mathcal{M}\theta+[k\mathcal{M}+4(C-1)+(N-C)\mathcal{M}]\alpha}{c_{f,p}}\\T^k_{f\_replica}&=\frac{k\mathcal{M}\theta+[4(C-1)+(N-C)\mathcal{M}]\alpha}{c_{f,r}}\end{aligned} \tag{9}$$

where $c_{f,p}$ and $c_{f,r}$ represent the computational speed of the primary and replica nodes in the final shard (DC group) process. Therefore, the validation time for each request in the final consensus can be obtained from (10).

$$T^k_{f\_val} = \frac{1}{\mathcal{M}} \max(T^k_{f\_primary}, T^k_{f\_replica}) \tag{10}$$

Next, the message propagation delay of the final shard is computed the same way the propagation delay of the intra-shard consensus was processed. Finally, the propagation delay for final consensus is given as

$$\begin{aligned} T_{f\_prop^k} &= \frac{1}{\mathcal{M}}(T^k_{f\_request} + T^k_{f\_preprepare} + T^k_{f\_prepare} \\ &\quad + T^k_{f\_commit} + T^k_{reply}) \\ &= \frac{1}{\mathcal{M}}(\min\left\{\max_{i=1,\cdots,k;j=1,\cdots,N_i;l=1,\cdots,C} \frac{\mathcal{M}B}{R_{n_{i,j},n_{f,l}}}, \zeta\right\} + \\ &\quad \min\left\{\max_{l\neq p} \frac{\mathcal{M}B}{R_{n_{f,p},n_{f,l}}}, \zeta\right\} + \\ &\quad \min\left\{\max_{u\neq p;u,l=1,\cdots,C} \frac{\mathcal{M}B}{R_{n_{f,u},n_{f,l}}}, \zeta\right\} + \\ &\quad \min\left\{\max_{u\neq l} \frac{\mathcal{M}B}{R_{n_{f,u},n_{f,l}}}, \zeta\right\} + \\ &\quad \min\left\{\max_{i=1,\cdots,k} \frac{k\mathcal{M}B}{R_{n_{f,u},n_{i,j}}}, \zeta\right\}) \end{aligned} \tag{11}$$

where $n_{f,p}$ and $n_{f,u}$ represents the node index of the primary and replica of the final shard, $T^k_{f\_request}$ is the propagation delay for the final shard to get the local block from the intra shard, and T^k_{reply} is the time consumed to have the final merged block broadcasted throughout the network. Finally, the total consensus time of the sharded blockchain system can be expressed in the following form.

$$\begin{aligned} T^k_{con} &= T^k_{intra} + T^k_{final} = (T^k_{in\_prop} + T^k_{in\_val}) \\ &\quad + (T^k_{f\_prop} + T^k_{f\_val}) \\ &= \frac{1}{\mathcal{M}}(\max_{i=1,\cdots,k}(T^i_{in\_replica}, T^i_{in\_primary}) \\ &\quad + \max(T^k_{f\_primary}, T^k_{f\_replica})) \\ &\quad + \frac{1}{\mathcal{M}} \max_{i=1,\cdots,k}(\min\left\{\max_{j\neq p} \frac{\mathcal{M}B}{R_{n_{i,p},n_{i,j}}}, \zeta\right\} \\ &\quad + \min\left\{\max_{j\neq l} \frac{\mathcal{M}B}{R_{n_{i,j},n_{i,l}}}, \zeta\right\} \\ &\quad + \min\left\{\max_{j\neq l} \frac{\mathcal{M}B}{R_{n_{i,j},n_{i,l}}}, \zeta\right\}) \\ &\quad + \frac{1}{\mathcal{M}}(\min\left\{\max_{i=1,\cdots,k;j=1,\cdots,N_i;l=1,\cdots,C} \frac{\mathcal{M}B}{R_{n_{i,j},n_{f,l}}}, \zeta\right\} \\ &\quad + \min\left\{\max_{l\neq p} \frac{\mathcal{M}B}{R_{n_{f,p},n_{f,l}}}, \zeta\right\} \\ &\quad + \min\left\{\max_{u\neq p;u,l=1,\cdots,C} \frac{\mathcal{M}B}{R_{n_{f,u},n_{f,l}}}, \zeta\right\} \\ &\quad + \min\left\{\max_{u\neq l} \frac{\mathcal{M}B}{R_{n_{f,u},n_{f,l}}}, \zeta\right\} \\ &\quad + \min\left\{\max_{i=1,\cdots,k} \frac{k\mathcal{M}B}{R_{n_{f,u},n_{i,j}}}, \zeta\right\}) \end{aligned} \tag{12}$$

The transaction consensus latency should be finished within a number of consecutive block intervals (u) to satisfy the finality property of the blockchain [23]. Summing all the time factors up, the constraint on the total latency can be described as in (13).

$$T_{latency} = T^I + T^k_{con} \leq uT^I, k = 1, 2, \cdots, K \tag{13}$$

C. Security Analysis

Security constraints differ depending on the type of consensus. Because the blockchain consensus algorithms adopt a PoW or PoS approach, if a group of nodes obtains more than 51% of the hash power or stake, then that group could have its nodes collude to take control of the blockchain consensus decisions [34]. In addition, these competition-based consensus methods do not guarantee the finality of the consensus and may cause forks in the blockchain. Forking may be needed when multiple chaining block options have to be (temporarily) kept valid until a consensus can be reached on what block to officially chain to the blockchain is decided. PoW based consensus methods select the longer chain to preserve the liveness of the blockchain rather than the finality [35]. On the other hand, the PBFT based algorithm uses the individual voting rights of each replica to reduce individual centralization. The PBFT consensus scheme can have up to f malicious nodes among N nodes based on the relation that $(3f + 1) \leq N$ is satisfied in order to guarantee safeness of the consensus process [15]. The proposed DQNSB scheme is a shard based blockchain that applies a parallel PBFT consensus process in each shard. The nodes are evenly divided into k shards,

and the DC that is in charge of the shard configuration and final consensus consists of $C = \lfloor \frac{N}{k+1} \rfloor$ members, then a total of $k+1$ groups are formed among the N validator nodes. In this case, the PBFT security bound can be defined based on each shard and DC group. Note that all k shards are randomly distributed by the characteristics of the one-way hash output, and C of the DC members aim to divide N nodes into $k+1$ shards evenly. In this case, the following lemmas are established for the shard based PBFT consensus algorithm.

