





Article

KNN-Based Consensus Algorithm for Better Service Level Agreement in Blockchain as a Service (BaaS) Systems

Qingxiao Zheng ¹, Lingfeng Wang ¹, Jin He ^{1,*} and Taiyong Li ^{2,*}¹ Industry College of Blockchain, Chengdu University of Information Technology, Chengdu 610225, China² School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu 611130, China

* Correspondence: hejin@cuit.edu.cn (J.H.); litaiyong@gmail.com (T.L.)

Abstract: With services in cloud manufacturing expanding, cloud manufacturers increasingly use service level agreements (SLAs) to guarantee business processing cooperation between CSPs and CSCs (cloud service providers and cloud service consumers). Although blockchain and smart contract technologies are critical innovations in cloud computing, consensus algorithms in Blockchain as a Service (BaaS) systems often overlook the importance of SLAs. In fact, SLAs play a crucial role in establishing clear commitments between a service provider and a customer. There are currently no effective consensus algorithms that can monitor the SLA and provide service level priority. To address this issue, we propose a novel KNN-based consensus algorithm that classifies transactions based on their priority. Any factor that impacts the priority of the transaction can be used to calculate the distance in the KNN algorithm, including the SLA definition, the smart contract type, the CSC type, and the account type. This paper demonstrates the full functionality of the enhanced consensus algorithm. With this new method, the CSP in BaaS systems can provide improved services to the CSC. Experimental results obtained by adopting the enhanced consensus algorithm show that the SLA is better satisfied in the BaaS systems.

Keywords: BaaS system; blockchain consensus algorithm; KNN; service level agreement; transaction priority



Citation: Zheng, Q.; Wang, L.; He, J.; Li, T. KNN-Based Consensus Algorithm for Better Service Level Agreement in Blockchain as a Service (BaaS) Systems. *Electronics* **2023**, *12*, 1429. <https://doi.org/10.3390/electronics12061429>

Academic Editor: Ping-Feng Pai

Received: 4 February 2023

Revised: 28 February 2023

Accepted: 14 March 2023

Published: 16 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Blockchain, business analytics, and the Internet of Things (IoT) are the emerging industry trends to which scholars and practitioners have paid much attention in recent years. The state-of-the-art research related to these technologies has been summarized by Zhang and Chen [1]. Blockchain as a Service (BaaS) is a new technology that combines cloud computing and blockchain technology. As a third-party service, BaaS provides customers with the ability to create and manage blockchain-based networks through cloud technology. It is a relatively new technology trend that provides third-party services within the blockchain technology domain. Blockchain applications are more than just cryptocurrency transactions. They have expanded to encompass all types of secure transactions. As a result, hosting services are increasingly in demand. Blockchain technology has been used to provide services to more customers as a service model through the cloud. This model works similarly to SaaS, PaaS, and IaaS models, which support the usage of cloud-based applications and storage. Blockchain technology is complex, and much effort is required to build, maintain, and monitor a blockchain system when applied. In order to increase the accessibility of the blockchain and distributed ledgers, we need to leverage blockchain with lower costs and less overhead, especially for businesses. BaaS is a promising technical option that can meet these goals [2]. However, critical issues in current public blockchain systems prevent them from being used as a generic platform for different services and applications. Bitcoin can handle about 5.5 transactions per second (TPS), and Ethereum can process about 20 TPS, which is far below the mainstream payment systems. There is

no silver bullet that solves all of these problems due to the Trilemma, as mentioned by the founder of Ethereum, Vitalik Buterin: public blockchain systems can have only two of the following three properties: decentralization, scalability, and security [3].

Most previous studies have not solved the scalability issue well. It is difficult for cloud service providers (CSPs) to guarantee effective SLA with cloud service consumers (CSCs). There are some studies that achieve better data query/sharing services based on blockchain service, such as BlockShare [4], Verifiable Query Layer (VQL) [5], and vChain+ [6], but these services cannot solve the SLA issue. To address the SLA issue in the BaaS environment, this paper proposes a novel KNN-based consensus algorithm by classifying the transactions with priority. Any factor that impacts the priority of the transaction can be used to calculate the distance in the KNN algorithm. Such factors include the SLA definition, the smart contract type, the CSC type, and the account type.

This paper has three main contributions: (1) A simple supervised learning method, KNN, is used to build a consensus algorithm for the first time. (2) With the realization of the full functionality of the enhanced consensus algorithm, the CSP in the BaaS systems can provide improved services to the CSC. (3) Experimental results demonstrate that the SLA is better satisfied in the BaaS systems. The transaction with higher priority that arrives later is executed early.

We have organized the rest of the paper as follows. Section 2 provides a review of related work. Section 3 depicts the problem studied in this paper. Section 4 describes preliminaries, such as BaaS, cloud computing SLA in BaaS, and the KNN algorithm. The proposed KNN-based consensus algorithm is detailed in Section 5. Section 6 reports and analyzes the experimental results. Section 7 concludes this paper.

2. Related Work

2.1. Evolution of Consensus Algorithms

In decentralization, any node in a blockchain can submit a transaction to be stored in the system, so it is important that there are processes that can ensure that each node reaches a consensus to accept or reject the submitted transactions. These processes are essentially considered consensus algorithms.

PoW is the first consensus protocol used in blockchain. It works with Bitcoin and Ethereum, among others. In each round of consensus, PoW uses computational power competition to decide which node can pack recent transactions into a new block. PoW guarantees eventual consistency based on the major distributed nodes with high computational power in reaching a consensus. It is a probabilistic-finality consensus protocol [7].

PoS was created to overcome shortages that occur when PoW consumes too much computational power. In each round of consensus, PoS considers not only the computational power but also the stake held when deciding which node can pack recent transactions into a new block. The difference between PoS and PoW is the importance of the amount of stake (coins) and of how many times the nonce is adjusted. PoS is also a probabilistic-finality consensus protocol [7].

Raft reaches a consensus by an elected leader. A node in a blockchain system with Raft is either a leader or a follower and can be a candidate in an election scenario when a current leader is unavailable. The Raft leader has the responsibility of logging replications to the followers, and it periodically notifies the followers of its alive state by sending a heartbeat message. Raft implements a consensus based on the leader schema. The whole blockchain system has only one elected leader, which has full responsibility for managing logged replications to followers.

PBFT provides a practical Byzantine state machine replication that tolerates the Byzantine Generals' Problem caused by malicious nodes. It assumes that these malicious nodes have independent failures and send manipulated messages. Distributed nodes in a blockchain system with PBFT are appointed as leaders, in turn, and others are appointed as backup nodes. All nodes in the blockchain system assume that all honest nodes will make an agreement by using predefined rules when communicating with each other.

The above consensus algorithms are the main types of consensus algorithms used in the blockchain system. They have different decentralization and transaction throughput capabilities, and these consensus algorithms have their own application scenarios based on the requirement of decentralization and performance grades.

The data structure of the transaction in most blockchains is simple. It includes a receiver address, transaction amount, etc. In a typical blockchain system, such as Bitcoin, the receiver address is located in the “Locking-Script” field of a transaction output, and the transaction amount is located in the “Amount” field of a transaction, as shown in Tables 1 and 2. The blockchain node verifies the validity and effectiveness of the transaction, while transactions are not classified or processed with priority in the consensus procedure since there is no field in the transaction data structure to describe the transaction priority or type [8]. There is an opportunity for optimization by classifying and processing transactions with priority. The method introduced in this paper uses a strategy that ensures that transactions with higher priority can be processed in a timely manner.

Table 1. The structure of a transaction in Bitcoin.

| Size | Field | Description |
|--------------------|----------------|--|
| 4 bytes | Version | Specifies which rules this transaction follows |
| 1–9 bytes (VarInt) | Input Counter | How many inputs are included |
| Variable | Inputs | One or more transaction inputs |
| 1–9 bytes (VarInt) | Output Counter | How many outputs are included |
| Variable | Outputs | One or more transaction outputs |
| 4 bytes | Locktime | A Unix timestamp or block number |

Table 2. The structure of a transaction output in Bitcoin.

| Size | Field | Description |
|--------------------|---------------------|---|
| 8 bytes | Amount | Bitcoin value in satoshis |
| 1–9 bytes (VarInt) | Locking-Script Size | Locking-Script length in bytes, to follow |
| Variable | Locking-Script | A script defining the conditions needed to spend the output |

2.2. QoS Assurance

Previous studies show that most of the recently developed public blockchain systems focus on increasing transaction throughput to improve scalability. Even if the existing consortium blockchain TPS is improved compared with public blockchains, the efficiency of the consensus algorithm is still low, and its fault tolerance is still poor [9].

