Hindawi
Scientific Programming
Volume 2022, Article ID 6129150, 14 pages
https://doi.org/10.1155/2022/6129150

Research Article

Optimized Bandwidth Allocation for MEC Server in Blockchain-Enabled IoT Networks

Shengcheng Ma,[1] **Wei-Tek Tsai**,[1,2] **Shuai Wang**,[1] **Yan Liang**,[3] **and Dong Yang**[1]

[1]*Beihang University, No. 37 Xueyuan Road, Beijing 100191, China*
[2]*Digital Society & Blockchain Laboratory, No. 37 Xueyuan Road, Beijing 100191, China*
[3]*Beijing Information Science and Technology University, No. 35 the 4th Ring Middle Road, Beijing 100101, China*

Correspondence should be addressed to Shengcheng Ma; mashengcheng@163.com

Received 22 March 2022; Revised 20 April 2022; Accepted 29 April 2022; Published 16 May 2022

Academic Editor: Jiwei Huang

Powered by the development of the fifth-generation mobile communication technology (5G), the Internet of things (IoT) has been widely applied in people's life. Due to the limitation of storage and computing power, the data transmission of IoT devices faces challenges in terms of security and privacy. Therefore, many researchers provide the conjunction of blockchain and mobile edge computing (MEC) to make up for the lack of computing and security. MEC can meet the storage and computing requirements for IoT devices. Blockchain can provide a decentralized, antitamper solution that can help devices overcome security deficiencies, whereas the speed of blockchain communication is not fast enough because of the consensus mechanism. In this article, we focus on the permissioned blockchain and propose an optimized bandwidth allocation algorithm to promote the performance of consensus communication. The algorithm contains an In-Network control ideology and supports deployment on MEC servers. Deep reinforcement learning (DRL) is employed to perform the computation of available bandwidth in our scheme. We implement a prototype system in the testbed and perform a simulation, and the results show the advantages compared with the current widely used algorithm. By applying our method, the Internet of things devices can transmit data safely and efficiently.

1. Introduction

With the increasing popularity of the 5G, the IoT has been widely applied in people's life. 5G network provides infrastructure for the application of IoT, such as smart city [1], Internet to Vehicle (IoV) [2], and health monitoring [3]. However, the data generated by the Internet of things devices has security and privacy challenges. The devices only have limited storage and computing capability, which makes it difficult to ensure the safety of the data transmission.

To meet these challenges, a fusion paradigm of blockchain and MEC is proposed to strengthen the transmission of IoT [4, 5]. Blockchain can improve the security for IoT devices, and MEC offers the computing and storage resources for blockchain network. MEC is a distributed system deployed at the edge of network to process the offloading tasks from the mobile devices. Applying MEC can bring many benefits, including decreased transmission delay [6], provided more computing power [7], improved user experience [8], and reduced the energy cost [9]. Because of sinking the computing and storage resources close to the end devices, it is possible to improve the security of IoT data transmission by using blockchain.

Blockchain as a distributed network with decentralized ledger is widely used in many scenarios such as finance, e-business, and IoT. Every node on the blockchain reaches a consensus for every transaction, and each block is dependent on the previous one in a cryptographic hash way. All blocks are linked by hash timestamp like a chain. Attributed to these structures and organizations, blockchain could prevent data from being tampered with [10] and provide transactions with a traceable path [11].

Although blockchain brings many benefits to applications, it consumes huge resources to ensure data consistency.

All peers need to synchronize the information when a new block is generated. This will lead to communications between all nodes on the blockchain. In particular the classic Byzantine Fault Tolerance (BFT) consensus algorithm used by the permissioned blockchain causes too many communications among the peers [12]. When the number of peers on the blockchain is N, the communication boundary of this algorithm will be $O(N^2)$ [13]. Therefore, the performance of consensus in the blockchain is an urgent problem to solve.

To improve the performance of blockchain, many platforms attempt to optimize the process of the transaction. Some platforms add a database layer above the blockchain to speed up the operation of transactions [14]. And some researchers try to simplify the consensus mechanism to reduce the calculated workload [15]. However, these methods sacrifice parts of consistency to improve performance.

In this paper, we focus on the permissioned blockchain which employs BFT based consensus mechanism and propose an optimized bandwidth allocation strategy to improve the performance. The permissioned blockchain usually does not use Proof of Work (PoW) [16] or Proof of Stake (PoS) [17] as the consensus protocol. Therefore, the transaction process in this kind of blockchain depends more on the network. The performance of the network can directly affect the performance of the blockchain. Then it affects the data transmission of IoT network. In addition, the permissioned blockchain can be customized, so the deployment of the permissioned blockchain is easier than public blockchain on MEC servers. Hence, we consider the permissioned blockchain as the target scenarios and optimize the network to improve the performance of BFT consensus communication.

Our main contributions of this paper can be summarized as follows:

(i) First, we design blockchain platform in the MEC servers which can strengthen the IoT communication. And we analyze some current bandwidth congestion control algorithms and provide an available bandwidth notification (ABN) algorithm. The ABN algorithm combines the advantages of Bottleneck Bandwidth and Round-trip propagation time (BBR) and explicit congestion notification (ECN).

(ii) Second, according to the different deployment methods, we consider the application scenarios of private network and public network, respectively, and use the deep reinforcement learning method to solve the bandwidth allocation method in the public network with the competition.

(iii) Third, our prototype has run in the testbed and the results show the improvement compared with current widely used techniques.

The rest of this paper is organized as follows: Section 2 summarizes related work of blockchain optimizations. Next, we present a blockchain-enabled IoT network model to optimize the performance of consensus communication in Section 3. In particular, we illustrate the architecture and implementation of our new method in this section. Then, we provide the solution of bandwidth allocation for different scenarios in Section 4. In Section 5, we introduce the experimental environment and analyze the result to prove the improvement of performance. Finally, we conclude this paper in Section 6.

2. Related Work

2.1. Cost of Blockchain. The discussion concerning the cost of blockchain has always been a hot topic. Catalini and Gans consider the verification cost and network cost by blockchain technology from an economics perspective [18]. Sukhwani et al. use Stochastic Reward Nets (SRN) to simulate the PBFT consensus process [19] and investigate why the process could be a performance bottleneck in the network with a large number of peers. In their SRN model, conclusion is that the transmission delays can impact the average time to consensus. Marko from IBM discusses PoW and BFT blockchains focusing on overcoming and scalability [20]. In his conclusion, the performance of PoW-based blockchain depends on the computational power and scale of nodes.

In addition, scalability is a weakness in this kind of blockchain. A review on the cost of IoT application based on blockchain is presented in [4], which concludes that the reduction in overhead is carried out by removing the PoW consensus mechanism and choosing the right cryptographic scheme.

Sekaran et al. research the integration of blockchain and IoT technologies [21]. By addressing the shortcomings and limitations of IoT, they summarized the state of the art of high-level solutions. Li et al. propose a blockchain-based data security scheme for IoT networks in 6G [22]. The fusion technology of AI and blockchain is developed to evaluate and optimize the quality of service in IoT networks.

2.2. Attempts to Promote the Performance of Blockchain. Many researchers and companies have made efforts to improve performance and reduce costs in the blockchain. Some researchers analyze the relationship between performance and storage structure, and they change the size of block or database to optimize the blockchain performance [23, 24]. Sharding is an interesting technology that can promote the parallel processing capability of the blockchain. Choosing an optimal shard size or employing a suitable sharding protocol can maximize performance and improve scalability [25, 26].