Lemma 1. *The number of shards k is bounded by $k < \frac{N(1-3p)-1}{3Np+1}$ to guarantee a honest consensus for all shards.*□

Proof. For a blockchain based on a total of N nodes, where the probability of a malicious node is p and the number of shards is k, if the number of total nodes and faulty nodes in shard i are given as N_i and $f_i (i = 1, \cdots, k)$, respectively, then each shard should satisfy $3f_i + 1 \leq N_i$ to satisfy the PBFT security bound. In addition, when the number of malicious nodes in the DC group is f_{DC}, then the $3f_{DC} + 1 \leq C$ constraint is required to satisfy the security bound. When the network has Np malicious nodes in the network, the worst case would occur when all malicious nodes are completely grouped in the DC. In this case, each shard and DC group require $3Np+1 \leq N_i$ and $3Np+1 \leq C$ to satisfy the security bound. Since C is given as $\lfloor \frac{N}{(k+1)} \rfloor$ and $N_i \geq \lfloor \frac{(N-C)}{k} \rfloor$, therefore, $C \leq N_i$. Finally, $3Np + 1 \leq C = \lfloor \frac{N}{(k+1)} \rfloor$ becomes the final security constraint. This implies $3Np+1 \leq \lfloor \frac{N}{(k+1)} \rfloor < \frac{N}{(k+1)}$ and when the number of nodes in a shard is k, the requirement to satisfy the security bound for a safe consensus becomes $k < \frac{N(1-3p)-1}{3Np+1}$. ■

Lemma 2. *The number of shards k is bounded by $k < \frac{2N}{3(Np+1)} - 1$ to prevent a malicious block from being chained.*□

Proof. In the PBFT consensus process, when one-third of the nodes in a shard are maliciously colluding, they can reject a honest block. In addition, when two-thirds of the nodes in a shard are maliciously colluding, they can approve a faulty block such that it is chained to the blockchain. So, the quorum of nodes ratio required for a proper PBFT consensus is two-thirds of the nodes. When the total number of nodes is N, the probability of a malicious node is p, and the number of shards is k, based on the PBFT consensus quorum, $Np \leq \frac{2}{3}N_i - 1$ and $Np \leq \frac{2}{3}C - 1$ are required to satisfy the security bound. Based on lemma 1, $C = \lfloor \frac{N}{(k+1)} \rfloor$, $N_i \geq \lfloor \frac{N-C}{k} \rfloor$, and $C \leq N_i$. This leads to $Np \leq \frac{2}{3}C - 1 = \frac{2}{3}\lfloor \frac{N}{(k+1)} \rfloor - 1 < \frac{2N}{3(k+1)} - 1$. Therefore, when the number of nodes in a shard is k, the requirement to satisfy the security bound for a safe consensus becomes $k < \frac{2N}{3(Np+1)} - 1$. ■

Recall that the worst case of the shard distribution is when all malicious nodes in the network are members of the DC or form a majority in a shard. In this case, the number of faulty nodes f need to be less than one-third of the number of nodes in each shard to ensure that safe agreements are reached. Let each security constraints (i.e., $k < \frac{N(1-3p)-1}{3Np+1}$ and $k < \frac{2N}{3(Np+1)} - 1$ respectively from lemmas 1 and 2) be assigned as constraints $S_1 = \frac{N(1-3p)-1}{3Np+1}$ and $S_2 = \frac{2N}{3(Np+1)} - 1$.

TABLE I: Security performance metric for blockchain consensus

		Consensus Result	
		Valid(Accept)	**Invalid(Reject)**
Ground truth of the validator	**Valid**	True Negative (TN)	False Positive (FP)
	Invalid	False Negative (FN)	True Positive (TP)

S_1 is a stricter constraint than S_2, because it prevents malicious nodes from acquiring one-third of a single shard. In this case, even if a malicious node becomes a block producer, no false consensus occurs by the honest majority, and the consensus of honest blocks by a honest block producer is also not hindered. S_2 is a constraint that prevents malicious nodes from occupying two-thirds of each shard, which prevents malicious blocks from being chained to the blockchain. In Table 1, the security performance metrics used in the consensus evaluations are presented, where the definition of the parameters are defined below.

Consensus Success Probability (CSP): CSP represents the probability (i.e., TN) of a honest block that is a valid block from a honest block producer that is approved (i.e., agreed upon via majority vote) through the consensus process. Therefore, CSP is an indicator of how dominant the honest nodes are in the consensus process, therefore, CSP is proportional to the throughput performance.

False Consensus Probability (FCP): FCP represents the probability of a malicious shard clustering. A *malicious shard* is a shard that has lost its capability to reliably make a honest consensus, which is due to malicious nodes interfering with the creation of a honest block (FP), or the chaining of a faulty block is decided by the malicious nodes that are a majority in the shard (FN). For example, if a block created by a malicious block producer has been accepted by consensus, it is counted as FN. If the honset block generated by the honest block producer is rejected by consensus, it is counted as FP. The FCP is an indicator of how dominant the malicious nodes are in the consensus decision. Finally, the consensus security level of a blockchain can be expressed by the malicious shard (FN+FP) ratio compared to all cases (TN+TP+FP+FN). In each episode, K+1 consensus occurs (K intra shard consensus and one final shard consensus), and FCP is computed by counting cases corresponding to FN and FP, given as $FCP = \frac{FN+FP}{TN+TP+FN+FP}$.

V. Trust Computation on Consensus in Shard Based Blockchain Systems

In this section, the trust computation method applied to the consensus process is described. The worst case occurs when the malicious nodes are a majority and collude to overtake consensus decisions. Such an event results in honest blocks being rejected, and faulty blocks obtain consensus and are chained to the blockchain. Therefore, if the block validator acts maliciously during the consensus process, the consensus opinions will be inconsistent (i.e., there will be many mixed

votes to $accept$ and $reject$ the new block). This inconsistency in the local consensus in each shard will be recorded, where the individual and aggregated inconsistency level of the shards and aggregated inconsistency level of the blockchain will be used in the trust level evaluations. The total consensus trust evaluation is used to estimate the malicious ratio ($\bar{p}$) of nodes in the network.

The blockchain network is assumed to have an average of Np malicious nodes, where p is the probability of a node being malicious. In the proposed DQNSB scheme, the DRL agent of the network does not know the probability p value or which nodes are malicious. The DRL agent conducts a faulty probability estimation based on the consensus history, and based on this estimation, the number of shards $\dot{K}$ is derived based on lemmas 1 and 2.

A. Adversary model

The types of malicious node attacks that can harm the blockchain network are described in [24]. In this paper, the Naive Malicious Attack (NMA) model is applied. In the NMA model, malicious nodes always behave opposite to the honest nodes. The NMA node will vote to reject a commit result made by a honest block and accept the commit result of a faulty block generated by a malicious node. If the malicious node is selected to be the block producer, it will make a fake block containing false information, and attempt to have this block be approved from the consensus process, and be chained to the blockchain.

B. Network Trust and Faulty Probability Estimation

In PBFT, more than two-thirds of the node agreements are needed to reach a consensus on a new block. In other words, more than one-third of malicious nodes can collude to negate an ongoing block's consensus. Every shard is allowed a maximum of one-third of its nodes to be malicious nodes and still make correct consensus decisions. All replica nodes can check the consensus results of other replica nodes through PBFT procedures of each shard. Each replica node is evaluated and a decision is made by the shard nodes whether to accept or reject the proposed block [15]. The commit results are all exchanged during the PBFT process, and each message is authenticated by the MAC and signature, so it is assumed that the non-repudiation property is maintained.