Blockchain technology plays an important role in supporting Service Level Agreements (SLAs) that guarantee quality of service (QoS) standards for various service providers. Meanwhile, although smart contracts are applied in traditional cloud providers, SLAs are rarely used to provide improved service [10].

The blockchain data are used in the BaaS system to provide a range of operational services, such as search queries and task submission on the blockchain [11]. Driven by BaaS, the content of a cloud service becomes more abundant, and the CSC increases its requirements for QoS [12,13]. In order to solve the QoS assurance problem between the CSP and CSC, one method is proposed to support the cloud computing service level agreement. The purpose of this agreement is to create a healthy environment for operations on the network so that the CSC can enjoy not only the service promised verbally by the CSP but also a service that is regulated and fully protected [14].

Existing research recognizes the critical role played by the service provider [15], but it lacks a valid method that enhances the consensus algorithm with improved QoS assurance. Since smart contracts stand on the application layer, providing QoS assurance for smart

contracts is relatively inefficient and is not the best choice; it is better to put this assurance in the kernel module of the BaaS system for all transactions.

According to the above studies, the existing consensus algorithms cannot provide effective support for SLAs between a CSP and a CSC. It is important to provide QoS assurance in a consensus algorithm, and how various transactions are classified is key in supporting QoS. As the KNN is one of the simplest classification methods, it was chosen here for classifying transactions. The main aim of a KNN is to find k training samples that are closest to the new sample and assign the majority label of the k samples to the new sample. Despite its simplicity, the KNN has been successful in solving a wide range of regression and classification problems, including handwritten characters and image recognition scenarios. As a non-parametric approach, it often succeeds in classification situations where the decision boundary is highly irregular [16].

In this paper, we introduce a KNN-based consensus algorithm for improved service level agreements in BaaS systems. Even with the efficiency or poor fault tolerance in BaaS systems, the QoS assurance between the CSC and the CSP is better achieved with the enhanced consensus algorithm.

3. Problem Definition

Performance and scalability are always key non-functional requirements in application systems, and such application systems generally achieve extremely high transaction throughput. China's central bank digital currency, DCEP, for example, has about 220,000 TPS. While blockchain systems or BaaS achieve a lower transaction throughput, Bitcoin has 5.5 TPS, and Ethereum has 20 TPS on average. The CSP in BaaS can only provide a similar transaction throughput performance; it cannot meet the requirements of the CSC in the SLA due to the limitation of throughput [17].

The two major challenges of blockchain, scalability and throughput issues, have been studied and improved extensively as the below methods.

Consortium blockchain does not use high-power consensus algorithms such as PoW. They consume much effort and have a complicated consensus process. Hyperledger Fabric is a typical consortium blockchain that uses a Raft or PBFT consensus algorithm [18] to reach a consensus faster than a public blockchain that uses PoW or PoS. It can achieve higher throughput than a public blockchain, and its throughput is 3500 TPS on average [17].

The Ethereum community scheduled a scaling method that performs sharding to improve Ethereum's scalability and capacity. It splits Ethereum data horizontally to spread the load. After Ethereum upgrades to 2.0 with sharding, it is expected to reach 100,000 TPS [17].

NeuChain utilized an ordering-free consensus that makes ordering implicit through deterministic execution to markedly improve the throughput of the blockchain system. The distributed experimental results show that NeuChain can achieve 47.2–64.1X throughput improvement over HyperLedger Fabric [19].

Some hardware methods to improve blockchain performance have been proposed. For instance, a FPGA-based NoSQL caching system with high performance was proposed to improve the throughput and scalability of the blockchain system, and this can increase the throughput to about 10,000 TPS when a cache hit occurs [20].

Except for the above performance optimization for consensus algorithms, some proposals for the optimization of other aspects related to the blockchain system and the blockchain-based framework have also been researched. For some special scenarios, such as confidential transactions, the SymmeProof method, used to reduce the transmission cost, was proposed, and it can improve communication efficiency and indirectly improve the transaction throughput for special types of transactions [21]. A mechanism where full nodes can be compensated fairly for their full blockchain data storage and where clients can query blockchain data effectively was constructed [22]. LineageChain provides an innovative method to support efficient provenance and history data query processing [23]. The secure performance of the blockchain-based federated learning framework has been proposed to be optimized [24].

Due to the need to establish trust between completely anonymous entities, a time-consuming mining-based consensus mechanism was used. Thus, it takes a long time to achieve transaction finality and results in much lower transaction throughput. The limitation of throughput can be increased by using the methods mentioned above. However, compared to traditional e-business application systems that do not adopt blockchain technology, the optimized blockchain still presents a gap between throughput performance and the requirements of e-business scenarios. Although some of the methods mentioned above can improve the throughput of the blockchain system to different degrees, they generally cannot be applied for most scenarios.

Considering the existing studies on blockchain performance optimization, the throughput of a blockchain system cannot reach the same magnitude as traditional e-business application systems. Therefore, another approach where the CSP of BaaS provides an SLA that meets the CSC's requirements is needed.

4. Preliminaries

4.1. BaaS

Blockchain as a Service (BaaS) is a service provided by third parties that create and manage cloud-based networks for customers building their own blockchain applications. The decentralization of blockchain, the pervasiveness of IoT, and the high computing power of cloud computing are combined in BaaS, while the transparency and openness of the system are ensured. The main functional behaviors of blockchain, such as off-chain and on-chain synchronization, node validity, consensus, and forking, are managed by BaaS. The CSC can fully outsource the technical overhead to the CSP [25].

BaaS inherits blockchain's challenges, synchronization mechanism, transaction throughput, storage space, network congestion, accessibility, and cost issues, among others. As discussed in Section 3, the transaction throughput of the blockchain system cannot be improved to match the magnitude of traditional e-business application systems. BaaS also has a transaction throughput issue that cannot be completely resolved. This paper depicts a method to optimize SLAs for key transactions when transaction throughput cannot be further promoted in BaaS.

4.2. Cloud Computing SLAs in BaaS

BaaS is introduced as an important part of the cloud service platform of several giant enterprises that can provide a trustworthy decentralization service, such as the Alibaba-built BaaS system on Kubernetes, the IBM-built BaaS system on Bluemix, and the Microsoft-built BaaS system on the Microsoft Azure cloud platform [2].

An SLA formally defines the relationship between two or more parties in BaaS, one of which is the CSC and one of which is the CSP. It specifies what CSCs can be served by a CSP, the obligations that both the CSC and the CSP shall fulfill, the objectives of the service related to performance, availability, and security, and the processes that guarantee compliance with SLAs. In general, an SLA includes the following typical components: the type of service to be provided; the desired performance level of the service; the reporting process that occurs when the service is unstable or unavailable; the time frame for responding and issuing a problem resolution; the schema for monitoring and reporting the service level; the consequences that result when the CSP does not fulfill its promises; termination clauses; and constraints of service. The SLA is used to evaluate the QoS provided by the CSP in BaaS, as in IaaS, PaaS, and SaaS.

4.3. KNN

As a typical supervised learning method in machine learning, the k -nearest neighbors algorithm (KNN) has shown its advantages for both classification and prediction [26–28]. It is a supervised learning classifier and is used to classify or predict the grouping of an individual data point according to the distance between different feature vectors. KNN has two main phases: (1) the training phase, in which feature vectors are stored and labels of

the training samples are classified, and (2) the classification phase, in which an unlabeled vector is classified by assigning the most frequent label among the k training samples that are nearest to that vector. Although it can be used in either regression or classification, it is typically used as a classification algorithm, as in this paper.

The parameter k of the KNN has an extraordinary impact on the classification result, and the data impact the best choice of k . In general, a larger k reduces the effect of noise on classification, but it is then less distinct among class boundaries. Cross-validation is used when assigning different k values to different test samples in previous solutions. A kTree method that learns different optimal k values for different tests of individual data points is proposed in the training stage during kNN classification [29].

Although KNN was developed by Joseph Hodges and Evelyn Fix in 1951 [30], due to its simple implementation and relatively excellent performance, it, along with its improved methods, has been widely used in the applications of several industries in the last three years, including cancer diagnosis in medicine [31], gas-bearing reservoir prediction in geophysics [32], and antenna optimization and design in the electronic industry [33]. This paper applies the KNN to classify the priority of the transaction in BaaS, and transactions are executed with different priorities based on priority classification. It should be noted that the KNN can be replaced by other classification models in practice.