2.3. Optimization from the Perspective of Network. With respect to performance optimization for blockchain, the consensus algorithm evolves many branches. As an infrastructure, the increased network throughput will benefit all types of blockchain. Many researchers applied Artificial Intelligence (AI) to optimize the performance for IoT network. Huang et al. [27] studied the federated learning with the Matrix Factorization method and optimized the recommendation system for IoT network. Chen et al. [5] dealt

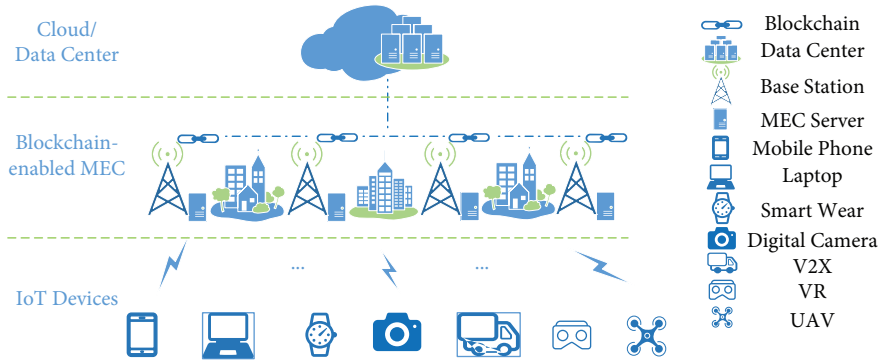

Figure 1: Blockchain-enabled IoT networks.

with task offload problem and used DRL method in MEC to improve the performance of IoT. The authors of [28] improved the performance of blockchain by enabling distributed NFV-MANO framework. Huang et al. proposed the joint admission control and computation resource allocation in the MEC server [29]. Their method reached the balance tradeoff between the utility and the queue length in small cell network. Huang's team proposed a BoR (Blockchain over RDMA) framework that can reduce the consuming time when a new node joins the blockchain network [30]. However, these methods need to add some hardware to ensure network achievement. This will lead to a sharp rise in network deployment costs. Jointly considering the cost and compatibility, we plan to optimize the congestion control of TCP in network to enhance the performance.

3. System Model

To illustrate our system, we first describe the network application scenarios. We classify the network flows according to the requirements in the scenarios. Then we build a model to represent the system operation mechanism and provide an optimal goal for blockchain consensus communication. Finally, we use the deep reinforcement learning method to optimize the bandwidth for blockchain business and introduce how feedback mechanism can avoid congestion.

3.1. Network Application Scenarios. We consider a blockchain-enabled IoT network, which consists of three layers, the IoT layer, the blockchain and MEC layer, and the cloud/data center layer. As shown in Figure 1, permissioned blockchain platform is deployed on the MEC servers, because MEC servers provide sufficient storage and computing capacities. A typical communication in this scenario is that the IoT devices transmit data to cloud server or data center via MEC servers. MEC servers deal with a large amount of offloading task from IoT devices. To enhance the security of data from IoT, blockchain collects the data as transactions from MEC servers. All blockchain nodes communicate with each other and participate in consensus. After the consensus is achieved, the transaction will be uploaded to blockchain as a new block. Then, the IoT data is protected against tampering.

To simplify the problem, we define three types of network elements according to network behavior in our system. *Sender*. The network node that sends data is defined as the sender. It is similar to a mobile phone in an IoT network. *Receiver*. The network node receiving data becomes the receiver, such as data center or cloud service. *Middlebox*. The network node that forwards data and has a buffer capacity is called middleboxes. It resembles the MEC servers in this scenario. This scenario can be shown as Figure 2.

The sender and the receiver can be converted to each other according to the data transmission direction. In Figure 2, senders sends data to receiver through middlebox and maybe many other middleboxes. This scenario can represent all network transmission forms.

In this scenario, the efficiency of data transmission is subject to these three types of nodes. The sender's congestion control strategy affects the sending speed. The middlebox's forwarding speed and buffer length determine whether it will become a bottleneck in the path. The receiver's feedback information ACK controls the sender's available windows, and it also decides whether retransmission is needed. Moreover, different types of data flows will also influence the utilization of transmission.

3.2. Network Flow Categories. According to the characteristics of connection, we classify network flows into three categories. The first type is fat flow; it usually is used for a large amount of data transmission with high bandwidth, for example, video streaming, large file download, and so on. The second type is slim flow; it is used for applications that send fewer data and close quickly. The most typical instance is web browsing and blockchain consensus communication. The third type is control flow; it is the ACK from receiver to sender. These three types of flows share the bandwidth of the network.

Each direction in middlebox has a specific bandwidth. The middlebox locates in a critical position when many flows go through it. It can be a bottleneck if the data inflow is greater than its processing capacity. Normally, the forward performance of the middlebox limits the receiving ability. If a great number of flows coming from senders fill the buffer of the middlebox, the effective bandwidth of the middlebox

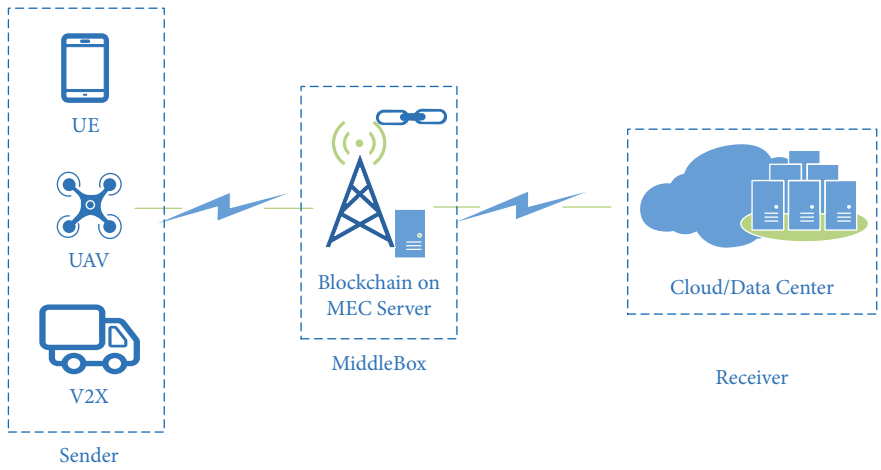

Figure 2: Three types of network elements.

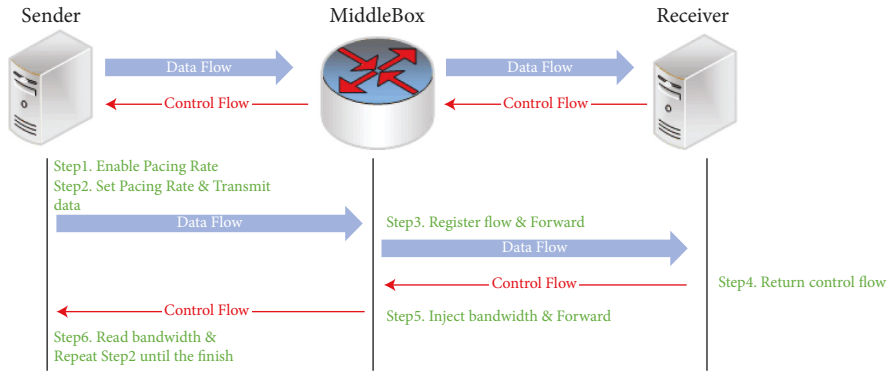

Figure 3: The process of available bandwidth notification.

will be exhausted. Then, congestion will form and delay will increase.

In order to avoid congestion, we propose a new method called available bandwidth notification.

3.3. Available Bandwidth Notification. Our available bandwidth notification is different from the existing methods. The traditional network is the end-to-end model. The senders do not understand the situation in the network. BBR's estimates of the fixed bandwidth and min RTT to operate congestion are like a guess [31]. Even the ECN method only tells the sender that congestion happens [32]. The sender does not know what speed is suitable, so it decreases the congestion window to the lowest value. The Software Defined Network (SDN) can organize the network topology and learn about the status of the network, but it needs a controller device and the standards of various manufacturers have not yet been unified. The SDN has not been widely used in many realistic production environments. Our method is designed to deploy in the traditional network for wider applicability; furthermore, it breaks through the end-to-end mode.