The consensus inconsistency is computed using a normalized entropy value, which is a measure of uncertainty of the different probability of the consensus status. For example, if the consensus for block validation (i.e., consensus to decide if the block is valid or invalid) results in a divided decision with an equal number of valid and invalid votes, then the normalized entropy value will be 1. On the other hand, if a consensus results in an unanimous vote (of either all valid or all invalid), the normalized entropy value will be 0. This inconsistency value is computed first at each local shard. Then, network trust is computed by averaging the normalized entropy values of all shards. Let p_m^i and p_M^i respectively be the total ratio of minor and major consensus opinions in the ith shard's consensus. Then, p_M^i is given as $(1-p_m^i)$, and the parameters that represent the level of consensus inconsistency in the ith shard's consensus using the normalized entropy can be defined as in (14).

$$\begin{aligned} I_i &= -p_m^i log_2(p_m^i) - (1-p_m^i) log_2(1-p_m^i) \\ I_{DC} &= -p_m^{DC} log_2(p_m^{DC}) - (1-p_m^{DC}) log_2(1-p_m^{DC}) \end{aligned} \quad (14)$$

In (14), I_i and I_{DC} respectively denote the inconsistency of the consensus in the ith shard and DC, and p_m^{DC} indicates the ratio of the minor consensus opinion in the DC. The shards conduct the consensus process during the intra-shard consensus period, while the DC group conducts its consensus during the final shard consensus stage. The consensus of all shards and DC should be honest, therefore the inconsistency of all groups where consensus occurs is computed. The consensus inconsistency of each group was computed in the form of entropy [40]. Through the entropy calculation, if a honest consensus has been reached (unanimous voting for a generated block), the entropy value will be zero, otherwise the entropy value will be in the range of (0,1] (if accept and reject votes for a generated block is divided into half, then the entropy value will be 1).

Then, the total consensus trust U is obtained by averaging the normalized entropy value of each shard.

$$U = \frac{1}{k+1}((\sum_{i=1}^{k} I_i) + I_{DC}) \quad (15)$$

In (15), U serves as an indicator of the lack of trust of the overall consensus process of the blockchain.

The proposed DQNSB scheme attempts to estimate the proportion of malicious nodes in the entire network, based on the consensus result of each shard. However, only the minor and major ratio of each shard's consensus opinion can be known, because there is no exact information on which node is malicious or honest. Therefore, the overall malicious ratio is estimated based on the following assumptions. First, the average of the inconsistency of each shard is approximately equal in level of inconsistency in terms of the overall network. In this case, if the ratio of faulty nodes in the network is $\bar{p}$, an estimate of ($\bar{p}$) can be obtained using the normalized entropy.

$$\begin{aligned} U &\approx -\dot{p} log_2 \dot{p} - (1-\dot{p}) log_2 (1-\dot{p}) \\ \bar{p} &= \min\{\dot{p}, (1-\dot{p})\} \end{aligned} \quad (16)$$

This estimation assumes that the honest nodes are a majority of the total network, and therefore, the lower value among $\dot{p}$ and $(1-\dot{p})$ is the estimated malicious node probability $\dot{p}$. Based on the estimation of $\dot{p}$, the value $\dot{K}$, which is the largest positive integer that satisfies $\dot{K} < \frac{N(1-3\bar{p})-1}{3N\bar{p}+1}$, can be obtained. Based on $\dot{K}$, the set of possible numbers of shards the blockchain may use can be defined as $K^* = \left\{1, 2, \ldots k \ldots, \dot{K}\right\}$. If the actual malicious node probability is p, there may be a difference between p and the estimated $\bar{p}$ value. Based on the inequality relation of the malicious node probability p derived in lemma 2, the boundary condition of $\bar{p}$ can be obtained as in lemma 3.

Lemma 3. *The estimated malicious node probability* $\bar{p}$ *is bounded by* $\frac{1}{k+1}\left\{(\sum_{i=1}^{k}\frac{p_m^i}{p_m^i+p_M^i})+\frac{p_m^{DC}}{p_m^{DC}+p_M^{DC}})\right\} \leq \bar{p} \leq \frac{1}{3}$.□

Proof. If there are no malicious nodes in the blockchain network, the consensus decision would be an unanimous agreement. If an unanimous agreement is not reached, the next best case would be when each shard reaches a majority consensus based on the opinions of the honest nodes. In other words, when considering the entire blockchain consensus process, the best result is obtained when malicious nodes can only influence a minority of the opinions in each of the k shards and the final shard, which is the DC group. Since $(\sum_{i=1}^{k}\frac{p_m^i}{p_m^i+p_M^i})+\frac{p_m^{DC}}{p_m^{DC}+p_M^{DC}}$ denotes the ratio of minor opinions of the entire consensus process, the estimated malicious node probability $(\bar{p})$ lower bound can be derived from the average of the minority of the opinions in each of the k shards and the final shard. On the other hand, if the ratio of malicious nodes become 1/3 of the number of nodes in the blockchain network, it can be regarded as the worst case, as it exceeds the PBFT consensus bound. ■

VI. DRL BASED SECURE SHARD BLOCKCHAIN PERFORMANCE OPTIMIZATION

To take into account the dynamic state of the IoT environment, the proposed DQNSB scheme uses a DRL agent that applies DNN based Q-learning to perform complicated function approximation more effectively, which is known as DQN technology [36]. DQN has the ability to approximate the value function accurately while addressing a large volume of state space problems, which can be applied to the learning process of multiple blockchain parameters and shard formation scenarios.

To obtain a long-term reward (e.g., higher and stable TPS throughput), optimal blockchain parameter selection is conducted considering the latency and security constraints. The security constraints are considered to prevent consensus failures that can occur during consensus in shard based blockchains. The proposed DQNSB scheme partially limits the actions of DRL by estimating the number of malicious nodes based on the consistency of previous consensus history records.

A. State Space

The discrete time epoch t is the basis for each decision of the DRL process, $\boldsymbol{R} = \{R_{i,j}\}$ is the data transmission rate of the link between node i and j, c is the computing capability of the blockchain nodes, and H is the consensus history. Then, based on epoch t, the state space S^t can be denoted as

$$S^t = [R, C, H, \bar{p}]^t \tag{17}$$

where the estimated malicious node probability $\bar{p}$ is computed using the previous consensus history H. The consensus history $H = \{H_i\}$ of node i is based on binary encoding, where, H_i is set to 1 if the consensus opinion of node i is valid, and it is set to 0 when it is invalid. Using the computed H, the consensus trust is derived based on the normalized entropy of the consensus opinion ratio. Finally, $\bar{p}$ is computed based on the network trust level, where the range of the number of shards to ensure secure sharding is obtained.

B. Action Space

The proposed DQNSB blockchain performance optimization framework aims to maximize the long term TPS. The DRL agent selects the block size B, block interval T^I, and number of bounded shards K^* for each epoch t. Then, the action space (A^t) of epoch t can be represented as

$$A^t = \left[B, T^I, K^*\right]^t \tag{18}$$

where the block size space $B \in \left\{1, 2, \cdots, \dot{B}\right\}$ and block interval $T^I \in \left\{1, 2, \cdots, \dot{T}^I\right\}$ has a maximum block size limit $\dot{B}$ and maximum block interval $\dot{T}^I$. Based on the shard space numbering set $K^* \in \left\{1, 2, \cdots, \dot{K}\right\}$, where $\dot{K}$ is the largest integer satisfying the security constraint S_l (l=1,2). In addition, $\dot{K}$ is bounded by the maximum allowable shard number ($\bar{K}$) based on $\dot{K} \leq \bar{K}$.