5. KNN-Based Consensus Algorithm

We propose a KNN-based consensus algorithm in this paper, and we describe this algorithm in three subsections: the Priority-Queue-Enabled Transaction Pool, Attribute Selection in the KNN-Based Consensus Algorithm, and Transactions Classified to a Different Priority Queue by Adopting the KNN-Based Consensus Algorithm. The first subsection introduces how the existing consensus algorithm puts newly received transactions into the transaction pool and points out that our optimization aim is to classify newly received transactions and determine their priority according to the classification results. The second subsection explains how classification attributes are selected, and the third subsection introduces how transactions are classified and how the transaction pool is filled according to the classified priority queue.

5.1. The Priority-Queue-Enabled Transaction Pool

The blockchain system only allows a limited number of transactions since a block can only contain a limited number of transactions. Transactions that exceed the limit of arrival should not be included in the block. For example, in a four-node consortium blockchain system, a Practical Byzantine Fault Tolerance algorithm [34] (also known as PBFT) is applied, where multiple clients connect to the blockchain system, and the transaction count limit of the transaction pool is set to 1000. As the leader node, Node4 picks the transactions to be sent to the Transaction Pool. However, once it reaches the pool limit, the transactions that arrive later will not be sent to the transaction pool. The existing PBFT consensus protocol can be illustrated in Figure 1.

A new priority-queue-enabled transaction pool is introduced in this paper. Based on the attributes, received transactions should be classified into queues of different priorities, and each queue is a first-in first-out queue. The transactions that arrive later are cached once the queue is full. The details on how the attributes are selected and how the incoming transactions are classified will be presented in the following subsections.

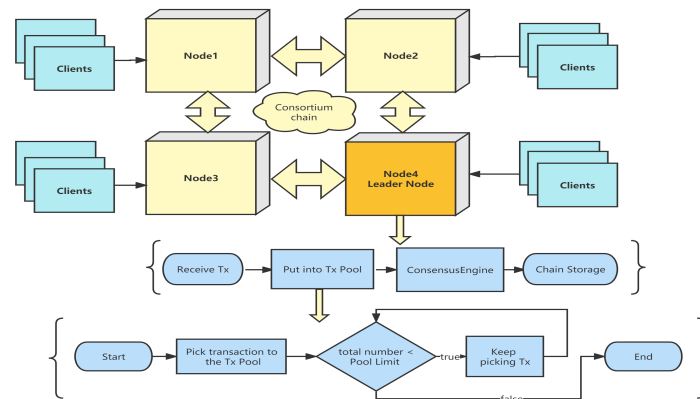


Figure 1. The PBFT consensus protocol.

5.2. Attribute Selection in the KNN-Based Consensus Algorithm

The account type, the CSC (cloud service consumer) type, and the contract type i chosen as the attributes to be used in the KNN algorithm to calculate the distance.

- (1) Account type. There are different kinds of roles in the blockchain system. Roles based on access control should usually be defined in the system, such as chain administrators, system administrators, and ordinary accounts. Chain administrators have access control permissions, that is, grant permissions. System administrators need to manage permissions related to system functions, and each permission should be granted independently, including contract deployment, user table creation, node management, and system parameter modification. Chain administrators can authorize other accounts to be chain administrators or system administrators or authorize ordinary accounts to write table lists. Table 3 lists the permissions related to the roles.

Table 3. Permissions related to the roles.

| Permission Type |
|--|
| Permission of chain administrators |
| Permission of system administrators |
| Permission to deploy contracts |
| Permission to create user tables |
| Permission to manage nodes |
| Permission to modify system parameters |
| Permission to write user tables |

- (2) CSC type. Transactions from different CSCs have varying priorities. Since CSC clients run similar CSCs, they should have similar priorities. However, if a CSC client experiences limitations in terms of CPU, memory, storage, or network resources, it may need to adjust the CSC priority accordingly.
- (3) Contract type. Besides contracts relating to chain management, there are many smart contracts. Some of them handle time-critical applications, some of them handle applications that are not so critical but are urgent, and some of them do not care about the timing. In the consortium Blockchain FISCO BCOS [35], for example, there are many contract types in the consortium Blockchain system, as shown in Tables 4 and 5.

Table 4. Address range in FISCO BCOS [35].

| Address Use | Address Range |
|-------------------------------------|---------------|
| Ethereum precompiled | 0x0001–0x0008 |
| Reserve | 0x0008–0x0fff |
| Precompiled contracts in FISCO BCOS | 0x1000–0x1006 |
| Reserved in FISCO BCOS | 0x1007–0x5000 |
| Interval assigned by user | 0x5001–0xffff |
| Reserved for CRUD | 0x10000+ |
| Used by Solidity | Other address |

Table 5. Precompiled contracts in FISCO BCOS [35].

| Address | Feature |
|---------|------------------------------------|
| 0x1000 | managing system parameters |
| 0x1001 | contract of the table factory |
| 0x1002 | implementing CRUD operations |
| 0x1003 | managing consensus nodes |
| 0x1004 | Contract Name Services |
| 0x1005 | managing storage table authorities |
| 0x1006 | configuring parallel contracts |

The chain management contracts have higher priority when we send transactions to the Tx pool. For those fundamental contracts, such as the table factory and CRUD operations, we cannot determine the priority that depends on the CSC's request. The contract type can be used to calculate the priority.

5.3. Transactions Classified to a Different Priority Queue by Adopting the KNN-Based Consensus Algorithm

A KNN-based consensus algorithm is proposed to select the transactions for the queue. When a CSC is registered, we can obtain the key properties of the transactions in this CSC, which may impact its transaction priority, such as the SLA type, the contract type, the account type, the CSC type, the CPU type, the memory size, the storage type, and the network bandwidth.

Classification is an important task in machine learning. The KNN algorithm is simple and accurate and is used for regression models and pattern classification [36]. The term “non-parametric” is used when there are no parameters, or there is a fixed number of parameters, regardless of data size. The size of the training dataset determines the parameters, although no assumptions need to be made about the underlying data distribution. Therefore, KNN is probably the best choice for any classification study that involves little or no prior knowledge of the data distribution. KNN is also a lazy learning method, which means it stores all training data and waits to generate test data without creating a learning model [37]. This is the reason why the KNN algorithm was chosen to optimize the consensus algorithm.

The KNN algorithm classifies as follows: there is an existing set of sample data or a training set. All of these data have been labeled, and we know the class of each piece of data. When a new piece of data has no label, we compare that new piece of data with every existing piece of data. We then take the nearest neighbors and check their labels. We look at the k data that are most similar to the known dataset, which is what k represents. Finally, we perform a majority vote on the similar k data, and the label of the winning vote is selected as the new class to be assigned to the new piece of data. The detailed steps to calculate distance and determine the k value are listed below:

- (1) The distance calculation and normalization procedure is as follows: We can use

$$d(p, q) = \sqrt{\sum_{i=0}^n (p_i - q_i)^2}$$

to calculate the Euclidean Distance between input data and existing data. Which term in the above equation makes the most difference? It must be the one with the largest magnitude. To reduce the impacts of the magnitude, we need to normalize the sample data to give all factors an equivalent weight. In this paper, every attribute is scaled from 0 to 1, which can be formulated as

$$\text{newValue} = (\text{oldValue} - \text{min}) / (\text{max} - \text{min}).$$

- (2) The KNN algorithm does not need a training procedure. However, the selection of k is important for accuracy. Basically, k should be an integer between 1 and 20. We divided the sample data into two portions: 90% of them was used for the known set, while the remaining 10% was for testing. We increase k successively and calculate the accuracy. The k value that achieves the highest accuracy is chosen for classifying the transactions from the incoming client in the final algorithm. Algorithm 1 describes the procedure by which the incoming transaction is classified into different priority queues, while Algorithm 2 details how transactions are collected and sent to T_x Pool. In the system, there are N queues starting from Q_1 to Q_N , where Q_N and Q_1 have the highest and lowest priority, respectively. n is in $1..N$, and $Q_n.size$ denotes the number of transactions in Q_n .

Algorithm 1 Transaction classification.

Input: T_x : The incoming transaction; $Q_1 \dots Q_N$: The priority Q list from Q_1 to Q_N ; *sample*: The sample dataset;

Output: updated Q_n

- 1: Initial *trainingData* = prepareLabeledDataset(*sample*); ▷ Prepare training dataset
 - 2: $n = \text{classify}(T_x, \text{trainingData}, k)$ ▷ Classify T_x with given training data and k value, n means which Q_n it belongs to
 - 3: **if** ($Q_n.size < MAX_SIZE$) **then**
 - 4: PUSH T_x to Q_n ; ▷ PUSH T_x to corresponding Q_n once the Q isn't full
 - 5: **else**
 - 6: Save T_x to memory pool; ▷ Otherwise temporarily save the T_x to memory pool
 - 7: **end if**
 - 8: **return** Q_n
-

Algorithm 2 Prepare the T_x Pool.

