Proceedings of 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN 2018)

Chameleon: A Scalable and Adaptive Permissioned Blockchain Architecture

Guobiao He
School of Electric and Information Engineering
Beijing Jiaotong University
Beijing, China
17111014@bjtu.edu.cn

Wei Su
School of Electric and Information Engineering
Beijing Jiaotong University
Beijing, China
wsu@bjtu.edu.cn

Shuai Gao
School of Electric and Information Engineering
Beijing Jiaotong University
Beijing, China
shgao@bjtu.edu.cn

***Abstract*—Designing blockchain architecture is still an open question and encounters a lot of challenges such as scalability, security, high utilization and so on. In this paper, we propose Chameleon, a scalable and adaptive permissioned blockchain architecture. We adopt the principles of non-forking, high security, scalable and high utilization to design the Chameleon to be suitable for next generation blockchain architecture. In Chameleon, we introduce credit value which can only be acquired through honest behavior to enhance the security of the consensus algorithm. We also introduce the QoS of transactions to meet the various needs of different users. While previous work using sharding to improve the scalability, they either store all transactions in every area or just work independently, it either causes too much redundant data or low resource utilization. Combined with the cloud storage, Chameleon partition the nodes into different area according to different scenario, every area process their own transactions and can also cooperate with each other dynamically to improve transaction throughput and resource utilization.**

Keywords—permissioned blockchain, scalable, adaptive, QoS

I. Introduction

Blockchain is essentially a distributed ledger, Blockchain technology have a lot of significant features such as decentralized, transparent, can't be tampered, etc. These features can be used in a variety of applications like financial transactions, supply chain management, data provenance, credential management, etc. The essential driving force behind blockchain technology is people's needs of autonomy, openness, transparency, high efficiency and trust. Blockchain defines the ownership of data property which is the cornerstone in Internet of Value, this will greatly unleash the potential ability of data. However, designing a scalable, secure and high utilization blockchain architecture to meet various application scenario is still an open question.

Recently, the popularity of CryptoKitties game in Ethereum cause the network congestion and the important money transfer transaction have to wait for almost half a day. This because that current transaction has no priority and it also don't have effective measures to cope with large amount of bursting transactions. Thus the quality of transaction service won't be guaranteed under the overload situation. We focus on solving these challenges using the advantages of the permission blockchain architecture. Different from the open blockchain, a permissioned blockchain have the control layer to govern or coordinate the network behavior.

This paper presents Chameleon, a dynamic adaption and scalable permissioned blockchain architecture. There are four layers in Chameleon: control and authentication layer, cloud storage layer, consensus and processing layer and access layer.

The control and authentication layer have two functions, they are issuing certifications and load balancing. In previous work, no architecture had considered classifying different transactions and perform transaction-level load balancing. We introduce the QoS of transaction and perform transaction-level load balancing to meet different kind of requirements and greatly improve resource utilization.

The cloud storage layer to solve the scalability problems in specific scenarios , the cloud can be a local cloud, edge cloud or core cloud. The consensus nodes only have to store one epoch of transaction block and the previous transaction block will be stored in the cloud. Every epoch can be one day or one week. This will greatly reduce the storage burden and the storage capacity requirement of the consensus node.

The consensus and processing layer is mainly used to process transactions and reach consensus, the nodes are divided into different areas according to different scenarios. Different area can adopt the different consensus protocol to meet their own specific needs ,this can improve the resource utilization of the system.

We introduce an improved byzantine agreement protocol called RLSCV(Random Leader Selection based on Credit Value) inspired by Algorand which guarantee the security of system by the overall balance in consensus nodes. Our improved byzantine agreement protocol choose leader randomly based on the credit value to replace the previous the deterministic and predictable leader election method. This method improve the security of the system from two aspects: one is that leader selection based on credit value which can only be acquired by honest behavior, the higher the credit value, the higher probability to be selected as a leader; another aspect is that the random selection of the leader can resist the Dos attack during the election.