C. Reward Function

The reward function is designed to maximize the blockchain system throughput (TPS) while satisfying the latency and security constraints through DRL adaptation. The objective function and constraint can be summarized as follows

$$\begin{aligned} &Objective\text{: } \max_{A} Q(S, A) \\ &Constraint\,1\text{: } T_{latency} = T^I + T_{con}^k \leq uT^I \\ &Constraint\,2\text{: } \dot{K} < S_l, l = 1, 2 \end{aligned} \tag{19}$$

where $Q(S, A)$ is the action-value function of the DQN [36]. The intermediate reward function $\mathfrak{R}^t$ at epoch t can be computed as $\mathfrak{R}^t = \mathfrak{R}(S^t, A^t) = \frac{k\lfloor(B-B_H)/b\rfloor}{T^I}$ if constraints 1 and 2 are satisfied, otherwise it will be zero. The action-value function of the DRL agent is set to maximize the long term aggregated rewards. Therefore, an optimal action-value function $Q^*(S, A)$ can be computed as $Q^*(S, A) = \max_\pi \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t\mathfrak{R}(S^t, A^t) \mid S^{(0)} = S, A^{(0)} = A, \pi]$, where γ is the discount factor ($\gamma \in (0, 1]$) and π denotes the behavior policy.

D. Workflow of DQNSB

In Fig. 6, the workflow of DQNSB consists of two environments: Background blockchain environment (*BCenv*) and the DQN Agent. Through interaction between *BCenv* and the DQN agent, optimal parameter selection is performed while maintaining the security level adaptably.

In the background blockchain environment, the shard formation process begins. Initially, malicious nodes are selected according to the ratio *p*, and are randomly distributed according to the shard formation rule (section 3-B). During the first epoch, *BCenv* cannot receive any learned action from the agent, therefore it performs arbitrary action and proceeds with the consensus process. At this time, malicious nodes perform a consensus attack (based on the NMA scenario).

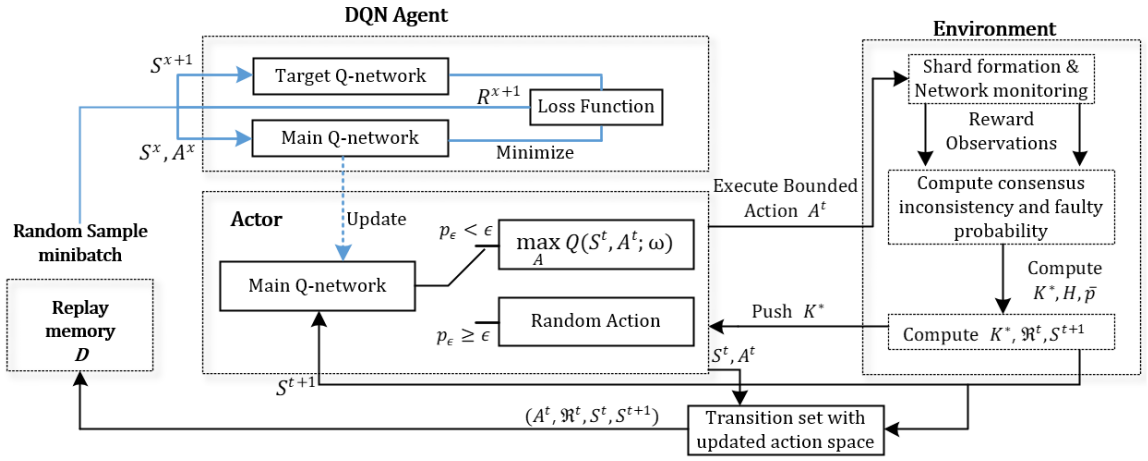

Fig. 6: Interaction of the DQN agent and network of the shard based blockchain system.

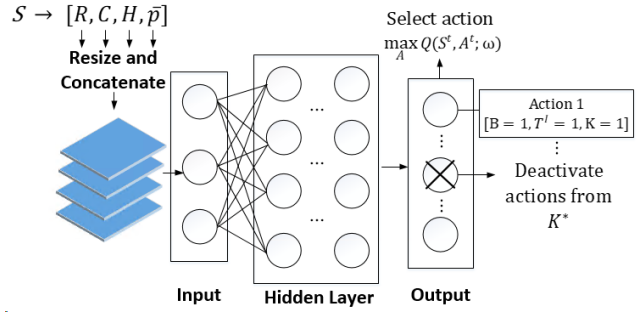

Fig. 7: DNN architecture of DQNSB

Once the consensus process is complete, *BCenv* can collect reward information, consensus history H, and the nodes faulty probability $\bar{p}$ through trust computations. Finally, the system will compute the next state (S^{t+1}), and *BCenv* push K^* to the DQN agent.

The DQN agent uses the conventional DNN architecture, where the input of the DNN are the state parameters of S^t and the output is the probability of each action space A^t at each epoch t [29]. The actor network executes actions by exploration or exploitation by probability p_ϵ. The exploitation process finds the optimal-value function $Q^*(S, A)$ based on the Bellman equation. To improve the stability of reinforcement learning, DQN adopts the following two improvements, which include 1) experience replay and 2) separate target network [29].

1) In every time step, the DQN agent stores the transition set $(A^t, \mathfrak{R}^t, S^t, S^{t+1})$ in its replay memory D, and randomly extracts a minibatch to train the DNN. Through this method, the strong correlation between the samples is alleviated which also reduces the variance of the update.

2) The target networks are cloned by the main network in every $\mathbb{C}$ step, minimizing the loss using the action-values of the target networks. This approach prevents too frequent updates, and reduces the divergence and oscillation of the training.

The training process of DNN is described in Fig. 7. To fit the size of the DNN input, the state components $R, C, H, \bar{p}$ are resized and concatenated [23]. The hidden layers are composed of Rectified linear units (ReLUs) by using the fuction $f(x) = max(0, x)$, and a batch normalization process is performed at the end of each layer. The output node returns the probability for each action when the state is entered. At this time, the action that does not satisfy the K^* obtained by *BCenv* is deactivated. Through this, unnecessary exploration to enable fast convergence is reduced. The bounded action that satisfies K^* and the maximum Q-value is determined, and this action is finally propagated through out the blockchain environment. The main networks are trained in each step to minimize the loss function $L(w)$, which is given as

$$L(w) = E[(\mathfrak{R}^x + \gamma \max_{A'} Q^*(S^{x+1}, A'; \omega^*) - Q(S^x, A^x; \omega))^2] \tag{20}$$

where both ω and ω^* are the weights of the DNN in the main network and the target networks, respectively. During the training process, the target network is periodically updated, and all the actions selected from the Q-values are bounded action from the constraints of the load bounded shard space K^*. The training process of the overall workflow is shown in Algorithm 1.