Input: $Q_1 \dots Q_N$: The priority queue list from Q_1 to Q_N ; *POOL_LIMIT*: The limit on the number of transactions that can be accommodated in the T_x_Pool ;

Output: T_x_Pool

- 1: Set T_x_Pool to empty, $j = N$; ▷ $j = N \rightarrow$ start to pick up priority queue item from Q_N to Q_1
 - 2: **while** ($T_x_Pool.size < POOL_LIMIT$) and ($j > 0$) **do** ▷ Keep filling the pool in order of priority of the queue items until $j \leq 0$ or T_x_Pool is full
 - 3: Fill T_x_Pool with Q_j
 - 4: $j - -$ ▷ Move to the next lower priority queue item
 - 5: **end while**
 - 6: Save remaining queue items to memory pool if the T_x_Pool is full;
 - 7: **return** T_x_Pool
-

With the KNN algorithm, the consensus algorithm can be optimized with SLA assurance. Any transaction that is classified with higher priority can be handled earlier. Figure 2 shows the data flow through which transactions are selected and sent to the transaction pool.

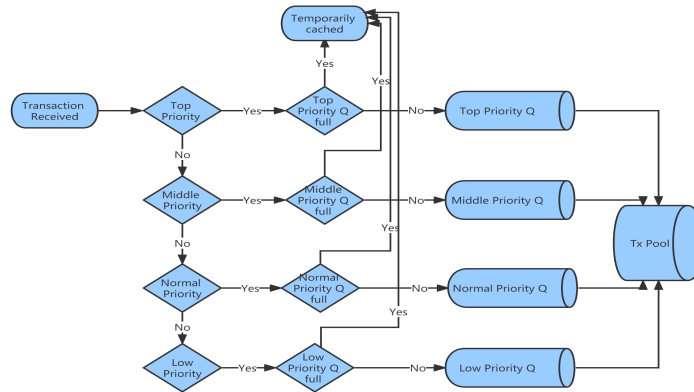


Figure 2. Data flow through which transactions are selected and sent to the transaction pool.

Figure 3 provides the overall framework of the enhanced KNN-enabled consensus algorithm. As shown in the figure, the newly added KNN-enabled transaction classification module is a new concept in the BaaS system. Any CSP can integrate this module into its BaaS framework when it wants to provide a guaranteed SLA to the CSC. Some minor changes are required when preparing the transaction pool, which picks up transactions in the order of priority. If the SLA of one transaction is 1 s, 2000 transactions come in within 1 s, the TPS of the BaaS system is 1000, and the CSP receives this transaction with a sequence number 1100. Only transactions with a sequence number smaller than 1000 can be handled, so this transaction cannot meet the requirements of the SLA. With this KNN-enabled consensus algorithm, since it has a higher priority, it can be sent to the transaction pool with a smaller sequence number (e.g., 100), and it can be handled earlier within the SLA.

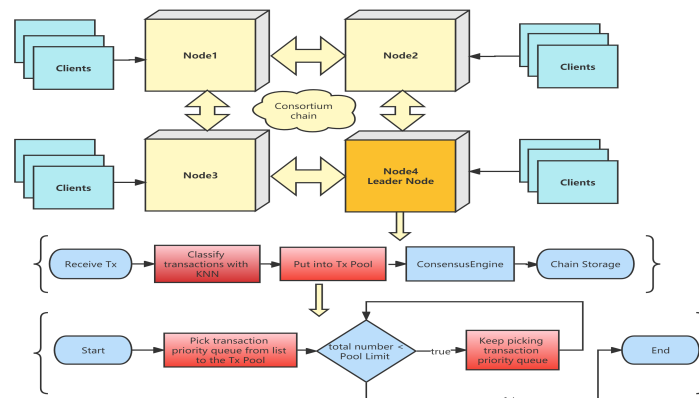


Figure 3. The KNN-enabled PBFT consensus algorithm.

6. Simulation Experiments and Analysis

6.1. TPS Limit Measured from the Existing BaaS System

To evaluate the proposed consensus algorithm, we ran a performance test on the well-known FISCO-BCOS Consortium Blockchain system. The flowchart of the test process is shown in Figure 4.

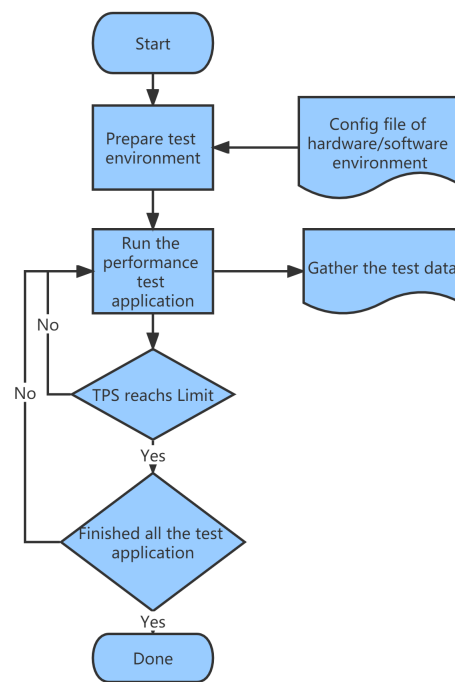


Figure 4. Performance test flow chart.

We deployed a Cloud Virtual Machine standard type S3 on the Tencent Cloud to simulate the BaaS system. Tables 6 and 7 list the details of the hardware and software environment, respectively.

Table 6. Virtual machine hardware configuration.

| Hardware Type | Hardware Configuration |
|-----------------------|---|
| CPU | Intel Xeon Cascade Lake 8255C (2.5 GHz) |
| vCPU | 4 Core |
| Memory (GB) | 4 GB |
| System disk type | High-performance cloud disk |
| System disk size (GB) | 50 GB |
| Bandwidth | 100 Mbps |

Table 7. Virtual machine software configuration.

| Software Type | Software Version |
|---------------|---|
| FISCO BCOS | V2.7.2 |
| OS | CentOS 7.6 64 bit |
| WeBase | V1.5.1 |
| JAVA | jdk1.8.0 |
| IDE | IntelliJ IDEA 2022.2.2 (Ultimate Edition) |

A JAVA performance testing application [38] was used to measure the TPS on the BaaS simulation system. It started at 1000 transactions and set the TPS limit from 10 to 100 with a step of 10. Figure 5 shows the Actual TPS/TPS Limit results. The TPS limit setting is the maximum number of transactions that the testing application is allowed to send, and the actual TPS is the actual number of transactions that the testing application sends. If the actual TPS is smaller than the TPS limit setting, then the testing application has reached the maximum TPS supported by the BaaS system.

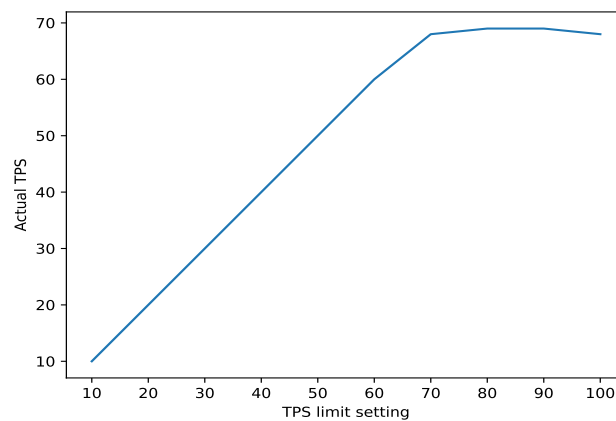


Figure 5. TPS limit of the performance evaluation system.

6.2. Simulation Experiments with the Existing Consensus Algorithm

We considered transaction type, account type, and CSC type as the input features of the KNN. The normalized values are from 0 to 1, where 0 is the highest priority and 1 is the lowest priority.

Without a KNN-based consensus algorithm, the transactions should be handled in a FIFO way. In this way, the transaction that arrived early will be served early. We generated 1000 transactions with different transaction types, account types, CSC types, and arrival times. Table 8 shows part of the transaction data. Algorithm 3 illustrates how the transaction pool picks up the transactions in a FIFO way. Correspondingly, Figure 6 shows a scatter diagram of the handled transactions.