In our ABN method, we let middleboxes participate in congestion control. Middlebox can sense the real condition of the network in the self-system. All types of flows go through the middlebox, and the bandwidth utilization and queueing status can be read by the system of the middlebox. Then, the middlebox can calculate the most appropriate bandwidth value according to this information. With the help of the control flow returned from the receiver, the bandwidth value is put into control flow by the middlebox and delivered to the sender. When the sender receives the control flow, it will adjust the send window and sequence number by ACK information. Besides, it will set its pacing rate according to the value notified by the middlebox. In this way, the sender can adjust the sending speed in time conforming with the frequency of ACK return. The whole process of our available bandwidth notification method is shown in Figure 3. The details of operation steps are as follows.

(i) Step 1. The sender enables the pacing rate system setting. That means the sender will transmit data at a smooth setting speed without burst.

(ii) Step 2. The sender reads the pacing rate and transmits data. The pacing rate will be set to the initial value.

(iii) Step 3. The middlebox senses the data flow from the sender and forwards the data flow to the receiver. At

the same time, the middlebox registers the flow information for available bandwidth calculation.

(iv) Step 4. The receiver accepts the data flow and returns the ACK information by the control flow.

(v) Step 5. The middlebox searches the registration information according to the five-tuple of the control flow and finds out which one is the corresponding sender. It injects the available bandwidth value into control flow after calculation, and it forwards the flow to the sender.

(vi) Step 6. The sender receives the control flow and parses the available bandwidth value from it and then repeats the operation in Step 2. The sender sets the pacing rate according to the bandwidth value and continues to send until the transmission task is accomplished.

After such a series of operations, the sender can reasonably arrange the sending speed according to the actual situation in the network. Because the situation of the network is as feedback from the devices inside the network, the devices inside the network participate in congestion control, which breaks the end-to-end mode. Next, we will analyze how the devices in the network, the middlebox, calculate the available bandwidth.

3.4. Bandwidth Allocation Model. The total bandwidth of a middlebox is a fixed value. This is determined by the service of ISP (Internet Service Provider) and the performance of equipment itself. In this fixed bandwidth, there is a part of the traffic that belongs to non-TCP protocol. It is not affected by the congestion avoidance algorithm, so our method ignores the bandwidth of this part of the data flow.

According to our previous classification of flow, control flow, fat flow, and slim flow all belong to TCP protocol. Although the control flow belongs to TCP protocol, the timely transmission of control flow is very important to restore the sender's window which improves the efficiency of the whole network. Therefore, this kind of flow should be given high priority to obtain the required bandwidth.

After removing the bandwidth occupied by non-TCP traffic and control flow, the remaining bandwidth of the middlebox is shared by fat flows and slim flows competitively. Our method should reasonably allocate the remaining bandwidth, calculate a suitable bandwidth for senders, and ensure that slim traffic representing blockchain communication gets better service.

In our bandwidth allocation model, the whole bandwidth of the middlebox is defined as B_{all}. The non-TCP traffic obtains the bandwidth presented as G, and the control flow's bandwidth is presented as C. Excluding these two parts of bandwidth, the actual bandwidth that can be allocated is B. And B can be derived as (1).

$$B = B_{\text{all}} - G - C. \tag{1}$$

In the middlebox, the number of flows changes with time. Combined with the strategy of middlebox adjustment, we define a timer t_i to represent the period of bandwidth adjustment, and $i \in \{1, \ldots, N\}$. In order to obtain a steady-state of the system operation, we define the observation time as T, and it contains N timers which can be expressed as

$$T = \sum_{i=1}^{N} t_i. \tag{2}$$

When a new flow comes in, the middlebox will register the information of this flow on the flow table. After the timer t_i expires, the middlebox checks the flow table. If there is no data belonging to a flow, it is considered that this flow already has been processed. Consequently, the total number of flows stored in the flow table will be reduced by the number of processed flows. The detail of the flow table is shown in Table 1.

Table 1: Flow information table.

Item	Description
src_ip	Source IP address
dst_ip	Destination IP address
src_port	Source port
dst_port	Destination port
Timer	Number of occupied timer cycles
data_inflow	The variable shows data inflow
Rate	The real transmit rate of flow
Count	The number of flows

Based on the previous classification, the B bandwidth is allocated for fat flow and slim flow. We denote fat flow as a set $F_j = \{m_j, d_j, r_j\}$, $j \in \{1, \ldots, N^F(t)\}$, and $N^F(t)$ is the total number of fat flows through the middlebox in time phase t. It represents a single large amount of data transmission in the middlebox. In this set, m_j denotes the whole data size needed to transmit in flow F_j, d_j denotes the maximum tolerable time to complete the transmission, and r_j denotes the reward for completing transmission of this flow, which means the system should finish the work of the j th flow within d_j time.

Similar to the fat flow, the slim flow is defined as $S_k = \{m_k, d_k, r_k\}$, $k \in \{1, \ldots, N^S(t)\}$, where m_k is the data size of the flow S_k, d_k is the time requirement of this flow, and r_k is the reward of finishing this flow. These variables represent the same meaning as fat flow. The $N^S(t)$ is the total number of slim flows through the middlebox in time slot t. According to the actual situation, the value of data size m_k in the slim flow may be substantially smaller than in the fat flow.

For our method in the middlebox, the most critical work is to allocate bandwidth to each flow. The bandwidth allocated to the flow will change with the timer, so we consider the bandwidth of the flow as a function. We denote $B_j^F(t)$ as the bandwidth which allocated for fat flow F_j during phase t. The same to the slim flow, we use $B_k^S(t)$ to represent the bandwidth of slim flow S_k in time slot t.

The job of our approach is to find the available bandwidth for all flows through the middlebox. The available bandwidth for each flow should meet the processing time requirement and be limited by the whole actual bandwidth

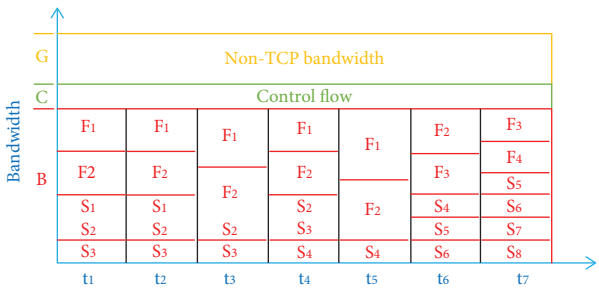

Figure 4: Bandwidth allocation for each flows.

of the middlebox B. At time slot t_i, we assume that the middlebox has $N^F(t_i)$ fat flows and $N^S(t_i)$ slim flows. The total fat flow's bandwidth can be denoted as $\sum_{j=1}^{N^F(t_i)} B_j^F(t_i)$, and all the slim flow's bandwidth can be denoted as $\sum_{k=1}^{N^F(t_i)} B_k^S(t_i)$. The holistic bandwidth of the flows can not be beyond the B of the middlebox, which can be represented by

$$\sum_{j=1}^{N^F(t_i)} B_j^F(t_i) + \sum_{k=1}^{N^S(t_i)} B_k^S(t_i) \leqslant B. \tag{3}$$

The timer of bandwidth regulation is a fixed value, so every time period is the same in our system. We assume that a slim flow S_k has completed transmission in X_k^S time periods, and the bandwidth allocated for S_k in each time slot t_i is denoted as $B_k^S(t_1), B_k^S(t_2), \ldots B_k^S(t_{X_k^S})$.