VII. Simulation Results

The simulation environment is based on 500 IoT devices and K block producers of each shard among total 200 validators are elected randomly for every epoch. The DRL framework used to optimize the proposed DQNSB blockchain system was implemented with PyTorch, which is a fast and concise deep learning framework [37]. The parameters used in the simulation are summarized in Table 2. The performance was analyzed in terms of TPS, which represents the throughput of the blockchain under the restrictions of the security constraints. For every episode, shards are redistributed and a consensus process is conducted for each shard. Among 200 validators, probability p is assigned as a malicious ratio, and NMA attacks were performed in the consensus process. *BCenv* checks the inconsistency of the consensus of each episode and forms state information S. Then, the agent trains the DNN of the DQN from each state using the replay memory as an input. In the DNN architecture, 3 hidden layers and a batch size of 256

Algorithm 1 DQNSB TPS Throughput Optimization

Initialization DQN parameters

Initialize replay memory D with the size of $\dot{D}$
Initialize action-value function Q with random weights ω (main Q network)
Initialize target action-value function Q^* with weights $\omega^* \leftarrow \omega$
Load initial state space data and use it as the input of the actor network

Deep Q-learning

for *each decision epoch t* **do**

Load bounded shard space K^* and push it to the Action Space (A^t)
If $p_\epsilon \geq \epsilon$, random action selection with probability ϵ
Otherwise select action $A^t = \arg\max_A Q(S^t, A^t; \omega)$
Execute action A^t and observe reward $\mathfrak{R}^t$ and proceed to next state S^{t+1}
Load $\bar{p}$ in S^{t+1} and update the bounded shard space K^*
Store transition $(S^t, A^t, \mathfrak{R}^t, S^{t+1})$ in replay memory D
Randomly sample minibatch of transitions $(S^x, A^x, \mathfrak{R}^x, S^{x+1})$ from memory D
Set $y^x =$

$$\begin{cases} \mathfrak{R}^x & \text{if episode terminates at step } x+1 \\ \mathfrak{R}^x + \gamma \max_{A'} Q^*(S^{x+1}, A'; \omega^*) & \text{otherwise} \end{cases}$$

to compute the target Q-value from the target Q network by y^x
Update target Q network by minimizaing the loss function $(y^x - Q(S^x, A^x; \omega))^2$ for every $\mathbb{C}$ step

end

TABLE II: Simulation Parameters

Symbol	Parameters	Value
N	Number of nodes	200
b	Average transaction size	200 Bytes
B_H	Block header size [8]	80 Bytes
$\dot{B}$	Maximum block size of each shard	8 MB
$\dot{T}^I$	Maximum block interval	16s
$\bar{K}$	Maximum allowable shard number	8
$R_{i,j}$	Data transmission rate between node i and j [23]	10-100 Mbps
c_i	Computing resource of node i [23]	10-30 GHz
θ	Computing cost for verifying the signature [33]	2 MHz
α	Computing cost to generate and verify the MAC [33]	1 MHz
u	Consecutive block interval to satisfy finality property [39]	6
$\mathcal{M}$	Batch size [38]	3
$\mathbb{C}$	Step interval for target network update	10

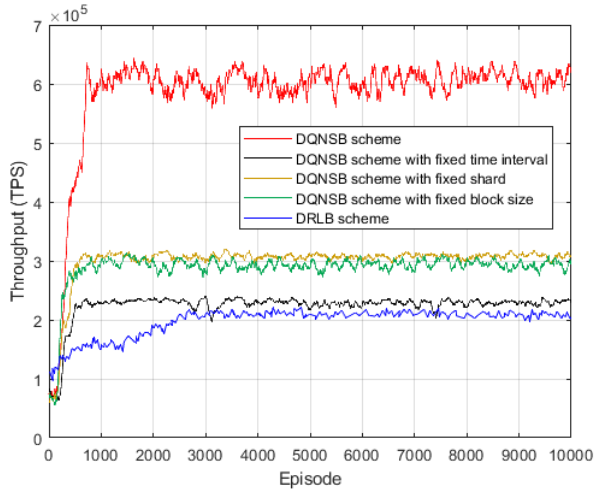

Fig. 8: TPS performance and convergence trend analysis.

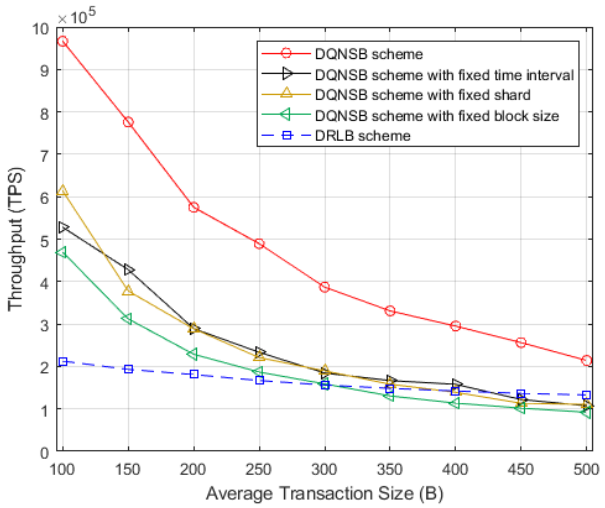

Fig. 9: TPS performance based on transaction size.

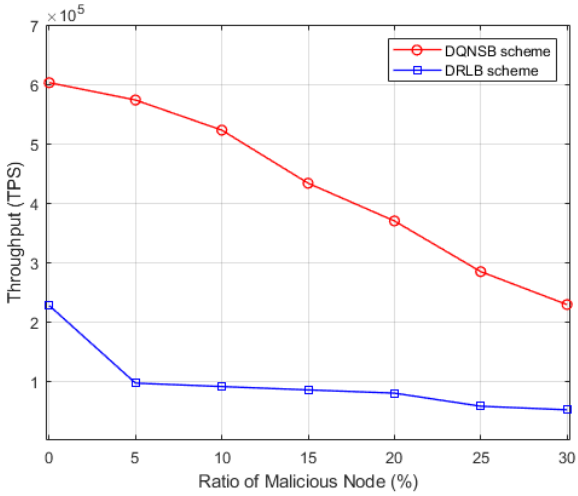

Fig. 10: TPS performance with malicious node injection.

was used. Target network updates were conducted every 10 episodes, and the RMSProp was used as the optimizer [42].

A. Throughput performance analysis

The performance of the proposed DQNSB scheme is compared to the optimized DRLB scheme of [23] in terms of throughput in this section. The performance analysis is based on the following four benchmarking schemes.

1) DQNSB with fixed time interval: The block producing interval is fixed to 4 seconds.
2) DQNSB with fixed shard: The number of shards for parallel processing is fixed to 4 shards.
3) DQNSB with fixed block size: The size of the block created in the shard is always the same 4 MB.
4) DRLB scheme: The DRL based performance optimized blockchain scheme of [23].