Table 8. Samples of transactions.

| Transaction Type | Account Type | CSC Type | Arriving Time (ms) |
|------------------|--------------|----------|--------------------|
| 0.503 | 0.536 | 0.592 | 506 |
| 0.014 | 0.399 | 0.454 | 539 |
| 0.655 | 0.547 | 0.991 | 52 |
| 0.328 | 0.095 | 0.579 | 152 |
| 0.271 | 0.810 | 0.502 | 822 |
| 0.734 | 0.675 | 0.667 | 391 |
| ... | ... | ... | ... |
| 0.608 | 0.726 | 0.806 | 84 |
| 0.472 | 0.610 | 0.017 | 935 |
| 0.410 | 0.239 | 0.538 | 257 |
| 0.870 | 0.715 | 0.617 | 23 |

Algorithm 3 Transactions selected with the FIFO method.

Input: T_x_Table : A 2D array as the transaction table, one row presents one T_x ;

Output: Q : An FIFO Q with all transactions;

- 1: Set Q to empty;
- 2: **while** ($Q.size < T_x_MAX$) and (T_x_Table has unused transaction) **do** ▷ Keep picking up T_x from T_x_Table until Q is full or T_x_Table has no unused transaction
- 3: Get a new T_x from T_x_Table
- 4: Insert T_x to Q
- 5: **end while**
- 6: **return** Q

Figure 6 shows that, with the FIFO method, transactions that arrive earlier will be handled earlier, even if it has a lower priority classified by their attributes. The transaction start time is irrelevant to its priority, and the higher priority transaction will not be handled earlier.

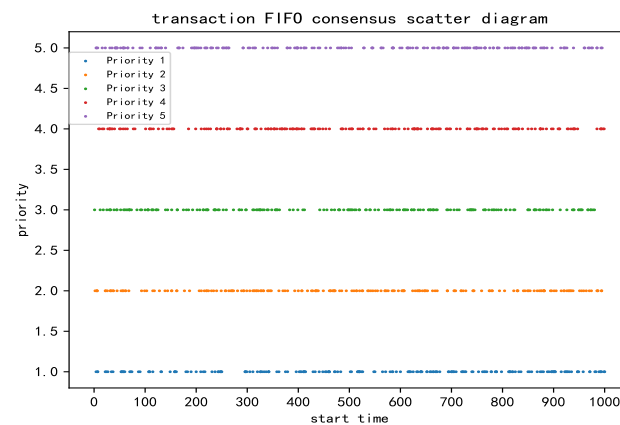


Figure 6. FIFO way transaction scatter diagram.

6.3. Simulation Experiments with the KNN-Enabled Enhanced Consensus Algorithm

6.3.1. First Round of k Value Selection

In this paper, Algorithms 1 and 2 are used to classify 1000 transactions into 5 priority queues using the KNN algorithm, in which the number of nearest neighbors, k is an important parameter. First, we apply Algorithm 4 to the selection of the best k value. It adopts the KNN classification using Scikit-learn in python. Generally, the dataset is split into a training set and a testing set. We then run the KNN classifier with different k values. The accuracy score is used to check the accuracy of our KNN model and the k value. The k value with the highest accuracy score should be selected as the best k value for handling unknown incoming transactions and checking its target priority.

Algorithm 4 KNN k value selection with a single training/test set split.

Input: *Training_Data*: The prepared transaction training data; *Training_Target*: The target priority of the prepared transaction training data; k : The k value used in KNeighborsClassifier function;

Output: *accuracy_score*: The accuracy score of the given k value;

- 1: $X_{train}, X_{test}, y_{train}, y_{test} = \text{train_test_split}(Training_Data, Training_Target, test_size = 0.3)$; ▷ The dataset is split into a training set and a testing set, and the testing set size is 30 percent
 - 2: $knn = \text{neighbors.KNeighborsClassifier}(k, weights = "distance")$ ▷ Prepare the KNN classifier by using the Scikit-learn module with the specified k value
 - 3: $knn.fit(X_{train}, y_{train})$ ▷ Fit the KNN classifier with the split training set
 - 4: $y_{prediction} = knn.predict(X_{test})$ ▷ Predict the target priority of the split testing set
 - 5: $accuracy_score = \text{accuracy_score}(y_{test}, y_{prediction})$ ▷ Check the accuracy of the target priority of the testing set
 - 6: **return** *accuracy_score*
-

We plotted the accuracy of different k values ranging from 1 to 20 in Figure 7. It demonstrates that $k = 12$ can achieve the highest accuracy among all k values. Therefore, we used $k = 12$ for KNN in all remaining experiments. A flowchart for classifying all 1000 transactions into the 5 priority queues once k is fixed is shown in Figure 8.

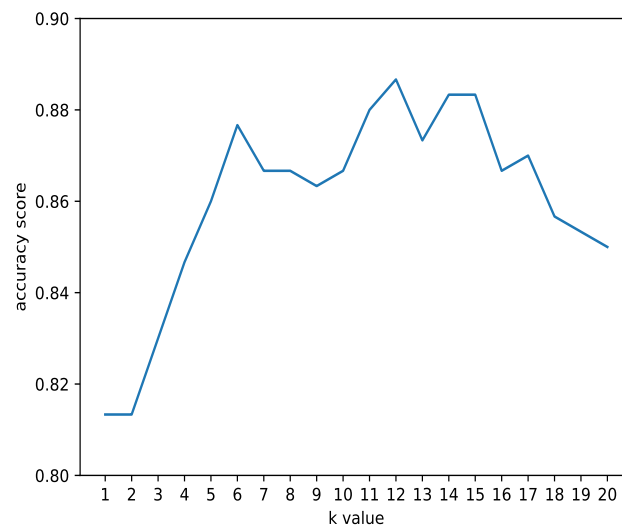


Figure 7. *k* value selection.

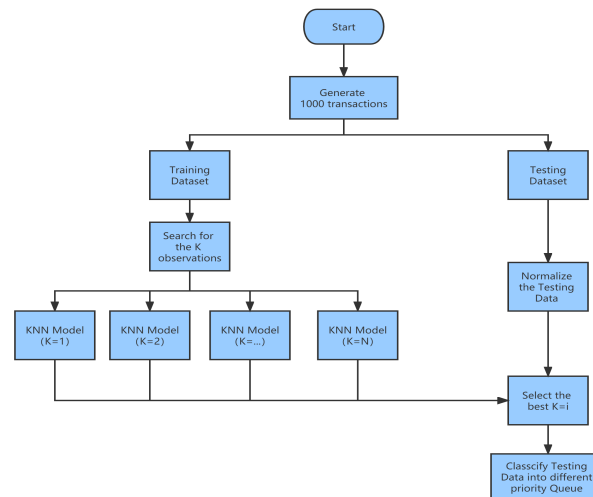


Figure 8. Transaction classification.

6.3.2. Choose a *k* Value in K-Fold cross-Validation

In Section 6.3.1, we presented the initial method for selecting the best value of *k*. However, is this the optimal *k* value? In the first round’s KNN *k* value, we only use a single training/test set split. The test set will only include a small portion of randomly selected data. In this scenario, the test set may not accurately represent “new unseen data”, which could lead to an overestimation of performance if it is used alone (due to potentially significant variability in the test results). By using cross-validation, all available data can be used for testing purposes, thereby ensuring that “bad” observations also play a role during the testing process. The train–test split and *k*-fold cross-validation are both examples of resampling methods in statistics. Resampling methods involve taking a sample from a dataset and using it to estimate unknown quantities. These techniques are particularly useful in machine learning and data analysis when a limited amount of data is available for model training and evaluation. The *k* value generated by using only one training/test set split will change due to the selection of the training/test set. We must use *k*-fold cross-validation to eliminate this effect, so we use the following algorithm to ensure that we consider all of the elements in the dataset. We finally obtain a *k* value of 4, as shown in Figure 9 below.

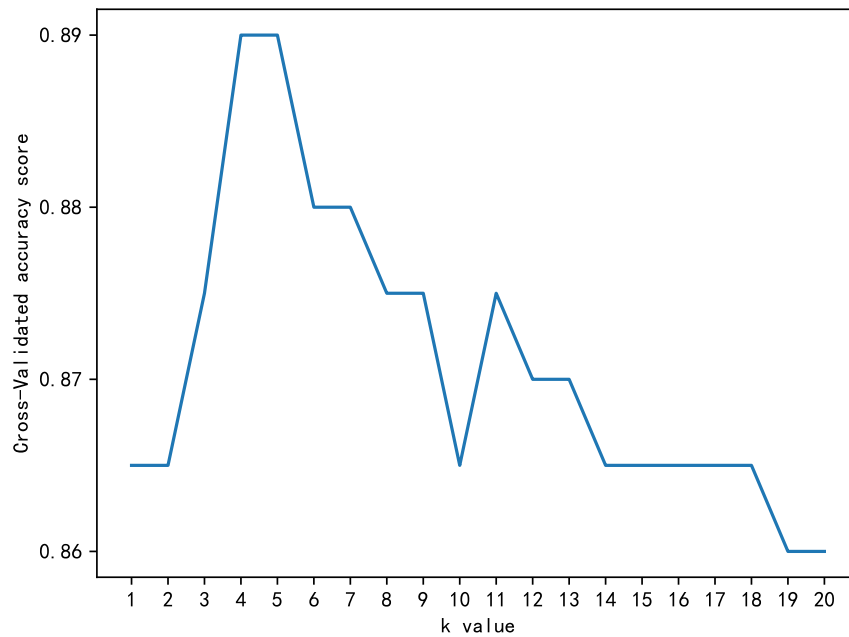


Figure 9. Cross-validated accuracy score scatter diagram.