The access layer allows client to register corresponding services and obtain certificate in control and authentication layer.

Contributions: we claim the following contributions:

- We introduce the QoS of transaction and perform transaction-level load balancing to meet different kind of requirements and greatly improve resource utilization.
- We introduce improved byzantine consensus protocol RLSCV which can greatly guarantee the security and the scalability of the system
- We introduce a collaborative mechanism interacting with the cloud storage infrastructure to solve the problem of storage scalability.

II. RELATED WORK

There are many kinds of open and permissioned blockchain architecture. Here, we present the most popular ones to analyze the challenges and unsolved problems of these architectures.

A. *Elastico:*

Elastico is a sharding protocol for open blockchain. It use PoW to generate identity of processor, these processors are organized into committee according to some certain identity features in every sharding [1].

There are also a lot of challenges in Elastico :

1) All nodes must store all transaction data in every sharding, it is a great challenge for the storage ability.

2) Elastico changes committee every epoch ,so it was not possible to process transactions during the time of changing epoch.

3) Every sharding in Elastico haves to be the same to ensure the normal operation of the system ,it can't meet the different requirements of different scenarios.

B. *Omniledger*

Different from Elastico's periodically reconfiguring committees based on PoW, Omniledger periodically reconfigures committees based on RandHerd. Although Omniledger can scale in nodes and transactions, There are several problems unsolved[2].

1) Omniledger makes sharding randomly, It is based on the assumption that all nodes have the same kind of ability and all sharding deal with same kind of transactions. It can't guarantee the QoS of different transactions.

2) Omniledger which is client-driven atomic commit protocols is vulnerable to DoS, client can flood transaction to every sharding since cross-sharding transactions are randomly assigned to sharding for processing.

C. *Algorand:*

In Algorand, users check for themselves whether or not they should play a role in the committee for the next round by seeing if the output of a verifiable random function is less than a certain value. As participants start playing their roles, they can include information in their messages that allows other participants to check that they are in fact eligible[3].

Although Algorand is not susceptible to either targeted compromises or DoS attacks and improve the throughput greatly compared with bitcoin, it has many other limitations.

1) Algorand rely on a large number of nodes to participate in and the honest majority of money to guarantee system security, but as the transaction volume increases sharply every day, more and more nodes will not be able to participate in the system operation after a long period because of the limitation of storage ability.

2) Algorand is designed to solve the scalability of cryptocurrency, it don't suitable for many other scenarios.

D. *Hyperledger Fabric*

Fabric have two kinds of nodes, these are peer and order. Peer nodes in charge of endorsement and storing the blocks, order nodes in charge of ordering the transactions. Fabric can be divided into different channels according to different scenarios[4].

There are a several problems unsolved in fabric:

1) Fabric order nodes run the classic or simple BFT consensus which is vulnerable to DoS attack because the leader selection was predictable.

2) Different channel work separately in fabric and can't cooperate each other to improve the transaction performance

3)In fabric, the client can't participate the consensus process, consensus only happens in peer and order nodes.

III. SYSTEM ARCHITECTURE

This section presents our system architecture and our design concept.

A. *Overview of Chameleon*

There are four layers in our permissioned blockchain architecture Chameleon as shown in Fig.1: control and authentication layer, cloud storage layer, consensus and processing layer and access layer.

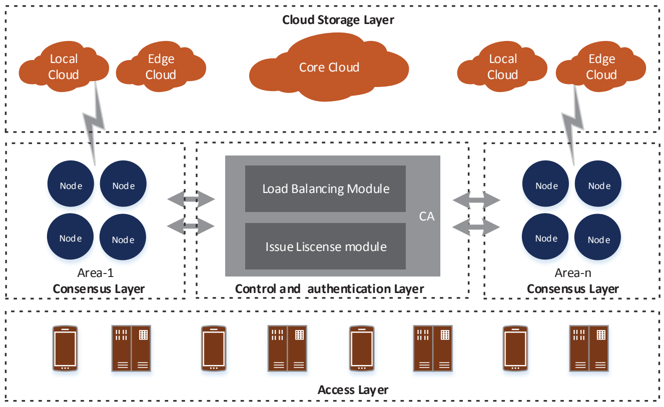

Fig. 1 System architecture of chameleon

Our design concept:

1) QoS of transaction: to meet the requirements of variety of applications, blockchain must be divided into several areas according to applications scenarios to process transactions efficiently. Transaction must have different kinds of priority to guarantee the QoS.