Figs. 8-10 represents the simulation results of the throughput performance (TPS) based on different blockchain network environments. Fig. 8 shows the throughput performance of the DQNSB scheme compared to the DRLB scheme of [23]. The simulation results show that the TPS is low at the beginning of the learning process, but gradually increases while the DQN agent finds the optimal blockchain parameters. The proposed DQNSB scheme results in a higher TPS in almost all intervals compared to the DRLB scheme, and the convergence speed is also much faster. The reason that the proposed DQNSB scheme has a fast convergence speed is due to the DNN architecture. DQNSB uses a deactivation module to turn off actions that violate security constraints, thereby preventing unnecessary exploration of output actions (to expedite the convergence speed) as shown in Fig. 7. This approach is similar with reference [41], which enables convergence in fewer episodes through guided actions. On the other hand, the DRLB scheme has the lowest convergence speed among the benchmarked schemes, because it has a large action space due to the block producer selection and do not use any approach to reduce the training time. The DQNSB scheme with a fixed shard and fixed block size shows a similar level of TPS performance. In the case of limiting the shard, the agent increases the block size or decrease the time interval to increase the TPS. In the case of limiting the block size, the agent increases the shard size or decrease the time interval to achieve a higher TPS. In the case of limiting the time interval, even if the block size and the number of shards are optimized, the block issuing time can become too long, which will restrict the throughput performance. In the case of DRLB, it performs DRL optimization without any scalable approach (e.g., sharding), thus showing the lowest TPS performance among all benchmarked schemes. Fig. 9 shows the TPS change as the average transaction size increases. As the average transaction size increases, the number of transactions contained in one block decreases. The fixed time interval, shard size, and block size cases of the DQNSB scheme result in a similar efficiency performance compared to the DRLB scheme, which is near the 500 bytes transaction size, because the TPS decreases with the increasing transaction size. When the DQNSB scheme is used without any limiting factors, the TPS is much higher than the reference DRLB scheme, as it is able to select the needed blockchain parameters to achieve a higher throughput.

Fig. 10 shows the throughput changes when a malicious node is injected into the blockchain validators. As the ratio of the malicious nodes in the network increases, the probability of selecting a malicious node increases, resulting in a faulty block that causes the throughput to drop. The proposed DQNSB scheme achieves consensus through shard based parallel processing, so the throughput is still high even though the TPS performance degrades due to the malicious block producer. When the ratio of malicious nodes is up to 30%, compared to the performance of when there are no malicious nodes, the DRLB scheme's performance degrades to its 27% level and the proposed DQNSB scheme degrades to its 38% level. However, even at the malicious node ratio of 30%, the DQNSB scheme has an approximate 4.8 times higher TPS performance compared to the DRLB scheme.

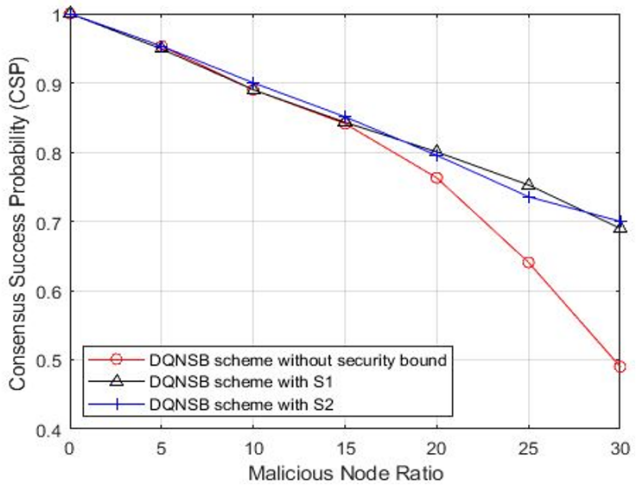

Fig. 11: Consensus success probability performance.

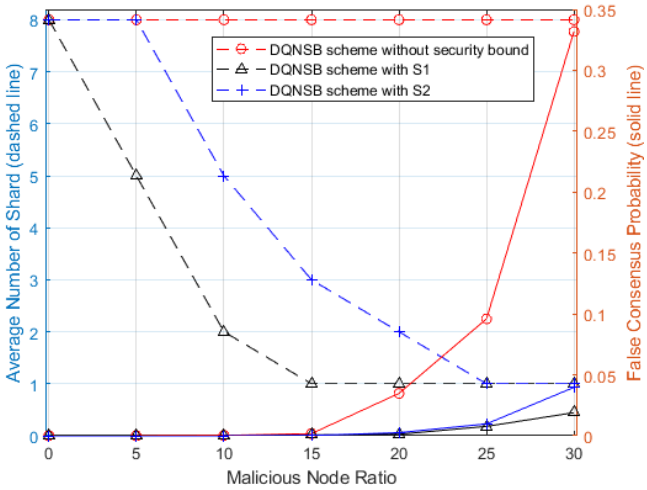

Fig. 12: False consensus probability performance.

B. Security performance comparison with security bound

Figs. 11 and 12 shows the simulation results of the security performance according to security bounds S_1 and S_2, which were derived in lemmas 1 and 2, respectively. S_1 is the upper bound that represents the optimal number of shards, such that

less than one-third of the nodes in a shard are malicious nodes, in order to preserve a honest decision from the PBFT process.

S_2 bound is a less stringent security condition than S_1. S_1 is a condition that prevents a malicious validator from achieving one third of the consensus by *lemma 1*. There are two cases in which consensus problems arise: 1) a faulty block is accepted (FN) and 2) honest blocks are rejected (FP). For case 1, more than 2/3 of the faulty nodes in the shard are required. For case 2, more than 1/3 of the faulty nodes are required (more than one-third of the malicious nodes mean that there are less than two-thirds of honest nodes, preventing the PBFT agreement). S_1 guarantees less than 1/3 faulty nodes for all shards, so it is safe in both cases 1 and 2. On the other hand, S_2 is bounded to prevent 2/3 of malicious nodes in each shard. This is only safe in case 1 (FN). When there are fixed malicious nodes in the sharded blockchain system, the larger the number of shards, the higher the malicious ratio becomes in a single shard. Therefore, S_2 allows more faulty nodes in each shard than S_1, allowing a wider range of shard numbers under an equivalent malicious ratio. Therefore, S_2 decreases the number of shards more slowly than S_1 in a situation where the malicious ratio increases.

Fig. 11 shows the consensus success probability (CSP) as the malicious node rate increases. It is assumed that a block producer can be a malicious node, and the probability of a block producer being malicious also increases with an increase in the rate of malicious nodes, so there is a performance drop in the CSP. If no security bounds are applied, the DRL agent chooses an action to improve only the throughput, so the CSP drops sharply as the malicious rate increases. On the other hand, when S_1 and S_2 are applied, the decrease of CSP due to an increase in the malicious node rate is smaller than that without applying the security bounds. It can be seen that there is not much of a difference in the CSP performance (TN) between S_1 and S_2, because both S_1 and S_2 do not affect the TN performance. S_1 prevents FN and FP cases, and S_2 prevents the FP case. The TN performance drop is directly caused by a faulty block from the malicious block producer, so it is independent of the consensus result.