6.3.3. Performance Optimization and Evaluation

When we obtain an optimal k value, we use 1000 transactions as the training set, and their layout is shown in Figure 10. In the figure, different categories of data in the training set data sometimes overlap (meaning that the categories of this part of the data are blurred). This part of the data will cause some model overfitting. Based on the learning curve in Figure 11, we know that there are still opportunities to optimize performance. One idea is to directly remove this part of the overlapping data, which is referred to as a clipping method.

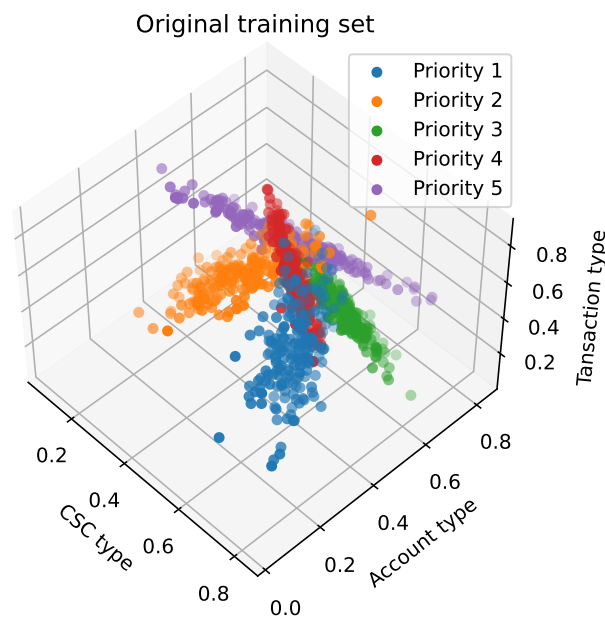


Figure 10. Original training set scatter diagram.

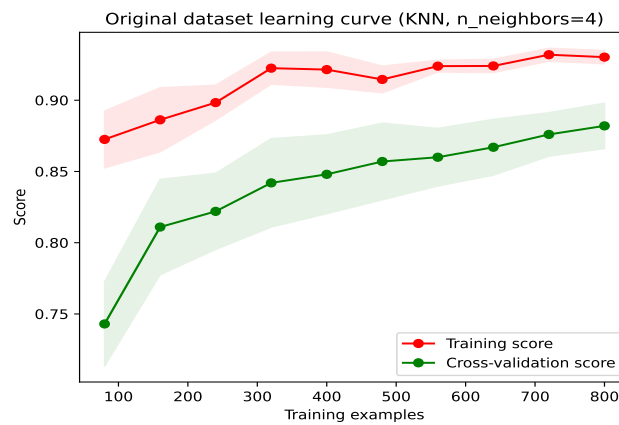


Figure 11. Original training set learning curve scatter diagram.

The clipping method randomly divides the training set, D , into two parts. One part is used as a new training set, and the other part is used as a test set. Based on the new training set, the KNN method is used to classify the test set, and the misclassified samples are removed from the entire training set. Since the division of the training set D is randomly divided, it is difficult to ensure that the samples in the overlapping part of the data will be eliminated in the first clip. After obtaining the new training set, the above operations can be repeated, and clearer class boundaries can be obtained. We can obtain its layout image (Figure 12) and learning curve (Figure 13), as shown below. Compared with the original training set, we achieved improved performance with a smaller size.

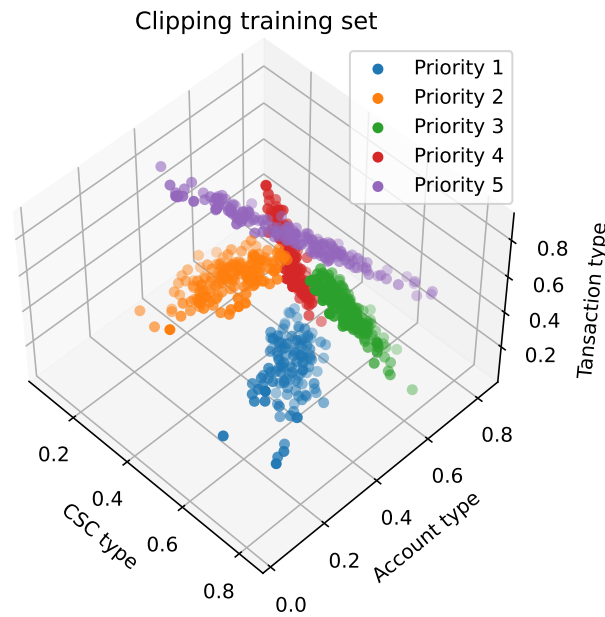


Figure 12. Clipping training set scatter diagram.

By observing the learning curve optimized by the clipping method, it can be seen that when the number of samples is around 300, it already has a good fitting performance. At the same time, as shown by the layout of samples in Figure 12, there are a large number of samples in the center of each class, indicating that we can reduce the size of the training set by compressing the KNN training set. The compressing method is used when a large number of samples of the same type are concentrated in the center of the cluster, and these concentrated samples have little effect on classification, so these samples can be discarded. The training set is divided into two parts in this method. The first part is a store that contains a portion of the samples, and the second part is a grab bag that contains the remaining samples. The store is used for the training set of the KNN model, and the grab bag is

used for the test set. The misclassified samples are moved from the grab bag to the store. The store continues to be used with increased samples, and the grab bag with decreased samples is used to train and test the KNN model again until all samples in the grab bag are correctly classified or until the number of samples in the grab bag is 0. After compression, the store keeps a portion of the randomly selected samples at initialization as well as the misclassified samples in each subsequent cycle. Since the clipping method removes all outliers, these selected misclassified samples are concentrated at the edge of the cluster and are considered correct samples with a large classification effect. The final training set is smaller. We can see its layout in Figure 14. The learning curve in Figure 15 shows that the training set still has a similar accuracy to that of the clipping training set.

Each transaction will be executed with its priority, and arrival time is only used when the transactions have the same priority. If two transactions have the same priority, the transaction that arrived earlier will be executed earlier. Table 9 describes the priority and new start time of each transaction based on its attributes.

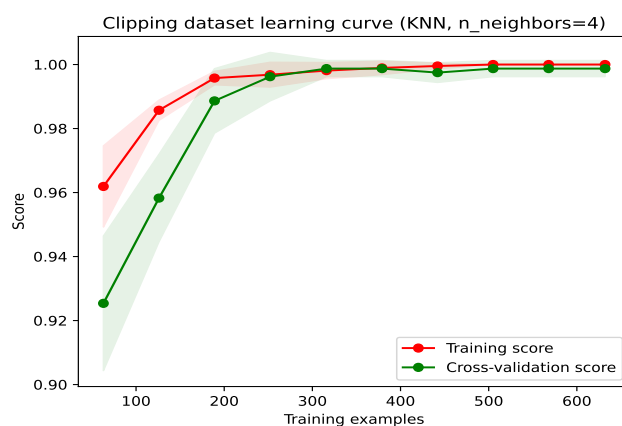


Figure 13. Clipping training set learning curve scatter diagram.

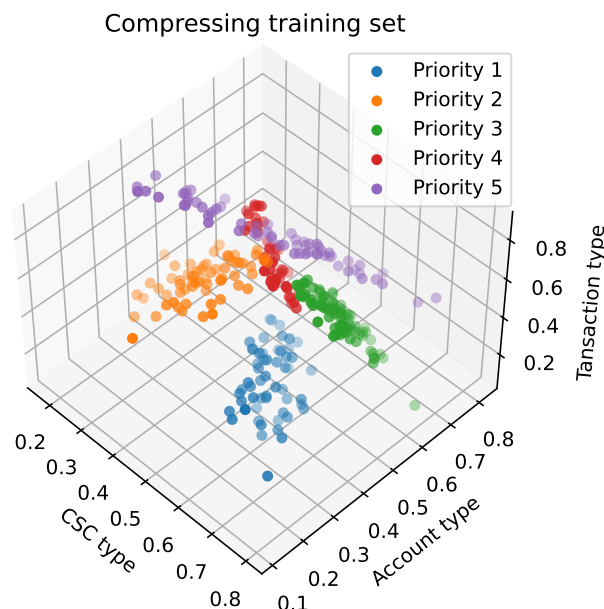


Figure 14. Compressing training set scatter diagram.