2) Dynamic Adaption: Different area can adjust the sub-area or cooperate with other area dynamically to deal with the large amount of burst transaction in certain period. This can improve the resource utilization and enhance the ability of blockchain automation management.

3) Based on Trust: we believe the essential driving force behind blockchain technology is people's needs of autonomy, openness and transparency. Blockchain will play a fundamental role in the future value internet and trust will be the base stone, so we use the credit value of all honest nodes to guarantee the trust and security in Chameleon.

4) Combining with Cloud. Blockchain is not meant to subvert all existing internet infrastructures. In most conditions, it only needs to enhance the properties of Blockchain and maximize the cooperation with existing systems such as cloud services and edge computing facilities. This can reduce the excessive redundant data and solve the storage scalability problems in blockchain.

System assumption: we assume the honest nodes is more than 2/3 in every area, all honest nodes is well connected and will receive the message within a known maximum delay.

Processing flow:

1) The client in the accessing layer send request to the control and authentication layer (CA) to sign up the corresponding service, client can choose to participate into the consensus process or not, the CA will endorse the transaction according to the registration information. The client receive enough endorsement and send transaction to the corresponding area.

2) Every area will run improved byzantine consensus protocol RLSCV to generate transaction block and can request to the CA for dividing into different sub-areas to speed up transaction processing. After the consensus, the block will be stored in every nodes. The blocks will be stored in the Cloud side after one epoch such as one day or one week. Only the previous epoch block header will be recorded in every nodes to guarantee the consistency.

3) The local, edge and the core cloud will send ACK to confirmation the storage information and the node in every area will run RLSCV consensus to make sure the ACK message have been received by most of the nodes and then start the next epoch.

4) The CA will collect all the load and resource utilization in every area and trigger load balancing mechanism when load exceeds a certain threshold.

B. Transaction Format

We define the transaction format in TABLE I to realize QoS and transaction level load balancing in Chameleon.

TABLE I. TRANSACTION FORMAT

Transaction Type	**Transaction ID**	**Area ID**	**Subarea ID**	**Destination Area**	**QoS**	**Endorsement of CA**	**Content**
1）ordinary transaction 2）cross-area transaction 3）cooperative transaction 4）sub-area transaction	Transaction- x	Area-n	Subarea m	for cooperative and cross-area transaction use	1) network control 2) instant processing 3) accelerate 4) ordinary 5) to be reserved	Signature of CA	Specific content

Transaction Type is used to distinguish different kind of transactions and the area will trigger corresponding processing. There are four transaction type in Chameleon. Ordinary transaction can only be processed in one specific area. Cross-area transaction is use to address the needs of data trading or data sharing between different area. Cooperative transaction is to address the needs of transaction level resource scheduling under the situation of overload. Sub-area transaction is used to address the needs of dividing into several sub-area to improve the transaction throughput.

Transaction ID is used to identify the transaction, Area ID and sub-area ID is used to identify the area and sub-area, if there is no sub-area in one area, the sub-area ID is zero by default.

Destination Area is used to specific the light load area or specific the area for cross-area transaction. When there are a large amount of overload burst transactions in one area, CA will trigger the load balancing mechanism to share some transaction into the other light load area.

QoS is use to meet the different requirement of transaction such as delay-sensitive transactions must be processed instantly. We reserve the 8 bit for QoS, i.e. 64 kinds of transaction type just as the same with Differentiated Services Code Point in the Internet. Now, there are four kinds of transaction type ,the others is for reservation in the future.

Endorsement of CA is used to issue a certificate for the transaction.