Fig. 12 shows the false consensus probability (FCP) performance with the security bounds applied. The FCP value was calculated by averaging 10,000 episodes. The left axis shows the average number of shards that change as the malicious node ratio increases, and the right axis represents the FCP performance. When no security bounds are applied, the DRL agent's action will always increase the TPS, resulting in the maximum number of shards to be used regardless of the malicious node ratio. In this case, malicious nodes will be distributed among multiple shards, and the vulnerability and FCP of the blockchain will increase. On the other hand, when the security bounds S_1 and S_2 are used, the probability of false consensus can be effectively limited. However, the false consensus case could not be completely prevented because there was an error in predicting the ratio of malicious nodes in lemma 3. The S_1 security bound is much stricter than S_2, and as a result the security performance of S_1 is twice of S_2 at the 30% malicious node ratio in Fig. 12. The average number of shards decrease as the malicious node ratio increases when the DQNSB scheme is used, when the security bounds are applied. This shows that reducing the number of shards to an appropriate level can increase the FCP security performance in a shard-based blockchain system. When the percentage of malicious nodes reaches 15%, the proposed DQNSB scheme with S_1 stops sharding and forms a single blockchain network. On the other hand, the proposed DQNSB scheme with the S_2 bound will stop sharding and form a single blockchain network when the percentage of malicious nodes reaches 25%. This is because the DQN agent decreases the number of shards as the malicious rate increases, and S_2 allows more faulty nodes in each shard than S_1. Therefore, S_2 will commonly set a larger shard size than S_1 at the same malicious rate, but the FCP will be higher when using S_2 compared to when using S_1. The proposed DQNSB scheme with S2 will stop sharding and form a single blockchain network when the percentage of malicious nodes reaches 25%, because at this condition, sharding the blockchain makes its more vulnerable to colluding attacks. Therefore, the blockchain will sacrifice its efficiency (i.e., its TPS performance) and scalability to achieve a higher security level. The agent cannot increase the shard and block size infinitely to increase the TPS, because an increase in shard and block size causes higher latency [43]. Since the agent does not get any reward for actions outside the constraint, it trains the model in a way that maximizes the long-term reward.

VIII. Conclusion and Future work

In this paper, a DQNSB blockchain scheme that maximizes the TPS throughput and security performance by incorporating DRL adaptive control of the shards is proposed. The proposed DQNSB scheme improves the TPS throughput when compared to the DRLB scheme under various attack environments. In order to solve the security problems that can be caused by applying sharding, an action bounded reinforcement learning approach that uses an agent to select the appropriate action for the blockchain network (based on its trust using recent consensus history) was proposed. The DQN agent maximizes the TPS throughput performance by selecting the optimal configuration actions (e.g., block size, time interval, and number of shards) to be applied to the blockchain. The simulation results also show that the proposed DQNSB scheme results in an improved security performance under malicious attack scenarios and also achieves a higher TPS throughput performance compared to the DRLB scheme of [23]. The attack model applied in this paper uses a consensus layer attack model that directly affects the TPS performance. On the other hand, attacks can be applied to other domains of the blockchain, in which novel countermeasure are needed [44]. Future research needs to be conducted on artificial intelligence (AI)-based blockchain framworks that can adaptively defend various new attack types while enhancing the scalability and TPS performance.

References

[1] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," in *Proc. IEEE Int. Congr. Big Data*, Honolulu, HI, USA, Jun. 2017, pp. 557–564.

[2] S. Guo, X. Hu, S. Guo, X. Qiu, and F. Qi, "Blockchain Meets Edge Computing: A Distributed and Trusted Authentication System," *IEEE Trans. on Ind. Inform.*, vol. 16, no. 3, pp. 1972–1983, Mar. 2020.

[3] J. Wan, J. Li, M. Imran, D. Li, and Fazal-e-Amin, "A Blockchain-Based Solution for Enhancing Security and Privacy in Smart Factory," *IEEE Trans. on Ind. Inform.*, vol. 15, no. 6, pp. 3652–3660, Jun. 2019.

[4] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource Trading in Blockchain-Based Industrial Internet of Things," *IEEE Trans. on Ind. Inform.*, vol. 15, no. 6, pp. 3602–3609, Jun. 2019.

[5] J. Yun, Y. Goh, J.-M. Chung, O. Kim, S. Shin, and Y. Kim, "MMOG User Participation Based Decentralized Consensus Scheme and Proof of Participation Analysis on the Bryllite Blockchain System," *KSII Trans. on Internet and Inform. Syst.*, vol. 13, no. 8, pp. 4093–4107, Aug. 2019.

[6] "Gartner Top 10 Strategic Technology Trends for 2020," Oct. 21, 2019. [Online] Available: https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020/

[7] A. S. Sani, D. Yuan, W. Bao, P. L. Yeoh, Z. Y. Dong, B. Vucetic, and E. Bertino, "Xyreum: A High-Performance and Scalable Blockchain for IIoT Security and Privacy," in *Proc. 2019 IEEE 39th Int. Conf. on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, 2019, pp. 1920–1930.

[8] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2009. [Online] Available: http://www.bitcoin.org/bitcoin.pdf

[9] "Bitcoin daily transactions," Mar. 3, 2020. [Online] Available: https://www.blockchain.com/explorer?view=btc_txperday

[10] "Ethereum project," Mar. 3, 2020. [Online] Available: https://www.ethereum.org/

[11] O. Vashchuk and R. Shuwar, "Pros and cons of consensus algorithm proof of stake. Difference in the network safety in proof of work and proof of stake," *Electronics and Information Technologies*, vol. 9, pp. 106–112, 2018.

[12] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," Jan. 2016. [Online] Available: https://lightning.network/lightning-network-paper.pdf

[13] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," 2017. [Online] Available: https://www.plasma.io/plasma-deprecated.pdf

[14] "The Raiden network," 2015. [Online] Available: https://raiden.network/

[15] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proc. 3rd Symp. Oper. Syst. Design Implement.*, New Orleans, LA, USA, Feb. 1999, pp. 173–186.

[16] 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.*, New York, NY, USA, Oct. 2016, pp. 17–30.

[17] ZILLIQA, "The ZILLIQA Technical Whitepaper v0.1," Aug. 2017. [Online] Available: https://docs.zilliqa.com/whitepaper.pdf

[18] G. Lee, "EOS.IO Technical White Paper v2," Mar. 2018. [Online] Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[19] T. Duong, L. Fan, T. Veale, and H. S. Zhou, "Securing Bitcoin-like Backbone Protocols against a Malicious Majority of Computing Power," 2016. [Online] Available: https://allquantor.at/blockchainbib/pdf/duong2016securing.pdf

[20] J. Ray. Sharding Introduction R&D Compendium. [Online] Available: https://github.com/ethereum/wiki/wiki/Sharding-introduction-R&D-compendium#information

[21] "The Advantages and Disadvantages of Off-Chain to On-Chain Scaling," [Online] Available: https://quantus.biz/crypto/the-advantages-and-disadvantages-of-off-chain-to-on-chain-scaling/

[22] M. Conti, E. Sandeep Kumar, C. Lal, and S. Ruj, "A Survey on Security and Privacy Issues of Bitcoin," *IEEE Commun. Surv. & Tut.*, vol. 20, no. 4, pp. 3416–3452, 4th Quarter, 2018.