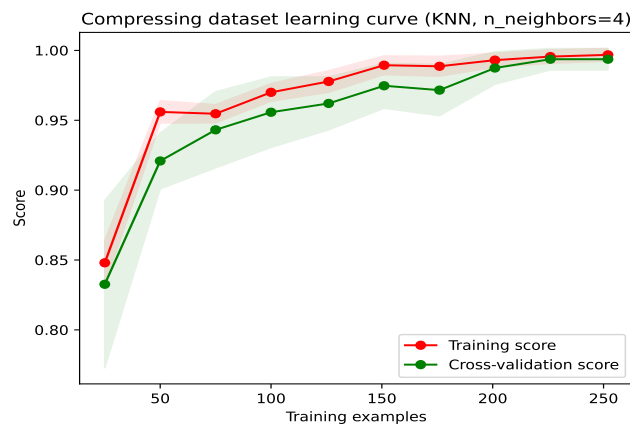


Figure 15. Compressing training set learning curve scatter diagram.

Table 9. Transactions with priority and new start time.

| Transaction Type | Account Type | CSC Type | Arrival Time (ms) | Priority | Start Time (ms) |
|------------------|--------------|----------|-------------------|----------|-----------------|
| 0.077 | 0.132 | 0.058 | 396 | 1 | 1 |
| 0.133 | 0.029 | 0.226 | 330 | 1 | 2 |
| 0.031 | 0.007 | 0.213 | 500 | 1 | 3 |
| 0.208 | 0.017 | 0.174 | 132 | 1 | 4 |
| 0.116 | 0.069 | 0.007 | 326 | 1 | 5 |
| ... | ... | ... | ... | ... | ... |
| 0.067 | 0.306 | 0.009 | 499 | 1 | 12 |
| 0.034 | 0.143 | 0.545 | 331 | 2 | 13 |
| 0.496 | 0.066 | 0.171 | 393 | 2 | 14 |
| 0.248 | 0.371 | 0.122 | 788 | 2 | 15 |
| ... | ... | ... | ... | ... | ... |

With the proposed KNN consensus algorithm, the scatter diagram of the transactions is shown in Figure 16, where 1 is the highest priority, and 5 is the lowest priority. Differently from the start time that only relates to the arrival time in the FIFO method, as shown in Figure 6, the start time with the KNN-based consensus algorithm relates to the priority of the transaction, which introduces the QoS method to the consensus algorithm and helps to better achieve SLA requirements and provide BaaS users an improved experience.

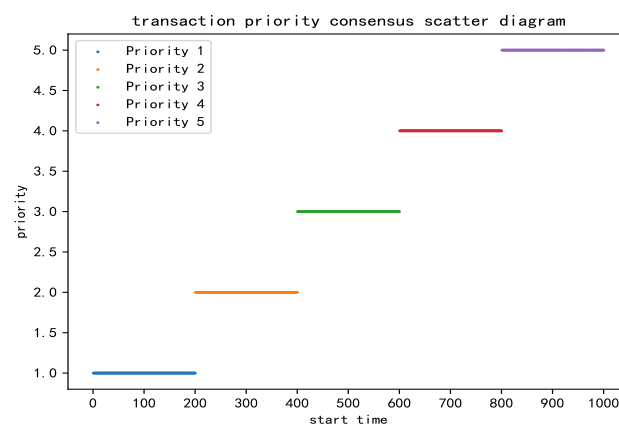


Figure 16. Prioritized transaction scatter diagram.

When a new transaction needs to be added to the transaction pool, it needs to be classified by the KNN algorithm. The prediction is only determined by the number of sample points in the training set, which is a constant value once the training set is finalized. The time complexity of this algorithm is $O(1)$, and the space complexity is also $O(1)$, which

is irrelevant to the number of transactions in the transaction pool. After adopting the clipping and compressing algorithms, the number of samples in the training set is greatly reduced while ensuring a good fitting performance. The given example reduces the number of samples from 1000 to 200+. Algorithm 5 describes how a new transaction is added to the priority queue with the new compressed training set.

Algorithm 5 New transaction classification.

Input: T_x : The new transaction; $Q_1 \dots Q_N$: The priority Q list from Q_1 to Q_N ; k : The best k in K-fold Cross-Validation *Training_data*: The compressed training data; *Training_target*: The compressed training target;

Output: Updated Q_n

- 1: Initial $KNN_clf = neighbors.KNeighborsClassifier(n_neighbors = k)$; ▷ Prepare KNeighborsClassifier with the best k value
 - 2: $KNN_clf.fit(Training_data, Training_target)$ ▷ Fit the compressed training data and target to the KNeighborsClassifier
 - 3: $predict = KNN_clf.predict(T_x)$ ▷ Predict the class of new incoming T_x
 - 4: **if** ($predict == n$) and ($Q_n.size < MAX_SIZE$) **then**
 - 5: PUSH T_x to Q_n ; ▷ PUSH T_x to Q_n once the Q isn't full
 - 6: **else**
 - 7: Save T_x to memory pool; ▷ Otherwise temporarily save the T_x to memory pool
 - 8: **end if**
 - 9: **return** Q_n
-

Compared with existing blockchain consensus algorithms, the proposed KNN-based consensus algorithm guarantees that higher priority transactions are executed earlier. Table 9 shows that the CSC type is important for calculating the priority. If a CSC has a short SLA requirement, its CSC type should be assigned with a high priority. This helps to deliver services to the CSCs within the SLA limitation in the BaaS system. Considering the transaction with attributes {0.034, 0.143, 0.545} in Table 9 as an example, its arrival time is 331. Without the proposed KNN-based optimization consensus algorithm, the transaction pool assigns it with a sequence number of 331. If the SLA of this transaction has a short duration, the transaction may miss the SLA. With the KNN-based consensus algorithm, however, it should be classified into a higher target priority queue. In this way, the transaction pool assigns it with a sequence number of 13 and, therefore, is more likely to satisfy the SLA.

7. Conclusions

Most existing consensus algorithms do not consider the priority. If a high-priority transaction comes late, it needs to wait until other, lower-priority transactions are handled. Due to the TPS limitation, it is difficult to meet SLA requirements in the BaaS system. This paper proposes a KNN-based consensus algorithm to enhance the SLA handling in the BaaS system. With the KNN-based consensus algorithm, each transaction is handled based on its priority. The transactions that arrive late but have high priority can be handled early. In this way, the BaaS system can better satisfy the SLA between the CSP and the CSC. The proposed KNN-based blockchain consensus algorithm is a common solution, and we only choose three attributes for classification. The experimental results illustrate the advantages of the proposed algorithm. In the future, we will consider more attributes for classification and try using other classification methods that can outperform the KNN.

Author Contributions: Conceptualization, Q.Z., L.W. and J.H.; formal analysis, Q.Z., L.W. and J.H.; investigation, Q.Z. and L.W.; methodology, Q.Z., L.W. and T.L.; project administration, L.W. and J.H.; resources, J.H. and T.L.; software, Q.Z., T.L. and L.W.; supervision, J.H.; validation, Q.Z. and T.L.; writing—original draft preparation, Q.Z., L.W. and J.H.; writing—review and editing, Q.Z., L.W. and T.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Education of Humanities and Social Science Project (grant no. 19YJAZH047), the Ministry of Science and Technology Innovation Method Work

Special Project (grant no.2017IM030100), Sichuan Provincial Higher Education Talent Training Quality and Teaching Reform Project (grant no. JG2021-995), Sichuan Provincial Higher Education Talent Training Quality and Teaching Reform Project (grant no. JG2021-1016), and the Social Practice Research for Teachers of Southwestern University of Finance and Economics (grant no. 2022JSSHSJ11).