C. Consensus Layer

PBFT(Practical Byzantine Fault Tolerance) consensus is first proposed in 1999 and widely used in permissioned blockchain architecture[5]. However, leader selection in PBFT consensus are according to the order of node number. As the leader selection can be predicted, it’s easy to be the adversary start the DoS attack to influence the leader selection and threaten the security of the system.

We introduce improved PBFT Consensus RLSCV to improve the byzantine protocol security. RLSCV have two features. One is the credit value in consensus which can only be obtained through honesty behavior, all nodes credit value is first initialized to one. The other is that the leader is randomly selected by credit value, the higher the credit value, the greater the probability that the node is selected to be the leader.

Leader Selection In RLSCV (inspired by Algorand), the Block form in Chameleon: $Br = (r, Cr, Q_r, H(Br\text{-}1)$.r represents r round block, Cr represents all credit value in r round, Qr is random seed which is carefully constructed and can be hard for powerful adversary to manipulate. H(Br-1) is the hash of the previous block. A potential leader of round r is a node i based on the following function :

$$.H(SIGi\ (r,Qr\text{-}1))\leq p. \tag{1}$$

$$p=Ni\_credit\_value/Sum\_credit\_value. \tag{2}$$

$$Ni\_credit\_value=\log_2(Bn) \tag{3}$$

Here, Ni_credit_value represents the credit value of node i, Sum_credit_value represents the overall credit value of the consensus nodes. The higher the credit value, the easier it is to be selected as a leader. In order to ensure that the credit value is not always concentrated in a few nodes, we construct a credit value growth function $\log_2(Bn)$ based on the fact that the log function have the slow growth character in the long run . Bn is the sum of the block generated by the nodes as a leader.

Qr-1 is part of block Br-1, SIGi（r, Qr-1）is a binary string uniquely associated to i and r . Since H is a hash function which have the character of randomness, H (SIGi (r, Qr-1)is a random long string uniquely associated to i and r.

The symbol "." in front of H (SIGi (r, Qr-1) is the decimal point, so that Ri=.H (SIGi (r, Qr-1) is a uniform distribution between 0 and 1 that uniquely associated to i and r. Thus the probability that Ri is less than or equal to p is essentially p.

Note that, since node i is the only one capable of computing his own signatures, it alone can determine whether it is a potential leader. However, by revealing his own credential SIGi(r, Qr-1), node i can prove to anyone to be a potential leader of round r.

The leader Lr is the one whose hashed credential is smaller than the hashed credential of all other potential leader j: that is, H (SIGi (r, Qr-1) ≤H (SIGj(r, Qr-1) . If There are two many nodes in the area. We can also choose k smallest H (SIGi (r, Qr-1) to participate into the consensus.

A node i can be a potential leader after participating in the system at least k rounds, This reduce the risk of a large number of malicious nodes joining the system suddenly to influence the leader selection results and the Qr result. In fact the potential leader determine the Qr.

D. Store Layer

Currently, most blockchain store all data in every nodes to guarantee the safety of the system. Take bitcoin network for example, from the Fig.2, we can see that the cumulative block size is 160G bytes up to 2018 in bitcoin network. Most mobile device can't store such a large volume of data and can't participate into the consensus process. From the cumulative block size of Fig.2 and the transaction rate of Fig.3, we can also find that the cumulative block size increase linearly with the transaction rate.