We make an instance to explain our model for bandwidth allocation. As shown in Figure 4, there are two fat flows (F_1, F_2) and three slim flows (S_1, S_2, S_3) in slot time t_1. We can derive that $N^F(t_1) = 2$ and $N^S(t_1) = 3$ in phase t_1. The height of each block represents their bandwidth at time t_1, so the height of S_1 equals $B_1^S(t_1)$ in time t_1. For example, in Figure 4, the slim flow S_1 takes two time phases to finish the data transmission; hence $X_1^S = 2$ in this flow.

For a data flow, the time to complete all the data transmission must be less than the required time. Bandwidth multiplied by time represents the amount of data that can be transmitted during this period of time. Then in the whole process, the total processing capacity of allocated bandwidth for slim slow S_k can be expressed as

$$P_{S_k} = \sum_{i=1}^{X_k^S} B_k^S(t_i) \cdot t_i, \tag{4}$$

where P_{S_k} is the total data size that the allocated resource can handle. P_{S_k} should be larger than the required data size of slim flow, which can be represented as

$$P_{S_k} \geqslant m_k. \tag{5}$$

The total processing time should be less than the required time of slim flow, which can be represented as

$$\sum_{i=1}^{X_k^S} t_i \leqslant d_k. \tag{6}$$

In order to optimize blockchain communication, we use slim flow to represent the network traffic in blockchain applications. Under these constraints, the object of our bandwidth allocation strategy is to serve as many slim flows as possible. In our model, we assume that the system will get a certain reward when a flow task completes. In the T period of observation, the situation of completed flow can be denoted as a Boolean array $I = \{1, 1, 1, 0, \ldots, 0\}$, and the rewards of all fat flows R^F can be represented as

$$R^F = \sum_{j=1}^{M} r_j \cdot I_j^F, \tag{7}$$

where M is the total number of fat flows that go through the middlebox during T time, I_j^F is the status of fat flow F_j, the situation $I_j^F = 1$ means the flow has been transmitted, and the situation $I_j^F = 0$ means the flow has not been served or has not finished. The rewards of all slim flows R^S can be represented as

$$R^S = \sum_{k=1}^{L} r_k \cdot I_k^S, \tag{8}$$

where L is the total number of fat flows, and the service status of slim flows can be represented by I^S.

The object of our model is to maximize rewards by allocating bandwidth to serve more data flows. By integrating all the limited conditions, our bandwidth allocation model can be defined as follows:

$$\begin{aligned}
&\mathbf{P1}: \max R^F + R^S, \\
&s.t.\ C1: P_{F_j} \geqslant m_j, \\
&\quad C2: P_{S_k} \geqslant m_k, \\
&\quad C3: \sum_{i=1}^{X_j^F} t_i \leqslant d_j, \\
&\quad C4: \sum_{i=1}^{X_k^S} t_i \leqslant d_k, \\
&\quad C5: \sum_{j=1}^{N^F(t_i)} B_j^F(t_i) + \sum_{k=1}^{N^S(t_i)} B_k^S(t_i) \leqslant B,
\end{aligned} \tag{9}$$

where $P1$ represents the goal of our model which means the rewards from fat flows and slim flows. Constraints $C1$ and $C2$ represent that the amount of transmission capability provided by allocated bandwidth should be greater than the amount of data required by the flow itself. $C1$ is for fat flows, and $C2$ is for slim flows. Constraints $C3$ and $C4$, respectively, represent the cost time of flow transmission within the required time. The constraint $C5$ shows the bandwidth division. In every certain time slot, the sum of the bandwidth of fat flows and slim flows should be less than the actual available bandwidth of the middlebox.

The design of this model considers blockchain communication in a general network environment. In the shared network environment, blockchain data flows compete with other network flows for bandwidth. We use slim flow to represent blockchain communication in our model, because

slim flow conforms to the properties of consensus communication in blockchain applications. It is a type of frequent data communication with small bandwidth. The communication of blockchain needs to obtain enough bandwidth resources to ensure the timeliness of consensus. Consequently, we define a fat flow to simulate the competition between other network flows and blockchain data flows in our model.

4. Solution and Optimization for Blockchain Communication

In order to optimize the performance of blockchain communication, we propose a model of available bandwidth allocation. By solving this model, we can accelerate the speed of blockchain network traffic. For our bandwidth allocation model, we have two types of network environments to consider. The first case is the private network, where the consensus nodes of blockchain are connected by private networks. The second case is the public network, where the data of the blockchain is transmitted in a competitive environment. We will analyze these two cases respectively.

4.1. Private Network. For the private network, all the network resources are used for blockchain services. It usually is deployed by consortium blockchain or private blockchain.

In this type of network, all the data flows are blockchain communications. It means that these flows do not need to share bandwidth with other services. In our model, if there are only slim flows, this situation can be represented. All the slim flows should guarantee quality of service. The rewards of these flows are the same, and the bandwidth is exclusively used by slim flows without the competition of fat flow.

The solution of resource allocation in a private network can be simplified as average allocation. The average allocation method is to equally allocate the current network bandwidth to the flows through the middlebox.

Although it is an average allocation, there are still three factors that can change the allocated bandwidth. The first factor is the incoming of the new flows in the middlebox. The second factor is that the leaving flows after the transmission are finished. The third factor is the urgency of flow that needs to complete the service in time.

For the first factor, the flow information table updates when a new flow comes into the middlebox. The current total number of flows will increase. This will result in less bandwidth allocated to each flow. For the second factor, once a flow is finished, there is no more data belonging to this flow coming into the middlebox. Then the flow information table will delete the information of this flow, and at the same time, it will reclaim the bandwidth occupied by this flow. The reclaimed bandwidth is allocated to other data flows; this will increase the bandwidth to each flow. The third factor is a special case, which occurs only when the flow is about to time out. The system will allocate enough bandwidth to this flow so that it can accomplish the data transmission in the required time.

Combined with these three factors, we propose a bandwidth allocation algorithm based on the private network. Our priority of consideration is the third factor. To find out the flows that are about to time out, we look up the flow information table. The value of the timer in table X_k^S is the number of time slots that has been spent by this flow. The variable d_k is the required time slots of the flow k. If $d_k - X_k^S$ equal to 1, that means this flow has only one time slot to finish the transmission. This is the condition to determine whether a flow is about to time out. Next, we need to calculate the residual amount of data to be transmitted. We assume variable r_k is the nontransmitted data, where $r_k = m_k - \sum_i^{X_k} B_k(t_i) \cdot t_i$. Then r_k/t_{i+1} is enough bandwidth that can satisfy the time requirement.

For the first factor and the second factor, we can combine them to deal with. By calculating the difference between the new and reduced flows, the total number of the current flows through the middlebox is updated. The system sets the total number as the count value of the flow information table.

When calculating the bandwidth for each flow, we first give the bandwidth to the flow that is about to time out. Then, the total number of flows is subtracted from the number of flows about to timeout, and the difference is obtained for the average allocation of bandwidth. The bandwidth allocation algorithm for the private network is shown in Algorithm 1.

4.2. Public Network. In the more widely used public network environment, it is not only blockchain communication, but also any other type of transmission. Network resources are no longer exclusive to blockchain services but compete with various data flows. In order to reflect the real environment of blockchain communication, we use the fat flow and slim flow competition model and define the state space and action space.

We choose the Deep Q Network (DQN) as DRL method to optimize the bandwidth allocation. DQN has some advantages: first, it only needs an incentive mechanism to evaluate the behavior so that it does not need data sets for training. Better results can be obtained by training according to the environment observation and action of the agent. Second, when the number of flows increases, the DQN can well deal with the large-dimensional state action space. Third, DQN has an experience replay mechanism, which reuses the samples to improve the learning efficiency. In this case, we use DQN algorithm to solve the problem.