Data Availability Statement: All the data in this paper are publicly available. Please contact the corresponding author to obtain them.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Zhang, C.; Chen, Y. A review of research relevant to the emerging industry trends: Industry 4.0, IoT, blockchain, and business analytics. *J. Ind. Integr. Manag.* **2020**, *5*, 165–180. [CrossRef]
2. Song, J.; Zhang, P.; Alkubati, M.; Bao, Y.; Yu, G. Research advances on blockchain-as-a-service: Architectures, applications and challenges. *Digit. Commun. Netw.* **2021**, *8*, 466–475. [CrossRef]
3. Buterin, V. A next-generation smart contract and decentralized application platform. *White Pap.* **2014**, *3*, 1–36. Available online: <https://ethereum.org/en/whitepaper/#a-next-generation-smart-contract-and-decentralized-application-platform> (accessed on 28 January 2023).
4. Peng, Z.; Xu, J.; Hu, H.; Chen, L. BlockShare: A Blockchain Empowered System for Privacy-Preserving Verifiable Data Sharing. *IEEE Data Eng. Bull.* **2022**, *45*, 14–24.
5. Wu, H.; Peng, Z.; Guo, S.; Yang, Y.; Xiao, B. VQL: Efficient and Verifiable Cloud Query Services for Blockchain Systems. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 1393–1406. [CrossRef]
6. Wang, H.; Xu, C.; Zhang, C.; Xu, J.; Peng, Z.; Pei, J. vChain+: Optimizing Verifiable Blockchain Boolean Range Queries. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia, 9–12 May 2022; pp. 1927–1940. [CrossRef]
7. Sayeed, S.; Marco-Gisbert, H. Assessing Blockchain Consensus and Security Mechanisms against the 51% Attack. *Appl. Sci.* **2019**, *9*, 1788. [CrossRef]
8. Akcora, C.G.; Gel, Y.R.; Kantarcioglu, M. Blockchain networks: Data structures of Bitcoin, Monero, Zcash, Ethereum, Ripple, and Iota. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2022**, *12*, e1436. [CrossRef] [PubMed]
9. Du, M.; Chen, Q.; Ma, X. MBFT: A New Consensus Algorithm for Consortium Blockchain. *IEEE Access* **2020**, *8*, 87665–87675. [CrossRef]
10. Li, D.; Deng, L.; Cai, Z.; Souiri, A. Blockchain as a service models in the Internet of Things management: Systematic review. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e4139. [CrossRef]
11. Samaniego, M.; Jamsrandorj, U.; Deters, R. Blockchain as a Service for IoT. In Proceedings of the 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, 15–18 December 2016; pp. 433–436. [CrossRef]
12. Ardagna, D.; Casale, G.; Ciavotta, M.; Pérez, J.F.; Wang, W. Quality-of-service in cloud computing: Modeling techniques and their applications. *J. Internet Serv. Appl.* **2014**, *5*, 1–17. [CrossRef]
13. Viriyasitavat, W.; Da Xu, L.; Bi, Z.; Hoonsopon, D.; Charoenruk, N. Managing qos of internet-of-things services using blockchain. *IEEE Trans. Comput. Soc. Syst.* **2019**, *6*, 1357–1368. [CrossRef]
14. Tan, W.; Zhu, H.; Tan, J.; Zhao, Y.; Xu, L.D.; Guo, K. A novel service level agreement model using blockchain and smart contract for cloud manufacturing in industry 4.0. *Enterp. Inf. Syst.* **2022**, *16*, 1939426. [CrossRef]
15. Rashid, A.; Chaturvedi, A. Cloud computing characteristics and services: A brief review. *Int. J. Comput. Sci. Eng.* **2019**, *7*, 421–426. [CrossRef]
16. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
17. Kshetri, N. The Economics of Central Bank Digital Currency [Computing’s Economics]. *Computer* **2021**, *54*, 53–58. [CrossRef]
18. Yang, G.; Lee, K.; Lee, K.; Yoo, Y.; Lee, H.; Yoo, C. Resource Analysis of Blockchain Consensus Algorithms in Hyperledger Fabric. *IEEE Access* **2022**, *10*, 74902–74920. [CrossRef]
19. Peng, Z.; Zhang, Y.; Xu, Q.; Liu, H.; Gao, Y.; Li, X.; Yu, G. NeuChain: A Fast Permissioned Blockchain System with Deterministic Ordering. *Proc. VLDB Endow.* **2022**, *15*, 2585–2598. [CrossRef]
20. Sanka, A.I.; Cheung, R.C. Efficient High Performance FPGA based NoSQL Caching System for Blockchain Scalability and Throughput Improvement. In Proceedings of the 2018 26th International Conference on Systems Engineering (ICSEng), Sydney, NSW, Australia, 18–20 December 2018; pp. 1–8. [CrossRef]
21. Gao, S.; Peng, Z.; Tan, F.; Zheng, Y.; Xiao, B. SymmeProof: Compact Zero-Knowledge Argument for Blockchain Confidential Transactions. *IEEE Trans. Dependable Secur. Comput.* **2022**, *1*. [CrossRef]
22. Cai, C.; Xu, L.; Zhou, A.; Wang, C. Toward a Secure, Rich, and Fair Query Service for Light Clients on Public Blockchains. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 3640–3655. [CrossRef]

23. Ruan, P.; Chen, G.; Dinh, T.T.A.; Lin, Q.; Ooi, B.C.; Zhang, M. Fine-Grained, Secure and Efficient Data Provenance on Blockchain Systems. *Proc. VLDB Endow.* **2019**, *12*, 975–988. [[CrossRef](#)]
24. Peng, Z.; Xu, J.; Chu, X.; Gao, S.; Yao, Y.; Gu, R.; Tang, Y. Vfchain: Enabling verifiable and auditable federated learning via blockchain systems. *IEEE Trans. Netw. Sci. Eng.* **2021**, *9*, 173–186. [[CrossRef](#)]
25. Onik, M.M.H.; Miraz, M.H. Performance Analytical Comparison of Blockchain-as-a-Service (BaaS) Platforms. In *Emerging Technologies in Computing*; Miraz, M.H., Excell, P.S., Ware, A., Soomro, S., Ali, M., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 3–18.
26. Samet, H. K-nearest neighbor finding using MaxNearestDist. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *30*, 243–252. [[CrossRef](#)]
27. Martínez, F.; Frías, M.P.; Pérez-Godoy, M.D.; Rivera, A.J. Dealing with seasonality by narrowing the training set in time series forecasting with kNN. *Expert Syst. Appl.* **2018**, *103*, 38–48. [[CrossRef](#)]
28. Li, T.; Qian, Z.; Deng, W.; Zhang, D.; Lu, H.; Wang, S. Forecasting crude oil prices based on variational mode decomposition and random sparse Bayesian learning. *Appl. Soft Comput.* **2021**, *113*, 108032. [[CrossRef](#)]
29. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Wang, R. Efficient kNN Classification With Different Numbers of Nearest Neighbors. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 1774–1785. [[CrossRef](#)] [[PubMed](#)]
30. Fix, E. *Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties*; USAF School of Aviation Medicine: Dayton, OH, USA, 1985; Volume 1.
31. Mahfouz, M.A.; Shoukry, A.; Ismail, M.A. EKNN: Ensemble classifier incorporating connectivity and density into kNN with application to cancer diagnosis. *Artif. Intell. Med.* **2021**, *111*, 101985. [[CrossRef](#)] [[PubMed](#)]
32. Song, Z.H.; Sang, W.J.; Yuan, S.Y.; Wang, S.X. Gas-Bearing Reservoir Prediction Using k-nearest neighbor Based on Nonlinear Directional Dimension Reduction. *Appl. Geophys.* **2022**, 1–11. [[CrossRef](#)]
33. Cui, L.; Zhang, Y.; Zhang, R.; Liu, Q.H. A Modified Efficient KNN Method for Antenna Optimization and Design. *IEEE Trans. Antennas Propag.* **2020**, *68*, 6858–6866. [[CrossRef](#)]
34. Castro, M.; Liskov, B. Practical byzantine fault tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, LA, USA, 22–25 February 1999; Volume 99, pp. 173–186.
35. FISCO BCOS Platform. Available online: <https://github.com/fisco-bcos> (accessed on 28 January 2023).
36. Abu Alfeilat, H.A.; Hassanat, A.B.; Lasassmeh, O.; Tarawneh, A.S.; Alhasanat, M.B.; Eyal Salman, H.S.; Prasath, V.S. Effects of distance measure choice on k-nearest neighbor classifier performance: A review. *Big Data* **2019**, *7*, 221–248. [[CrossRef](#)]
37. Wettschereck, D.; Aha, D.W.; Mohri, T. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artif. Intell. Rev.* **1997**, *11*, 273–314. [[CrossRef](#)]
38. FISCO BCOS Performance Demo Program. Available online: <https://github.com/FISCO-BCOS/java-sdk-demo/blob/main/src/main/java/org/fisco/bcos/sdk/demo/perf/PerformanceOk.java> (accessed on 28 January 2023).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.