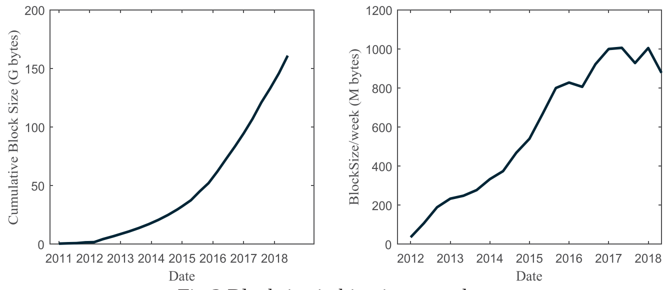

Fig.2 Blocksize in bitcoin network

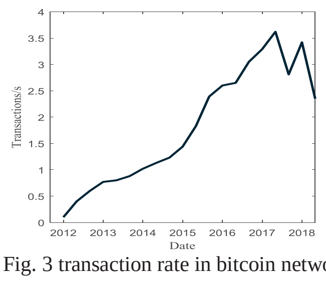

Fig. 3 transaction rate in bitcoin network

Blockchain network's storage scalability is still an open problem. There are several methods try to solve the scalability problems, Here we present one typical solutions:

A trust to trust internet architecture blockstack try to solve the storage scalability by storing the hash of the immutable data in the blockchain and the actual data are stored in the cloud[6]. If the consensus process need the recent historical data, it will be low efficiency to get the data from the cloud. Blockstack runs on bitcoin network , bitcoin is concentration on mining power and the traffic is also concentrated on a few autonomous area, It is easy to suffer BGP hijack[7] .

In chameleon, we combine the advantage of Edge Cloud storage, Central Cloud and the local storage without limit the scope of consensus. There are two storage modes can be selected for the area in Chameleon according to specific scenarios.

1) Store All transaction in every nodes, this can be used in financial scenarios.

2) Store one epoch block data such as one day or one week and the previous block will be stored in the cloud, this method is mainly used in IoT or mobile device scenarios.

As the first modes is widely used now ,let's explain the principle of the second mode. Each node in the area can choose to saves the data of the most recent epoch such as one day or one week, this is ideal for mobile and IoT scenarios, since that the device in these two scenarios have very limited storage capacity and must cooperate with the cloud. Just as shown in Fig.2, the cumulative block size is almost 1 G bytes per week, if the node just store one week or one day data, most mobile device can participate into the consensus process.

The previous data will be stored in the local cloud, edge cloud and the core cloud simultaneously in an encrypted manner . The last hash value of previous block will be stored in the node to maintain consistency. When Cloud databases accept the block ,it will check the hash of the received block and compare with the hash in the block, if it is the same ,the cloud database will send a confirmation to the all the nodes in the area. The nodes will run consensus protocol to make a consensus of the cloud confirmations and then start the next epoch.

There are several advantage of the second modes :

1) Block data is concentrated store on the cloud and it will reduce excessive redundant data in the nodes. In many scenarios , the data maybe not related to the previous data or barely related to previous data ,if the node still store all the data into the node , it will be a waste of the storage ability. In this condition store the data on the cloud is a better and economy choice. As long as the nodes participate into consensus process, the data can't be tampered and the safety is not been trade-off.

2) As the block is stored on the local, edge and core cloud synchronously, even if one or two cloud have been failure ,the system can also operate well. In addition, edge cloud and local cloud is widely deployed in the communication network, it's easy and convenient for the node to store and get the historical block data timely.

E. Transaction Load balancing

In chameleon, we divide into several area, every area can use different consensus protocol. To cope with the emergence of unexpected peak transaction requirements, clients often need to obtain resources that are several times or even more than ten times higher than usual at peak business hours, which puts pressure on the resource utilization of area. Its marginal cost of guaranteeing service quality is getting higher and higher if we simply improve the capacity of area. In Fabric, we have conducted an simulation, when the transaction is overloaded, it will start the view change because of many transactions are not committed in certain period. The view change will degrade the processing performance greatly and the system will be in a vicious circle .We do believe that the transaction level load balancing is very crucial for blockchain to cooperate with other area to share the unexpected peak transaction load. To reduce the processing delay, we assume that the transaction load balancing only happened in adjacent area. When the load exceed the threshold which is dynamic changed according to the different kind of transactions, the load balancing mechanism will be started. There are two phrases in load balancing. First is to select the select light load areas to share the transaction load and the second is to conduct the load balancing mechanism.

Load is defined by the following parameters:

1) Storage utilization ratio(S), $0 \le S \le 1$ # S= Storage utilization /Storage capacity.