4.2.1. State Space. For the public network, the model contains slim flows representing blockchain communications and fat flows representing other application flows. We define the network resource and flow status as the environment in reinforcement learning and the state of the environment is changing along with time. The flow information table contains all the flows state and updates in each time slot. Except for srcip, dstip, sport, and dport variables, other variables change with time. And the total number of data flows in the table also changes along with time.

Input: srcip, dstip, sport, dport
Output: bandwidth
Caculate an available bandwidth for the flow identified by srcip, dstip, sport, and dport
load data from flow information table into flowList
create a timeoutList
for each S_k in flowList **do**
$X_k^S \leftarrow S_k$.timer
if $d_k - X_k^S = 1$ **then**
timeoutList.add (S_k)
end if
end for
use r_k to store the residual data for flow k
use B_timeout to accumulate the bandwidth of the timeoutList
$B\_timeout \leftarrow 0$
for each S_k in timeoutList **do**
calculate the residual data for each flow
$r_k \leftarrow m_k - \sum_i^{X_k} B_k(t_i) \cdot t_i$
$B_k^S \leftarrow r_k / t_{i+1}$
B_timeout$\leftarrow B$_timeout + B_k^S
end for
$B_k^S \leftarrow B - B$_timeout/flowList.count − timeoutList.count
update B_k^S for each flow to flowList
bandwidth ←flowList.lookup(srcip, dstip,sport, dport)
return bandwidth

Algorithm 1: Bandwidth allocation algorithm for private network.

According to the bandwidth allocation model and flow information table, the environment has three parts, the fat flow state, the slim flow state, and the whole number of flows. The fat flow state can be represented as $F_j(t) = [d_j(t), m_j(t)]$. And $d_j(t)$ is a function that changes with time t; it represents the remaining available time slots of the flow j. Function $m_j(t)$ denotes the remaining data to be transmitted. For fat flows j, $j \in \{1, 2, 3, \ldots N^F(t)\}$ and the number of fat flows is $N^F(t)$. Same to the fat flow, $S_k(t) = [d_k(t), m_k(t)]$ denotes the state of slim flow k. The number of slim flows is denotes as $N^S(t)$, and $k \in \{1, 2, 3, \ldots N^F(t)\}$. The total number of flows through the middlebox is the sum of fat and slim flows. It can be denoted as $N^F(t) + N^S(t)$, so the state space can be represented as

$$\text{State}(t) = \left[F_j(t), S_k(t), N^F(t) + N^S(t)\right]. \tag{10}$$

4.2.2. Action Space. In the public network, the decision made by the middlebox in each time slot is the calculation of the bandwidth value for each data flow. According to our bandwidth allocation model, the actions in each time slots can be denoted as

$$\text{Action}(t) = \left[B_j^F(t), B_k^S(t)\right], \tag{11}$$

where $B_j^F(t)$, $j \in \{1, 2, 3, \ldots N^F(t)\}$ refers to the bandwidth of fat flow j, and $B_k^S(t), k \in \{1, 2, 3, \ldots N^S(t)\}$ refers to the bandwidth of slim flow k. The total bandwidth allocated in each action should meet the constraint C5 in equation (9).

The action includes all flows' bandwidth value in time slot t. The value computed by the middlebox is the available bandwidth for the sender. It will ensure that congestion is avoided and bandwidth utilization is improved. The action will affect the transition of state and the accumulation of reward.

4.3. Reward Function. The goal of our model is to maximize the total reward through bandwidth allocation in a certain period of time. The object is to maximize the rewards, which also means that our allocation strategy serves more data flows in the observation time.

We choose the RL method to optimize the bandwidth allocation, because it only needs an incentive mechanism to evaluate the behavior. We build such a mechanism to train the strategy of bandwidth allocation. If a flow is not completed within the required time due to improper bandwidth allocation, the reward is negative. If the bandwidth allocation is appropriate and the flow is completed within the specified time, a positive reward is obtained. We add a punish rule in the reward function, so it is a little different from the model definition in equations (7) and (8).

The details of the reward rules can be described as follows:

(i) In a time slot t, suppose the state space has a flow state $F_j(t) = [d_j(t) = 0, m_j(t) > 0]$, where $d_j(t) = 0$ indicates that it has timed out and $m_j(t) > 0$ indicates that there is still data that has not been served. Such a state is a bad situation and should be punished. We use $\theta_j(t) = -1$ to represent this state, and the reward of this flow $F_j(t)$ should be $r_j \cdot \theta_j(t)$.

(ii) In a certain time slot t, suppose the state space has a flow state $S_k(t) = [d_k(t) > = 0, m_k(t) = 0]$, where $m_k(t) = 0$ indicates that the data has been served, and $d_k(t) > 0$ indicates that the data has been completed before the timeout. Else $d_k(t) = 0$ indicates that the flow finished in time. This state is a good situation and it should be rewarded with the value of $r_k \cdot \theta_k(t)$ and $\theta_k(t) = 1$.

(iii) In other states, there is no change in the number of data flows, so no reward can be obtained in this situation. The corresponding parameter $\theta(t)$ equals zero in this state.

(iv) To optimize blockchain communication, the absolute value of the reward of slim flow should be greater than that of fat flow. It can be shown as $|r_k| > |r_j|, \forall k \in \{1,2,3,\ldots N^S(t)\}, j \in \{1,2,3,\ldots\ N^F(t)\}$.

The reward function $R(t)$ is the sum of rewards of all the flows at time slot t.

$$R(t) = R^F(t) + R^S(t) = \sum r_j \cdot \theta_j(t) + \sum r_k \cdot \theta_k(t),$$
$$\theta_{j,k}(t) = \begin{cases} -1 & , d_{j,k}(t) = 0, m_{j,k}(t) > 0, \\ 1 & , d_{j,k}(t) \geq 0, m_{j,k}(t) = 0, \\ 0 & , \text{others.} \end{cases} \quad (12)$$

After each action of bandwidth allocation, the system calculates the state of each flow. When the system has finished updating the state and the total number of data flows, it will enter the next state. According to the state of served data and the residue time requirement, the system decides whether to get a reward.

4.4. DQN. Our system needs to compute proper bandwidth for data flows in each time slot. The principle of proper bandwidth calculation is to maximize the rewards in the whole observation time period. When the number of flows increases or the network bandwidth to be allocated is large, the space of states and actions becomes difficult to calculate. Due to the large scale of state-changing of data flows in the middlebox, we employ the DQN method to deal with the high-dimensional state spaces and action spaces.

DQN is a reinforcement learning algorithm with the advantages of neural networks [33]. The agent of DQN can replace the role of the middlebox system and complete the bandwidth allocation according to learning from the environment. For our model, the long-term value function represents the accumulated reward from the data flow service. It can be denoted as

$$Q(\text{State}(t), \text{Action}(t)) \leftarrow (1-\alpha) Q(\text{State}(t), \text{Action}(t)) + \alpha [R(t) + \gamma \max Q(\text{State}(t+1), \text{Action}(t+1))]. \quad (13)$$

In this function, α is the learning rate that is used to balance the current and future reward value, and γ is the discount rate. The DQN method for bandwidth allocation is shown as Figure 5.

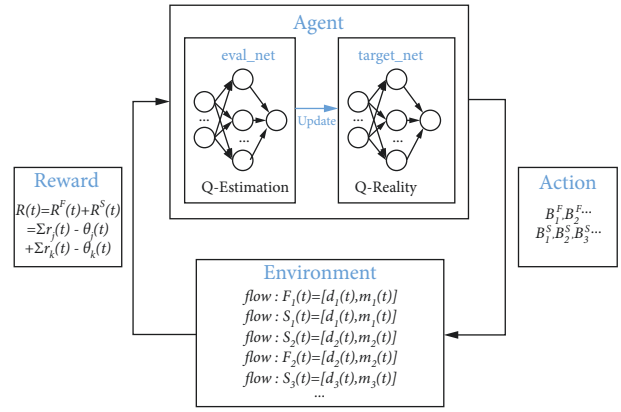

Figure 5: DQN method for bandwidth allocation.