[23] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Performance Optimization for Blockchain-Enabled Industrial Internet of Things (IIoT) Systems: A Deep Reinforcement Learning Approach," *IEEE Trans. on Ind. Inform.*, vol. 15, no. 6, pp. 3559–3570, Jun. 2019.

[24] J. Yun, Y. Goh, and J.-M. Chung, "Trust-Based Shard Distribution Scheme for Fault-Tolerant Shard Blockchain Networks," *IEEE Access*, vol. 7, pp. 135164–135175, 2019.

[25] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things," *IEEE Trans. on Ind. Inform.*, vol. 14, no. 6, pp. 3690–3700, Aug. 2018.

[26] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous Reputation System for IIoT-Enabled Retail Marketing Atop PoS Blockchain," *IEEE Trans. on Ind. Inform.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.

[27] D. Miller, "Blockchain and the Internet of Things in the Industrial Sector," *IT Professional*, vol. 20, no. 3, pp. 15–18, May 2018.

[28] R. Sutton and A. Barto. (2017). *Reinforcement Learning: An Introduction (DRAFT).* [Online] Available: http://www.incompleteideas.net/book/bookdraft2017nov5.pdf

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[30] C. H. Liu, Q. Lin, and S. Wen, "Blockchain-Enabled Data Collection and Sharing for Industrial IoT With Deep Reinforcement Learning," *IEEE Trans. on Ind. Inform.*, vol. 15, no. 6, pp. 3516–3526, Jun. 2019.

[31] C. Qiu, H. Yao, F. R. Yu, C. Jiang and S. Guo, "A Service-Oriented Permissioned Blockchain for the Internet of Things," *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 203-215, 1 March-April 2020.

[32] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu and C. Zhao, "Blockchain-Based Software-Defined Industrial Internet of Things: A Dueling Deep Q -Learning Approach," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4627-4639, June 2019.

[33] A. Clement, E.Wong, L. Alvisi, and M. Dahlin, "Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults," in *Proc. 6th USENIX Symp. Netw. Syst. Des. Implementation*, Boston, MA, USA, Apr. 2009, pp. 153–168.

[34] R. Shrestha and S. Y. Nam, "Regional Blockchain for Vehicular Networks to Prevent 51% Attacks," *IEEE Access*, vol. 7, pp. 95033–95045, 2019.

[35] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *J. ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," in *Proc. NIPS Deep Learn. Workshop*, Lake Tahoe, USA, 2013, arXiv:1312.5602.

[37] A. Paszke, "Reinforcement Learning (DQN) Tutorial," 2018. [Online] Available: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

[38] A. Singh, T. Das, P. Maniatis, P. Druschel, and T. Roscoe, "BFT Protocols Under Fire," in *Proc. 5th USENIX Symp. Netw. Syst. Des. Implementation*, Berkeley, CA, USA, Jan. 2008, pp. 189–204.

[39] "Blockchain confirmation" [Online] Available: https://en.bitcoin.it/wiki/Confirmation

[40] X. Chen, S. Zhu and Y. Ji, "Entropy based uncertainty measures for classification rules with inconsistency tolerance," *in Proc. IEEE International Conference on Systems, Man, and Cybernetics*, Nashville, TN, USA, 2000, pp. 2816-2821.

[41] I. J. Sledge, M. S. Emigh and J. C. Prncipe, "Guided Policy Exploration for Markov Decision Processes Using an Uncertainty-Based Value-of-Information Criterion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2080-2098, June 2018.

[42] T. Tieleman and G. Hinton, "Lecture 6.5-RMSPROP: Divide the gradient by a running average of its recent magnitude," *COURSERA, Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 2631, Oct. 2012.

[43] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*,Toronto, Canada, Oct. 2018, pp. 931-948.

[44] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and D. Mohaisen, "Exploring the Attack Surface of Blockchain: A Comprehensive Survey," in *IEEE Commun. Surv. & Tut.*, to be published. doi: 10.1109/COMST.2020.2975999.

JUSIK YUN (awp212@yonsei.ac.kr) received a B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, Republic of Korea in 2016. He is currently pursuing a combined M.S. and Ph.D. degree in electrical and electronic engineering at Yonsei University, where he is a researcher of the Communications and Networking Laboratory (CNL). His current research interests include blockchain, deep reinforcement learning, network intelligence, trust management network security systems, and 5G mobile edge computing.

YUNYEONG GOH (rhdbsdud@yonsei.ac.kr) received a B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, Republic of Korea in 2019. He is currently pursuing a combined M.S. and Ph.D. degree in electrical and electronic engineering at Yonsei University, where he is a researcher of the CNL laboratory. His current research interests include blockchain, AI, trust network security systems, and augmented reality.

JONG-MOON CHUNG (jmc@yonsei.ac.kr) received B.S. and M.S. degrees in electronic engineering from Yonsei University and Ph.D. in electrical engineering from the Pennsylvania State University. Since 2005, he has been a professor in the School of Electrical and Electronic Engineering at Yonsei University, where he is currently the Associate Dean of the College of Engineering and professor of the Department of Emergency Medicine in the College of Medicine at Yonsei University. From 1997 to 1999, he was an assistant professor and instructor at the Pennsylvania State University in the Department of Electrical Engineering. From 2000 to 2005, he was with the Oklahoma State University (OSU) as a tenured associate professor in the School of Electrical & Computer Engineering. Currently he serves as a Vice President of the IEEE Consumer Electronics Society, Editor of the *IEEE Transactions on Vehicular Technology*, Associate Editor of the *IEEE Transactions on Consumer Electronics*, Section Editor of the *Wiley ETRI Journal*, and Co-Editor-in-Chief of the *KSII Transactions on Internet and Information Systems*. In 2019, he received the Minister Award from the Ministry of the Interior and Safety and an Appreciation Award from the National Police Agency of the Republic of Korea. In 2019, 2018, and 2008 he received Outstanding Accomplishment Faculty Awards, and in 2019, 2014, 2009, and 2007 he received Outstanding Teaching Awards from Yonsei University. In 2012, he received the Republic of Korea government's Defense Acquisition Program Administration Award. As a tenured associate professor at OSU, in 2005 he received the Regents Distinguished Research Award and the Halliburton Outstanding Young Faculty Award. In 2004 and 2003, respectively, he received the Technology Innovator Award and the Distinguished Faculty Award from OSU, and in 2000 he received the First Place Outstanding Paper Award at the IEEE EIT 2000 conference held in Chicago, USA. He is also a pledge book award winning member of the Eta Kappa Nu (HKN) Epsilon Chapter. In addition, Dr. Chung's 12 Coursera (www.coursera.org) courses focus on deep learning, big data, cloud computing, 5G & 4G mobile communications, Wi-Fi, Bluetooth, augmented reality (AR), internet of things (IoT), Skype, YouTube, TCP/IP, as well as hardware and software of smartphones and smartwatches. Dr. Chung will be serving as the General Chair of IEEE ICCE 2022 and IEEE ICCE-Asia 2020, and was the General Chair of several conferences including IEEE MWSCAS 2011.