2) Transaction utilization ratio (T). $0 \le T \le 1$ # T =Current transactions rate/Transaction throughput

Definition of Load:

$$L=u1*S+ +u_2*T \tag{4}$$

The target optimization function LB is defined by following parameters: In order to ensure that the entire network uses the same metric, the parameters need to be normalized.

1) Processing capacity that can be offered (P)

2) Time delay (T), time delay include the time delay between two areas and the time delay of acquire the previous transaction record in Cloud

3) Storage capacity that can be provided

$$LB=w_1*Pn+ w_2*Tn+ w_3*Sn \tag{5}$$
$$Pn=P/Pmax ,Tn=T/Tmax, Sn=Sn/Smax \tag{6}$$

w_i is decided by the empirical value and the specific scenarios.

Global traffic balancing is an optimization problem. First of all, the CA's balancing algorithm check the load in each area. When load in in one area exceeds the set threshold in continuous time period, The CA will start the transaction load balance mechanism. The algorithm will first search for the maximum load area, and then find the adjacent area to share the transaction load, taking into account factors such as Processing capacity that can be offered , Time delay (T) between the overload area and the corresponding destination area, Storage capacity that can be provided in adjacent area. If the capacity of the destination area can't meet the requirement of the overload area, then the algorithm will find several adjacent area to share the transaction loading. At last, transactions load will be allocated to other areas and the utilization of the entire network resources will increase.

The follow are the pseudo code of the transaction load balancing of Chameleon:

Algorithm of Transaction Load balancing

```
do
    for(i=1;i<=n;i++)                  #there are n area in blockchain network
        Calculate_Load()                  #calculate the load in every area
            Li=u1*S +u2*T                 #the area I transaction load
        if Li>threshold in continuous time period
           return the maximum Li and the adjacent Lj(j=1,2…k) of Li
        do
           for(j=1,j<k,j++)        #k is the number of adjacent area of maximum Lj
               load_balance()
               LBj=w1*Pn+ w2*Tn+ w3*Sn #LBj is the optimization function of area j
               return minimum Lj             # destination area to be diverted
                execute load allocation
         while(( Li-threshold)> (threshold-Lj))  /** make a judgment whether adjacent area
     can provide enough processing power**/
while(Li<threshold)           # update the information and run the algorithm again
```

If there are hundreds of area, we can optimize the algorithm, use the sliding windows methods such as we calculate the five maximum at once and at least five destination area to be diverted at one time. It will reduce the algorithm complexity by five times.

The second phrase is to synchronize the data to the destination areas. Here we explain the mechanism in fig.4:

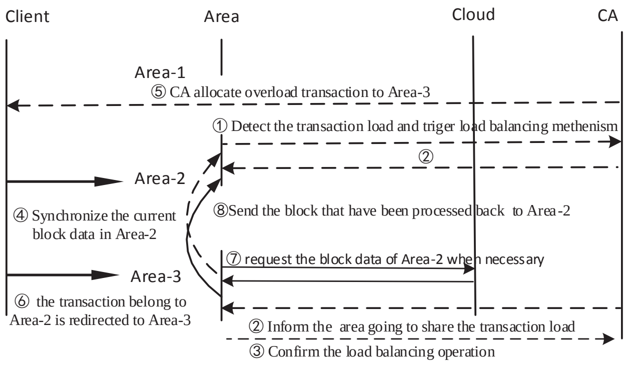
Client
Area
Cloud
CA
Area-1
⑤ CA allocate overload transaction to Area-3
① Detect the transaction load and triger load balancing methenism
②
Area-2
④ Synchronize the current block data in Area-2
⑧Send the block that have been processed back to Area-2
⑦ request the block data of Area-2 when necessary
Area-3
⑥ the transaction belong to Area-2 is redirected to Area-3
② Inform the area going to share the transaction load
③ Confirm the load balancing operation