In the DQN method, two neural networks are used to train the agent to do bandwidth allocation. One neural network is called eval_net that trains the parameter of agent to estimate Q value. The other neural network is called the target_net. It is used to save the target Q value. The eval_net and the target_net are with the same structure but different parameters. When training the agent to allocate bandwidth, the experience replay is well utilized to learn from the past operations. By random sampling of the experiences, the neural network will update more efficiently. The eval_net represents the Q-estimation which contains the new parameter. The target_net predicts Q-reality, but its parameter is not new. After some steps, the target_net will be updated according to the parameters of the eval_net. This mechanism, which is called fixed Q-targets, can overcome the data correlation and ensure the convergence of parameters. The algorithm of DQN for bandwidth allocation is present in Algorithm 2.

5. Experimental Environment and Results

In this section, we first introduce our experimental environment. Then we discuss performance evaluation.

5.1. Experimental Environment. In our test environment, we use VMware ESXi virtual machine system to build a local network. This network has 4 middleboxes and 4 endpoints to act as senders and receivers. These middleboxes can play the role of consensus nodes in blockchain communication. Each node is a virtual machine with 2 gigabit netcards and 4 GB memory. The operating system is Ubuntu 18.04 LTS with 4.15 version Linux kernel.

For throughput test, we use the network testing tool IPerf3. To simulate a complex network environment, TC (Traffic Control) tool is employed. Netem (Network emulation) is a kernel module controlled by TC. We use Netem to set packet loss and delay in the middlebox. In order to make a bottleneck in the network path, HTB (Hierarchy Token Bucket) is utilized to do shaping and policing for network traffic. It is also a type of queue managed by TC and can set a rated speed for a network interface.

In order to make the system support our available bandwidth notification method, we create a new kernel

(1) Initialize memory M for experience replay
(2) Initialize eval_net and $Q_{\text{estimation}}$ function with weight parameter ω
(3) Initialize target_net and Q_{reality} function with weight parameter $\omega^{-} = \omega$
(4) **for** each episode **do**
(5) Initialize State(t)
(6) **for** each time slot t **do**
(7) Generate a random number λ using ϵ – greedy policy for balancing exploration and exploitation
(8) **if** $\lambda < \epsilon$ **then**
(9) Randomly choose a series of bandwidth $\{B_1^F, B_2^F, \ldots, B_1^S, B_2^S, \ldots\}$ as the Action(t) for current flows.
(10) **else**
(11) Action(t) = avg max Q(State(t), Action(t); ω)
(12) **end if**
(13) Execute Action(t)
(14) Obtain $R(t)$ and State$(t+1)$
(15) Save {State(t), Action(t), $R(t)$, State$(t+1)$} into memory D
(16) Randomly select batch of experience data $\{\text{State}_i, \text{Action}_i, R_i, \text{State}_{i+1}\}$ from D
(17) **if** $State_{i+1}$ is the terminal state **then**
(18) set $y_i = R_i$.
(19) **else**
(20) set $y_i = R_i + \gamma \max_a Q_{\text{reality}}(\text{State}_{i+1}, a)$
(21) **end if**
(22) Perform gradiant descent on $(y_i - Q_{\text{estimation}}(\text{State}_i, \text{Action}_i; \omega))^2$ with ω
(23) Update Q_{reality} parameter with $Q_{\text{estimation}}$
(24) **end for**
(25) **end for**

Algorithm 2: DQN algorithm for bandwidth allocation in public network.

module called "tcp_abn" following the example of "tcp_bbr" that is developed by Google. Some kernel files of the TCP stack have been modified, while the compatibility has been kept. As our method needed, the modified files contain two types. One type is deployed on the middlebox. This type of file contains sch_htb.c. In sch_htb.c, the calculation of available bandwidth is added, and the bandwidth value is injected into the corresponding ACK packet. The other type is deployed on the sender. This type of file contains tcp.h, tcp_input.c, tcp_ipv4.c, tcp_output.c, and tcp_abn.c. The tcp_abn.c file is the major part of the "tcp_abn" module. Other files make the system suitable for the "tcp_abn" module and support applying the bandwidth values from the ACK packet.

5.2. Results. For the private network, we only initiate blockchain communication in our local network. In the middlebox, we set the upper limit of network bandwidth to 2 Gbit/s by HTB policing function. We use the sysctl command to change the kernel file "/proc/sys/net/ipv4/tcp_wmem" as "4096 16384 100000000" in each sender. This operation can make the system provide better performance which avoids limiting network speed due to insufficient send windows.

In this network configuration, we compare our ABN method with two recently popular algorithms, Cubic and BBR. Cubic is the default TCP congestion control algorithm in Linux system, and BBR is a Google provided algorithm that is also supported in Linux kernel.

In order to get a more significant effect, we set the delay changing in a certain range. It is hard for the RTT probing

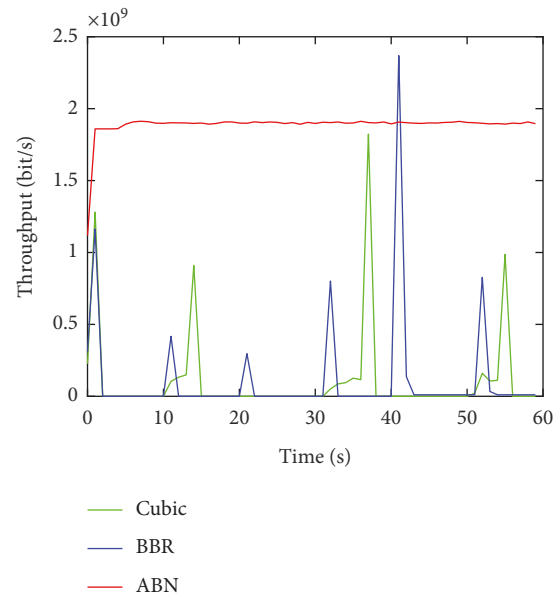

Figure 6: Throughput with 50 ms delay.

mechanism to measure an accurate value in variable delay. We use Netem tool to add latency in the network, and the delay range is 50 ms, 100 ms, and 200 ms, respectively. We use IPerf3 to transmit data in TCP protocol where the packets go through the middlebox. The middlebox will be deployed on Cubic, BBR, and our ABN. We use tcpdump to catch all the traffics generated by IPerf3 during 60 seconds and save then into a pcap file. Then, we convert these pcap files to csv format files and draw Figures 6–8.

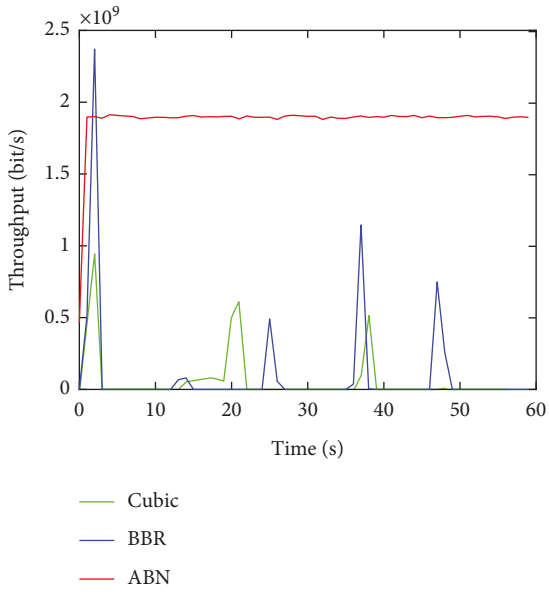

Figure 7: Throughput with 100 ms delay.

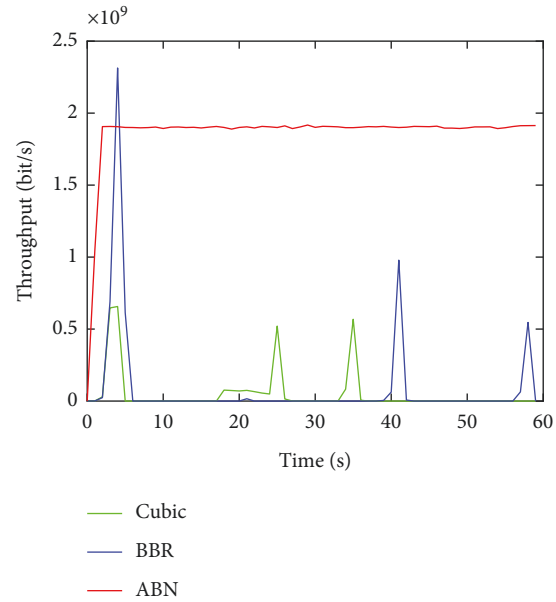

Figure 8: Throughput with 200 ms delay.

The throughput with 50 ms range delay is shown in Figure 6. We can find that our ABN method has a relatively stable speed close to the ceiling bandwidth. BBR has 6 spikes and Cubic has 4; in addition, the average height of spikes of BBR is higher than Cubic. It means the throughput of BBR is better than Cubic. The throughput results during 60 s with 50 ms range of delay are Cubic : 235 Mbit/s, BBR : 1370 Mbit/s, and ABN : 1990 Mbit/s. At the position of 40 s in Figure 6, the throughput is higher than the limit capacity. That is because BBR will probe the available bandwidth and sometimes overestimate it. Then it will occupy the network buffer of the middlebox.

The throughput with 100 ms and 200 ms range of delay is shown in Figures 7 and 8. With the delay increasing, we can see that the number of spikes of BBR and Cubic gets smaller. The throughput of our ABN decreases minorly and still stays at a stable speed. That is attributed to the bandwidth notification from the middlebox. The speed of ABN accords with the feedback of the network environment, not probing or guessing. BBR will drop the speed when it perceives the delay increasing in probeRTT phase. Cubic sends packets continually, so the spike of Cubic usually lasts longer than BBR. When delay increases and leads to packet drop occurrence, Cubic sharply reduces the transmission speed. If the packet loss state continues, Cubic will always maintain the low speed state. The throughput of Cubic with 100 ms range of delay is 205 Mbit/s. BBR is 1120 Mbit/s, and ABN is 1990 Mbit/s. With 200 ms range of latency, the throughput of Cubic drops to 93.2 Mbit/s. BBR's throughput shrinks to 465 Mbit/s. The average speed of ABN during 60 seconds is 1970 Mbit/s. The performance analyses demonstrate that the ABN method can stand delay influence and improve the network bandwidth utilization.

For the public network, we use a DQN model to do the performance simulation. There are two neural networks in DQN, eval_net and target_net, each network has two layers, and each layer has 20 neurons. The input of neural network is set according to the amount of flows and the size of bandwidth allocation action space. Tensorflow 1.12.0 with python 3.7 is employed to do the training on Windows 10 system. In the simulation, 4 blockchain nodes with PBFT consensus initiate slim flows through a middlebox. In order to highlight the support of the ABN method for blockchain communications, we add two fat flows with large bandwidth as the background traffic to compete with blockchain communications, and the agent tends to allocate bandwidth to blockchain flow for more rewards. The detail of simulation parameters can be shown in Table 2.

The loss of DQN model is as shown in Figure 9. We can find that the loss is getting lower with the training steps increasing. The loss value tends to be stable which means the model is available. In Figure 10, the reward variation with different learning rates is presented. The learning rate affects the rewards during the training. When the learning rate is 0.1 and 0.01, the reward curve oscillates violently and gets the optimal value occasionally. When the learning rate is 0.001, the reward curve is relatively convergent, but it does not converge to the optimal value. In our simulation, the learning rate of 0.0001 obtains better performance than others. As shown in the figure, the learning speed is acceptable, and the reward curve stably converges to the optimal value. The rule of reward function is to accumulate the reward of each served flow.

The comparison of reward between ABN, BBR, and Cubic is presented in Figure 11. In this training, we send one slim flow and fat flow at the beginning and add slim flows during 200 s. To simulate PBFT communications, we add slim flows as a multiple of 4 which is the number of consensus nodes. In the case of background traffic, ABN can obtain more rewards with the increase of flows. Although the reward of Cubic is the same as others at the beginning, it will

Table 2: The parameter settings in the simulation.

Parameter	Value	Description
Bandwidth	10 Mbit/s	The whole bandwidth in the network
Fat flow	1000 Mbits, 200 s, 80	A type of fat flow as background traffic
Slim flows	1 Mbits, 1 s, 10	Four slim flows in a group to represent blockchain communications
T	200 s	Total time of observation
Node	4	The number of blockchain nodes that initiate PBFT consensus communications
γ	0.9	The discount factor

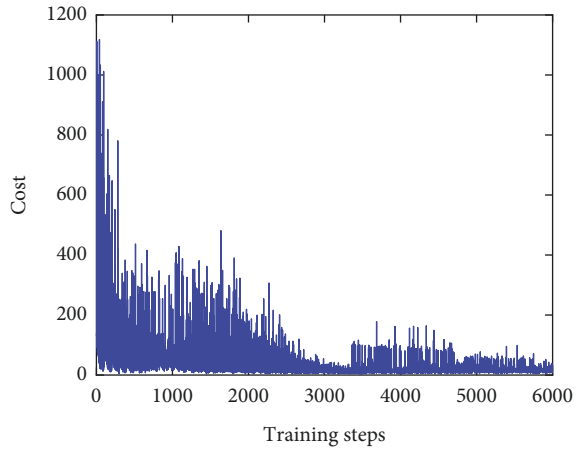

Figure 9: The learning loss of the DQN model for ABN.

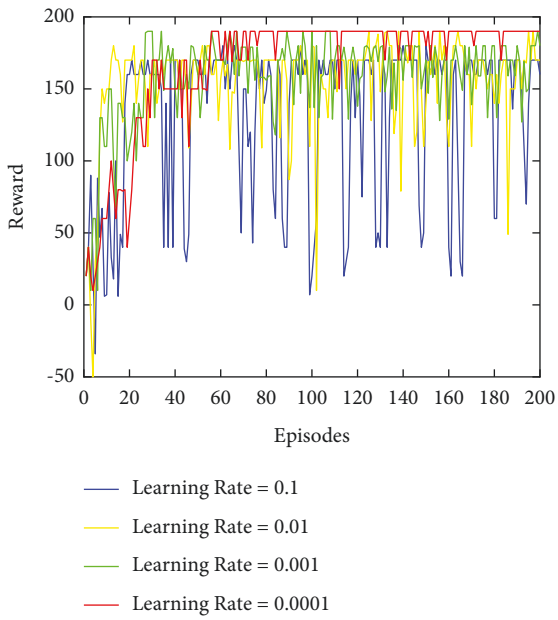

Figure 10: The rewards of DQN model for ABN with different learning rate.

be punished when congestion occurs with the increase of flows. The result is the slow growth of reward for Cubic. BBR sometimes can get a better reward, because it attempts to fill the pipe to probe bandwidth, and it possibly occupies the buffer. We can find that when the amount of flow is 32, the