Fig. 4 the transaction load balancing process

Area-1 and Area-3 are adjacent to area-2. First, CA will monitor the load in every area, when the Area-2 is overloaded, CA will trigger load balancing mechanism and choose area 3 to share the transaction load. Second, CA will inform the Area-3 to prepare for processing the transaction form Area-2 and also inform the Area-2 of the area going to share the transaction load. Third, Area-3 confirm the load balancing operation and tell the CA it's ready. Fourth, Area-3 will synchronize the current block data in Area-2.it is similar with the Elastic State Machine Replication[8].The leader in area2 will send the data using multiple signatures to the leader in the area3 to guarantee the integrity and the correctness, and the destination area will broadcast the data to all nodes. Fifth, the CA will allocate overload transactions to Area-3 through changing the transaction type to cooperative transaction shown in TABLE I. Sixth, Client will send the cooperative transactions to Area-3. Seventh, if the transactions are related with the previous transactions, Area-3 will get previous transactions from cloud. Finally, the block that have been processed by Area-3 will be sent back to Area-2 with multiple signatures. Area-2 will validate the block and store the block into the node. Repeat the above process until the there is no overload transactions in Area-2.

IV. SIMULATION

We conduct a group of simulations to verify the feasibility and efficiency our load balancing process in Chameleon on the MATLAB platform. In our simulation, we use Poisson process which is widely used in client and server architecture to simulate the transaction arriving rate. To be simple, Load is positive correlation function related with transaction arriving rate per second, if the load exceed 80% in certain period, the load balancing mechanism will be triggered. We also assume all area have same processing ability, i.e.400 transactions per second, a realistic parameter come from realistic testing on Fabric with four nodes using the PBFT consensus algorithm. We testing the transaction throughput in our laptop using Ubuntu 16.04.03 on VMware Workstation14, our laptop is configured with intel i7-7700HQ.

We assess the performance and efficiency of the load balancing mechanism in Chameleon by comparing the load distribution before and after load balancing in every area under different load conditions. As load is constructed by the storage and the transaction utilization ratio, which also refer the resource utilization ratio. There are five areas in our simulation and they are adjacent to each other.

The following figures are the simulation results:

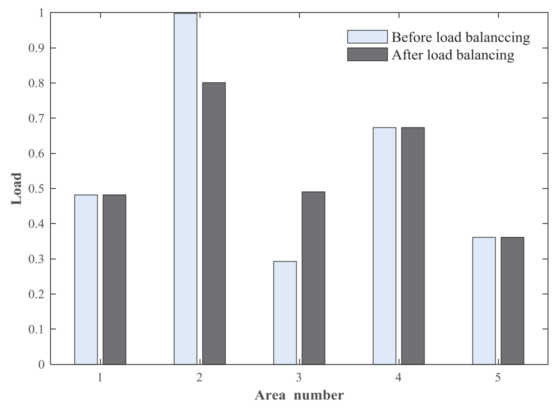

Fig.5 Transaction rate is 400 in Area2

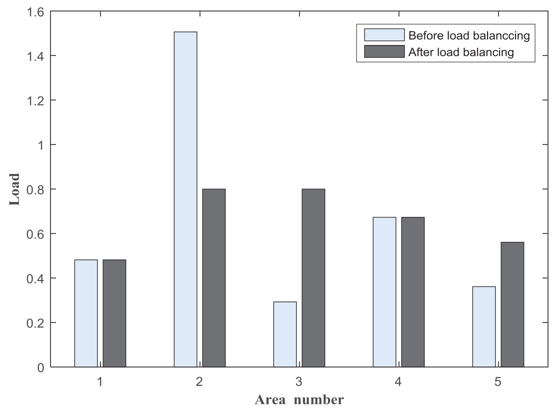

Fig. 6 Transaction rate is 600 in Area-2

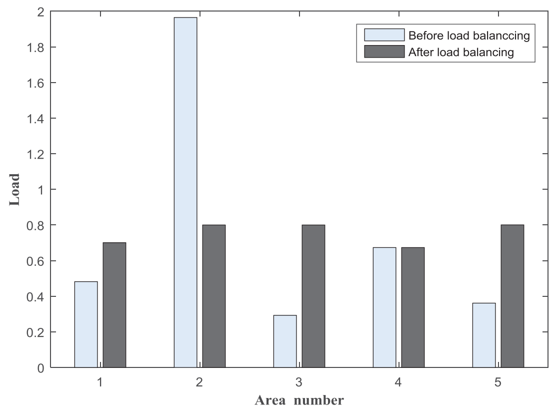

Figure 7 Transaction rate is 800 in Area-2

As shown in Fig.5, Transaction rate Area-2 is 400 which is the maximum transaction processing rate, obviously, it is overloaded. The system will trigger the load balancing mechanism and will chose the Area-3 to share the overload transactions because the load in Area-3 is the smallest and it can provide the maximum processing capacity than any other areas. We can find the load in Area-2 have drop to 0.8 and the load in Area-3 have increased to 0.5 after load balancing.

When the load continue grown to 600 transaction rate in the Fig.6, It will trigger the load balancing mechanism as well, the CA will calculate the load balancing area to share the overload transactions in Area-2. Just like the Fig.5, the CA will choose Area-3, but found that the Area-3 can’t offer enough processing ability and have to choose another area, that’s the Area-5. After this, the overload transactions in area-2 will be shared to Area-3 and Area-5. We can find that after load balancing, the load in Area-2 have drop to the 0.8 and the load in Area-3 and Area-5 have increase.

As shown in Fig.7, The overload area-2 is 800, it’s the double of the process ability in Area-2 which is maybe happen in certain situation. The CA will calculate the load balancing area to share the overload transactions after detecting the overload. Finally, it will choose the area-3, area-5 and area-1 to share the load just as shown in Fig.7. After load balancing we can find that the load is evenly distributed in every area.

As shown in Fig.5, Fig.6 and Fig.7, The entire load balancing process is very clear, i.e. which area is the first to choose, which area is the last to choose. The comparison of the load distribution before and after the load balancing in simulation result show that our load balancing mechanism in Chameleon can cope with the large amount of bursting transactions and can also greatly improve the resource utilization .

V. CONCLUSION

In this paper, we introduce the a scalable and adaptive permissioned blockchain architecture Chameleon. First, We design the transaction format to guarantee the QoS of transactions ,so that the nodes can process the transactions according to their priority. Second, we improved the PBFT by introduce the randomness in leader selection based on credit value , this greatly improve the security of consensus process. Third, we combine the advantage of cloud to reduce the excessive block data in node, this is very useful in many non-financial scenarios. Finally, we introduce the transaction level load balancing mechanism and have a simulation to verify the performance of our system.

Although Chameleon have many advantages compared with existing block chain architecture, it is still a proof-of-concept. We leave to future work of the implementation of Chameleon. Additionally, how to reduce the overhead when one area share the transactions with the other area under the overload situation need to be further studied in the future. Furthermore, to avoid the CA becoming the bottleneck of the system, parallel processing in the CA also need to be further researched in the future work.

VI. ACKNOWLEDGMENT

This work is supported by the Fundamental Research Funds for the Central Universities under Grant No.2018YJS009 and 973 program under Grant No. 2013CB329101.

References

[1] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 17-30.

[2] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, and B. Ford, "OmniLedger: A Secure, Scale-Out, Decentralized Ledger," IACR Cryptology ePrint Archive, vol. 2017, p. 406, 2017.

[3] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in Proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 51-68.

[4] C. Cachin, "Architecture of the Hyperledger blockchain fabric," in Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.

[5] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in OSDI, 1999, pp. 173-186.

[6] M. Ali, "Trust-to-trust design of a new Internet," Princeton University, 2017.

[7] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in Security and Privacy (SP), 2017 IEEE Symposium on, 2017, pp. 375-392.

[8] A. Nogueira, A. Casimiro, and A. Bessani, "Elastic state machine replication," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 2486-2499, 2017.