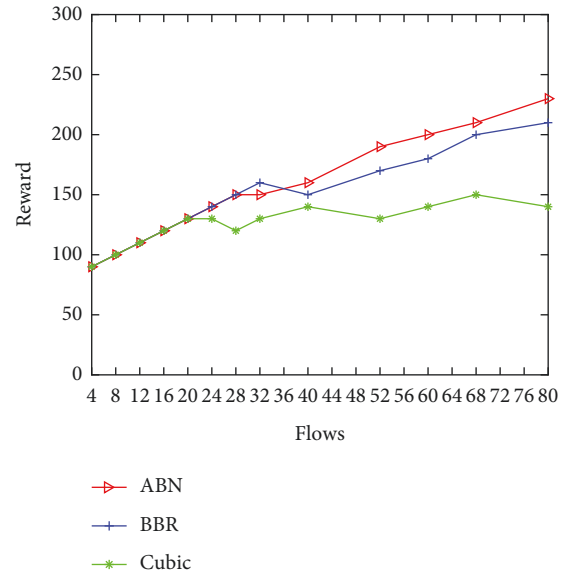

Figure 11: The reward comparison of ABN with BBR and Cubic.

reward of BBR is a little higher than ABN. But this situation cannot be persistent; it will drop the speed when BBR is in the drain phase. As we can see, ABN can get a more stable reward augmentation than BBR in a long-term running.

6. Conclusions

In this paper, we investigate a blockchain-enabled IoT network deployed on MEC servers and analyze the characteristics of the network flow of blockchain in consensus communication. After researching the relationship between consensus communication and performance in the permissioned blockchain, we determine to optimize the performance by focusing on network bandwidth utilization. We proposed an optimal allocation algorithm called available bandwidth notification. The ABN method can provide efficient bandwidth utilization and avoid congestion; it will improve the speed of blockchain consensus, so as to ensure IoT data is safe and fast. The results derived from the evaluation demonstrate the high performance of our method. Compared with the current TCP network, our method is more friendly to the communication of blockchain. With the further development of the research, we would consider the data privacy protection function of blockchain-enabled IoT network.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] X. Xu, Z. Fang, J. Zhang et al., "Edge content caching with deep spatiotemporal residual network for iov in smart city," *ACM Transactions on Sensor Networks*, vol. 17, no. 3, pp. 1–33, 2021.

[2] J. Huang, C. Zhang, and J. Zhang, "A mqueue approach of energy efficient task scheduling for sensor hubs," *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 242–247, 2020.

[3] X. Xu, H. Tian, X. Zhang, L. Qi, and Q. He, "Discov: distributed covid-19 detection on x-ray images with edge-cloud collaboration," *IEEE Transactions on Services Computing*, 2022.

[4] T. M. Fernandez-Carames and P. Fraga-Lamas, "A review on the use of blockchain for the internet of things," *IEEE Access*, vol. 6, pp. 32979–33001, 2018.

[5] Y. Chen, W. Gu, and K. Li, "Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning," *International Journal of Communication Systems*, 2022.

[6] J. Feng, F. R. Yu, Q. Pei, J. Du, and L. Zhu, "Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 4321–4334, 2020.

[7] Y. Chen, F. Zhao, Y. Lu, and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Science and Technology*, 2021.

[8] Y. Chen, Y. Zhang, Y. Wu, L. Qi, and X. X. Chen, "Joint task scheduling and energy management for heterogeneous mobile edge computing with hybrid energy supply," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8419–8429, 2020.

[9] Y. Chen, H. Xing, and Z. Ma, *Cost-Efficient Edge Caching for NOMA-Enabled IoT Services"*, China Communications, 2022.

[10] Q. F. Shao, C. Q. Jin, Z. Zhang, W. N. Qian, and A. Y. Zhou, "Blockchain: architecture and research progress," *Chinese Journal of Computers*, vol. 41, no. 8, pp. 969–988, 2018.

[11] T. Mitani and A. Otsuka, "Traceability in permissioned blockchain," in *Proceedings of the IEEE International Conference on Blockchain*, Atlanta, GA, USA, July 2019.

[12] M. Castro, "Barbara and Liskov, "Practical byzantine fault tolerance," in *Proceedings of the third symposium on Operating systems design and implementation*, pp. 173–186l, New Orleans Louisiana, USA, 1999.

[13] T. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K. L. Tan, "Blockbench: a framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1085–1100, Chicago Illinois, USA, May 2017.

[14] M. El-Hindi, C. Binnig, A. Arasu, D. Kossmann, and R. Ramamurthy, "BlockchainDB," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1597–1609, 2019.

[15] J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, "A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 429–445, 2019.

[16] S. Nakamoto, *Bitcoin: A Peer-To-Peer Electronic Cash System"*, 2008.

[17] V. Buterin, *A Next-Generation Smart Contract and Decentralized Application Platform"*, 2014.

[18] C. Catalini and J. S. Gans, "Some simple economics of the blockchain," *Communications of the ACM*, vol. 63, no. 7, pp. 80–90, 2020.

[19] H. Sukhwani, JM. Martinez, X. Chang, KS. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *Proceedings of the IEEE 36th Symposium on Reliable Distributed Systems*, Hong kong, China, September 2017.

[20] M. Vukoli, "The quest for scalable blockchain fabric: proof-of-work vs. bft replication," in *Proceeding of the Open Problems in Network Security*, Zurich, Switzerland, October 2015.

[21] R. Sekaran, R. Patan, A. Raveendran, F. Al-Turjman, M. Ramachandran, and L. Mostarda, "Survival study on blockchain based 6G-enabled mobile edge computation for IoT automation," *IEEE Access*, vol. 8, pp. 143453–143463, 2020.

[22] W. Li, Z. Su, R. Li, K. Zhang, and Y. Wang, "Blockchain-based data security for artificial intelligence applications in 6G networks," *IEEE Network*, vol. 34, no. 6, pp. 31–37, 2020.

[23] I. G. AK. Gemeliarana and RF. Sari, "Evaluation of proof of work (pow) blockchains security network on selfish mining," in *Proceedings of the IEEE Internasional Seminar on Research of Information Technology & Intelligent Systems*, Yogyakarta, Indonesia, November 2018.

[24] T. Nakaike, Q. Zhang, Y. Ueda, T. Inagaki, and M. Ohara, "Hyperledger fabric performance characterization and optimization using goleveldb benchmark," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, Toronto, Canada, May 2020.

[25] NN. Lan, T. DT. Nguyen, TN. Dinh, and MT. Thai, "Optchain: optimal transactions placement for scalable blockchain sharding," in *Proceedings of the IEEE 39th International Conference on Distributed Computing Systems*, Dallas, TX, USA, July 2019.

[26] S. Kantesariya and D. Goswami, "Determining optimal shard size in a hierarchical blockchain architecture," ", in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, Toronto, Canada, May 2020.

[27] J. Huang, Z. Tong, and Z. Feng, "Geographical POI recommendation for internet of things: a federated learning approach using Matrix factorization," *International Journal of Communication Systems*, 2022.

[28] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Performance optimization for blockchain-enabled distributed network function virtualization management and orchestration," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6670–6679, 2020.

[29] J. Huang, B. Lv, Y. Wu, and Y. X. Chen, "Dynamic admission control and resource allocation for mobile edge computing

enabled small cell network," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1964–1973, 2022.

[30] B. Huang, L. Jin, Z. Lu, X. Zhou, P. CK. Hung, and Bor, "Toward high-performance permissioned blockchain in rdma-enabled network," *IEEE Transactions on Services Computing*, vol. 13, no. 2, 2020.

[31] N. Cardwell, "Bbr: congestion-based congestion control," *ACM Queue*, vol. 60, no. 2, 2017.

[32] S. Floyd, "Tcp and explicit congestion notification," *ACM Computer Communications Review*, vol. 24, no. 5, 1994.

[33] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.